

**CSE 731: Software Testing**  
**Term I 2024-25: Project Work**

**Mutation Testing**

**Srishti Yadav - MT2023125**  
**Subhankhi Maiti - MT2023113**

# **Contents**

1. **Project Objective**
2. **Code Tested**
3. **Testing Strategies**
4. **Testing Tools Used**
  - JUnit 5
  - PIT (Pitest)
  - TestNG
5. **Mutation Testing Process and Results**
  - Detailed Process
  - Mutation Operators Used
  - PIT Coverage Images and Reports
6. **Contributions**
7. **References**

## Project Objective

The goal of this project is to self-learn and understand the practical aspects of software testing by employing open-source tools and test case design strategies as covered in the course.

Specifically, this project involves implementing Mutation Testing for assessing and improving the robustness of a ticket price calculation system.

## Code Tested

Mutation testing was applied to the **TicketPriceCalculator** project, focusing on fare calculation and sorting logic across various transportation modes.

### **Key Details:**

- **Objective:** Evaluate the robustness of the test suite by introducing deliberate mutations into the program and observing if test cases could identify them.
- **Test Scope:** Included edge cases, invalid inputs, family discounts, and senior citizen pricing.

[\[GitHub Link\]](#)

## Testing Strategies

Mutation testing evaluates the effectiveness of a test suite by introducing minor changes (mutations) into the code and verifying whether the test cases can detect these changes. This approach helps measure the robustness of the tests and uncovers potential weaknesses.

- **Strong Killing:** A mutant is strongly killed if the test case not only detects the mutation but also triggers a distinctly incorrect behavior, such as a failed assertion or an unexpected output.
- **Weak Killing:** A mutant is weakly killed if the mutation is detected during execution (e.g., an intermediate state change) but does not lead to observable failures, such as incorrect final results.

In this project, we aimed for strong killing of mutants in the TicketPriceCalculator system, ensuring that our test cases could detect and manifest failures for any injected mutations. This approach aligns with the goal of creating a robust and comprehensive test suite.

# Testing Tools Used

## JUnit 5

- **Purpose:**  
JUnit 5 is a widely adopted Java testing framework used to write and execute unit tests. In our project, it played a pivotal role in crafting robust test cases for evaluating the **TicketPriceCalculator** system and achieving strong mutation testing results.
  - **Key Features:**
    1. **Annotations:**  
Facilitated structured test cases using annotations such as **@Test**, **@BeforeEach**, **@AfterEach**, and **@ParameterizedTest**.
    2. **Assertions:**  
Provided a rich set of assertions (**assertEquals()**, **assertTrue()**, **assertThrows()**) to validate expected outputs effectively.
    3. **Parameterized Testing:**  
Enabled tests to be executed with multiple inputs, improving coverage and robustness.
    4. **Integration:**  
Worked seamlessly with PIT for mutation testing, enhancing our ability to assess test quality comprehensively.
- 

## Mutation Testing - PIT

- **Purpose:**  
PIT is a specialized mutation testing tool for Java. It evaluates the quality of test cases by introducing intentional code changes (mutants) and checks whether the test suite can detect these alterations. It is an essential tool for enhancing the robustness of the test suite.
- **Functionality:**
  1. **Mutant Generation:**
    - Automatically generates modified versions of the original code, called mutants.
    - Applies mutation operators such as:
      - Changing arithmetic operations (e.g., replacing **+** with **-**).
      - Inverting conditions (e.g., replacing **==** with **!=**).

