**A PROPOSED OFFERING OF A GYM MANAGEMENT**

**SYSTEM FOR ANYTIME FITNESS GYM**

A Technical Documentation 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:

Revilla, James Andrei N.

**INTRODUCTION**

This section introduces the Gym Management System (GMS) and explains the purpose, overview, and scope of this technical documentation. It provides the foundation for understanding why the system was developed and how this document should be used by developers, testers, administrators, and other stakeholders.

The purpose of this technical documentation is to serve as a comprehensive reference guide for the Gym Management System. It is intended to cover every aspect of the software, from its architecture and database schema to installation procedures, configuration guidelines, and operational instructions. This document will ensure that anyone who needs to work with the system—whether for development, deployment, or daily use—has a clear and structured resource.

The Gym Management System is a Windows-based application designed for Anytime Fitness Gym. It was developed using Visual Basic 2010 (VB.NET) and utilizes MySQL as its backend database, hosted on the XAMPP stack. The primary goal of this system is to automate and streamline gym operations that were previously handled manually. This includes staff authentication, member registration and tracking, attendance monitoring, equipment inventory, payment management, and generating administrative reports.

By implementing the GMS, the gym is expected to experience increased efficiency, better accuracy of records, reduced human error, and improved data security. Furthermore, the centralized system allows for more effective monitoring and decision-making by administrators.

The scope of this documentation is broad. It covers the system design and architecture, installation and configuration, database documentation, user manual, troubleshooting, coding standards, testing methodology, and maintenance. Each section is carefully structured to give both technical and non-technical stakeholders the knowledge they need to operate, maintain, and improve the system.

**SYSTEM OVERVIEW**

This section is about providing a **general overview of the Gym Management System’s architecture, components, and deployment strategy**. It describes how the system is structured internally and how its different parts interact with each other.

**System Architecture**

The Gym Management System employs a monolithic client-server architecture. The desktop application written in Visual Basic 2010 encompasses presentation and business logic layers while communicating directly with a MySQL database backend via ADO.NET. The absence of intermediate web APIs simplifies the architecture and facilitates direct, performant communication between the client and the data repository.

The architecture distinctly separates concerns into three layers:

1. The Presentation Layer provides intuitive user forms for staff interaction, including login, dashboards, data entry, editing, and reporting.
2. The Business Logic Layer encapsulates validation rules, workflows, membership status determinations, payment processing, and analytical computations.
3. The Data Access Layer handles all interactions with the MySQL database, executing parameterized SQL queries and managing transactions to maintain data consistency.

**High-Level Components and their Interactions**

Core system modules include Authentication, Member Management, Equipment Management, Attendance Tracking, Payment Handling, Dashboard Reporting, and Staff Account Administration. Each module interacts cohesively within the application to provide seamless workflows. For instance, the process of adding a new member combines data validation, payment confirmation, database insertion, and immediate dashboard update.

**Deployment Architecture**

Deployment is realized on Windows client machines where the VB.NET application runs locally, connecting over a secure TCP/IP channel to the MySQL database hosted on a local or networked server under the XAMPP stack. Network access is protected by firewalls and optionally VPNs to safeguard against unauthorized external access.

**Communication Protocols and Interfaces**

Communication between the client and database employs MySQL’s native protocol over port 3306. Security is enforced through encrypted password storage, two-factor authentication leveraging stored security questions, and parameterized queries that mitigate risks of SQL injection attacks.

Advanced features such as connection pooling enable efficient handling of multiple database operations, improving scalability. Error handling is integrated at both code and interface layers to ensure robust system stability and user guidance in case of failures.

**INSTALLATION GUIDE**

This section is about the installation process for the Gym Management System, including system requirements, installation steps, and configuration options. It ensures that the system can be deployed in a standardized and reliable manner.

**System Requirements**

The system requires both client and server environments.

| **Component** | **Minimum Specification** |
| --- | --- |
| Operating System | Windows 10 or later |
| Framework | .NET Framework 4.x |
| Processor | Dual-core or better |
| Memory | 4 GB RAM |
| Storage | 2 GB free disk space |

**Table 1.** Client Requirements

| **Component** | **Minimum Specification** |
| --- | --- |
| Operating System | Windows/Linux with XAMPP |
| Database | MySQL via XAMPP |
| Memory | 8 GB RAM |
| Storage | SSD recommended |
| Network | LAN with firewall enabled |

**Table 2.** Server Requirements

**Installation Steps**

1. Install XAMPP on the server machine and enable the MySQL service.
2. Create a database named gymdb and import the provided schema.
3. Install the GMS executable file on client machines.
4. Configure the connection string in the application’s config file to match the server database.
5. Test the installation by logging in with the default administrator account.

**Configuration Settings**

**Configuration is an essential part of deployment. The administrator must:**

* **Set database credentials in the encrypted config file.**
* **Add staff accounts through the Account Management module.**
* **Define membership plans, payment options, and equipment records.**
* **Configure backup scripts for the database.**
* **Enforce password policies and two-factor authentication for users.**

**CONFIGURATION GUIDE**

This section explains the customization and configuration options of the Gym Management System (GMS). After installation, administrators must configure the system to align with gym policies and operational needs.

**Staff Accounts**

* **Account Creation** – Administrators can create new staff accounts through the Account Management module.
* **Role Assignment** – Each account can be assigned a role (e.g., Admin, Manager, Staff) to enforce role-based access control.
* **Status Management** – Accounts can be set to *Pending*, *Active*, or *Suspended* depending on approval and usage.
* **Two-Factor Authentication (2FA)** – Administrators can enable 2FA for sensitive accounts.
* **Security Questions** – Configurable during account creation for password recovery.

**Membership Plans**

* **Predefined Plans** – The system includes Monthly, 3-Months, 6-Months, and Annual plans.
* **Customization** – Duration and pricing can be modified by administrators.
* **Promotions/Discounts** – Plans can be updated or extended to include seasonal offers.

