SDV502 Assessment One

White box approaches to testing Nelson State Cinema’s ticket pricing system.

Bob Win-Donnelly 12785549

Contents

[Introduction 2](#_Toc81982974)

[Equivalence Partitioning and Boundaries 2](#_Toc81982975)

[Unit Tests 4](#_Toc81982976)

[Summary 8](#_Toc81982977)

[References 8](#_Toc81982978)

# Introduction

This document is the test schedule for SDV502 Assessment 1 – Unit Testing. Conducting black box testing one Nelson State Cinemas ticket pricing. All functions referenced can be found in sdv502-assessment-one-bob-donnelly/cinema\_functions in sdv502-assessment-one-bob-donnelly repository.

The goal is to use black box equivalence boundary testing (*What Is Boundary Value Analysis and Equivalence Partitioning?*, n.d.) to inform the range of tests conducted in unit testing by using decision tables to determine which tests are necessary to test Nelson State Cinema’s ticket pricing. Then complete unit testing with a white box methodology on a schedule based on equivalence boundary partitioning to try and catch any errors created by the functions.

# Equivalence Partitioning and Boundaries

|  |  |  |
| --- | --- | --- |
| Adult before 5pm |  |  |
|  | Acceptable | Unacceptable |
| Person | Adult | Student  Senior  Child  Family Pass |
| Quantity | >=1 | <= 0 |
| Day | Mon  Wed  Thurs  Fri  Sat  Sun | Tues |
| Time | < 5pm | >= 5pm |

|  |  |  |
| --- | --- | --- |
| Adult After 5pm |  |  |
|  | Acceptable | Unacceptable |
| Person | Adult | Student  Senior  Child  Family Pass |
| Quantity | >=1 | <= 0 |
| Day | Mon  Wed  Thurs  Fri  Sat  Sun | Tues |
| Time | > 5pm | <= 5pm |

|  |  |  |
| --- | --- | --- |
| Adult Tuesday |  |  |
|  | Acceptable | Unacceptable |
| Person | Adult | Student  Senior  Child  Family Pass |
| Quantity | >=1 | <= 0 |
| Day | Tues | != Tues |
| Time | > 5pm | <= 5pm |

|  |  |  |
| --- | --- | --- |
| Children |  |  |
|  | Acceptable | Unacceptable |
| Person | Child | Adult  Student  Senior  Family Pass |
| Quantity | >=1 | <= 0 |
| Day | Any Day |  |
| Time | Any Time |  |

|  |  |  |
| --- | --- | --- |
| Seniors |  |  |
|  | Acceptable | Unacceptable |
| Person | Senior | Adult  Student  Child  Family Pass |
| Quantity | >=1 | <= 0 |
| Day | Any Day |  |
| Time | Any Time |  |

|  |  |  |
| --- | --- | --- |
| Students |  |  |
|  | Acceptable | Unacceptable |
| Person | Student | Adult  Child  Senior  Family Pass |
| Quantity | >=1 | <= 0 |
| Day | Any Day |  |
| Time | Any Time |  |

|  |  |  |
| --- | --- | --- |
| Family Pass |  |  |
|  | Acceptable | Unacceptable |
| Person | Family Pass | Adult  Child  Student  Senior |
| Quantity | >= 1 | < 4 |
| Day | Any Day |  |
| Time | Any Time |  |

|  |  |  |
| --- | --- | --- |
| Chick Flick Thursday |  |  |
|  | Acceptable | Unacceptable |
| Person | Adult | Adult  Child  Student  Senior |
| Quantity | >= 1 | <= 0 |
| Day | Thurs | != Thurs |
| Time | Any Time |  |

|  |  |  |
| --- | --- | --- |
| Kids and Careers |  |  |
|  | Acceptable | Unacceptable |
| Person | Adult  Child | Adult  Student  Senior |
| Quantity | >= 2 | < 2 |
| Day | Wednesday && != Public Holiday. | != Wednesday | | is Public Holiday |
| Time | Any time |  |

# Unit Tests

Unit testing is the act of testing the smallest part of a program possible, usually a function but can be classes or other objects. Each of the nine functions tested below are wrapped in a program which controls each function. (Atlassian, n.d.)

We tested this program by testing each individual function in the attached C# solution. A note on these tests is that passing does not mean that the functionality tested is a positive outcome. A passed test just means that the expected result matched the actual result of the unit test. This is to test the business logic as well as to make sure only intended results are being output from the developed logic.

