**A PROPOSED OFFERING OF WEB-BASED RESERVATION SYSTEM FOR THE ADDLIB DANCE STUDIO AT DATAMEX COLLEGE OF SAINT ADELINE VALENZUELA BRANCH**

A Research Project Presented to the

Faculty of Datamex College of Saint Adeline, Inc.

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in Information Technology

Bacani, Princess Mahallalel  
Delos Angeles, Arjay O.  
Mejila, Steven Carl A.  
Norte, Sakura Ann

**TECHNICAL DOCUMENTATION**

**INTRODUCTION**

**Purpose of the Document**

The purpose of this document is to present a complete and detailed explanation of the Adlib Dance Studio Reservation System. It aims to serve as a reference for users, developers, and administrators by providing a clear understanding of how the system functions, how it was developed, and how it will be maintained. This documentation also ensures that future improvements or updates can be implemented efficiently by giving a well-structured guide to the system’s design and operation.

In addition, this document aims to support the academic purpose of the project by showing the technical and practical aspects of the system. It highlights how technology can be used to simplify studio reservation processes and promote a more organized management system for dance studios.

**Overview of the Software System**

The Adlib Dance Studio Reservation System is a web-based application created to improve the reservation process for dance studios. The system allows clients to book available studio slots using a QR code that redirects them to the online reservation form. This modern method replaces the traditional manual booking process, reducing human error and saving time for both customers and administrators.

The system also integrates a GCash payment feature, which enables users to make secure, cashless payments directly through the platform. This feature not only increases convenience but also supports a safer, faster, and more reliable transaction process.

For the backend, the system uses Supabase as its main database to store and manage user information, booking records, and payment details securely. Through this technology, data is synchronized in real time, ensuring accuracy and consistency across devices.

**Scope of the Technical Documentation**

This technical documentation includes all important aspects of the Adlib Dance Studio Reservation System. It discusses the system’s structure, features, technical components, and maintenance plans to ensure smooth operation.

* The system design and how each component interacts.
* The software and hardware requirements necessary for deployment.
* The database setup and configuration, including how data is managed using supabase.
* The integration process of QR code and GCash payment systems.
* The procedures for maintaining, updating, and securing the system.
* The guidelines for troubleshooting common issues and performing regular system checks.

**SYSTEM OVERVIEW**

**System Architecture**

The Adlib Dance Studio Reservation System is a web-based application designed to streamline the process of booking dance studio sessions. The system follows a client-server architecture, where users interact with the web interface (client side), and all data transactions are processed and stored in a cloud-based database (server side) using Supabase.

The system architecture ensures that users can make reservations, process payments, and receive confirmation messages efficiently through an internet connection. The design prioritizes real-time data synchronization, meaning that any updates made by users or administrators are automatically reflected in the system without delay.

The architecture also supports secure communication between the user interface and the database to protect sensitive information, such as user profiles and GCash payment data.

**High-Level Components and Their Interactions**

The system is composed of several main components that work together to provide a smooth and reliable reservation experience:

**User Interface (Front-End):**  
The front-end is the part of the system that users interact with. It is developed using HTML, CSS, and JavaScript, providing a simple and responsive design. Through the interface, users can scan a QR code to access the reservation form, select available studio slots, and confirm payments using GCash.

**Application Logic (Back-End):**  
The back-end handles all processes that occur behind the scenes. It manages booking requests, verifies payment transactions, and communicates with the database. It ensures that each reservation follows the correct process—from submission to confirmation.

**Database Management (Supabase):**  
Supabase serves as the system’s centralized database. It stores all important data, including user profiles, reservation details, schedules, and payment records. Supabase also provides authentication services for secure login and user verification.

**Payment Integration (GCash):**  
The system integrates with GCash to enable cashless and secure payments. Once a user completes a transaction, the payment details are sent to the Supabase database for confirmation and recordkeeping.

**QR Code Module:**  
This feature generates and scans QR codes for easy access to the reservation page. It helps users quickly navigate to the booking system without manually entering URLs or credentials.

**Admin Dashboard:**  
The admin panel allows authorized personnel to manage bookings, monitor payment records, update schedules, and handle system maintenance. It serves as the control center for all operational activities within the system.

**DEPLOYMENT ARCHITECTURE**

The deployment architecture of the system follows a cloud-hosted structure to ensure accessibility, scalability, and reliability. The system is deployed using Supabase Hosting for the web application and Supabase Realtime Database for data storage.

