Test Plan

Group Assignment

# **Document Control**

|  |  |
| --- | --- |
| **Document Name** | Test Plan |
| **Reference Number** | - |
| **Version** | V1.0 |
| **Project Code** | UECS2354 Software Testing Group Assignment |
| **Status** | Completed |
| **Date Released** | 12 September 2025 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Position** | **Signature** |
| Prepared By: | **Edmund Chan Chee An** | **Team Member** | ***Edmund*** |
| Reviewed By: | **Quak Jing** | **Team Member** | ***Quak*** |
| Approved By: | **Leon Siow Yi Hong** | **Group Leader** | ***Leon*** |
| Approved By: | **Narvin A/L Chandrasegaran** | **Team Member** | ***Narvin*** |

# **Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Release Date** | **Section** | **Amendments** |
| 1.0 | 22 Aug 2025 | Introduction | None |
| 2.0 | 29 Aug 2025 | Test Plans | None |
| 3.0 | 12 Sept 2025 | Document Review | Test Plans Items |

Table of Content

[Document Control 1](#_Toc127346001)

[Version History 2](#_Toc127346002)

[1.0 Introduction 1](#_Toc127346003)

[1.1 Objective 1](#_Toc127346004)

[1.2 Scope 1](#_Toc127346005)

[1.3 References 1](#_Toc127346006)

[2.0 Test Plans 2](#_Toc127346007)

[2.1 Test Items 2](#_Toc127346008)

[2.2 Features to be tested. 2](#_Toc127346009)

[2.3 Features not to be tested. 2](#_Toc127346010)

[2.4 Test Basis 2](#_Toc127346011)

[2.5 Test Conditions 2](#_Toc127346012)

[2.6 Entry Criteria 2](#_Toc127346013)

[2.7 Exit Criteria 2](#_Toc127346014)

# **Introduction**

## 1.1 Objective

The objective of this test plan is to define the testing approach and activities for the Travel Ticket Booking System. The main purpose of this project is to verify the system core functionalities including ticket booking, user registration, fare calculation, route distance handling, discount and surcharge application, and payment method adjustment.

This test plan is to ensure that the developed testing modules meet the specified functional requirements (FR1 – FR11) by identifying and addressing defects early in the development cycle. The testing objectives are:

* Verify that the system produces expected and correct output for valid input.
* Detect errors, failure and inconsistencies through systematic test design techniques.
* Determine the reliability, quality and stability of the module before integration with other systems.
* Ensuring the implemented system aligns with user requirement including distance-based fare rates, passenger-type discounts, time/date surcharges and payment method adjustments.

By executing this test plan, the development team would be able to validate the system function as intended, ensuring accuracy and reliability of system.

## 1.2 Scope

The scope of this test plan covers both unit testing and integration testing for the modules implemented in Travel Ticket Booking System. Each individual class and method will be tested to ensure correctness of output. Unit testing will be applying these test design techniques:

1. Equivalence Partitioning (EP) and Boundary Value Analysis (BVA) for input validation (e.g. distance and day/time)
2. Decision Table (DT) to handle discount and surcharge combination
3. Test Doubles (stub and mocks) using Mockito to simulate external dependencies
4. Parameterized tests with JunitParams to cover multiple input scenarios

**Unit Testing**

Unit testing will be focusing on verifying the small and isolated parts of an application call “Units”, specifically unit methods. Modules such as Booking.java. CalculateFare.java, FareAdjustment.java, FileFunctionality.java, GuestFile.java, RouteInfo.java and UserFile.java will be tested.

**Integration Testing**

Integration Testing will focus on verifying interaction between related modules such as Booking.java, CalculateFare.java, GuestFile.java and UserFile.java. The integration test will ensure that combined outputs match the expected results when modules interact.

## 1.3 Test Basis

The test condition and designs are based on following resources:

* UECS2354 Software Testing Group Assignment Details
* Feature listing table (FR1 to FR11)
* Decision Table and application code created by team members
* Development environment: Java Programming Language and Eclipse IDE
* External tools: Junit 4, Mockito, JunitParams

## 1.4 References

UECS2354 Software Testing Lecture Slide Chapter 5 Entry Criteria and Exit Criteria

# **2.0 Test Plans**

## 2.1 Test Items

1. Booking.java

1.1 public double getTotalFare()

1.2 public double getDiscountedFare()

1.3 public List<String> getDiscountDetails()

1.4 public void makePayment()

2. CalculateFare.java

2.1 public void calculateTotalFare(String startStation, String endStation)

2.2 public void calculateDiscountedFare(String travelDay, String travelTime, String startStation, String endStation, List<String> passengerType, List<Integer> passengerQuantity)

2.3 public void calculatePayment(String paymentMethod)

3. FareAdjustment.java

3.1 public String validatePassengerType(String passengerType)

3.2 public double passengerAdjustment(String passengerType)

3.3 public boolean isWeekend(String travelDay)

3.4 public int validateTravelTime(String travelTime)

3.5 public int dayTimeAdjustment(String travelDay, String travelTime)

3.6 public String validatePaymentMethod(String paymentMethod)

3.7 public double paymentMethodAdjustment(String paymentMethod)

4. RouteInfo.java

4.1 public Station validateStation(String stationName)

4.2 public double getRouteDistance(String startStation, String endStation)

5. FileFunctionality.java

5.1 public void writeToFile(String[ ] inputArray, String fileName)

5.2 public String[ ] readFromFile(String fileName)

6. GuestFile.java

6.1 public List<Guest> readGuestFromFile(String fileName)

