Software Testing Assignment 1

Alireza Dastmalchi Saei

Student ID: *993613026*

University: *University of Isfahan*

Course: *Software Testing Course*

November 24, 2023

1 Introduction

In this comprehensive report, we embark on an exploration and evaluation of a Train Reservation System using a dual-approach testing methodology. Leveraging the power of JUnit and Gherkin, we conduct a thorough examination of the system's functionalities, ensuring its robustness, reliability, and adherence to business requirements.

JUnit, a widely adopted testing framework for Java applications, provides a solid foundation for scenario-based testing. Through the creation of five detailed scenarios, we utilize JUnit to meticulously assess various aspects of the Train Reservation System. These scenarios encompass critical functionalities such as City and Train Management, Trip Management, Ticket Booking and Cancellation, Delay Management, and Exchange Management (With Gherkin).

Complementing our JUnit tests, we employ Gherkin, a language-agnostic format, to write three additional scenarios. Gherkin facilitates Behavior-Driven Development (BDD) by allowing us to describe system behavior in a natural language format that is easily understandable by both technical and non-technical stakeholders.

This integrated approach ensures a comprehensive evaluation of the Train Reservation System, covering both the low-level unit testing provided by JUnit and the high-level behavioral testing facilitated by Gherkin. By combining these two testing methodologies, we aim to provide a robust and thorough validation of the system's functionality, ensuring its readiness for deployment in real-world scenarios.

2 Test Scenario 1: City and Train Management

2.1 Test Case 1.1: Add a New City

Description: Verify that a new city can be added to the system.
Preconditions:

- The system is running.
- The user is authorized to add a new city.

Steps:

1. Add a new city.

Expected Result: The city should be added to the list of cities in the system.

2.2 Test Case 1.2: Add a New Train

Description: Verify that a new train can be added to the system.
Preconditions:

- The system is running.
- The user is authorized to add a new train.

Steps:

1. Add a new train.

Expected Result: The new train should be added to the list of trains in the system.

3 Test Scenario 2: Trip Management

3.1 Test Case 2.1: Create a New Trip

Description: Verify that a new trip can be added to the system considering the specific constraints.
Preconditions:

- The system is running.
- The user is authorized to add a new trip.

Steps:

1. Provide valid details for the new trip: Origin
2. Provide valid details for the new trip: Destination
3. Provide valid details for the new trip: Train
4. Provide valid details for the new trip: Departure Time
5. Provide valid details for the new trip: Arrival Time

Expected Result: The new trip should be created and registered in the system.

3.2 Test Case 2.2: Cancel Trip

Description: Verify that a trip can be canceled in the system.
Preconditions:

- The system is running.
- The user is authorized to cancel a trip.

Steps:

1. Select a trip to cancel

Expected Result: The selected trip should be canceled, and all associated tickets should also be canceled.

3.3 Test Case 2.3: Add a trip with conflicting timings to a train

Description: Verify that a trip with conflicting times cannot be added to the train trips list.
Preconditions:

- The system is running.
- The user is authorized to add a trip.

Steps:

1. Add a valid trip to train
2. Add a new trip that has intercepting times with previous trip

Expected Result: The second should not be added to the trips list of the train, and a trip exception must be thrown.

4 Test Scenario 3: Ticket Booking and Cancellation

4.1 Test Case 3.1: Book a Ticket

Description: Verify that a new ticket can be booked for a trip if the trip has not reached the maximum number of passengers.
Preconditions:

- The system is running.
- The user is authorized to book a ticket.
- There is an available trip.

Steps:

1. Select and available trip
2. Provide passenger name
3. Book a ticket

Expected Result: A new ticket should be booked for the selected trip.

4.2 Test Case 3.2: Cancel a Ticket

Description: Verify that a booked ticket can be canceled.
Preconditions:

- The system is running.
- The user has a booked ticket.

Steps:

1. Select a booked ticket

Expected Result: The selected ticket should be canceled, and the trip should be updated accordingly.

4.3 Test Case 3.3: Book a Ticket for a full trip

Description: Verify that a ticket for a full trip cannot be created.
Preconditions:

- The system is running.

Steps:

1. Create a ticket for a trip with max passengers

Expected Result: The ticket should not be booked and it must give an error.

5 Test Scenario 4: Delay Management

5.1 Test Case 4.1: Add Departure Delay to a Trip

Description: Verify that a departure delay can be added to a trip, and it updates the real departure time.
Preconditions:

- The system is running.
- There is a trip available for delay.

Steps:

1. Select a trip
2. Add departure delay

Expected Result: The departure delay should be added to the trip, and the real departure time should be updated accordingly.

5.2 Test Case 4.2: Add Arrival Delay to a Trip

Description: Verify that an arrival delay can be added to a trip, and it updates the real arrival time.
Preconditions:

- The system is running.
- There is a trip available for delay.

Steps:

1. Select a trip
2. Add arrival delay

Expected Result: The arrival delay should be added to the trip, and the real arrival time should be updated accordingly.

5.3 Test Case 4.3: Add Departure Delay More Than Duration to Pass Arrival Date

Description: Verify that a departure date cannot pass arrival time after being delayed.
Preconditions:

- The system is running.
- There is a trip available for delay.

Steps:

1. Select a trip
2. Add departure delay more that duration

Expected Result: The departure delay should not be added to the trip, or the arrival time must be updated.

6 Test Scenario 5: Exchange Management (Cucumber)

6.1 Gherkin 5.1: Find All Exchangeable Tickets

Someone wants to exchange his/her ticket, available tickets for exchanges will be shown
Given → Exchange requirements are met
When → Request to see available tickets for exchange
Then → A list of all available tickets are shown

6.2 Gherkin 5.2: Get previous and Next Trip of a Train

Someone has a trip of a train and wants to see previous and next trip of that train
Given → Train has more that three trips and has a previous and next trip
When → Request to see available earlier and later trips of a train
Then → The correct successor and predecessor trips are shown

6.3 Gherkin 5.3: Exchange a Ticket successfully

Someone wants to exchange his/her ticket, the exchanging process must be correct
Given → Exchanging requirements are met
When → Request to exchange a ticket from a trip
Then → Exchange must be done for a trip