**Add-On Services**

* **Service Setup** – Additional services (e.g., Personal Training, Nutrition Coaching, Locker Rent) are configurable.
* **Billing Options** – Each service can be billed *Per Session* or *Monthly*.
* **Quantity Tracking** – Services such as multiple training sessions can be assigned with quantities.

**Equipment Records**

* **Inventory Management** – Equipment details such as name, category, purchase date, and vendor can be recorded.
* **Condition Tracking** – Equipment status can be updated as *Good*, *Needs Repair*, or *Out of Service*.
* **Vendor Information** – Optional fields for supplier contact details support future maintenance needs.

**Security and Recovery**

* **Password Policies** – Strong password enforcement and expiry rules are configurable.
* **Account Suspension** – Inactive or compromised accounts can be disabled by administrators.
* **Data Recovery** – Security question–based recovery is available for users who forget credentials.

**Backup and Maintenance**

* **Database Backups** – Daily automated backups should be configured using XAMPP or MySQL tools.
* **Backup Location** – Backups can be stored locally or on external drives/servers.
* **Verification** – Regular restore tests are recommended to confirm data integrity.

**API DOCUMENTATION**

This section provides an overview of the **API capabilities** of the Gym Management System. It explains whether the system currently exposes APIs, outlines possible future integrations, and describes the standards and formats that will be followed. The purpose of this section is to guide developers and integrators on how the system can connect with external services such as payment gateways, SMS platforms, or email servers.

**List of Third-Party Services or APIs Integrated into the System**

Currently, the Gym Management System operates as a standalone VB.NET desktop application without integration of any third-party APIs or external services. All core functionalities—including user authentication, member management, payment tracking, and reporting—are handled internally within the application. While this approach simplifies deployment and minimizes external dependencies, it also limits scalability and access to advanced features such as online payment processing, automated notifications, or real-time analytics. Future enhancements may consider integrating reliable third-party services to improve functionality, efficiency, and user experience, provided they align with security, performance, and maintainability requirements.

**Description of Integration Points and Data Exchange Formats**

Potential future enhancements may include integration with external services such as various third-party APIs and platforms to expand system capabilities, improve user experience, streamline operational workflows, and enhance communication and payment processing functionalities:

* **SMS or Email Gateways:** For sending two-factor authentication codes, password reset tokens, or membership renewal notifications. Data exchanged could use JSON or XML payloads conforming to gateway provider standards.
* **Online Payment Gateways:** To accept electronic payments from members. Integration would involve secure RESTful API calls with encrypted payloads, supporting JSON-formatted requests and responses.

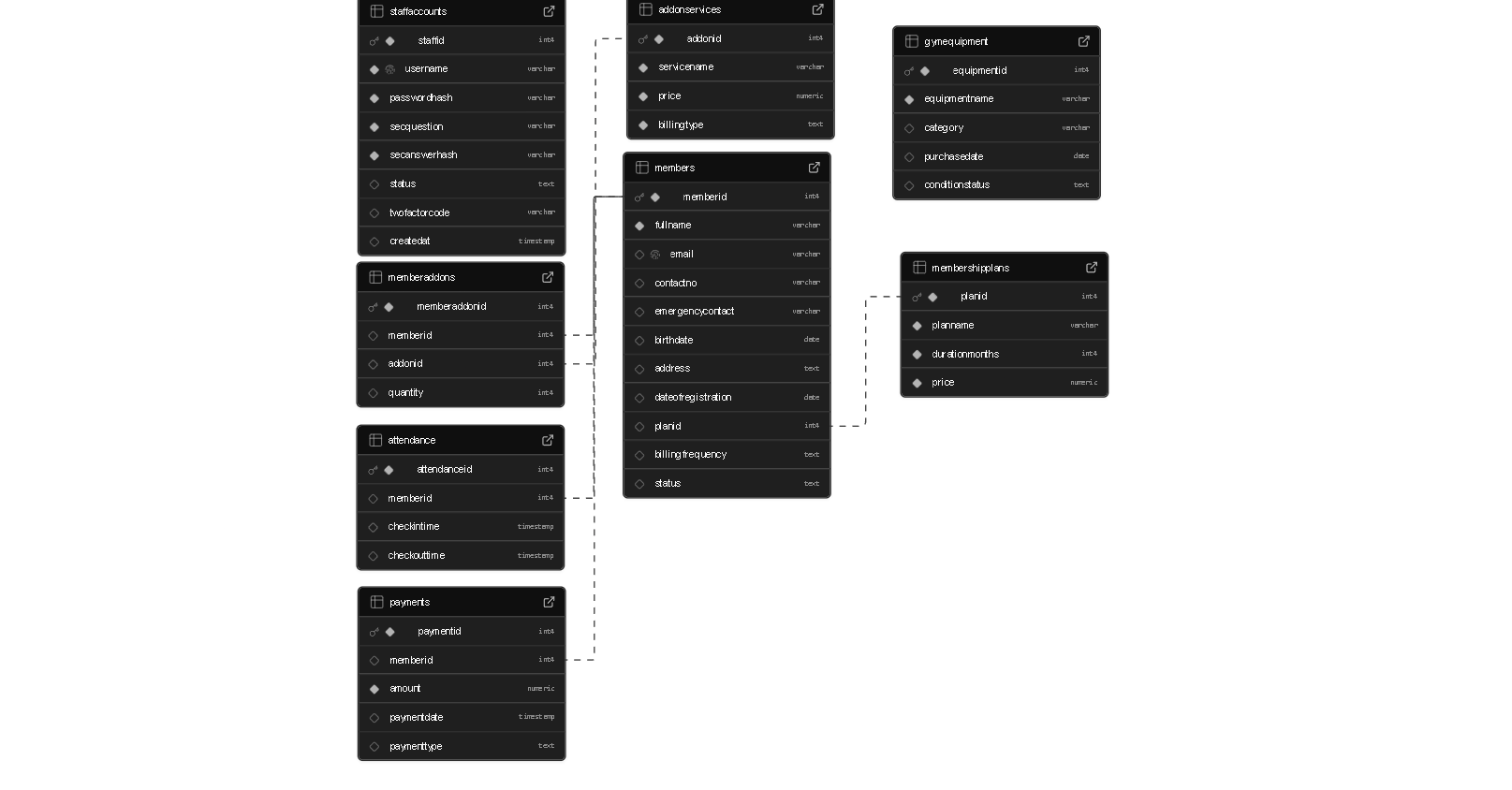
**DATABASE DOCUMENTATION**

The Gym Management System is supported by a relational database structured to store and manage critical data related to users, members, gym equipment, attendance records, and payment transactions. The ERD graphically depicts entities as tables, their primary keys, attributes, and relationships, ensuring a clear understanding of data flow and dependencies.