- Removing method calls or altering logical operators.
  - 2. **Test Execution:**
    - Executes the test suite against each mutant.
    - Verifies if the tests detect the introduced changes and behave accordingly (e.g., fail an assertion or produce incorrect output).
  - **Metrics:**
    1. **Mutation Coverage Percentage:**
      - Measures the proportion of mutants detected (killed) by the test suite, reflecting test effectiveness.
    2. **Killed Mutants:**
      - Mutants successfully detected and eliminated by the test cases.
      - Indicates a strong and effective test suite.
    3. **Survived Mutants:**
      - Mutants that the test suite fails to detect.
      - Highlights potential weaknesses in the test cases, prompting further improvement.
- 

## TestNG

- **Purpose:**

TestNG is a testing framework for Java designed to simplify and enhance the testing process, especially for unit testing and integration testing. It offers powerful features for test configuration, grouping, and parallel execution, making it an ideal choice for large-scale test suites and complex testing scenarios.
- **Key Features:**
  1. **Annotations:**

TestNG provides a rich set of annotations such as **@Test**, **@BeforeMethod**, **@AfterMethod**, **@BeforeClass**, **@AfterClass**, and **@DataProvider**. These annotations help structure and control the flow of tests, setup, and teardown operations.
  2. **Test Configuration:**

Allows for flexible configuration of tests, such as grouping tests, setting execution priorities, and specifying dependencies between tests. This helps organize tests based on different criteria, making the test suite easier to manage.
  3. **Parallel Execution:**

TestNG supports parallel test execution, enabling multiple tests or test classes to run simultaneously. This significantly reduces test execution time, especially for large test suites.

4. **Data-Driven Testing:**  
The **@DataProvider** annotation allows for running the same test with different sets of data, enhancing test coverage and robustness. This is useful for testing functions with multiple input combinations.
5. **Assertions:**  
TestNG offers a variety of assertions (e.g., **assertEquals()**, **assertTrue()**, **assertNotNull()**) to validate expected outcomes and conditions in the tests.
- **Integration:**  
TestNG can be seamlessly integrated with other tools and frameworks like JUnit 5, PIT (for mutation testing), and RESTAssured (for API testing). This makes it a versatile tool for various types of testing, including unit, integration, and functional tests.
- **Application in Our Project:**  
In our project, TestNG was used alongside JUnit 5 and PIT to create a comprehensive test suite for the **TicketPriceCalculator** system, ensuring both high-quality code and robust test coverage.

## Testing Process and Results

### Mutation Testing - Detailed Process Using PIT

- **Setup:** Configured PIT (Pitest) with Maven to perform mutation testing on the TicketPriceCalculator codebase. The configuration targeted both fare calculation logic and sorting functions.
- **Mutation Execution:** PIT generated mutants by introducing deliberate changes such as:
  - Negating conditional statements (e.g., `>` to `<=`).
  - Modifying arithmetic or logical operators (e.g., `+` to `-`, `&&` to `||`).
  - Changing return values or removing method calls.
- **Test Execution:** The existing test suite was executed against these mutants to determine which mutations were identified and killed.
- **Result Analysis:** Surviving mutants (those undetected by the test suite) were analyzed to identify gaps in test coverage. Enhanced test cases to ensure detection of previously surviving mutants, improving the robustness of the test suite.

**Programming Language:** Java

**IDE Used:** IntelliJ

### List of Mutation Operators Used

- **Unit Level Mutation Operators:** Focuses on changes within isolated methods

or functions, affecting individual logic components.

- **Integration Level Mutation Operators:** Introduces changes that impact interactions between multiple components, such as object returns or method calls affecting larger workflows.

PIT, by default provides a set of mutation operators. These operators are listed below:

BOOLEAN_FALSE_RETURN	INCREMENTS_MUTATOR
BOOLEAN_TRUE_RETURN	INVERT_NEGS_MUTATOR
CONDITIONALS_BOUNDARY_MUTATOR	MATH_MUTATOR
EMPTY_RETURN_VALUES	NEGATE_CONDITIONALS_MUTATOR
PRIMITIVE_RETURN_VALS_MUTATOR	VOID_METHOD_CALL_MUTATOR
NULL_RETURN_VALUES	

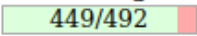
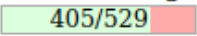
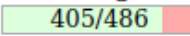
For more information on the mutation operators please refer to <https://pitest.org/quickstart/mutators/>