**Client-Side Deployment:**The web application can be accessed through any device with an internet connection and a web browser. Users scan the QR code, which directs them to the online booking page hosted on Supabase.

**Server-Side Deployment:**  
The back-end functions are handled through Supabase services, which include authentication, data storage, and cloud functions. These services manage real-time interactions and maintain system security.

**Payment Integration Deployment:**  
The GCash payment gateway is linked to the application through secure API connections. This ensures that payment information is encrypted and verified before confirmation.

**INSTALLATION GUIDE**

**System Requirements**

Before installing the Adlib Dance Studio Reservation System, it is essential to ensure that both the hardware and software meet the required specifications. Meeting these requirements will guarantee the smooth performance and compatibility of the system.

**Hardware Requirements**

|  |  |  |
| --- | --- | --- |
| Component | Minimum Requirement | Recommended Requirement |
| Processor | Intel Core i3 or equivalent | Intel Core i5 or higher |
| Memory (RAM) | 4 GB | 8 GB or more |
| Storage | 2 GB of free disk space | 5 GB of free disk space |
| Display | 1366×768 resolution | 1920×1080 resolution |
| Internet Connection | Stable connection | High-speed connection |

**Software Requirements**

|  |  |
| --- | --- |
| Software | Version/Details |
| Operating System | Windows 10 or higher / macOS / Linux |
| Web Browser | Google Chrome, Mozilla Firefox, or Edge (latest version) |
| Code Editor | Visual Studio Code or equivalent |
| Node.js | Version 18 or above |
| GCash API Access | Registered API account |

**Dependencies**

The following dependencies must be installed for the system to function properly:  
- Supabase SDK (for database, hosting, and authentication)  
- Express.js (for server-side functions)  
- QR Code Generator and Scanner library  
- GCash API module (for payment integration)  
- dotenv package (for environment variable management)

**Step-by-Step Installation Instructions**

Step 1: Download Project Files

Obtain the source code from the system repository or developer and save it to your computer.

Step 2: Install Node.js and Dependencies

Download Node.js from https://nodejs.org, then open Command Prompt and navigate to the project directory:  
 cd adlib-dance-studio-reservation  
Install dependencies using: npm install

Step 3: Configure Firebase

Create a Supabase project, enable Authentication, Realtime Database, and Hosting. Add configuration keys to your .env file.

Step 4: Set Up GCash API

Register a GCash API account, obtain credentials, and add API key and secret to the .env file.

Step 5: Generate QR Code

Use the built-in QR code generator to create a QR code linking to the reservation page.

Step 6: Launch the System

Start the server with: npm start  
Visit http://localhost:3000 to access the system or deploy using: supabase deploy.

**Configuration Settings and Options**

|  |  |  |
| --- | --- | --- |
| Configuration | Description | Default Setting |
| Database URL | Supabase Realtime Database connection | Provided by supabase |
| Authentication | Email and password / Google sign-in | Email and password |
| Payment Mode | Online payment through GCash API | Enabled |
| QR Code Link | URL embedded in QR code | Firebase-hosted link |
| Admin Access | System administrator credentials | Defined during setup |
| Backup Schedule | Automatic database backups | Weekly |

**CONFIGURATION GUIDE**

The configuration process involves setting up system parameters, API credentials, and interface customization.

Configuration File Format and Parameters:

• config.js – Contains Firebase and payment gateway credentials.

• app.json – Defines UI settings and environment configurations.

Best Practices: Always back up configuration files before making changes, and test modifications in a development environment first.

**API DOCUMENTATION**

**Introduction**

The API documentation provides detailed information about the interfaces and endpoints used in the Adlib Dance Studio Reservation System. This document aims to assist developers and administrators in understanding how to interact with the system’s backend services for managing reservations, payments, and user data.

The system uses Supabase as its primary database, offering secure data storage, authentication, and real-time synchronization. The APIs facilitate communication between the client-side application, the database, and third-party services such as GCash for digital payments.

**Overview of the API System**

The API acts as the communication bridge between the front-end interface and the Supabase backend. It manages user data, reservations, and payment requests. The design follows RESTful architecture, ensuring scalability, maintainability, and consistent data flow.

Key Functions:

* Handles user registration and login.
* Manages reservation data in real time.
* Processes payments through GCash.
* Generates and validates QR codes for reservation verification.

**Authentication and Authorization**

All API endpoints require user authentication through JSON Web Token (JWT) issued by Supabase Authentication.  
Only verified users can access restricted endpoints such as booking history, payment details, or reservation management.