Core entities include:

* **Staff Accounts**: Stores staff and administrative account information including unique identifiers, hashed passwords, security questions and answers, two-factor codes, and account status to control access privileges.
* **Membership Plans:** Defines predefined subscription options such as monthly, quarterly, semi-annual, and annual packages with set durations and prices.
* Add-On Services: Contains optional services like personal training, nutrition coaching, and locker rental, with billing types based on per-session or monthly charges.
* **Members:** Consolidates personal information, chosen membership plan, billing frequency, status flags (active, expired), and registration details.
* **Member Add-Ons**: Manages the many-to-many relationship between members and optional services, including service quantities purchased.
* **Payments:** Records all financial transactions including downpayments, balances, and add-on purchases, ensuring proper tracking of member financial activity.
* **Gym Equipment:** Maintains detailed records of gym assets, including equipment names, categories, purchase dates, and condition statuses for maintenance management.
* **Attendance:** Tracks member check-ins and check-outs to monitor engagement and usage patterns.

The schema adheres to third normal form eliminating redundancy by ensuring each non-key attribute depends solely on the primary key in its table. Transaction management within the database supports atomicity, ensuring critical operations involving multiple updates (such as adding members and recording payments) succeed or fail entirely, maintaining consistent states.



**Figure 1.** Database Schema

**Description of Database Tables, Fields, and Relationships**

The database consists of several interrelated tables that organize information about staff accounts, membership plans, optional services, members, payments, equipment, and attendance. Each table has a primary key to uniquely identify records and uses foreign keys to establish relationships, ensuring data consistency and integrity across the system.

* + **StaffAccounts Table**  
     Fields include StaffID (PK), username, password hash, security question and answer hash, account status, two-factor code, and date created. This table manages staff and administrator accounts, handling authentication and access control within the system.
  + **MembershipPlans Table**  
     Fields include PlanID (PK), plan name, duration in months, and price. This table defines the available membership packages such as monthly, quarterly, semi-annual, and annual plans, which members can select upon registration.
  + **AddOnServices Table**  
     Fields include AddOnID (PK), service name, price, and billing type. This table specifies optional services like personal training, nutrition coaching, and locker rentals, including their corresponding pricing and billing structure.
  + **Members Table**  
     Fields include MemberID (PK), full name, email, contact number, emergency contact, birthdate, address, date of registration, plan ID (FK), billing frequency, and status. This table stores member details and links them to their chosen membership plan through the PlanID foreign key.
  + **MemberAddOns Table**  
     Fields include MemberAddOnID (PK), member ID (FK), add-on ID (FK), and quantity. This table manages the many-to-many relationship between members and add-on services, recording which services are availed and in what quantity.
  + **Payments Table**  
     Fields include PaymentID (PK), member ID (FK), amount, payment date, and payment type. This table tracks all financial transactions including downpayments, balances, and add-on purchases made by members.
  + **GymEquipment Table**  
     Fields include EquipmentID (PK), equipment name, category, purchase date, and condition status. This table maintains the inventory of gym equipment and supports monitoring of usage and maintenance needs.
  + **Attendance Table**  
     Fields include AttendanceID (PK), member ID (FK), check-in time, and check-out time. This table records member attendance by tracking entry and exit times linked to each member.

**Relationships Summary**  
 The Members table connects directly to the MembershipPlans table through PlanID, ensuring each member is linked to a subscription plan. Members are also associated with Payments and Attendance through their MemberID, allowing the system to track both financial transactions and gym usage. The MemberAddOns table establishes a many-to-many relationship between Members and AddOnServices, recording which services are subscribed by which member. Together, these relationships integrate member information with plans, services, financial records, and attendance to provide a complete management system.

**USER MANUAL**

This section is about guiding end-users on how to operate the Gym Management System. It includes step-by-step instructions, navigation guidelines, and explanations of common workflows. The user manual ensures that staff can effectively use the system for day-to-day operations without requiring technical knowledge of the backend.

**System Login**

* **Process**: Users access the system via the login screen. Credentials (username and password) are entered, and if enabled, a two-factor authentication code must be provided.
* **Access Control**: User roles determine which modules can be accessed (e.g., Admin, Manager, Staff).
* **Error Handling**: Incorrect credentials prompt an error message; locked accounts require administrator intervention.

**Dashboard Overview**

* **Purpose**: The dashboard provides a quick summary of key gym operations, such as number of active members, recent payments, and equipment status.
* **Features**: Charts and tables display real-time statistics, enabling managers to make quick decisions.
* **Navigation**: From the dashboard, users can access modules such as Members, Attendance, Payments, and Equipment.

**Member Management**

* **Registration**: Staff can add new members by entering personal details, selecting a membership plan, and applying add-on services if required.
* **Editing Profiles**: Member records can be updated (e.g., contact details, address changes).
* **Status Updates**: Membership status automatically changes based on expiration, but administrators can manually suspend or reactivate accounts.
* **Search & Filter**: Users can search for members by name, plan type, or status.

**Attendance Tracking**

* **Check-In**: Members are logged in by entering their Member ID or scanning a barcode (if available).
* **Check-Out**: Staff can record check-out times when members leave.
* **Reports**: Attendance records can be filtered by date, member, or frequency.

**Payment Processing**

* **Adding Payments**: Staff can record membership payments, add-on purchases, or balance settlements.
* **Payment Types**: Options include Downpayment, Balance, and Add-On.
* **Receipts**: Printable receipts are generated for each transaction.
* **Reports**: Payment history can be reviewed by member, date range, or payment type.

