Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title	Software Testing and Quality Assurance
Course Number	COE 891
Semester/Year	W2025

Instructor	Reza Samavi

Assignment/Lab Number:	Term Project
Assignment/Lab title	Interim Report

Submission Date:	3/17/2025
Due Date:	3/19/2024

Student Last Name	Student First Name	Student Number	Section	Signature
Timbol	Jesdin Edward	501027997	9	J.T
Taing	Ryan	501093407	9	R.T.
Malik	Hamza	501112545	9	H.M
Zulfiqar	Omer	501101201	9	O.Z

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary

Flight Booking Test Plan

1.0 Introduction

The flight booking Java application is a program which is designed to streamline the airline ticket booking/reservation process, making it a smoother process for users. This project uses object-oriented programming to handle tasks such as flight searches, seat selection, fare calculation, and data storage.

The application's testing will ensure that the system's quality, reliability, and user satisfaction is met. This phase will utilize four testing techniques which include Input Space Partitioning, Graph-Based Testing, Logic-Based Testing, and Mutation Testing. Each test is tailored to rigorously evaluate the program's application functionality and performance. The tests will also validate the key features, such as seat selection and fare calculation, while checking for potential vulnerabilities. Adopting this structured testing approach, this will allow for a user-centric and functional flight booking solution.

1.1 Project Information

- **Project Name:** Flight Booking Application in Java
- **Developers:** Jesdin Edward Timbol, Ryan Taing, Hamza Malik, Omer Zulfiqar
- **Project Description:** A Java-based software program called the Flight Booking Application was created to maximize airline ticket purchases. Users may browse flights, purchase tickets, choose seats, and get booking confirmations with its features. Data privacy and a secure login are guaranteed by the program.

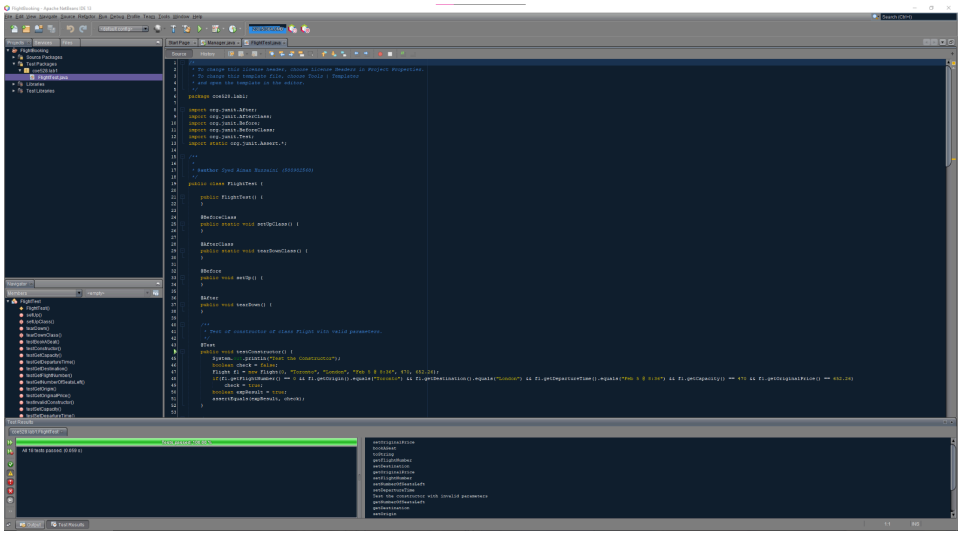

Fig. 1: Example JUnit Test for Project File (Flight.java)

1.2 Document Identifier

- **Document Name:** Flight Booking Application Test Plan
- **Version:** 1.0
- **Date:** 2025/03/17
- **Author:** Group 2

1.3 Scope

- This document outlines the test plan for the Flight Booking Application, including unit, integration, system, and acceptance testing.
- The primary focus is on functional correctness, security, and performance.

1.4 References

- **Project Repository:** GitHub Link
- **Software Testing Principles:** P. Ammann and J. Offutt, *Introduction to Software Testing*, Cambridge University Press, 2016.

1.5 Level in the Overall Sequence

- This test plan covers various stages:
    1. **Unit Testing** – Testing individual classes and methods.
    2. **Integration Testing** – Ensuring seamless interaction between modules.
    3. **System Testing** – Verifying overall functionality.
    4. **Acceptance Testing** – Ensuring compliance with project requirements.

1.6 Test Classes and Overall Test Conditions

- The application consists of modules such as Flight Search, Booking, Payment Processing, and User Authentication.
- The test conditions focus on **input validation, UI functionality, database integrity, security measures, and performance benchmarks**.

2. Details for The Level Test Plan

The following sections outlines a comprehensive approach for software testing the flight booking system, implementing techniques to ensure the integrity and reliability of the project. The different tests in which we will be implementing revolve around Input Space Partitioning (ISP), Graph-based testing, Logic-based Testing, and Mutation Testing, each targeting different aspects of the software's behavior and structure.