## PIT Coverage Images and Sample Reports

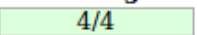
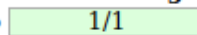
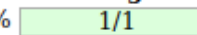
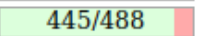
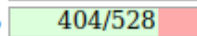
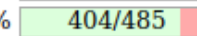
# Pit Test Coverage Report

### Package Summary

**com.thedeciders.pitest**

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
2	91% 	77% 	83% 

### Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
<a href="#">Passenger.java</a>	100% 	100% 	100% 
<a href="#">TicketPriceCalculator.java</a>	91% 	77% 	83% 

---

Report generated by [PIT](#) 1.9.11



# Passenger.java

```
1 package com.thedeciders.pitest;
2
3 public class Passenger {
4     private int age;
5
6     public Passenger(int age) {
7         this.age = age;
8     }
9
10    public int getAge() {
11        return age;
12    }
13
14    // public void setAge(int age) {
15    //     this.age = age;
16    // }
17
18 }
19
```

## Mutations

12 1. replaced int return with 0 for com/thedeciders/pitest/Passenger::getAge → KILLED

## Active mutators

- CONDITIONALS\_BOUNDARY
- EMPTY\_RETURNS
- FALSE\_RETURNS
- INCREMENTS
- INVERT\_NEGS
- MATH
- NEGATE\_CONDITIONALS
- NULL\_RETURNS
- PRIMITIVE\_RETURNS
- TRUE\_RETURNS
- VOID\_METHOD\_CALLS

### Tests examined

[illegible]

Report generated by PIT 1.9.11

## TicketPriceCalculator.java

```

1 package com.thedeciders.pitest;
2
3 import java.util.List;
4
5 public class TicketPriceCalculator {
6
7     public static int ADULT_AGE_BUS = 18;
8     public static int FREE_TICKET_AGE_BELOW_BUS = 3;
9     private static int SENIOR_CITIZEN_BUS = 60;
10    private static int ARMY_OFFICER_BUS = 18;
11    public static double FAMILY_DISCOUNT_BUS= 0.05;
12
13    private static int ADULT_AGE_TRAIN = 18;
14    private static int FREE_TICKET_AGE_BELOW_TRAIN = 3;
15    private static int SENIOR_CITIZEN_TRAIN = 60;
16    private static int ARMY_OFFICER_TRAIN = 18;
17    public static double FAMILY_DISCOUNT_TRAIN= 0.25;
18
19
20    private static int ADULT_AGE_AIRPLANE = 18;
21    private static int FREE_TICKET_AGE_BELOW_AIRPLANE = 3;
22    private static int SENIOR_CITIZEN_AIRPLANE = 60;
23    private static int ARMY_OFFICER_AIRPLANE = 18;
24    public static double FAMILY_DISCOUNT_AIRPLANE= 1;
25
26
27    private static int ADULT_AGE_TAXI = 18;
28    private static int FREE_TICKET_AGE_BELOW_TAXI = 3;
29    private static int SENIOR_CITIZEN_TAXI = 60;
30    private static int ARMY_OFFICER_TAXI = 18;
31    public static double FAMILY_DISCOUNT_TAXI= 0.15;
32
33 }