|  |  |  |  |
| --- | --- | --- | --- |
| Adult Before 5pm | Input | Expected | Result |
| An adult purchasing a ticket before at 3pm on Monday. | 1 | 14.5 | Pass |
| Five adults purchasing one ticket each on a Wednesday at 4pm | 5 | 72.5 | Pass |
| Testing to see if an Adult can buy a ticket on a Tuesday at 2pm. | 1 | -1 | Pass |
| Testing to see if zero people can buy a ticket. | 0 | -1 | Pass |
| Testing to see if minus one person can buy a ticket.. | -1 | -1 | Pass |
| Testing to see if a Senior can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Child can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Student can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Adult After 5pm | Input | Expected | Result |
| An adult purchasing a ticket at 6pm on Monday. | 1 | 17.5 | Pass |
| Three adults purchasing one ticket each on a Wednesday at 6pm | 3 | 52.5 | Pass |
| Testing to see if an Adult can buy a ticket on a Tuesday at 6pm. | 1 | -1 | Pass |
| Testing to see if zero people can buy a ticket. | 0 | -1 | Pass |
| Testing to see if minus one person can buy a ticket. | -1 | -1 | Pass |
| Testing to see if a Senior can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Child can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Student can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Adult Tuesday | Input | Expected | Result |
| An adult purchasing a ticket on Tuesday. | 1 | 13 | Pass |
| Two adults purchasing one ticket each on a Tuesday. | 2 | 26 | Pass |
| Testing to see if an Adult can buy a ticket on a Wednesday. | 1 | -1 | Pass |
| Testing to see if zero people can buy a ticket. | 0 | -1 | Pass |
| Testing to see If minus one person can buy a ticket. | -1 | -1 | Pass |
| Testing to see if a Senior can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Child can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Student can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Child Under 16 | Input | Expected | Result |
| A child purchasing a ticket on any day. | 1 | 12 | Pass |
| Two children purchasing one ticket each on any day. | 2 | 36 | Pass |
| Testing to see if zero children can buy a ticket. | 0 | -1 | Pass |
| Testing to see If minus one child can buy a ticket. | -1 | -1 | Pass |
| Testing to see if an Adult can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Student can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Senior can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Seniors | Input | Expected | Result |
| A Senior purchasing a ticket on any day. | 1 | 12.50 | Pass |
| Two Seniors purchasing one ticket each on any day. | 2 | 25 | Pass |
| Testing to see if zero Seniors can buy a ticket. | 0 | -1 | Pass |
| Testing to see If minus one Senior can buy a ticket. | -1 | -1 | Pass |
| Testing to see if an Adult can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Student can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Child can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |

|  |  |  |  |
| --- | --- | --- | --- |
| Students | Input | Expected | Result |
| A Student purchasing a ticket on any day. | 1 | 14 | Pass |
| Two students purchasing one ticket each on any day. | 4 | 56 | Pass |
| Testing to see if zero students can buy a ticket. | 0 | -1 | Pass |
| Testing to see If minus one student can buy a ticket. | -1 | -1 | Pass |
| Testing to see if an Adult can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a child can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a senior can buy a ticket at child prices in valid hours. | 1 | -1 | Pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Family Pass | Quantity | Adult | Child | Expected | Result |
| A Family purchasing a ticket on any day. | 1 | 2 | 2 | 46 | Pass |
| A Family purchasing one ticket each on any day. | 1 | 1 | 3 | 46 | Pass |
| Test to see if you can process multiple families at the same time. | 4 | 2 | 2 | 184 | Pass |
| Test to replicate the above test with a different family makeup. | 2 | 1 | 3 | 92 | Pass |
| Test to see if two families of separate types can be processed at the same time. | 2 | 3 | 5 | 92 | Fail |
| Testing to see if the inverse of the above test is true. | 2 | 2 | 6 | -1 | Pass |
| Testing to see if zero Families can buy a pass. | 0 | 2 | 2 | -1 | Pass |
| Testing to see if minus one Family can buy a ticket. | -1 | 1 | 3 | -1 | Pass |
| Testing to see if an adult and a senior can buy a family pass. | 1 | “adult” | “senior” | -1 | Fail  Int32  Error |

Family Pass Note –

The first failed test was due to trying to see if families of different sizes can get tickets processed simultaneously as you can do with families of the same size or multiple tickets for the same family.

Due to me expecting -1 instead of a type error the second test failed. I left it in as it shows my methodology even if it was a mistake.

|  |  |  |  |
| --- | --- | --- | --- |
| Chick Flick Thursday | Input | Expected | Result |
| An adult purchasing a ticket on Thursday. | 1 | 21.50 | Pass |
| Two adults purchasing one ticket each on a Tuesday. | 2 | 43 | Pass |
| Testing to see if an Adult can buy a ticket on a Wednesday. | 1 | -1 | Pass |
| Testing to see if zero people can buy a ticket. | 0 | -1 | Pass |
| Testing to see If minus one person can buy a ticket. | -1 | -1 | Pass |
| Testing to see if a Senior can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Child can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |
| Testing to see if a Student can buy a ticket at adult prices in valid hours. | 1 | -1 | Pass |

Note – Kids and Careers

As the business requirement state that Kids and Carers get a discount on the first Wednesday of the month and not a public holiday this logic is flawed from the start. The reason is that there is no pr\_day < 6 catch error which would force Wednesday to be the first week of the month.

But these tests pass with the logic in its current state. I recommend that the logic be changed for this function.

|  |  |  |  |
| --- | --- | --- | --- |
| Kids and Careers | Input | Expected | Result |
| Kids and Carer duo purchasing a ticket on a Wednesday that is not a public holiday. | 1 | 12 | Pass |
| Two Kid and Carer duos purchasing a ticket on a Wednesday that is not a public holiday. | 2 | 24 | Pass |
| Kids and Carer duo purchasing a ticket on a Tuesday that is not a public holiday. | 1 | -1 | Pass |
| Kids and Carer duo purchasing a ticket on a Wednesday that is a public holiday. | 1 | -1 | Pass |
| Testing to see if 0 people can purchase a ticket on a Wednesday that is not a public holiday. | 0 | -1 | Pass |
| Testing to see if people can purchase a ticket on a Wednesday that is not a public holiday. | -1 | -1 | Pass |

# Summary

I tested nine functions with six to nine unit tests each for a total of sixty eight tests ran. Sixty six tests passed, and two tests failed which shows how equivalence boundary testing helps define exceptions before unit testing begins.

One failed test was possible, but not a probable exception in the business logic that was tested for thoroughness. The other test failed because the test schedule expected a result that was not possible due to the arguments being passed being a type error.

# References

Atlassian. (n.d.). *The different types of testing in software*. Atlassian. Retrieved 8 September 2021, from https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing

*What is Boundary value analysis and Equivalence partitioning?* (n.d.). Retrieved 8 September 2021, from https://www.softwaretestinghelp.com/what-is-boundary-value-analysis-and-equivalence-partitioning/