**A PROPOSED OFFERING OF BUS TICKETING AND BOOKING SYSTEM**

**FOR VICTORY LINER INC.**

A Project Proposal 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

By:

Arellano, Romeo C.

Estigoy, Ron Kyrie

Flores, Joserey G.

Gadi, Susana

October 2025

**TECHNICAL DOCUMENTATION**

**INTRODUCTION**

**Purpose of the Testing Phase:**

The purpose of testing the Bus Ticketing and Booking System is to ensure that all functionalities work as intended, providing users with a reliable, secure, and efficient platform for booking and managing bus tickets. The testing phase aims to identify and correct any defects before deployment.

**Objectives of the Testing Process:**

Verify that all system modules (user registration, booking, payment, and management) function correctly.

Ensure data accuracy, security, and system performance.

Validate that the system meets both functional and non-functional requirements.

Confirm compatibility across different devices and browsers.

**Scope of the Testing:**

The testing covers all major components of the system including user login, seat selection, payment processing, and report generation.

**In-Scope:** Functional testing, integration testing, performance testing, and user acceptance testing.

**Out-of-Scope:** Third-party payment gateway integration testing and hardware stress testing beyond standard configurations.

**Testing Environment**

**Hardware Specifications:**

Processor: Intel Core i5

RAM: 16 GB

Storage: 128 GB

Operating System: Windows 10 pro

Devices Used: Desktop computer and Android mobile phone

**Software Requirements:**

Web Server: Apache / XAMPP

Database: MySQL 8.0

Development Tools: PHP 8.2, HTML, CSS, JavaScript

Browser: Google Chrome, Mozilla Firefox

Testing Tools: Postman (API testing), Selenium (UI testing)

**Test Data:**

Sample data sets were created to simulate real-world usage, including:

Sample users with valid and invalid credentials.

Sample bus schedules with different destinations and times.

Booking data with various seat and date combinations.

Payment records with successful and failed transactions.

**Testing Methodology**

**Testing Approaches Used:**

**Black-box testing:** To verify system functionality without inspecting code.

**White-box testing:** Applied by developers to test internal logic and database queries.

**User acceptance testing (UAT):** Conducted by end users to ensure the system meets requirements and is user-friendly.

**Testing Tools and Frameworks:**

Postman for testing API endpoints and response handling.

Selenium WebDriver– for automated UI testing.

**Test Cases and Testing Criteria:**

**Login Module:** Verify successful and failed login attempts.

**Booking Module:** Validate seat availability and reservation confirmation.

**Payment Module:** Confirm successful and failed payment scenarios.

**Admin Dashboard:**Check correct display of data analytics and reports.

**Performance Testing:** Ensure the system handles multiple concurrent users without lag.aged relational DB with read replicas, multi-AZ for high availability; Redis (managed) for caching and locks; S3-compatible object storage for static assets.

**CI/CD:** Automated pipelines for build, test, and deploy; isolated staging and production environments; canary or blue/green deployments for zero-downtime releases.

**High Availability & Disaster Recovery:** Multi-AZ deployments, automated backups, point-in-time recovery for DB, and health checks with automatic failover strategies.

**Security:** VPC isolation, private subnets for databases, IAM roles, secrets management, WAF for protecting public endpoints, and regular security scans.

**Configuration Guide**

**Detailed instructions for configuring the software:**

The Bus Ticketing and Booking System can be configured by setting up the server environment, installing the required dependencies, and initializing the database. Configuration includes specifying database connection details (host, username, password, and database name) in the configuration file.

**Configuration file formats and parameters:**

`DB\_HOST` – Database server address

`DB\_USER` – Database username

`DB\_PASS` – Database password

`DB\_NAME` – Database name

`API\_KEY` – Authentication key for API access

`PORT` – Application running port

**Best practices for customization:**

Modify configuration files carefully and back them up before applying changes.

Avoid hardcoding sensitive credentials.

Use version control to track configuration updates.

Adjust settings according to the number of expected users to maintain performance and scalability.

**API Documentation**

**List of APIs exposed by the system:**

`/api/users` – Manages user registration and authentication.

`/api/buses` – Retrieves available bus schedules and details.

`/api/bookings` – Handles booking creation, updates, and cancellations.

`/api/payments` – Processes and verifies ticket payments.

**Database Documentation**

Users – Stores user details (user\_id, name, email, password).

Buses – Contains bus information (bus\_id, route, departure\_time, seats).

Bookings– Links users to their bus tickets (booking\_id, user\_id, bus\_id, seat\_number).

Payments – Records payment details (payment\_id, booking\_id, amount, status).

**Relationships:**

One User can have many Bookings.