```

```

100
101     public double calculatePriceBus(List<Passenger> passengers, int adultTicketPrice, int childTicketPrice) {
102         int totalPriceBus = 0;
103         int childrenCounterBus = 0;
104         int adultCounterBus = 0;
105         double resultBus=0.0;
106         int seniorCitizen = 0;
107
108         for (Passenger passenger : passengers) {
109             // Skip invalid passengers
110             if (passenger.getAge() > 150 || passenger.getAge() < 0) {
111                 System.out.println("Not Valid");
112                 continue; // Skip processing this passenger
113             }
114
115             // Senior citizen check (optional, for output only)
116             if (passenger.getAge() > 60 && passenger.getAge() < 100) {
117                 System.out.println("Person is senior citizen");
118                 seniorCitizen++;
119                 totalPriceBus+=adultTicketPrice;
120                 continue;
121             }
122
123             // Calculate ticket price based on age
124             if (passenger.getAge() > ADULT_AGE_BUS) {
125                 totalPriceBus += adultTicketPrice;
126                 adultCounterBus++;
127             } else if (passenger.getAge() > FREE_TICKET_AGE_BELOW_BUS) {
128                 totalPriceBus += childTicketPrice;
129                 childrenCounterBus++;
130             }
131
132         }
133
134         // Apply family discount if applicable
135         if (childrenCounterBus > 1 && adultCounterBus > 1) {
136             resultBus = (1 - FAMILY_DISCOUNT_BUS) * totalPriceBus;
137         } else {
138             resultBus = totalPriceBus;
139         }
140
141         return resultBus;
142     }

```

```

195 void mergeBus(int arr[], int l, int m, int r)
196 {
197     int i, j, k;
198     int n1 = m - l + 1;
199     int n2 = r - m;
200
201     int L[] = new int[n1];
202     int R[] = new int[n2];
203
204     for (i = 0; i < n1; i++)
205         L[i] = arr[l + i];
206     for (j = 0; j < n2; j++)
207         R[j] = arr[m + 1 + j];
208
209     i = 0;
210     j = 0;
211     k = l;
212     while (i < n1 && j < n2)
213     {
214         if (L[i] <= R[j])
215             arr[k] = L[i];
216             i++;
217         else
218             arr[k] = R[j];
219             j++;
220         k++;
221     }
222
223     while (i < n1)
224     {
225         arr[k] = L[i];
226         i++;
227         k++;
228     }
229
230     while (j < n2)
231     {
232         arr[k] = R[j];
233         j++;
234         k++;
235     }
236 }

```

## Mutations

	1. changed conditional boundary → KILLED
110	2. changed conditional boundary → SURVIVED
	3. negated conditional → KILLED
	4. negated conditional → KILLED
111	1. removed call to java/io/PrintStream::println → SURVIVED
	1. changed conditional boundary → SURVIVED
117	2. changed conditional boundary → SURVIVED
	3. negated conditional → KILLED
	4. negated conditional → SURVIVED
118	1. removed call to java/io/PrintStream::println → SURVIVED
119	1. Changed increment from 1 to -1 → SURVIVED
120	1. Replaced integer addition with subtraction → KILLED
125	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
126	1. Replaced integer addition with subtraction → KILLED
127	1. Changed increment from 1 to -1 → KILLED
128	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
129	1. Replaced integer addition with subtraction → KILLED
130	1. Changed increment from 1 to -1 → KILLED
	1. changed conditional boundary → KILLED
135	2. changed conditional boundary → KILLED
	3. negated conditional → KILLED
	4. negated conditional → KILLED
136	1. Replaced double subtraction with addition → KILLED
	2. Replaced double multiplication with division → KILLED
141	1. replaced double return with 0.0d for com/thedeciders/pitest/TicketPriceCalculator::calculatePriceBus → KILLED
	1. changed conditional boundary → KILLED
170	2. Replaced integer subtraction with addition → KILLED
	3. negated conditional → KILLED
171	1. Replaced integer multiplication with division → TIMED_OUT
	1. changed conditional boundary → SURVIVED
176	2. Replaced integer subtraction with addition → KILLED
	3. negated conditional → KILLED
177	1. Replaced integer multiplication with division → KILLED
	2. Replaced integer addition with subtraction → KILLED
182	1. Replaced integer addition with subtraction → KILLED
	2. Replaced integer subtraction with addition → KILLED
	1. Replaced integer multiplication with division → KILLED
184	2. Replaced integer addition with subtraction → KILLED
	3. Replaced integer subtraction with addition → KILLED
	4. Replaced integer subtraction with addition → KILLED
189	1. removed call to com/thedeciders/pitest/TicketPriceCalculator::mergeBus → KILLED
192	1. replaced return value with null for com/thedeciders/pitest/TicketPriceCalculator::mergeSortBus → KILLED
198	1. Replaced integer subtraction with addition → KILLED
	2. Replaced integer addition with subtraction → KILLED
199	1. Replaced integer subtraction with addition → KILLED
205	1. changed conditional boundary → KILLED
	2. negated conditional → KILLED
206	1. Replaced integer addition with subtraction → KILLED
207	1. changed conditional boundary → KILLED

