Gym Membership and Attendance Tracker

Higher National Diploma In Information Technology

SRI LANKA INSTITUTE OF ADVANCED TECHNOLOGICAL EDUCATION

GAMPAHA

2nd Year-1st Semester

group-2

Date of Submission-31/12/2025

Group members

GAM/IT/P/2324/P/0002-H.M.C.N.Herath
GAM/IT/P/2324/P/0014-R.P.S.L.Ranasingha
GAM/IT/P/2324/P/0042-R.P.A.V.D.Rajapaksha
GAM/IT/P/2324/P/0039-J.G.S.B.Dilshani
GAM/IT/P/2324/P/0025-K.P.R.Sathsarani
GAM/IT/P/2324/P/0035-A.E.P.B.Athukorala

1. Abstract

1.1.Overview

The Gym Membership & Attendance Tracker is a dedicated desktop application designed to modernize the management of fitness centers. By integrating member registration with real-time attendance tracking, the system ensures that gym operations are organized, secure, and data-driven.

1.2.Purpose of the System

The primary purpose is to provide gym administrators and instructors with a centralized tool to manage their member database and monitor daily check-ins. It aims to automate the calculation of membership status and provide analytical insights into gym usage through the AnalyticsService.

1.3.Problem it Solves

- ➢ Manual Inefficiency: Replaces traditional paper-based logs which are difficult to search and prone to physical damage.
- ➢ Data Security Risks: Addresses the danger of storing sensitive user information and passwords in plain text by implementing jBCrypt hashing.
- ➢ Unauthorized Access: Prevents non-members or expired members from utilizing facilities through an automated verification process.

1.4.Target Users

- ➢ Administrators**:** Users who require full access to reports, member registration, and system settings.
- ➢ Instructors**:** Staff members responsible for day-to-day attendance marking and managing specific member interactions via the InstructorDashboard.

1.5.Technologies Used

- ➢ Language**:** Java (JDK 17).
- ➢ Framework**:** JavaFX with FXML for the Graphical User Interface (GUI).
- ➢ Data Persistence: File-based storage using both text (members.txt) and binary serialization (attendance.bin).
- ➢ Security: jBCrypt library for secure password encryption.
- ➢ Build Tool: Maven for dependency management.

2. Requirements Specification

2.1.Functional Requirements

These requirements define the specific behaviors and services the system provides to the users.

- User Authentication & Authorization: The system must provide a secure login interface for Administrators and Instructors to access their respective dashboards.
- Member Profile Management: Authorized users must be able to add new members, update existing profiles, and remove member records from the system.
- Attendance Tracking: The system must allow staff to record daily check-ins and check-outs for gym members.
- Report & Analytics Generation: The system must process stored data to generate attendance summaries and analytical trends via the AdminReportsController and AnalyticsService.
- Email Notifications: The system should have the capability to send automated emails to members using the EmailService module.

2.2.Non-Functional Requirements

These requirements specify the quality attributes and constraints of the system.

- Security (Data Protection): All user passwords must be hashed using the jBCrypt algorithm before storage to prevent unauthorized access to credentials.
- Usability: The interface must be intuitive and responsive, utilizing JavaFX and CSS to provide a consistent user experience.
- Reliability: The system must handle data persistence errors gracefully, such as missing files in the data folder, without crashing the application.
- Performance: The system should quickly load member lists and process report summaries, even as the members.txt file grows in size.

3. System Architecture

3.1 Architecture Overview

The Gym Membership & Attendance Tracker is designed using a Multi-Layered Architecture. This approach separates the application into distinct layers, ensuring that the User Interface (UI) is independent of the business logic and data storage. By using this structure, the system becomes easier to maintain, test, and scale, as changes in one layer (such as switching from File storage to MySQL) do not require a complete rewrite of the UI.