**Equipment Management**

* **Adding Equipment**: New items can be added with details such as name, category, purchase date, and vendor.
* **Updating Status**: Condition can be marked as Good, Needs Repair, or Out of Service.
* **Inventory Tracking**: Staff can generate equipment reports to monitor asset health and usage.

**Staff Account Management (Admin Only)**

* **Creating Accounts**: Administrators add staff accounts with username, password, role, and security question.
* **Modifying Accounts**: User details and roles can be updated as needed.
* **Suspending Accounts**: Inactive or compromised accounts can be disabled.
* **Resetting Passwords**: Administrators can reset staff passwords upon request.

**Common Workflows**

* **New Member Registration** → Assign plan → Record payment → Activate account.
* **Membership Renewal** → Select member → Record new payment → Update expiration date.
* **Attendance Recording** → Select member → Log check-in and check-out → Generate report.
* **Equipment Update** → Add new equipment → Assign category → Track status changes.

**TROUBLESHOOTING GUIDE**

This section is about identifying and resolving common issues that may occur while using the Gym Management System (GMS). It provides error explanations, troubleshooting steps, and recovery options. The goal is to help staff and administrators quickly restore normal operations without requiring deep technical expertise.

**Common Issues and Solutions**

This subsection provides a list of frequent problems that users may encounter while using the system. Each issue includes a brief explanation, possible causes, and suggested resolutions. By following these guidelines, users can often resolve problems without escalating to technical support.

**1. Login Problems**

* **Issue**: User cannot log in with correct credentials.
* **Possible Causes**:
  + Account is *Pending* or *Suspended*.
  + Incorrect password entered.
  + Two-factor authentication code expired.
* **Resolution**:
  + Verify account status with an administrator.
  + Reset the password via security question.
  + Request a new 2FA code.

**2. Database Connection Failure**

* **Issue**: Application cannot connect to MySQL database.
* **Possible Causes**:
  + MySQL service in XAMPP is not running.
  + Incorrect connection string in configuration file.
  + Firewall blocking port 3306.
* **Resolution**:
  + Start MySQL service via XAMPP Control Panel.
  + Check and correct database connection settings.
  + Allow port 3306 through firewall.

**3. Missing or Corrupted Data**

* **Issue**: Records are not displaying correctly or appear corrupted.
* **Possible Causes**:
  + Manual database edits caused inconsistencies.
  + Backup not restored properly.
* **Resolution**:
  + Restore the latest verified backup.
  + Avoid direct database modifications outside the system.

**4. Application Crash**

* **Issue**: GMS closes unexpectedly during use.
* **Possible Causes**:
  + Corrupted installation files.
  + Outdated .NET Framework version.
* **Resolution**:
  + Reinstall the GMS application.
  + Ensure .NET Framework 4.x is installed and updated.

**5. Report Generation Errors**

* **Issue**: Reports do not load or generate incomplete data.
* **Possible Causes**:
  + Filters incorrectly applied.
  + Database indexes missing or corrupted.
* **Resolution**:
  + Reset filters and regenerate the report.
  + Rebuild indexes in the database.

**Error Message Reference**

This subsection provides a reference table of the most common error messages encountered in the system. It explains the likely cause of each error and gives clear instructions on how to resolve it.

| **Error Message** | **Possible Cause** | **Solution** |
| --- | --- | --- |
| “Invalid Username or Password” | Incorrect credentials entered | Re-enter credentials or reset password. |
| “Account Suspended” | Account disabled by administrator | Contact system administrator for reactivation. |
| “Database Connection Failed” | MySQL not running / wrong config | Start MySQL service, check connection string. |
| “Record Not Found” | Search filters too narrow | Adjust filters or verify data exists. |
| “Access Denied” | Insufficient role permissions | Request proper access role from administrator. |

**Table 3.** Common Error Messages and Fixes

**Contact for Technical Support**

If issues persist after following the troubleshooting guide, users should escalate the problem to the designated developer or support contact. For more complex technical issues such as database corruption, version upgrade failures, or critical security breaches, external IT support may be required.

**Technical Support Contact Information**

| **Role** | **Name** | **Email Address** | **Contact Number** |
| --- | --- | --- | --- |
| Solo Developer | James Andrei N. Revilla | [revillajamesandrei4@gmail.com](mailto:revillajamesandrei4@gmail.com) | 0907-116-9962 |

**Table 4.** Contact Information

Users should reach out via email for non-urgent issues, such as feature requests, usability questions, or minor bugs. For urgent issues affecting system availability like database connection failures, application crashes), direct phone contact is recommended.

**CODE DOCUMENTATION**

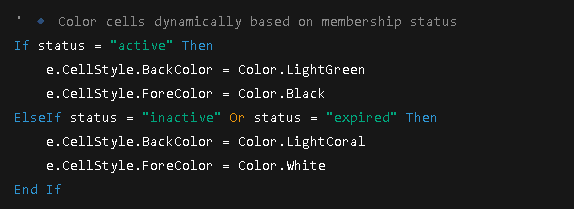
This section provides an overview of the **code structure, key modules, and coding standards** used in the Gym Management System (GMS). The purpose is to help future developers, maintainers, or collaborators understand the organization of the codebase and ensure consistency in development.

