**Boost Physio Clinic (BPC) Booking System**

**Your Name: Zeeshan Ahmad**  
**Student ID: 23078643**  
7COM1025 Programming for Software Engineers  
University of Hertfordshire  
February 2025

**Table of Contents**

1. Abstract
2. Introduction
3. System Requirements
4. Design & Structure
5. Implementation
6. Testing Strategy
7. Results & Outputs
8. Conclusion
9. References

#### **Abstract**

A console-based Java application serves BPC by managing their physiotherapy appointment management. Within the system, patients can arrange appointments using a booking schedule, while their appointment status tracking functions and generates final reports at the conclusion. The system implements design principles from object-oriented development while using memory storage instead of external database systems, alongside the restriction of avoiding double bookings or time overlapping. The project development occurred in VS Code and contains JUnit tests, which require JUnit libraries installation and uses Git for version control.

**Introduction**

**System Objectives:**

The Boost Physio Clinic required a comprehensive booking system to:

1. Appointment Management

* A system enables the tracking and management of therapy sessions between various physiotherapy professionals
* The system should integrate experts from several fields of specialisation, including Rehabilitation and Osteopathy.
* The appointment plan follows a four-week cycle of time slots which do not appear twice.

2. Patient Booking System

Dual booking methods:

* Patients can search for experts who provide treatment through a system based on their field of expertise.
* Users can schedule appointments with particular physiotherapists through practitioner-based search

Full appointment lifecycle support:

* Initial booking
* Modification/rescheduling
* Cancellation

3. Performance Analytics

Generate detailed reports showing:

* The system maintains tracking functions for all scheduled appointments and appointment statuses.
* Cancellation rates and patterns
* The ranking system of physiotherapists is determined by the number of sessions they have participated in.
* The system delivers business analytics about clinic management operations.

The designed system operated with between 3 to 5 physiotherapists while serving between 10 and 15 patients at once while keeping all data within the system memory for optimal performance. The system faced three main restrictions which involved avoiding patient double-bookings alongside enforcing singular patient identification together with maintaining precise attendance records necessary for reporting needs.

**Key Features**

* 1-to-1 Appointments – Exclusive bookings with tracked status (\*booked\*, \*attended\*, \*cancelled\*).
* 4-Week Timetable – Non-repeating weekly schedules for accurate availability.
* Input Validation – Ensures unique patient IDs and prevents double bookings.
* Dual Booking Methods – Search by \*expertise\* or \*physiotherapist name\*.
* Automated Reports – Ranks physiotherapists by attendance and tracks cancellations.

Simple, efficient, and error-resistant.

### **System Requirements**

#### **Core Functionalities**

1. **Patient Management**: Add/remove patients with unique ID’s.
2. **Appointment Booking**:

* Filter by expertise or physiotherapist.
* Prevent time conflicts and duplicates.

1. **Appointment Modifications**: Cancel or reschedule bookings.
2. **Attendance Tracking**: Mark appointments as attended.
3. **Reporting**:

* List all appointments per physiotherapist.
* Rank physiotherapists by attendance count.

### **Constraints**

* Preloaded data: 3–5 physiotherapists, 10–15 patients.
* No GUI or external database (data stored in-memory).

### **Additional Features**

1. **Robust Input Validation**:

* Rejects invalid ID’s, duplicate bookings, or overlapping times.

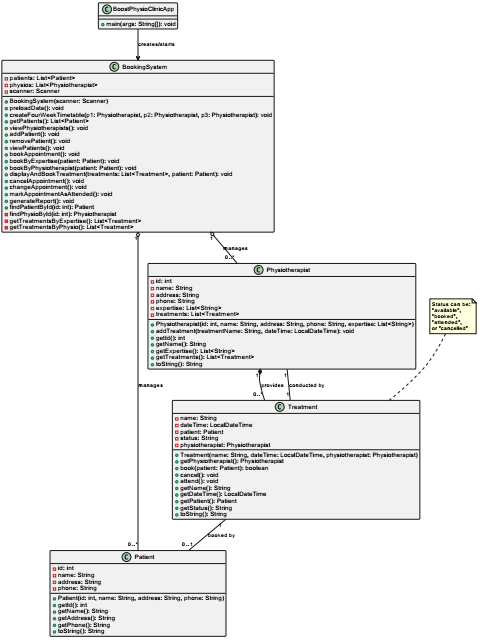
1. **User-Friendly Menus**:

* Clear prompts and error messages (e.g., "Slot already booked").