3.2 System Layers

Based on the project structure, the system is divided into the following four layers:

- User Interface (UI) Layer: This is the topmost layer where users interact with the system. It consists of JavaFX FXML files (e.g., LoginView.fxml, AdminDashboard.fxml) and their corresponding Controller classes (e.g., LoginController.java) which handle user inputs and button clicks.
- Service / Business Logic Layer: This layer contains the core rules and logic of the gym system. Classes such as AuthService.java, AttendanceService.java, and AnalyticsService.java process data, validate user credentials, and calculate gym usage trends.
- Data Access Layer (DAO): This layer acts as a bridge between the services and the stored data. The FileHandler.java class is responsible for reading from and writing to the physical files, ensuring the rest of the system does not need to handle low-level file I/O operations.
- Storage Layer: This is the physical location where data is kept. The system utilizes the data folder, containing users.txt for credentials, members.txt for profile data, and attendance.bin for serialized attendance records.

4. Object-Oriented Design

4.1.OOP Concepts Used

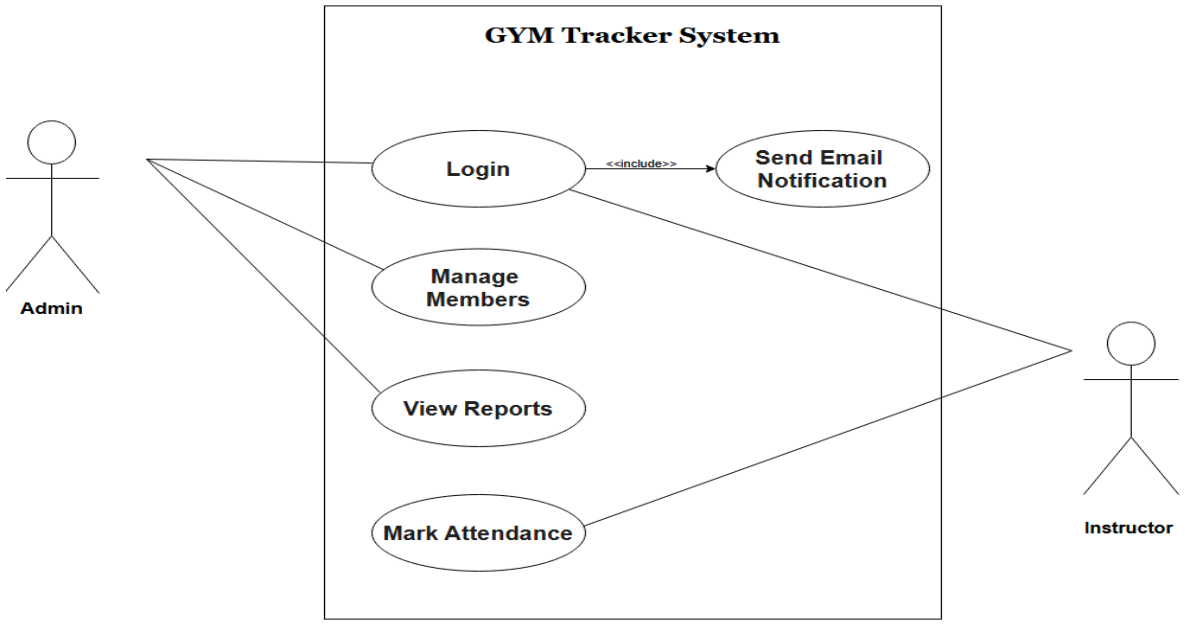
The system is built on the core pillars of Object-Oriented Programming to ensure code reusability and maintainability.

- Encapsulation: In classes like Member.java and Person.java, all data fields (e.g., name, email, memberID) are declared as private. Access to these fields is strictly controlled through public getters and setters, ensuring data integrity.
- Inheritance: The project utilizes a class hierarchy where Member.java extends the Person.java base class. This allows the Member class to inherit common attributes like name and contact info while adding gym-specific details like membership type.
- Abstraction: An interface named Billable.java is implemented to define essential methods for financial transactions. This hides the complex implementation details of fee calculations from the rest of the system.
- Polymorphism: Method overriding is used to provide specific implementations for inherited methods, such as custom toString() methods or specialized processing logic in the AttendanceRecord class.
- Collections: The ArrayList and HashMap structures are used within the service package to manage groups of members and attendance logs in memory before saving them to files.