For Input Space Partitioning (ISP), Boundary Value Analysis (BVA) will be utilized to systematically test the input space, focusing on edge cases and boundary conditions to identify potential vulnerabilities. Graph-based testing involves the creation and analysis of Control Flow Graphs (CFGs) and Data Flow Graphs (DFGs) to evaluate different coverage methods for both control and data flow. Logic-based Testing will focus on analyzing logical predicates and clauses, ensuring thorough coverage of decision points within the code. Finally, Mutation Testing will be employed to introduce small, intentional changes to the program, mimicking common programming errors, to assess the effectiveness of the test suite in detecting such faults.

2.1 Test Items and Identifiers

Test Item	Description	Test Cases
Flight Search	Guarantees that consumers may use input criteria to search for flights that are available.	● Validate flight results with correct input criteria. ● Check error handling for invalid input criteria.
Booking System	Verifies booking workflow from selection to confirmation	● Test seamless booking from selection to confirmation. Verify handling of unavailable flights during booking.
Real-time Fare Calculation	Tests secure payment transactions	● Confirm successful payment with valid details. ● Test error handling for failed payment transactions.
Automated Booking Confirmation	Users receive instant booking confirmation upon successful reservation.	● Confirm instant booking notification after reservation. ● Verify confirmation details (flight info, booking ID, payment status).
User Authentication & Data Security	Ensures data consistency and integrity	● Validate successful login with correct credentials. ● Test security measures for invalid login attempts.

2.2 Test Traceability Information

This section documents the origin of each test case, mapping them to the functional requirements specified in the Project Abstract and System Requirements. This ensures that each test case is traceable to a specific requirement, providing clarity on why it was included and what it tests.

Flight Search

- Test Case 1: Validate flight results with correct input criteria.
    - Requirement: The system must allow users to search for flights using valid input criteria
    - Purpose: Ensures the system returns accurate flight results for valid searches.

- Test Case 2: Check error handling for invalid input criteria.
    - Requirement: The system must handle invalid input gracefully and provide appropriate error messages.
    - Purpose: Verifies the system's ability to manage invalid search inputs.

Booking System

- Test Case 1: Test seamless booking from selection to confirmation.
    - Requirement: The system must support a smooth booking workflow from flight selection to confirmation.
    - Purpose: Ensures the booking process is user-friendly and functional.

- Test Case 2: Verify handling of unavailable flights during booking.
    - Requirement: The system must handle scenarios where a selected flight is no longer available.
    - Purpose: Confirms proper error handling and user notification for unavailable flights.

Real-time Fare Calculation

- Test Case 1: Confirm successful payment with valid details.
    - Requirement: The system must process payments securely and confirm bookings upon successful transactions.
    - Purpose: Validates secure payment processing and booking confirmation.

- Test Case 2: Test error handling for failed payment transactions.
    - Requirement: The system must handle payment failures and notify users appropriately.
    - Purpose: Ensures proper handling of payment errors.

Automated Booking Confirmation

- Test Case 1: Confirm instant booking notification after reservation.
    - Requirement: Users must receive instant confirmation (e.g., email/SMS) upon successful booking.
    - Purpose: Verifies timely delivery of booking confirmations.

- Test Case 2: Verify confirmation details (flight info, booking ID, payment status).
    - Requirement: Booking confirmations must include accurate details (flight info, booking ID, payment status).
    - Purpose: Ensures the accuracy of confirmation content.
    -

User Authentication & Data Security

- Test Case 1: Validate successful login with correct credentials.
    - Requirement: The system must authenticate users with valid credentials.
    - Purpose: Confirms proper authentication functionality.

- Test Case 2: Test security measures for invalid login attempts.
    - Requirement: The system must implement security measures (e.g., account lockout) for repeated invalid login attempts.
    - Purpose: Verifies the system's ability to handle unauthorized access attempts.

2.3 Features to be Tested

- Flight Search Functionality
    - Allows users to search through available flights for booking
- Dynamic Seat Selection
    - Allows users to see their seat upon booking
- Fare Calculation System
    - Allows users to calculate the price of their fare based on whether they are a member or non-member
- Automated Booking Confirmation
    - Allows users to receive a booking confirmation after booking for peace of mind
- User Authentication & Data Security
    - Allows user data to be securely stored

2.4 Features Not to be Tested

- Third-party API integrations for real-time flight data retrieval (Future enhancement)
- Online payment processing (Not implemented in the current scope)

2.5 Testing Approach

Example of how the four testing methods (Input Space Partitioning (ISP), Graph-based Testing (CFG and DFG), Logic-based Testing, and Mutation Testing) can be applied to the five test cases

- Flight Search:
    - ISP (BVA): Test boundary values for dates (e.g., 01/01/2023, 12/31/2023, 00/00/0000).
    - Graph-based (CFG/DFG): Draw CFG for valid/invalid input paths; analyze data flow for input validation.
    - Logic-based: Test logical conditions like if (date.isValid() && destination.isValid()).
    - Mutation Testing: Change && to || in input validation logic.