**Major Modules**

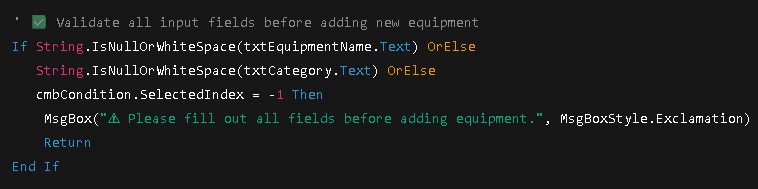
1. **Membership Management**
   * **Forms:** MembershipForm, downpaymentFrm, memberlist
   * **Responsibilities:**
     + Handle member registration and billing.
     + Calculate plan fees and add-ons.
     + Record down payments and track balances.
     + Display member lists with contact details, emergency contacts, and balances.
   * **Key Features:**
     + memberInfo dictionary transfers structured data between forms.
     + ClearForm() method resets all form controls after submission.
2. **Attendance Management**
   * **Forms:** CheckInOut, ViewAttendance
   * **Responsibilities:**
     + Track member check-ins and check-outs.
     + Prevent multiple check-ins without checkout.
     + Assign attendance labels (Day, Week, Month) based on historical data.
     + Display latest attendance records in a styled DataGridView.
     + Enable live search functionality for quick member lookups.
   * **Key Features:**
     + Parameterized MySQL queries prevent SQL injection.
     + Attendance data is updated dynamically and displayed with clear formatting.
3. **Equipment Management**
   * **Forms:** equipmentList, GymEquipmentAdd
   * **Responsibilities:**
     + Load and display gym equipment in a DataGridView.
     + Add new equipment to the database.
     + Apply visual styling, alignment, and column formatting.
   * **Key Features:**
     + Conditional formatting for the Condition column.
     + Input validation ensures completeness before database insertion.
4. **Account Management**
   * **Form:** AccountManage
   * **Responsibilities:**
     + Load system accounts for administrators and staff.
     + Approve or reject account registration requests.
     + Display account information including username, security questions, and status.
   * **Key Features:**
     + Status updates (Accepted/Rejected) executed via parameterized SQL queries.
     + DataGridView styling ensures consistency across modules.
5. **Member Status Monitoring**
   * **Form:** MemberStatus
   * **Responsibilities:**
     + Load and display member status (Active, Inactive, Expired) in real-time.
     + Dynamically color-code cells based on status.
     + Provide a centralized view of member health and subscription status.
   * **Key Features:**
     + Conditional formatting for clarity.
     + Hides sensitive columns like MemberID.
6. **Database Connectivity**
   * Centralized OpenConnection() and CloseConnection() functions manage MySQL connections efficiently.
   * Parameterized queries enhance security against SQL injection.
   * Transactions ensure data integrity during critical operations.
7. **User Interface**
   * Interactive controls: buttons, radio buttons, checkboxes, textboxes, DataGridViews, and date pickers.
   * Consistent visual design using custom fonts (e.g., "Poppins ExtraBold") and color schemes.
   * Dynamic DataGridView styling improves readability and accessibility.

**Inline Comments**

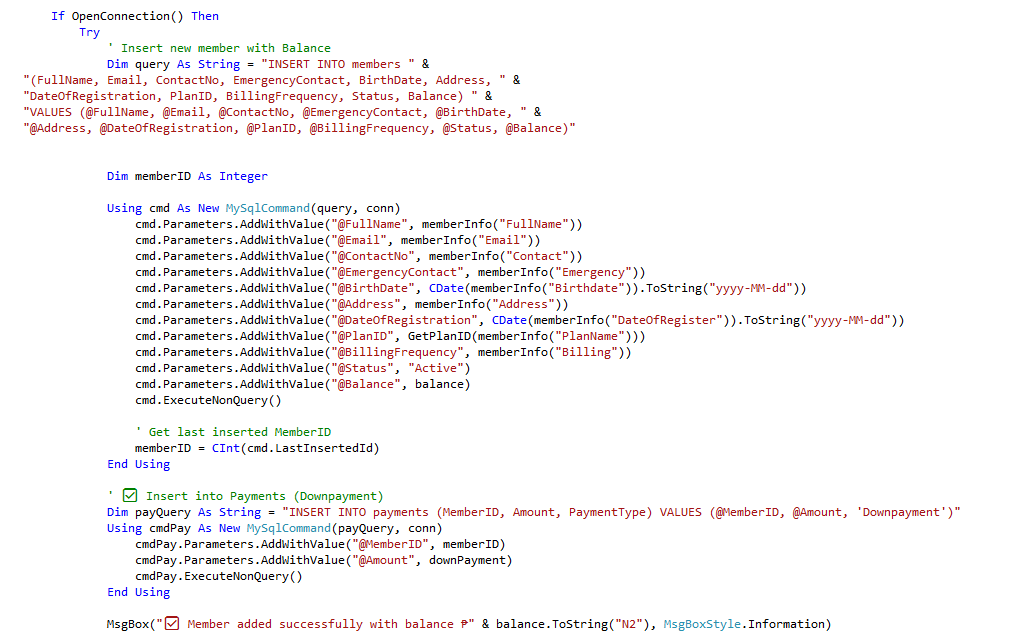
Inline comments are included to explain logic, algorithmic decisions, and workflows, making the code easier to maintain and extend.



**Figure 2. Status Code**

****

**Figure 3.** Add Equipment

****

**Figure 4.** Add Member & Make down payment

**Coding Standards & Conventions**

To ensure code readability, maintainability, and security, the following coding standards and conventions are strictly observed throughout the development of the Gym Management System (GMS).

**Naming Conventions:**

* Classes & Forms: PascalCase (MembershipForm, CheckInOut, MemberStatus)
* Methods: PascalCase (LoadAttendance, MemberCheckIn, LoadAccounts)
* Variables: camelCase (totalAmount, memberInfo)
* Constants: UPPERCASE (if applicable)

**Indentation and Formatting:**

* 4-space indentation.
* Logical separation of code blocks for readability.
* SQL queries formatted across multiple lines for clarity.

**Error Handling:**

* Try-Catch-Finally blocks for database operations and critical logic.
* User-friendly error messages display while maintaining security.
* Errors are logged or shown via MsgBox for debugging.

**Security Practices:**

* Parameterized queries prevent SQL injection.
* Input validation ensures required fields and correct data types.
* Sensitive data (passwords, security answers) is handled securely.

**Best Practices:**

* Modular design allows easy maintenance and future enhancements.
* Dictionaries like memberInfo enable structured data transfer between forms.
* DataGridViews are dynamically formatted for readability and accessibility.
* Live search functionality enhances usability.
* Consistent styling across all forms ensures a professional interface.

**TESTING DOCUMENTATION**

This section outlines the procedures used to test the Gym Management System (GMS). Testing ensures that all features function as intended, the system is secure, reliable, and user-friendly, and that it meets the defined requirements prior to deployment.

