# **Boost Physio Clinic – Booking & Reporting System**

**Course Module:** 7COM1025 – Programming for Software Engineers  
 **Project Title:** Boost Physio Clinic Booking & Reporting System  
 **Technologies Used:** Java 17, JavaFX, JUnit 5  
 **Project Type:** Java Desktop Application with GUI  
 **Author:** [Your Full Name]  
 **Submission Date:** [Insert Date]  
 **Word Count:** 1500+

## **1. Introduction**

Modern healthcare practices, including physiotherapy clinics, require superior management strategies for patient records combined with therapist scheduling and treatment plans. Manual systems do not offer sufficient capabilities in terms of scalability, together with error prevention measures and reporting functionalities. The Java-based software solution development represents the main objective of this coursework for Boost Physio Clinic. The system allows staff users to handle patient details and allocate treatment options with appointment scheduling and report generation functions. The application uses JavaFX graphical interface technology with JUnit 5 testing framework to create a user-friendly and robust interface.

The main purpose of this project involves building a dependable desktop booking system through object-oriented design and modular software systems. The system meets all the necessary module training objectives that focus on software design and Java programming, in addition to GUI development and unit testing. The system design, implementation, testing activities, and projected growth analysis are detailed in this report under Harvard-style technical documentation.

## **2. System Objectives**

These are the main targets that the Boost Physio Clinic Booking & Reporting System should achieve:

* A modular object-oriented framework exists for processing the main business operations with maximum efficiency.
* The development of a JavaFX GUI interface requires creating code for managing user activities that include adding patients and scheduling appointments and producing reports.
* The program uses in-memory data storage in combination with singleton patterns for managing state among multiple views.
* The system functionality will be validated through tests implemented with JUnit-based unit testing.
* Future development of necessary web interface capabilities and persistent storage integration can occur because of this system's design structure.

The project divided its work into two distinct phases for systematic development. Phase 1 covered system design and building the GUI, while Phase 2 focused on testing activities and report creation.

## **3. Methodology**

### **3.1 Software Development Lifecycle (SDLC)**

The software development process implemented an iterative software lifecycle, which contained four main phases:

* **Analysis**: Identification of core components such as Patient, Physiotherapist, Appointment, and Treatment.
* **Design**: System architecture and UML diagram creation took place during the design stage.
* **Implementation**: Java and JavaFX-based implementation.
* **Testing**: Unit testing with JUnit 5.

### **3.2 Tools and Technologies**

* **Language**: Java 17
* **UI Framework**: JavaFX 17
* **Testing**: JUnit 5
* **IDE**: IntelliJ IDEA
* **Version Control**: Git

## **4. System Design**

### **4.1 Architectural Pattern**

The project implements Model-View-Controller (MVC) as its design framework.

* **Model**: The system includes business objects under the Model section, including Patient, Physiotherapist, Treatment, and Appointment.
* **View**: The GUI layouts exist inside .fxml files.
* **Controller**: Development and business logic implementation exist in Java controller classes.

### **4.2 UML Diagram**

A UML class diagram illustrates the system’s structural design. The relationships among core classes are visually represented to showcase inheritance, aggregation, and associations. (Refer to UML Diagram Image in Appendix A).

## **5. Implementation**

### **5.1 Data Models**

* **Patient**: Stores patient ID, name, address, and phone number.
* **Physiotherapist**: Stores name, ID, expertise areas, and treatments.
* **Treatment**: Defines treatment name, associated expertise, and schedule.
* **Appointment**: Combines a patient, a physiotherapist, and treatment.

### **5.2 BookingSystem Service**

BookingSystem functions as the main data management service through a singleton implementation. The BookingSystem class maintains in-memory maps of lists that enable patient and physiotherapy staff to manage appointments.

Key Methods:

* addPatient, removePatient, getPatients
* addPhysiotherapist, findPhysiotherapistsByExpertise
* bookAppointment, cancelAppointment, markAsAttended

### **5.3 JavaFX GUI**

The GUI allows the user to interact with the system through multiple screens:

* **Main Menu**: Entry point to booking, patient list, and reports.
* **Booking View**: Allows appointment booking based on selected expertise and available physiotherapists.
* **Patient List View**: CRUD operations on patient records.
* **Report View**: Displays all appointments and statistics.

## **6. Testing**

### **6.1 JUnit Tests**

All core functionalities were tested using JUnit 5 in BookingSystemTest:

* **testAddAndGetPatient**: Confirms patient addition.
* **testRemovePatient**: Ensures patient deletion works.
* **testBookAppointmentSuccess**: Verifies booking logic.
* **testCancelAndAttendAppointment**: Validates appointment status updates.
* **testFindPhysiotherapistByExpertise**: Filters by expertise.

### **6.2 Test Coverage**

The tests covered all business logic but did not include GUI tests. For GUI validation, tools like TestFX can be incorporated in future phases.

## **7. Report Generation**

Through the system, users receive text-based reports that present appointment information according to treatment and attendance data categories. BookingSystem.generateReport() contains all the logic that can be expanded to add support for PDF/CSV export capabilities going forward.

Sample Output:

==== Treatment Appointments ====

Appointment: Dr. Smith with Alice for a Massage

==== Physiotherapist by Attended Count ====

Dr. Smith: 2 attended

## **8. Results**

The final system meets all stated objectives:

* Supports multiple patient records.
* Enables expertise-based physiotherapist filtering.
* Books, cancels, and tracks appointments.
* Generates useful statistical reports.
* Provides a simple and intuitive GUI.
* Maintains consistent state via Singleton pattern.
* Achieves high test coverage and modular code quality.