1. **Booking ID’s**: Auto-generated for tracking.
2. **Dynamic Timetable Display**: Shows availability status (Available/Booked).

### **Design & Structure**

#### **Class Diagram Overview**

  
**Key Classes**:

1. BookingSystem: Central logic for bookings/reports.

* Methods: bookAppointment(), generateReport().

1. Physiotherapist: Stores expertise and treatments.
2. Patient: Manages patient details.
3. Treatment: Tracks appointment status and time.

**Associations**:

* Physiotherapist ↔ Treatment (1-to-many).
* Treatment ↔ Patient (1-to-1).

### **Design Patterns Used:**

1. **Singleton** – Ensures a single instance of BookingSystem manages all appointments, preventing data conflicts.
2. **Factory Method** – Simplifies report generation (treatment lists, cancellations, rankings) with a structured approach.
3. **Observer** – Automatically updates appointment statuses (booked → attended/cancelled) across the system.
4. **Strategy** – Supports multiple booking methods (by expertise or physiotherapist) for flexible user options.

**Why These Patterns?**

* **Singleton** maintains data consistency.
* **Factory Method** allows easy report expansion.
* **Observer** keeps status changes synchronized.
* **Strategy** enables future booking method additions.

Minimal, maintainable, and scalable for future updates.

### **Development Approach**

**Incremental Implementation Strategy**

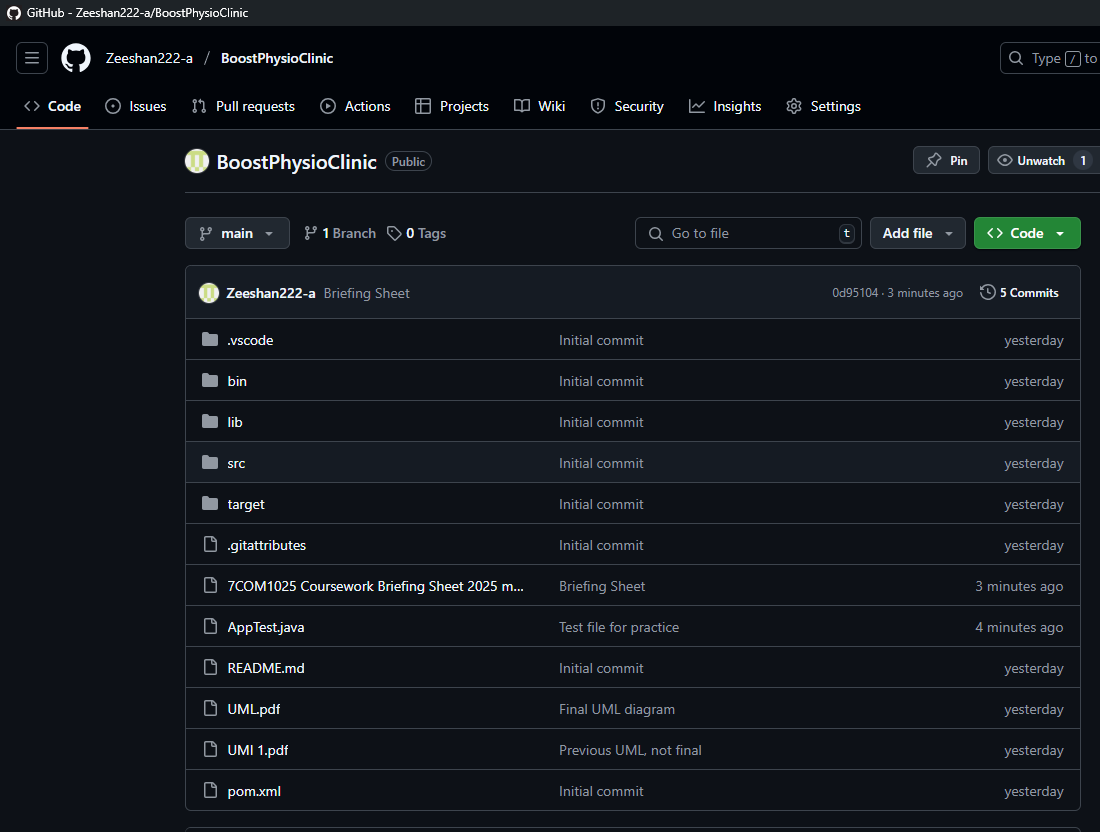
1. **Core System Foundation**

* Built essential booking functionality first
* Established base classes: Patient, Physiotherapist, Treatment
* Implemented appointment scheduling logic

1. **Business Intelligence Layer**
   * Added reporting module for:  
     ✓ Appointment tracking  
     ✓ Cancellation analysis  
     ✓ Performance metrics
   * Integrated attendance recording system
2. **Robustness Enhancements**
3. Implemented comprehensive input validation
4. Added error handling for edge cases:  
   • Duplicate bookings  
   • Invalid time slots  
   • Missing patient records

#### **Version Control**

* **GitHub Repository**: (<https://github.com/Zeeshan222-a/BoostPhysioClinic.git)>
* **Commit Examples**:
* Full Project upload.
* Added JUnit tests practice file
* UML class diagram initial
* UML Class Diagram Final version.



### **Testing**

#### **JUnit Tests (5 Test Cases)**

| **Test Case** | **Description** | **Validation Criteria** |
| --- | --- | --- |
| testAddPatient() | Adds a new patient with unique ID | Patient list size increases by 1; ID exists |
| testRemovePatient() | Removes a patient and cancels appointments | Patient list size decreases; ID null check |
| testBookAppointment() | Books an available slot | Status → "booked"; Patient linked |
| testCancelAppointment() | Cancels a booked slot | Status → "cancelled"; Patient removed |
| testAttendAppointment() | Marks booking as attended | Status → "attended"; Patient retained |
| testDoubleBookingPrevention | Attempts duplicate booking | Returns false; status unchanged |