**Testing Environment**

This section describes the hardware, software, and test data used during the testing phase. It ensures that testing was performed under controlled and well-documented conditions, making the results reproducible and reliable.

**Hardware Specification**

| **Device** | **Processor** | **RAM** | **OS** | **Storage** |
| --- | --- | --- | --- | --- |
| Laptop 1 | AMD Ryzen 3 3500U | 8 GB | Windows 10 | 256 GB SSD |
| Laptop 2 | AMD Ryzen 7 5500U | 16GB | Windows 11 | 512 GB SSD |

**Table 5.** Hardware Specification

**Software Requirements**

* **Development Environment**: Visual Basic 2010 (VB.NET)
* **Database Server**: MySQL 8.0 via XAMPP 8.2
* **Testing Tools**: Manual test execution, MySQL Workbench for query validation
* **Frameworks**: .NET Framework 4.x

**Test Data**

* Sample staff accounts (Admin, Staff).
* Dummy member records with varying membership plans and add-ons.
* Sample equipment entries (Treadmill, Dumbbells, Stationary Bike).
* Test payments (downpayment, balance settlement, add-on purchase).

**Testing Methodology**

This section outlines the approach used to ensure the system’s quality and reliability. It describes the testing techniques applied, including functional testing to verify features, structural testing to validate code logic, and user acceptance testing to confirm that the system meets end-user requirements.

**Testing Process**

The testing process follows a **hybrid approach**, meaning it combines several testing techniques to thoroughly evaluate both the internal logic and the external behavior of the system. This ensures that the software is not only technically sound but also practical and useful for real-world users.

1. **Black-Box Testing**

Black-box testing is used to verify the system’s functionality without examining the internal code structure. The focus is purely on the inputs and outputs—what the user sees and experiences.

**Purpose:** To confirm that the system behaves as expected when users interact with it.

Examples of Application:

* Validating the login process: ensuring the system accepts correct credentials, rejects incorrect ones, and displays proper error messages.
* Testing payment entries: checking that payment amounts entered by users are stored correctly and reflected in the system reports
* Confirming that the system navigates smoothly from one feature to another without unexpected errors.

1. **White-Box Testing**

White-box testing, also known as structural testing, focuses on examining the internal logic, conditions, and flow of the code. This method requires knowledge of the actual implementation.

**Purpose:** To ensure the system’s logic and data handling are reliable, efficient, and error-free.

Examples of Application:

* Checking conditional statements that handle exceptions, such as applying discounts only if criteria are met.
* Verifying that data flows correctly through the system like the member information is correctly updated in the database after editing a profile.

1. **User Acceptance Testing (UAT)**

User Acceptance Testing involves testing the system in a real-world environment by actual end-users, in this case, gym staff.

**Purpose:** To validate that the system meets operational requirements and is user-friendly.

How It’s Done:

* Staff members perform their everyday tasks using the system, such as registering new gym members, processing payments, booking classes, and checking reports.
* Feedback is collected on whether the system is intuitive, easy to learn, and efficient for daily operations.

**Testing Tools and Frameworks**

The testing process was carried out **purely through manual execution**, without the use of any automated testing tools or frameworks. Testers interacted directly with the system, running it multiple times under different scenarios to validate its functionality and stability.

This involved repeatedly performing typical user actions such as logging in, creating records, updating information, and processing transactions. Each feature was tested by providing both valid and invalid inputs to ensure the system responded correctly in all cases. For example, valid credentials were entered to confirm successful logins, while invalid credentials were used to test error handling. Similarly, various payment transactions were recorded to verify that the system properly saved and displayed them in reports.

By running the system multiple times, testers were able to observe how it behaved under different conditions, identify any inconsistencies, and record issues when they occurred. Since no automation tools were used, the focus was on **realistic user interactions** and **hands-on validation** of the system’s performance, accuracy, and reliability.

**Testing Criteria**

To ensure consistency and clarity in evaluating results, the following criteria were applied:

1. **Pass/Fail Status**

* Each test case had a clear expected result.
* If the actual result matched the expected result, the test was marked Pass.
* If the result did not match, it was marked Fail.

1. **Bug Logging and Severity Levels**
   * **Critical:** Prevents core functions of the system from working (e.g., inability to log in).
   * **High:** Affects important operations but does not completely stop system usage (e.g., payments recorded but reports not updating).
   * **Medium:** Functional errors with a moderate impact, usually involving secondary features.
   * **Low:** Minor issues such as typos, misaligned buttons, or small cosmetic problems.
2. **Readiness for Deployment**

* The system was declared ready for deployment only when all **Critical** and **High severity bugs** were resolved.
* Medium and Low severity issues could remain temporarily but were logged for future fixes and updates.

**Test Cases**

This section provides detailed test cases designed to validate core functionalities of the system. Each test case includes step-by-step actions, expected outcomes, and actual results, making it easier to track system performance and identify issues.

| **Test Case ID** | **Test Description** | **Test Steps** | **Expected Output** | **Actual Output** | **Status** | **Remarks** |
| --- | --- | --- | --- | --- | --- | --- |
| TC001 | Login with valid credentials | 1. Enter username  2. Enter password  3. Click login | User is redirected to dashboard | User is redirected to dashboard | Pass | N/A |
| TC002 | Login with invalid password | 1. Enter username  2. Enter incorrect password  3. Click login | Error message appears | No error message appeared | Fail | Bug identified |
| TC003 | Add new member | 1. Open Members form 2. Enter details 3. Save | Member record is added | Member record is added | Pass | N/A |
| TC004 | Record attendance | 1. Enter MemberID  2. Click Check-In  3. Verify Check-Out | Attendance logged with timestamps | Attendance logged successfully | Pass | N/A |
| TC005 | Payment transaction | 1. Open Payments form 2. Enter amount  3. Save | Payment recorded in database | Payment recorded correctly | Pass | N/A |

**Table 6.** Test Case Execution Log

**Bug Tracking & Issue Log**

This section documents all bugs or issues encountered during the testing phase. Each bug is carefully recorded with a unique identifier, description, severity, reporter, current status, and resolution. This structured approach ensures efficient tracking and prioritization of issues for timely resolution.