Authentication Details:

* Login Required: Yes
* Authorization Header: Bearer <access\_token>
* Token Expiration: 1 hour (auto-refresh available via Supabase Auth
* Security Protocol: HTTPS encryption for all data transmissions.

**User Management APIs**

| Endpoint | Method | Description | Request Parameters | Response |
| --- | --- | --- | --- | --- |
| /api/register | POST | Registers a new user to the system. | { "email": "", "password": "", "name": "" } | { "message": "Registration successful" } |
| /api/login | POST | Authenticates a user and returns an access token. | { "email": "", "password": "" } | { "token": "<jwt\_token>" } |
| /api/profile | GET | Fetches the logged-in user’s details. | Authorization: Bearer <token> | { "name": "", "email": "" } |

*Table 1. List of User Management APIs*

**Reservation APIs**

| Endpoint | Method | Description | Request Parameters | Response |
| --- | --- | --- | --- | --- |
| /api/reservations | GET | Retrieves all reservations for the current user. | Authorization: Bearer <token> | List of reservation records. |
| /api/reserve | POST | Creates a new dance studio reservation. | { "date": "", "time": "", "studio": "" } | { "message": "Reservation confirmed", "qr\_code": "<qr\_link>" } |
| /api/cancel/:id | DELETE | Cancels a specific reservation. | Reservation ID | { "message": "Reservation cancelled" } |

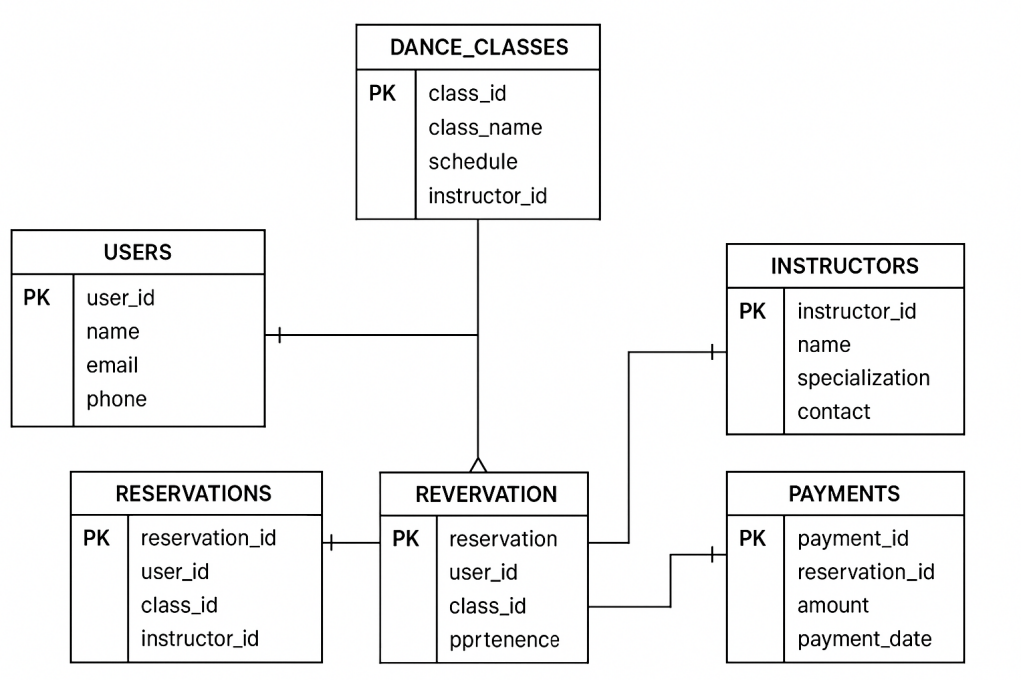
*Table 2 . List of Reservation APIs*

**DATABASE DOCUMENTATION**

The system uses Supabase as its primary database. It stores data related to users, reservations, payments, and schedules.The database structure includes the following collections: Users, Reservations, Payments, and Feedback.Backups are automatically stored in the Supabase cloud storage weekly.

**Entity-Relationship Diagram (ERD)**

The database of the Adlib Dance Studio Reservation System is designed using Supabase, a cloud-based database that provides real-time data handling and authentication features.  
The Entity-Relationship Diagram (ERD) shows how the main entities connect to each other.



*Image 1 . Entity Relationship Diagram*

**Main Entities:**

* Users – stores information about clients and staff members.
* Reservations – holds details of class bookings, including date, time, and chosen instructor.
* Payments – records transactions made through GCash.
* Dance\_Classes – includes data about the available dance sessions, types, and schedules.
* Instructors – keeps instructor profiles and class assignments.