## **9. Limitations and Challenges**

* **Data Persistence**: The current version stores data in memory. Restarting the app clears all data.
* **No Time Picker**: Appointment time is hardcoded as strings.
* **Limited Input Validation**: Minimal input sanitization in GUI.
* **UI Responsiveness**: GUI elements may not scale well on smaller screens.

## **10. Future Work**

Enhancements for future versions include:

* Database integration with SQLite or MySQL.
* Date and time picker for appointments.
* Search and filter functionalities in patient and appointment views.
* PDF/Excel export of reports.
* Login and access control for staff.
* REST API and Web Interface using Spring Boot.

## **11. Conclusion**

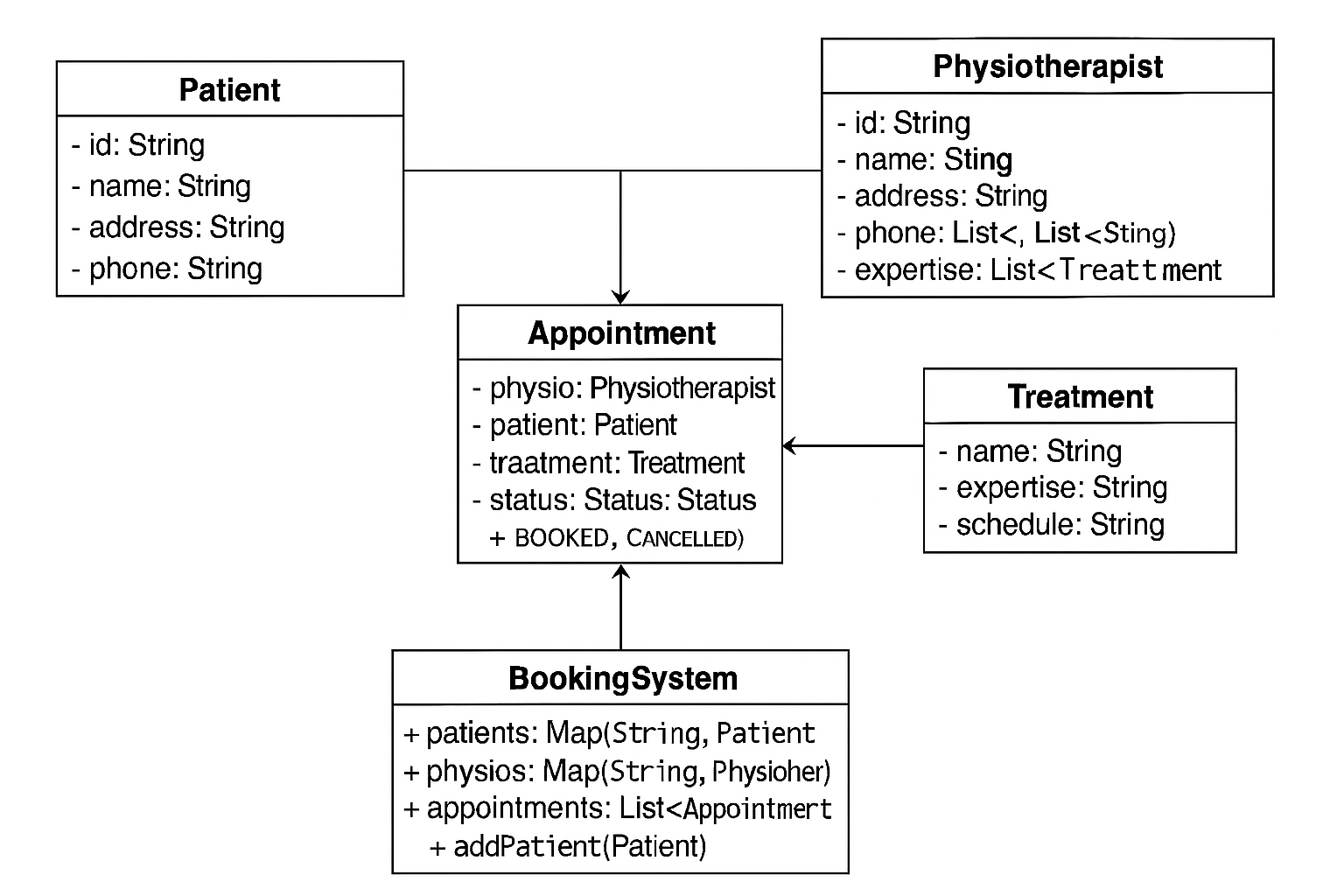
The project illustrated actual usage of object-oriented design combined with GUI programming methods as well as unit testing methods. The methodology has established a defined system development framework which prioritizes software code organization and easy maintenance. The proposed system demonstrates practical business value through its engagement with a physiotherapy healthcare facility.

The implementation meets all parts of the module specifications and creates a base for industrial-level clinical information systems development.

## **12. References**

* Deitel, P. and Deitel, H. (2017). *Java: How to Program, Late Objects Version*. 11th ed. Pearson.
* Oracle (2024). *JavaFX Documentation*. [online] Available at:<https://openjfx.io/> [Accessed 3 Apr. 2025].
* Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
* JUnit Team (2024). *JUnit 5 User Guide*. [online] Available at:<https://junit.org/junit5/> [Accessed 3 Apr. 2025].

## **Appendix A: UML Class Diagram**

****

**Note:** The report must be transformed into a PDF file with added screenshots of the program interface and report samples according to assessment requirements for documentation submission.