**Bug Severity Categories**

* **Critical:** Issues that prevent core system functionalities from working and must be fixed before deployment.
* **High:** Major issues affecting performance, stability, or usability.
* **Medium:** Moderate issues that do not halt operations but should be addressed.
* **Low:** Minor issues or cosmetic problems that do not impact core functionality.

**Status Tracking**

* **Open:** Issue reported but not yet addressed.
* **In Progress:** Developers are actively resolving the issue.
* **Resolved:** Bug has been fixed and verified by testers.

| **Bug ID** | **Description** | **Severity** | **Reported By** | **Status** | **Resolution** |
| --- | --- | --- | --- | --- | --- |
| B001 | Login page crashes on incorrect password | High | Tester 1 | Open | Pending fix |
| B002 | Payment receipt not printing | Medium | Tester 2 | Resolved | Updated print function |
| B003 | Attendance allows duplicate check-ins | Critical | Tester 1 | In Progress | Validation under review |

**Bug Tracking Log**

**Table 7.** Bug tracking Log

**User Acceptance Testing**

User Acceptance Testing (UAT) was performed to validate that the Gym Management System functions correctly under practical usage scenarios and meets user requirements. The testing focused on simulating typical gym operations to evaluate system usability, functionality, and performance.

**Test Scenarios for End-Users**

1. Registering new members and assigning appropriate membership plans.
2. Recording attendance for multiple members simultaneously.
3. Processing payments and generating receipts accurately.
4. Updating and tracking gym equipment records.

**Feedback from Actual Users**

* Staff found the interface intuitive and easy to navigate.
* Reports were accurate but exporting to Excel or PDF was slower than expected.
* Users requested the addition of a search filter by contact number for faster member lookup.

**Necessary Improvements or Fixes**

* Implement a member search feature using contact numbers.
* Optimize report export functions for faster processing.
* Add tooltips or guidance for dashboard icons to enhance usability.

**MAINTENANCE GUIDE**

The maintenance plan for the Gym Management System (GMS) provides a structured and systematic approach to ensure the system remains reliable, secure, and efficient throughout its operational lifecycle. It defines the strategies, procedures, and responsibilities for maintaining the system, addressing both expected and unforeseen issues. A well-defined maintenance plan guarantees that the GMS continues to support gym operations, reduces downtime, and maintains user confidence in system performance.

**Overall Maintenance Strategy**

The overall strategy for maintaining the GMS involves a combination of proactive, corrective, and adaptive measures. Maintenance activities are scheduled regularly and performed by designated personnel, ensuring the system is always operating at its optimal capacity. Key elements of the strategy include

* **Proactive Monitoring:** Continuously observing system health, performance metrics, and potential error logs to anticipate and prevent issues before they impact operations.
* **Timely Issue Resolution:** Rapidly addressing bugs, system errors, and performance bottlenecks to minimize downtime and operational disruptions.
* **Adaptation to Technological Changes:** Ensuring compatibility with updated software environments, operating systems, .NET Framework versions, and database updates.
* **User-Centric Improvements:** Incorporating user feedback and usage patterns into system enhancements to improve usability and operational efficiency.
* **Documentation and Training Updates:** Keeping all system manuals, user guides, and training materials current with the latest changes to prevent user errors and maintain workflow efficiency.

**Types of Maintenance**

The GMS maintenance plan includes multiple types of maintenance to address different aspects of system performance:

**Corrective Maintenance**

* **Purpose:** Detects and corrects faults, errors, or defects within the system that affect functionality or data integrity.
* **Examples**: Fixing login authentication errors, correcting inaccurate payment calculations, resolving attendance tracking discrepancies.
* **Outcome**: Ensures that the system operates correctly and reliably without recurring operational issues.

**Adaptive Maintenance**

* **Purpose:** Modifies and updates the system to maintain compatibility with evolving technical environments, software platforms, or business processes.
* **Examples:** Adjusting database connections after an OS update, upgrading the VB.NET runtime, adapting to new regulatory or compliance requirements.
* **Outcome:** Guarantees that the GMS remains functional in changing operational or technological contexts.

**Perfective Maintenance**

* **Purpose:** Improves or enhances the system to optimize performance, usability, or functionality without correcting defects.
* **Examples:** Enhancing dashboard reporting speed, improving search functionality in member lists, optimizing database queries for faster performance.
* **Outcome:** Increases system efficiency, reduces operational delays, and enhances user satisfaction.

**Preventive Maintenance**

* **Purpose:** Conducts scheduled activities to reduce the risk of future system failures and maintain long-term stability.
* **Examples:** Regular database optimization, cleaning obsolete logs, monitoring server performance, applying security patches.
* **Outcome:** Reduces the likelihood of unplanned downtime and ensures continuous availability of the system for gym operations.

**Maintenance Schedule**

A structured maintenance schedule ensures that all system components are regularly monitored, updated, and optimized. It provides clarity on tasks, responsibilities, and timing, helping prevent system failures and maximize uptime.

| **Task** | **Description** | **Frequency** | **Responsible Person** | **Status** |
| --- | --- | --- | --- | --- |
| Database Backup | Create full backups of the MySQL database, including member records, payment history, attendance logs, and equipment inventory. Verify backup integrity. | Weekly | Developer | Ongoing |
| Security Updates | Apply patches and updates to the system, including OS updates, XAMPP/MySQL security patches, and VB.NET runtime updates. | Monthly | Developer | Scheduled |
| Bug Fixes | Identify, analyze, and fix reported errors or software issues affecting system functionality. | As needed | Developer | Pending |
| System Performance Check | Monitor server and client performance, optimize database queries, clear obsolete logs, and ensure the application runs smoothly. | Quarterly | Developer | Not Started |
| User Training Refresh | Conduct refresher sessions for staff on system updates, new features, or operational changes. | Semi-Annually | Developer | Scheduled |

**Table 9.** Maintenance Schedule

**Issue Tracking & Bug Reports**

Effective issue tracking is critical to maintaining the reliability, security, and performance of the Gym Management System (GMS). All bugs, software errors, and user-reported issues are systematically recorded in a centralized tracking system. This approach ensures accountability, timely resolution, and clear communication between the support and development teams. A structured issue management workflow minimizes downtime, prevents recurrence of errors, and provides transparency for stakeholders.