4.2 UML Diagrams

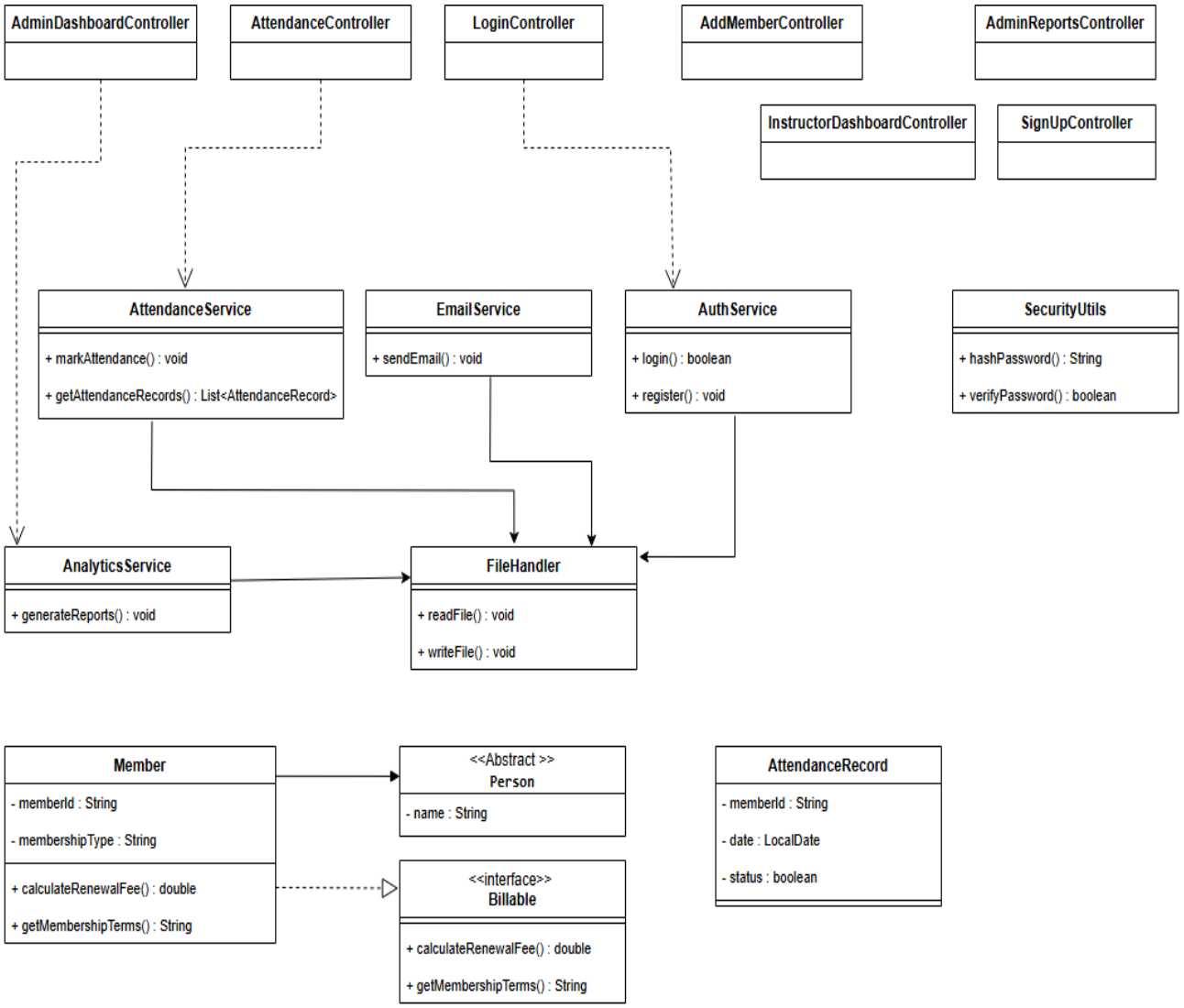
4.2.1.Use Case Diagram

This diagram shows the interactions between the Admin/Instructor and the system features like "Register Member," "Log Attendance," and "Generate Reports".