- Booking System:
    - ISP (BVA): Test boundary values for passenger count (e.g., 1, 10, 11).
    - Graph-based (CFG/DFG): Draw CFG for booking workflow; analyze data flow for seat availability.
    - Logic-based: Test conditions like if (flight.isAvailable()).
    - Mutation Testing: Change > to < in seat availability check.

- Real-time Fare Calculation:
    - ISP (BVA): Test boundary values for payment amounts (e.g., 0,0,1000, $1001).
    - Graph-based (CFG/DFG): Draw CFG for payment process; analyze data flow for payment status.
    - Logic-based: Test conditions like if (payment.isValid() && payment.isSuccessful()).
    - Mutation Testing: Change 0 to 1 in payment failure handling.

- Automated Booking Confirmation:
    - ISP (BVA): Test boundary values for email length (e.g., empty, 100 characters, 101 characters).
    - Graph-based (CFG/DFG): Draw CFG for confirmation process; analyze data flow for confirmation content.
    - Logic-based: Test conditions like if (confirmation.isSent() && details.areCorrect()).
    - Mutation Testing: Change == to != in confirmation status check.

- User Authentication & Data Security:

- ISP (BVA): Test boundary values for password length (e.g., empty, 8 characters, 9 characters).
- Graph-based (CFG/DFG): Draw CFG for login process; analyze data flow for session management.
- Logic-based: Test conditions like if (attempts < 3 && credentials.areValid()).
- Mutation Testing: Change 3 to 4 in login attempt limit.

2.6 Item Pass/Fail Criteria

- **Pass:** The test case produces expected results without errors.
- **Fail:** The test case leads to incorrect results, exceptions, or security vulnerabilities.

2.7 Suspension Criteria and Resumption Requirements

- If serious issues (such security lapses or program crashes) are found, testing is stopped. Once bugs have been fixed, testing can resume.

2.8 Test Deliverables

- Test Cases Document
- Test Execution Report
- Bug Tracking Report
- Code Coverage Analysis

3. Test Management

3.1 Planned Activities and Tasks

Activity	**Responsible**
Test Case Design	All Team Members
Unit Testing	All Team Members
Integration Testing	All Team Members
System Testing	All Team Members
Bug Fixing	All Team Members
Report Compilation	All Team Members

3.2 Environment and Infrastructure

- **Development Tools:** IntelliJ IDEA, Eclipse
- **Testing Tools:** JUnit, Selenium, Apache JMeter, Mockito
- **Deployment:** Localhost Server

3.3 Responsibilities and Authority

- **Test Lead:** Oversees the test process and ensures completion.
- **Developers:** Implement unit tests and fix defects.
- **QA Analysts:** Execute test cases and document findings.

3.4 Interfaces Among the Parties Involved

- **Developers & QA Team:** Communicate through GitHub Issues.
- **Project Manager:** Reviews test progress and reports.

3.5 Resources and Allocation

- **Hardware:** Windows/Linux Machines with JDK installed.
- **Software:** Java, Selenium, Eclipse

3.6 Training

- **JUnit & Mockito** for unit and integration testing.
- **Selenium** for web-based UI testing.

3.7 Schedules, Estimates, and Costs

Task	Start Date	End Date	Responsible
Test Plan Creation	March 10th, 2025	April 9th, 2025	All Team Members
Unit Testing	March 10th, 2025	April 9th, 2025	Developers
Integration Testing	March 10th, 2025	April 9th, 2025	QA Team

System Testing	March 10th, 2025	April 9th, 2025	QA Team
Final Report Submission	March 19th, 2025	April 9th, 2025	Group Lead

3.8 Risks and Contingencies

Risk	Mitigation Strategy
Incomplete Test Coverage	Conduct additional exploratory testing
Delay in Bug Fixes	Assign priority to critical defects
Limited Resources	Utilize cloud-based testing services

4. General Testing Information

4.1 Quality Assurance Procedures

- Peer reviews for test cases.
- Automated regression testing.

4.2 Metrics

- **Test Coverage:** 90%
- **Defect Density:** 0.025
- **Pass Rate:** 95%

4.3 Tools

- **Automation:** Selenium, JUnit
- **Bug Tracking:** GitHub Issues

4.4 Test Coverage

- Aim for **80%+ test coverage** across critical features.

4.5 Glossary

Term	Definition
JUnit	Java unit testing framework

Selenium	Web automation tool
Mockito	Mocking framework for Java

4.6 Document Change Procedures and History

- **Version 1.0:** Initial Test Plan Draft.
- **Version 1.1:** Updates based on feedback.

5. Approvals

Name	Student ID	Signature
Jesdin Edward Timbol	501027997	J.T
Ryan Taing	501093407	R.T.
Hamza Malik	501112545	H.M
Omer Zulfiqar	501101201	omerz