6.2 public void writeGuestToFile(Guest[ ] guestArray, String fileName)

7. UserFile.java

7.1 public List<User> readUserFromFile(String fileName)

7.2 public void writeUserToFile(User[ ] userArray, String fileName)

## 2.2 Features to be tested.

**1. Booking.java**

1.1 getTotalFare() - This method will get the total fare from CalculateFare.java

1.2 getDiscountedFare() – This method will get the total discounted fare from CalculateFare.java

1.3 getDiscountDetails() – This method will get the discount details from CalculateFare.java

1.4 makePayment() – This method will invoke calculatePayment() in CalculateFare.java

**2. CalculateFare.java**

2.1 calculateTotalFare() – This method calculates total fare using RouteInfo.java and FareAdjustment.java

2.2 calculateDiscountedFare() – This method will calculate discounted fare using FareAdjustment.java

2.3 calculatePayment() – This method will calculate payment for fare based on payment method using FareAdjustment.java

**3. FareAdjustment.java**

3.1 validatePassengerType() – This method will validate passenger type and return formatted passenger type

3.2 passengerAdjustment() – This method will do fare adjustment based on passenger type

3.3 isWeekend() – This method will verify travel day and return Boolean argument

3.4 validateTravelTime() – This method will validate travel time

3.5 dayTimeAdjustment() – This method will make payment adjustment based on day and time

3.6 validatePaymentMethod() – This method will validate the payment method

3.7 paymentMethodAdjustment() – This method make fare adjustment based on payment method

**4. RouteInfo.java**

4.1 validateStation() – This method will validate the station name and return Station object

4.2 getRouteDistance() – This method will find the distance between start and end station based on algorithm created

**5. FileFunctionality.java**

5.1 writeToFile() – This method will write the array of String into specified file

5.2 readFromFile() – This method will read data from specified file path and return array of String

**6. GuestFile.java**

6.1 readGuestFromFile() – This method will read guest data from specified file

6.2 writeGuestToFile() – This method will write new guest into specified file

**7. UserFile.java**

7.1 readUserFromFile() – This method will read user data from specified file

7.2 writesUserToFile() – Th is method will write new user into specified file

## 2.3 Features not to be tested.

1. Station.java – class to instantiate each station for path finding algorithm used in RouteInfo.java

2. Guest.java – class to store information such as name, email, phoneNo of a Guest

3. User.java – class to store information such as ID, name, email, phoneNo of a User

4. Payment.java – class created as a mock to interact with other core modules

5. emailNotification – an additional feature to email the receipt to the user/guest once payment is finished

## 2.4 Test Conditions

**1. Booking.java**

1.1 getTotalFare()

* To verify that calculateTotalFare is invoked

1.2 getDiscountedFare()

* To verify that calculateDiscountedFare is invoked

1.3 getDiscountDetails()

* To verify that getAdjustmentDetails is invoked and the details is not empty or null

1.4 makePayment()

* To verify that calculatePayment is invoked

**2. CalculateFare.java**

2.1 calculateTotalFare()

* To verify that total fare is calculated correctly based on starting station and ending station

2.2 calculateDiscountedFare()

* To verify that discounted fare is calculated correctly based on travel day, travel time, starting station, ending station, passenger type, and passenger quantity

2.3 calculatePayment()

* To verify that payment amount is calculated correctly based on payment method

**3. FareAdjustment.java**

3.1 validatePassengerType()

* To verify that a passenger type is correctly validated

3.2 passengerAdjustment()

* To verify that the fare adjustment for passenger type is correct

3.3 isWeekend()

* To verify that the travel day is correctly validated as a weekend

3.4 validateTravelTime()

* To verify that the travel time is correctly validated as a valid time

3.5 dayTimeAdjustment()

* To verify that the fare adjustment for travel day and travel time is correct

3.6 validatePaymentMethod()

* To verify that the payment method is correctly validated

3.7 paymentMethodAdjustment()

* To verify that the fare adjustment for payment method is correct

**4. RouteInfo.java**

4.1 validateStation()

* To verify that the station is correctly validated as a real station
  1. getRouteDistance()
* To verify that the distance calculated between stations is correct

**5. FileFunctionality.java**

5.1 writeToFile()

* To verify that the file is written correctly

5.2 readFromFile()

* To verify that the content read from a file is correct

**6. GuestFile.java**

6.1 readGuestFromFile()

* To verify that the content read from a guest file is correct

6.2 writeGuestToFile()

* To verify that the guest file is written correctly

**7. UserFile.java**

7.1 readUserFromFile()

* To verify that the content read from a user file is correct

7.2 writesUserToFile()

* To verify that the user file is written correctly

## 2.5 Entry Criteria

**System Testing can begin when**

* Bug tracking and test tracking system are in place and operational
* All components are under formal and automated configure
* Development Teams have completed all features and bug fixes scheduled for the release
* Development team deliver software build to the testing team at least 3 business days prior to System test start
* The project management team formally agrees in the system test phase entry meeting to proceed, with the following decisions made
  1. Confirming the code implementation and unit testing are complete
  2. Assign target fix dates for any detected bugs

## 2.6 Exit Criteria

**System Testing can be considered complete when**

* No design, code or feature changes have occurred in the prior 1 weeks
* No client system became inoperable due to failed updates during system test
* The development team has resolved all must fix bugs
* The project management team agrees that the project as defined during the final System Test cycle satisfies the customer reasonable expectations of quality
* A formal system test phase exit meeting was held, and the project management teams agreed that the system testing is complete