4.2.2.Class Diagram

This diagram illustrates the relationship between your classes (e.g., how Member inherits from Person and how FileHandler interacts with the Model classes).



5. Login & Security Implementation

5.1 Description of Login Process

The system features a mandatory secured login system that serves as the entry point for all administrative and instructional tasks. Upon launching the application via the Launcher.java class, the user is presented with the LoginView.fxml interface.

The login process is managed by the LoginController.java which captures the user's credentials and passes them to the AuthService.java for verification. The system validates the username and password against the records stored in users.txt. Access is only granted if the credentials match, ensuring that sensitive member data and attendance records remain protected from unauthorized access.

5.2 Password Storage Method

To comply with the requirement that passwords must not be stored in plain text, this project implements robust encryption:

- jBCrypt Hashing: The system utilizes the jBCrypt library (version 0.4) to hash passwords before they are written to the users.txt file.
- One-Way Hashing: jBCrypt provides a strong, one-way hashing algorithm that incorporates a "salt" automatically, protecting the system against rainbow table attacks.
- Verification: During the login attempt, the AuthService uses the BCrypt.checkpw() method to compare the plain-text input from the user with the hashed value stored in the database.

5.3 Security Measures

The following security best practices have been implemented to ensure system integrity:

- No Hard-coded Credentials: All user accounts are managed through the storage layer; no administrative usernames or passwords are hard-coded within the Java source files.
- Prevention of Decrypted Display: The system is designed to never display or log decrypted passwords in the UI or console.
- Role-Based Access: Based on the user's role defined in the AuthService, the system directs the user to either the AdminDashboard or the InstructorDashboard, limiting access to specific administrative functions.
- Secure Utilities: A dedicated SecurityUtils.java class in the util package centralizes security-related functions, ensuring consistent implementation across the application.

6. Data Persistence

6.1 File-Based Projects (Implementation)

The Gym Membership & Attendance Tracker utilizes a file-based storage system located within the data/ directory of the project root. This ensures that all member records, user credentials, and attendance logs are preserved even after the application is closed.

- File Types Used:
    - Plain Text (.txt): Used for members.txt and users.txt to store structured string data, such as member profiles and hashed user credentials.
    - Binary (.bin): Used for attendance.bin to store serialized Java objects, which allows the system to save and reload complex AttendanceRecord objects directly.
- File Read/Write Mechanism:
    - The system centralizes all storage logic within the FileHandler.java class in the service package.
    - It uses java.io classes such as BufferedReader/BufferedWriter for text files and ObjectInputStream/ObjectOutputStream for binary serialization.
- Handling Missing or Corrupt Files:
    - The FileHandler class implements checks to see if the data/ directory and required files exist upon startup.
    - If a file is missing, the system is designed to create a new, empty file gracefully or alert the user through the UI, preventing application crashes.

6.2 Data Organization

To maintain a clean system architecture, the data is separated by category:

- User Data: Stored in users.txt, containing unique usernames and BCrypt-hashed passwords for secure authentication.
- Member Data: Stored in members.txt, containing encapsulated information such as Member IDs, names, and membership status derived from the Member model.
- Attendance Logs: Stored in attendance.bin, capturing the timestamped check-ins for every member processed through the AttendanceService.

7. Processing & Business Logic

7.1 Logic & Rules Implementation

The system processes data through dedicated service classes to ensure that gym rules are applied consistently and calculations are accurate.

- Calculations & Decision Making: The AnalyticsService performs mathematical operations to determine member growth rates and average attendance figures. Decision-making logic is used during login to verify user roles (Admin vs. Instructor) and redirect them to the correct dashboard.
- Data Validations: The AddMemberController and SignUpController include validation logic to ensure that Member IDs are unique, email addresses are correctly formatted, and no mandatory fields are left empty before saving data to members.txt.
- Sorting & Searching: Using Java Collections, the system enables administrators to search for specific members by ID or name and sort attendance records by date for easier review.