**Relationships:**

* One User can make many Reservations.
* Each Reservation is linked to one Dance\_Class and one Instructor.
* Each Reservation has one Payment record.
* One Instructor can teach many Dance\_Classes.

**Description of Database Tables**

| Table Name | Description | Key Fields |
| --- | --- | --- |
| Users | Contains information about customers and staff. | user\_id (PK), name, email, phone, role |
| Dance\_Classes | Lists available dance classes and details. | class\_id (PK), class\_name, schedule, instructor\_id |
| Instructors | Stores instructor information. | instructor\_id (PK), name, specialization, contact |
| Reservations | Tracks all booking transactions. | reservation\_id (PK), user\_id (FK), class\_id (FK), instructor\_id (FK), date, status |
| Payments | Manages payment records via GCash. | payment\_id (PK), reservation\_id (FK), amount, payment\_date, payment\_status |

*Table 3. List of Database*

**Data Migration and Backup Procedures**

1. **Data Migration**

* Data from earlier versions or local storage can be imported into Supabase using CSV or SQL import tools.
* All records are validated before insertion to ensure correct formatting and no duplication.

1. **Backup Procedures**

* Automated backups are scheduled weekly through Supabase’s built-in backup system.
* Data is stored in encrypted cloud storage for security.
* In case of accidental data loss, recovery can be done by restoring the most recent backup from the Supabase dashboard.

**USER MANUAL**

**Introduction**

The Adlib Dance Studio Reservation System is designed to simplify the booking process for dance classes by allowing users to make reservations online through a QR code. The system integrates with GCash for secure and cashless payments and is supported by the Supabase database for efficient data management.

**2.1 System Access**

* Open the QR code provided by Adlib Dance Studio using your mobile device.
* Scan the QR code using your phone camera or any QR scanner app.
* You will be redirected to the studio’s official reservation webpage.

**2.2 Account Registration**

* Click Sign Up on the homepage.
* Provide your name, email address, and mobile number.
* Set a password and confirm it.
* After registration, you may log in using your credentials.

**2.3 User Login**

* On the main page, click Login.
* Enter your registered email and password
* Click Submit to access your user dashboard.

**User Interface Overview**

* Dashboard: Displays available classes, schedules, and your current reservations.
* Class List: Shows dance class options (e.g., Hip-hop, Ballet, Zumba) with available slots.
* Reservation Page: Allows users to book a session and confirm payment.
* Payment Section: Integrates with GCash for seamless payment transactions.
* Profile Section: Enables users to view and edit personal information.

**Common Tasks and Workflows**

**4.1 Making a Reservation**

* Log in to your account.
* Browse the available dance classes and select your preferred schedule.
* Click Reserve Now.
* Confirm your reservation details.
* Proceed to GCash Payment and complete the transaction.
* A confirmation message and receipt will appear once the payment is successful.

**4.2 Viewing Reservation History**

* Navigate to My Reservations from the dashboard.
* View all your active and past reservations, including payment status and class details.

**4.3 Canceling a Reservation**

* Go to My Reservations.
* Select the booking you wish to cancel.
* Click Cancel Reservation.
* A confirmation prompt will appear before completing the cancellation.

**4.4 Editing Profile Information**

* Access the Profile section from the menu.
* Update your name, contact number, or password.
* Click Save Changes to confirm.

**Support and Assistance**

If you encounter issues while using the system, you may contact:

Technical Support Team: letsdance@tadsph.com

Office Hours: Monday to Friday, 9:00 AM – 6:00 PM

**TROUBLESHOOTING GUIDE**

Common Issues and Fixes:

• Error: GCash payment not processing – Check internet connection or API key validity.

• Error: QR code not scanning – Ensure camera permissions are enabled.

If problems continue, contact the system administrator or support team for assistance.

**CODE DOCUMENTATION**

The code is organized into modules for easy maintenance: user management, reservation, and payment processing.

Inline comments explain the logic of each function, and standard coding conventions are followed for clarity and readability.

**TESTING DOCUMENTATION**

A test plan was created to ensure the system meets all functional and performance requirements.

Functional tests cover login, booking, and payment features, while non-functional tests focus on speed, security, and usability.

All identified defects are logged and tracked until resolved.

**MAINTENANCE GUIDE**

Maintenance includes regular software updates, bug fixes, and system performance checks.

Version control is managed through Git, with releases documented for tracking changes.

Enhancements and patches should be tested before implementation to avoid system disruptions.