## Active mutators

- CONDITIONALS\_BOUNDARY
- EMPTY\_RETURNS
- FALSE\_RETURNS
- INCREMENTS
- INVERT\_NEGS
- MATH
- NEGATE\_CONDITIONALS
- NULL\_RETURNS
- PRIMITIVE\_RETURNS
- TRUE\_RETURNS
- VOID\_METHOD\_CALLS

## Tests examined

- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForInvalidNegativeAgeBus(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testSeniorCitizenBoundaryConditions(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (0 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceAirPlaneSingleAdult(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForFamilyMovie(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortEmptyArray(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortTaxiWithZero(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortTrainBasic(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (6 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForFreeTicketNarrowCaseTRAIN(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceAirPlaneInvalidAgeNegative(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForSeniorCitizenBus(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForSeniorCitizenHotel(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortParkSmallArray(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortPlaneBasic(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForChildMovie(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceTrainInvalidAgeNegative(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortWithZero(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortWithEdgeCase(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceTrainWithFamilyDiscount(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortParkEdgeCases(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForChildNarrowCaseBUS(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForFreeTicketNarrowCaseMovie(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForFreeTicketNarrowCaseBUS(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForOneAdultMovie(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceTrainSingleChild(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortWithNegativeNumbers(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForChildNarrowCaseShip(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceHotelSingleAdult(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceAirPlaneWithFamilyDiscount(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForOneAdultBUS(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForFamilyShip(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForFamilyTAXI(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortParkLargeArray(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.shouldNotCalculatePriceForFamilyEdgeCaseWithAdultsShip(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.shouldNotCalculatePriceForFamilyEdgeCaseWithAdultsTAXI(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortTaxiBasic(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForChildNarrowCaseTAXI(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testCalculatePriceTaxiSingleAdult(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testMergeSortWithRepeatedElements(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.testNoFamilyDiscount(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (1 ms)
- com.thedeciderstest.pitest.TicketPriceCalculatorTest.calculatePriceForChildTRAIN(com.thedeciderstest.pitest.TicketPriceCalculatorTest) (2 ms)



# Contributions

## Subhankhi Maiti

- **Code Base Development:** Built and developed the core functionality of the **TicketPriceCalculator** and **Passenger** classes, implementing business logic for various transportation modes and pricing calculations.
- **Framework Setup:** Configured TestNG, JUnit 5, and PIT for automated testing, enabling smooth integration with the code base.
- **Mutation Testing:** Integrated PIT for mutation analysis, addressing survived mutants and improving the test suite to achieve strong mutation coverage.
- **Code Refactoring:** Refactored the code base to improve readability, enhance input validation, and ensure better testability of methods.

## Srishti Yadav

- **Code Base Contributions:** Contributed to the overall code architecture, ensuring modular design and maintainability of the **TicketPriceCalculator** system.
- **Test Case Development:** Designed and implemented robust unit and integration test cases using JUnit 5 and TestNG, covering edge cases, boundary conditions, and typical scenarios.
- **Coverage Analysis:** Analyzed line and mutation coverage reports to identify gaps in testing, implementing additional tests to strengthen the overall test suite.
- **Documentation:** Documented the testing approach, mutation results, and code base improvements, providing insights on test effectiveness and future enhancements.

# References

1. [PIT Documentation](#)
2. [JUnit 5 Documentation](#)
3. [TestNG Documentation](#)