7.2 Reports & Summaries

To meet the "Reports or Summaries" requirement, the system includes a dedicated reporting module:

- Admin Reports: The AdminReportsController generates summaries that allow administrators to view daily or monthly attendance trends.
- Attendance Tracking: The AttendanceService processes raw logs from attendance.bin to create readable summaries of who visited the gym during specific timeframes.

7.3 Exception Handling

The project implements robust error handling to manage unexpected scenarios without system failure:

- Custom Exceptions: The system is designed to throw specific exceptions, such as when a member is not found or when a database/file connection fails.
- Graceful Recovery: Use of try-catch blocks in the FileHandler ensures that if a data file is corrupted or missing, the system alerts the user via a JavaFX Alert dialog instead of crashing.

8.Exception Handling

The Gym Membership & Attendance Tracker implements a robust error-handling strategy to prevent system crashes and provide meaningful feedback to the user when unexpected events occur.

8.1 Custom and Built-in Exceptions

The system uses a combination of standard Java exceptions and custom logic within the service and controller layers to handle specific gym-related errors:

- MemberNotFoundException: (If implemented in your AttendanceService) Used when a member ID entered for attendance does not exist in members.txt.
- InvalidCredentialsException: Handled within the AuthService to manage failed login attempts.
- IOException: Extensively used in the FileHandler class to catch issues related to missing or inaccessible files like attendance.bin or users.txt.

8.2 Implementation Strategy

- Try-Catch Blocks: Critical operations, such as reading from the database files or parsing dates, are wrapped in try-catch blocks to ensure the application remains functional even if an error occurs.
- Graceful Recovery: Instead of terminating, the system is designed to recover gracefully. For example, if a data file is missing, the FileHandler is programmed to create a new default file rather than throwing a fatal error.
- User Alerts: When an exception is caught in the controller layer (e.g., AddMemberController), the system uses JavaFX Alert dialogs to inform the user of the specific problem, such as "Invalid Input" or "File Not Found," allowing them to correct the action.

8.3 Data Integrity

By handling exceptions during file writing, the system ensures that partial or corrupt data is not saved to the data/ folder. This is especially important for the attendance.bin file, where binary serialization requires strict error management to prevent the loss of historical logs.

9. Conclusion

9.1 Achievements of the Project

The development of the Gym Membership & Attendance Tracker has resulted in a fully functional, secure, and user-friendly desktop application. Key technical achievements include:

- Successful Data Persistence: The system reliably manages data across different formats, using plain text for member profiles and binary serialization for complex attendance logs.
- Robust Security: By integrating the jBCrypt library, the project ensures that administrative credentials are never stored in plain text, meeting high-standard security requirements.
- Modular Design: The use of a multi-layered architecture (UI, Service, and Data layers) allows for independent testing and maintenance of the system components.

9.2 OOP Concepts Successfully Implemented

The project serves as a practical application of core Object-Oriented Programming principles:

- Encapsulation: All model classes, such as Member and Person, use private fields with controlled access via getters and setters.
- Inheritance: We successfully implemented a hierarchical structure where Member inherits foundational attributes from the Person class, reducing code redundancy.
- Abstraction: The Billable interface was utilized to define a contract for payment processing, separating the "what" from the "how".
- Polymorphism: Method overriding was used in various controllers and models to provide specific behaviors for different user roles and data types.

9.3 Possible Future Improvements

While the current system is robust, the following enhancements are proposed for future iterations:

- Database Migration: Moving from file-based storage to a MySQL relational database to support larger datasets and complex queries.
- Expanded Communication: Fully activating the EmailService to send automated renewal alerts and registration confirmations to members.
- Graphical Analytics: Integrating JavaFX charts to provide a visual representation of attendance trends and membership growth in the AdminReportsView.
- Hardware Integration: Adding support for barcode or QR code scanners to automate the check-in process further.