One Bus can have many Bookings. One Booking has one Payment.

Description of database tables, fields, and relationships:

**Data migration and backup procedures:**

**Migration:**Use database migration tools or SQL scripts to transfer data between environments (e.g., from development to production).

**Backup:** Regular automated backups scheduled daily or weekly. Store backups securely using cloud storage or an external server.

**Restore:** In case of data loss, use the latest backup file to restore the database using restore commands or management tools (e.g., `mysqldump`, `pg\_restore`).ally, it is assumed that only authorized personnel, such as agents, managers, and administrators, will have access to the system

**User Manual**

**Instructions for using the software:**

Users can register or log in to the system, search for available buses, select travel dates, choose seats, and proceed with booking and payment. Admins can add or manage buses, schedules, and user accounts.

**User interface descriptions and navigation guidelines:**

Home Page: Displays available routes and upcoming trips.

Booking Page: Allows users to input travel details, select seats, and confirm booking.

Payment Page: Provides secure payment options.

Admin Dashboard:Displays reports, passenger lists, and system statistics.

**Common tasks and workflows:**

1. User logs in or registers an account.

2. User selects desired route and schedule.

3. System displays available seats.

4. User confirms booking and proceeds to payment.

5. Ticket confirmation and booking receipt are displayed.

**Troubleshooting Guide**

**Common issues and error messages:**

Invalid login credentials – User entered wrong username or password.

Seat already booked– Selected seat is no longer available.

Payment failed – Payment gateway issue or insufficient balance.

**Troubleshooting steps and resolutions:**

Verify credentials and reset password if necessary.

Refresh the booking page to view updated seat availability.

Check internet connection or retry payment with a different method.

**Contact information for technical support:**

Email: [support@busticketingsystem.com](mailto:support@busticketingsystem.com)

\* Hotline: +63 900 123 4567

**Code Documentation**

**Code structure and organization:**

The system follows the MVC (Model-View-Controller) architecture for maintainability and scalability.

**Model:** Handles database operations and logic.

View: Manages user interface and front-end components.

**Controller:** Connects model and view to process user requests.

**Inline comments explaining key functions and logic:**

All major functions are documented with inline comments to describe inputs, processes, and outputs for clarity.

**Coding standards and conventions:**

Variable names use camelCase.

Functions and classes follow clear naming conventions.

Code is formatted for readability and version-controlled via Git.

**Testing Documentation**

Test plan outlining testing objectives and strategies:

Testing ensures system reliability, usability, and security. The main objectives include verifying correct booking, payment processing, and user management functionalities.

**Test cases covering functional and non-functional requirements:**

Functional: Login authentication, booking flow, payment confirmation.

Non-functional: System response time, load testing, data accuracy.

**Test results and defect reports:**

Each test case includes expected and actual results. Any defects identified are logged with status (Pending, Fixed, Verified) and reassigned for resolution.

**Maintenance Guide**

Procedures for maintaining and updating the software:

Regular updates are released to improve system performance, security, and functionality. Maintenance tasks include data cleanup, database optimization, and dependency updates.

**Version control and release management practices:**

GitHub or GitLab repositories are used for version control.

Each release is tagged and documented with a changelog.

**Guidelines for handling bug fixes and enhancements:**

Issues are logged in a tracker system.

Critical bugs are prioritized and fixed before new features are deployed.

**12. Revision History**

**Revision History**

|  |  |
| --- | --- |
| Dates | Description of revision |
| 08-18-25 | Make an figure agile |
| 08-18-25 | Title of the System |
| 08-18-25 | The system make an offline |
| 08-18-25 | Make an times new roman font |
| 08-18-25 | Make an justify every sentence |

**Appendix**

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7th ed.). Pearson.

GeeksforGeeks. (2021, July 20). *Database management system (DBMS) tutorial*. GeeksforGeeks. https://www.geeksforgeeks.org/dbms/

Joyanes Aguilar, L. (2011). *Programming with Visual Basic .NET*. McGraw-Hill.

Microsoft. (n.d.). *ADO.NET overview*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/

Microsoft. (n.d.). *Visual Basic documentation*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/visual-basic/

Pressman, R. S. (2014). *Software engineering: A practitioner’s approach*(8th ed.). McGraw-Hill Education.

Stack Overflow. (n.d.). *VB.NET and SQL Server integration examples*. Stack Overflow. https://stackoverflow.com/

Tutorials Point. (n.d.). *VB.Net tutorial*. Tutorials Point. https://www.tutorialspoint.com/vb.net/

W3Schools. (n.d.). *SQL tutorial*. W3Schools. https://www.w3schools.com/sql