**Issue Tracking Log**

All reported issues are logged with unique identifiers, descriptions, severity levels, reporter details, and current status. The log serves as a historical record to monitor system stability, identify recurring problems, and track the effectiveness of fixes.

| **Issue ID** | **Description** | **Severity** | **Reported By** | **Date Reported** | **Status** |
| --- | --- | --- | --- | --- | --- |
| BUG001 | Login page does not load | High | User A | MM/DD/YYYY | Fixed |
| BUG003 | Attendance count mismatch | Medium | User C | MM/DD/YYYY | Pending |
| BUG004 | Member ID generation error | High | User D | MM/DD/YYYY | Pending |
| BUG005 | Equipment list not updating after addition | Medium | User E | MM/DD/YYYY | Pending |
| BUG006 | Dashboard statistics not refreshing | High | Admin F | MM/DD/YYYY | In Progress |

**Table 10.** Issue Tracking & Bug Reports

**Backup & Recovery Plan**

Ensuring data integrity and availability is critical for the Gym Management System (GMS). A structured backup and recovery plan minimizes data loss, reduces downtime, and ensures continuity of operations in the event of system failure, hardware malfunction, or accidental deletion. This plan outlines the procedures for creating, storing, verifying, and restoring backups to maintain the security and reliability of critical gym data, including member information, attendance logs, payment records, and system configurations.

**Backup Procedures**

Effective backup procedures are essential to protect the Gym Management System (GMS) data from loss or corruption. These procedures ensure that all critical information, including member records, financial transactions, and system configurations, can be securely restored in the event of hardware failure, software errors, or other unforeseen issues.

**Frequency of Backups**

* **Full Backups:** A complete snapshot of the MySQL database is performed **weekly**, capturing all data including members, payments, attendance, equipment inventory, and system settings.
* **Incremental Backups:** Daily incremental backups are performed to record only changes since the last full backup. This approach reduces storage requirements while ensuring that recent data modifications are preserved.
* **Configuration Files:** Backup copies of application configuration files, connection strings, and security settings are maintained weekly alongside database backups to facilitate full system restoration if needed.

**Storage Locations and Redundancy**

* **Primary Storage:** Full and incremental backups are saved on the local server, ensuring immediate accessibility for restoration.
* **Secondary Storage:** Copies of backups are transferred to external hard drives, which are stored offsite to protect against local hardware failures or disasters.
* **Optional Network Storage:** For additional redundancy, backups may also be stored on network-attached storage (NAS) devices within the gym’s secure network.

**Verification of Backups**

* Periodic test restores are conducted monthly to confirm the integrity of backup files and ensure they can be successfully used in a recovery scenario.
* Backup logs are reviewed and documented, including timestamps, file sizes, and successful verification checks, providing a clear audit trail.

**Security Measures for Backups**

* All backup files are encrypted and stored in restricted-access folders to prevent unauthorized access.
* Backup retention policies are enforced to maintain sufficient historical copies while managing storage capacity efficiently.

**Recovery Steps**

In the event of system failure, accidental data loss, or data corruption, a structured recovery procedure is essential to restore the Gym Management System (GMS) to normal operation with minimal disruption. The following steps ensure that data integrity is maintained and system functionality is quickly re-established.

1. **Identify Recovery Point:** Determine the most recent valid backup based on the defined Recovery Point Objective (RPO) to minimize data loss.
2. **Application Shutdown:** Stop the GMS application and ensure that no client or administrative sessions are actively connected to the database.
3. **Database Restoration:** Restore the database using **MySQL tools**, such as the XAMPP MySQL restoration utility. For incremental backups, apply sequential changes after restoring the last full backup.
4. **Data Validation:** Verify that critical data has been restored correctly, including member profiles, payment records, attendance logs, and equipment inventory. Any discrepancies should be reported and corrected immediately.
5. **System Restart:** Restart the GMS application and perform basic functional checks, including login, member management, attendance check-in/out, payment processing, and report generation, to confirm normal operation.
6. **Documentation and Reporting:** Record the restoration process in a recovery log, noting the date, backup version used, steps executed, and any issues encountered. Report the successful recovery to the system administrator and key stakeholders.

| **Role** | **Name** | **Email** | **Contact Number** |
| --- | --- | --- | --- |
| Developer | James Andrei N. Revilla | [revillajamesandrei4@gmail.com](mailto:revillajamesandrei4@gmail.com) | 0907-116-9962 |

**Table 11.** Technical Support Contact

**Performance Monitoring**

Continuous performance monitoring is critical to ensure that the Gym Management System (GMS) operates efficiently, reliably, and without unexpected downtime. Monitoring key performance indicators (KPIs) allows the IT team to detect early signs of system degradation, proactively address potential issues, and maintain a high-quality user experience for gym staff and management.

| **Metric** | **Description** | **Threshold** | **Monitoring Tool** |
| --- | --- | --- | --- |
| **Server Uptime** | Percentage of time the GMS server is fully operational and accessible by client machines | 99.9% | XAMPP logs, custom monitoring scripts, or Windows Event Viewer |
| **Response Time** | Time taken to load member records, dashboards, equipment lists, and reports | < 5 seconds | VB.NET profiler, performance logs, custom timing scripts |
| **Error Rate** | Percentage of failed database queries, unhandled exceptions, or application errors | < 1% | Centralized error log, log analyzer tools, real-time alerts |
| **Backup Integrity** | Validation that backup files are complete, consistent, and restorable | 100% | Periodic test restores, checksum verification, automated backup verification scripts |
| **Database Health** | Checks database size, index fragmentation, and query performance | Optimal operation | MySQL Workbench, XAMPP tools, custom SQL scripts |

**Table 12.** Key Performance Indicators