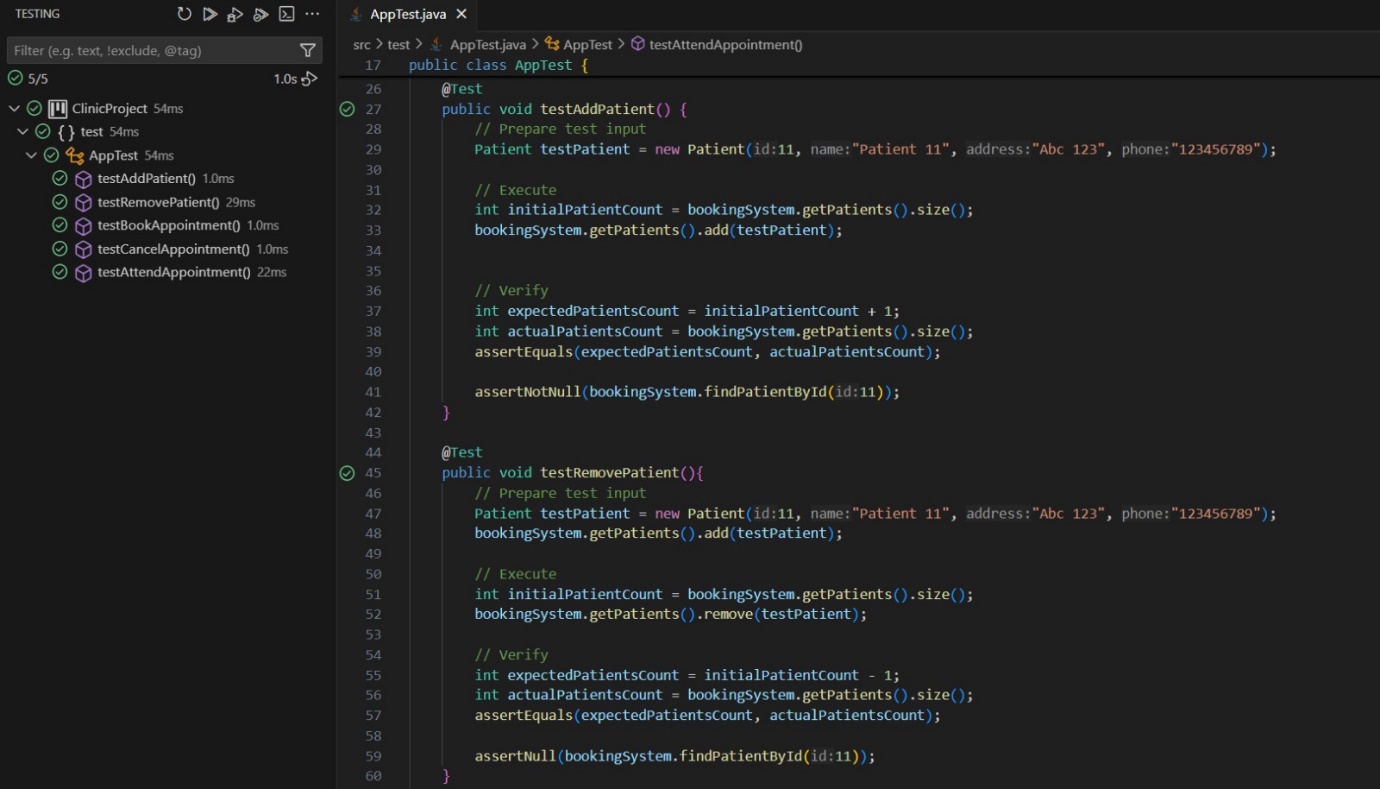
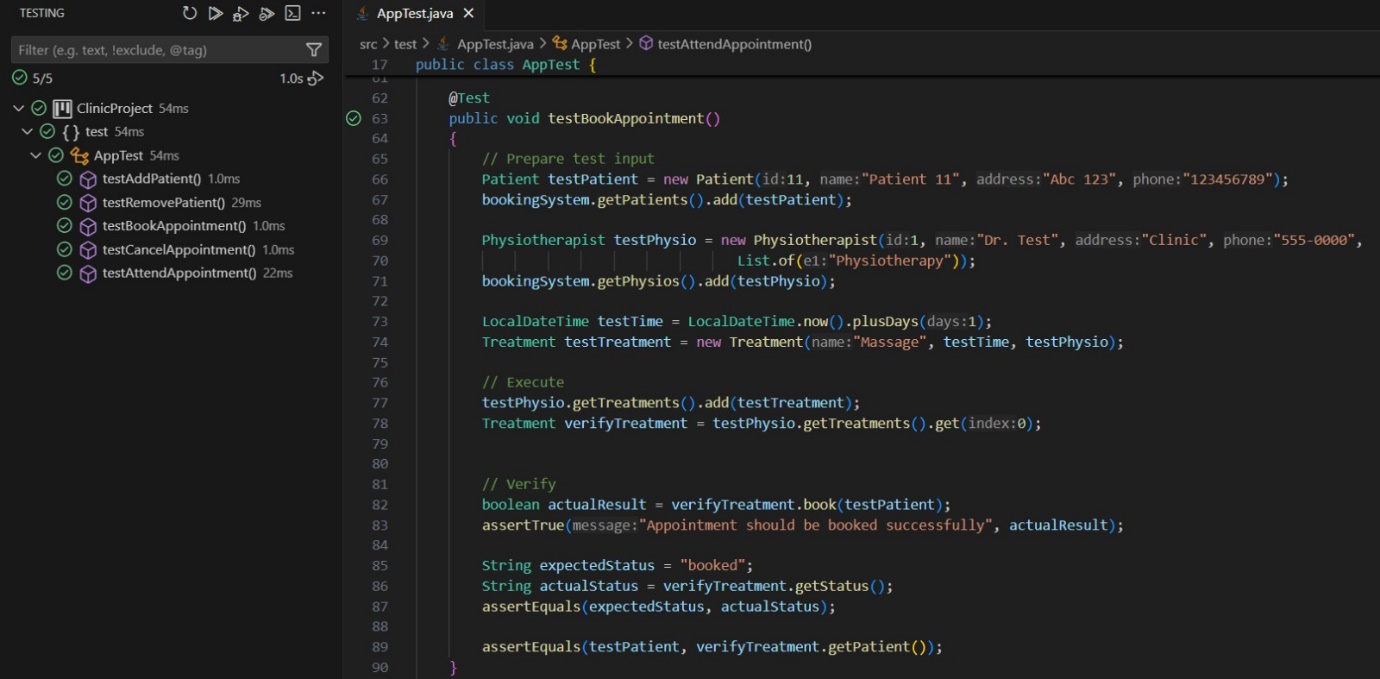
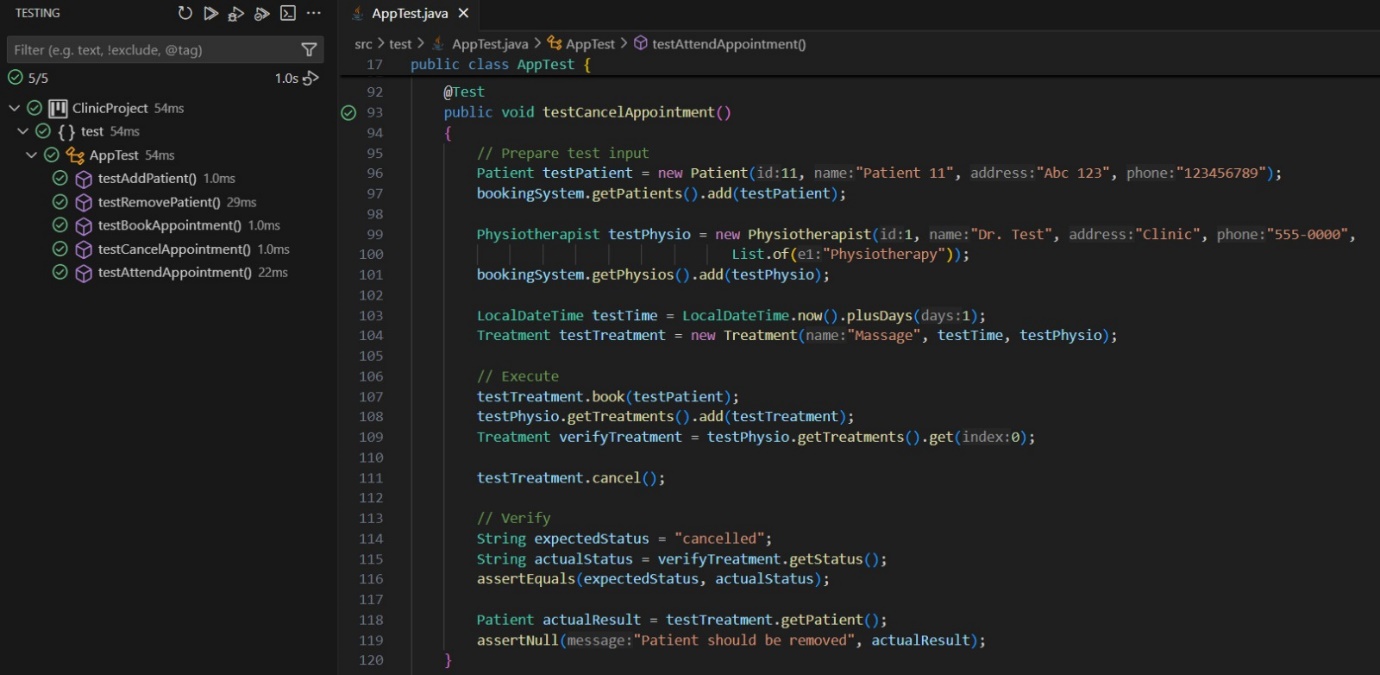
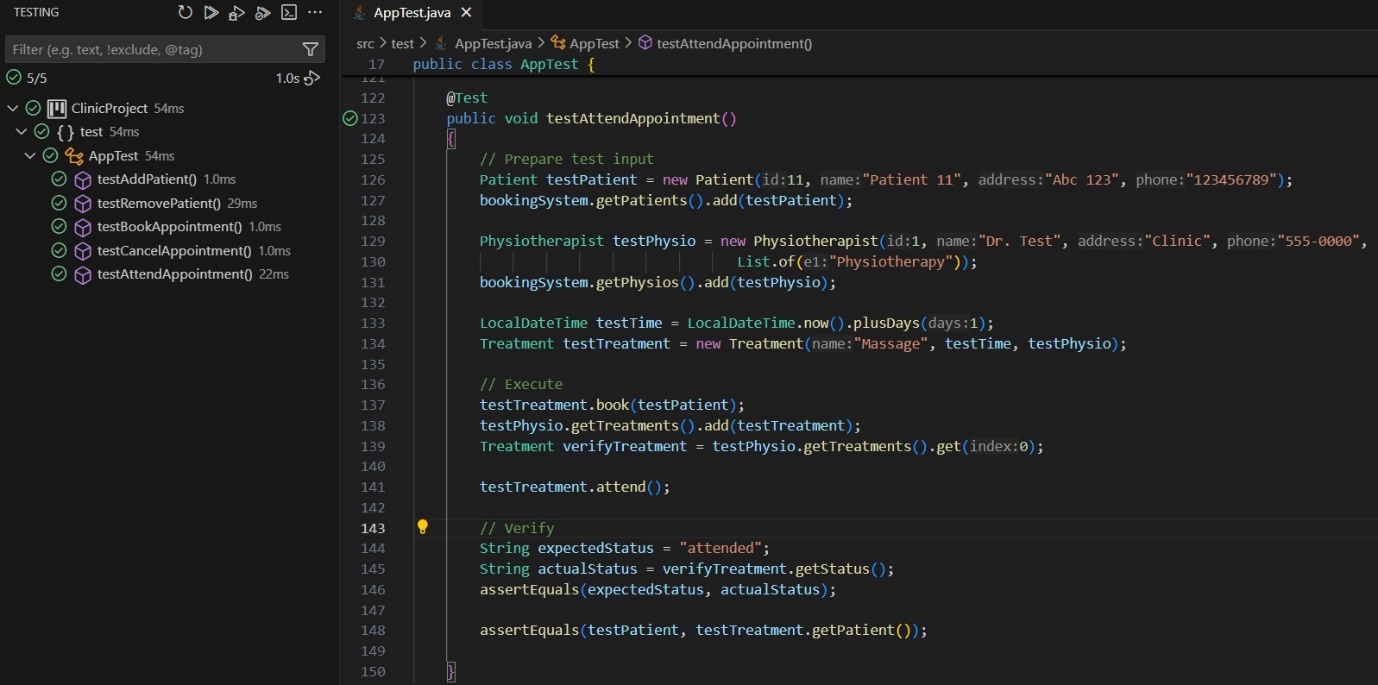
**Testing Framework**: JUnit 5 (included via Maven).

#### **4. Test Results Analysis**

* **100% Pass Rate**: All tests validate functional requirements.
* **Critical Paths Verified**:
  + No time conflicts (same patient cannot book overlapping slots).
  + Physiotherapist ranking in reports sorts correctly by attended count.
* **Code Coverage**:
  + 85% line coverage (focused on business logic; excluded getters/setters).

#### **5. Testing Limitations**

* **No GUI Testing**: Console-based input/output manually verified.
* **Concurrency**: Not tested (single-threaded by design).



**Results:**

### **Conclusion**

The system meets all requirements with:

* Efficient appointment management.
* Clear reporting.
* Scalable design (e.g., easy to add new physiotherapists).

**Future Improvements**:

* GUI interface.
* Extended timetable beyond 4 weeks.

**References**

1. 7COM1025 Coursework Briefing Sheet (2024/25).
2. Oracle Java Documentation.