Black Box test cases:

1. Story #1: As an employee, I want to be able to place an order for a specific book so it can be sold to a particular student.

Valid/Invalid Input/output

* + **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits)
  + **Valid Output:** Alphanumeric (Eg: “order placed, Order# 56690”, “unable to place order”, “employee/student num not found” etc)
  + **Invalid Input**: Characters a-z,A-Z, Special characters, Student Number (Numeric - 9 < digits < 9), ISBN-10 (Numeric- 10 < digits < 10), Employee Number (Numeric- 5 < digits < 5)
  + **Invalid Output:** program doesn’t proceed, Program crashing

Equivalent Classes

* + EC1 - Student Number [100000001, 999999999]
  + EC2 - ISBN-10 [1000000001, 9999999999]
  + EC3 - Employee Number [10001, 99999]

Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| Student # | 100000000 | 167934082 | 1000000000 |
| ISBN-10 | 1000000000 | 4672895719 | 10000000000 |
| Employee # | 10000 | 15561 | 100000 |

Steps for testing

* + Precondition: System (order placement software) is open/logged on to the main page
  + Input: Book ISBN-10, Student Number, Employee Number
  + Expected Output: Order Number
  + Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |  |  |
| --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 |
| Input 1: 167934082 | Input 1: 5 | Input 1: 5 | Input 1: 167937080 |
| Input 2: 4672895719 | Input 2: 2 | Input 2: 467985719 | Input 2: 4672235783 |
| Input 3: 15561 | Input 3: 5A7 | Input 3: 15781 | Input 3: 12181 |
| Output: “Order placed, Order# 56690” | Output: INVALID | Output: INVALID | Output: “Student Number Not Found” |

1. Story #2: As an employee, I want to be able to reserve an in-stock book for a student so they can come and purchase it later.

Valid/Invalid Input/output

* 1. **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits), E-mail (Alphanumeric – username@uwindsor.ca)
  2. **Valid Output:** Alphanumeric (Eg: “book reserved, reservation# 56690”, “unable to reserve item”, “employee/student num not found” etc), email reservation sent to username@uwindsor.ca.
  3. **Invalid Input**: Special characters other than @, Student Number (Numeric - 9 < digits < 9), ISBN-10 (Numeric- 10 < digits < 10), Employee Number (Numeric- 5 < digits < 5), emails outside of Uwindsor (@uwindsor.ca)
  4. **Invalid Output:** program doesn’t proceed, Program crashing

1. Equivalent Classes
   1. EC1 - Student Number [100000001, 999999999]
   2. EC2 - ISBN-10 [1000000001, 9999999999]
   3. EC3 - Employee Number [10001, 99999]
2. Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| Student # | 100000000 | 167934082 | 1000000000 |
| ISBN-10 | 1000000000 | 4672895719 | 10000000000 |
| Employee # | 10000 | 15561 | 100000 |

1. Steps for testing
   1. Precondition: System (order placement software) is open/logged on to the main page
   2. Input: Book ISBN-10, Student Number, Employee Number, e-mail address
   3. Expected Output: Reservation number, e-mail sent!
   4. Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 | TC5 |
| Input 1: 167934082 | Input 1: 5 | Input 1: 5 | Input 1: 167937082 | Input 1: 167937154 |
| Input 2: 4672895719 | Input 2: 2 | Input 2: 467985719 | Input 2: 4672235719 | Input 2: 4672235345 |
| Input 3: 15561 | Input 3: 5A7 | Input 3: 15781 | Input 3: 12181 | Input 3: 16475 |
| Input 4: abc12@uwindsor.ca | Input 4: abc12@gmail.com | Input 4: abc12@uwindsor.ca | Input 4: abc12@uwindsor.ca | Input 4: abcd@gmail.com |
| Output: “reservation made, reservation# 56690” | Output: INVALID | Output: INVALID | Output: “Student Num Not Found” | Output: INVALID |

1. Story #3: As an employee, I want to be able to reserve an out-of-stock book for a student so they can come and purchase it later.

Valid/Invalid Input/output

* 1. **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits), E-mail (Alphanumeric – username@uwindsor.ca)
  2. **Valid Output:** Alphanumeric (Eg: “book reserved, reservation# 56690, expected pickup date: 21-10-2022”, “unable to reserve item”, “employee/student num not found” etc), email reservation sent to username@uwindsor.ca.
  3. **Invalid Input**: Special characters other than @, Student Number (Numeric - 9 < digits < 9), ISBN-10 (Numeric- 10 < digits < 10), Employee Number (Numeric- 5 < digits < 5), emails outside of Uwindsor (@uwindsor.ca)
  4. **Invalid Output:** program doesn’t proceed, Program crashing

1. Equivalent Classes
   1. EC1 - Student Number [100000001, 999999999]
   2. EC2 - ISBN-10 [1000000001, 9999999999]
   3. EC3 - Employee Number [10001, 99999]
2. Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| Student # | 100000000 | 167934082 | 1000000000 |
| ISBN-10 | 1000000000 | 4672895719 | 10000000000 |
| Employee # | 10000 | 15561 | 100000 |

1. Steps for testing
   1. Precondition: System (order placement software) is open/logged on to the main page
   2. Input: Book ISBN-10, Student Number, Employee Number, e-mail address
   3. Expected Output: Reservation number, e-mail sent!
   4. Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 | TC5 |
| Input 1: 167934082 | Input 1: 5 | Input 1: 5 | Input 1: 167937082 | Input 1: 167937154 |
| Input 2: 4672895719 | Input 2: 2 | Input 2: 467985719 | Input 2: 4672235719 | Input 2: 4672235345 |
| Input 3: 15561 | Input 3: 5A7 | Input 3: 15781 | Input 3: 12181 | Input 3: 16475 |
| Input 4: abc12@uwindsor.ca | Input 4: abc12@gmail.com | Input 4: abc12@uwindsor.ca | Input 4: abc12@uwindsor.ca | Input 4: abcd@gmail.com |
| Output: “Reservation made, Reservation# 56690” | Output: INVALID | Output: INVALID | Output: “Student Num Not Found” | Output: INVALID |

1. Story #4: As an employee in the bookstore, I want to sell a book to a student so they may purchase a book.

Valid/Invalid Input/output

* 1. **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits), Student card code (Numeric-14 digits)
  2. **Valid Output:** Alphanumeric (Eg: “order placed, receipt”, “unable to place order”, “employee/student num not found” etc)
  3. **Invalid Input**: Characters a-z,A-Z, Special characters, Student Number (Numeric - 9 < digits < 9), ISBN-10 (Numeric- 10 < digits < 10), Employee Number (Numeric- 5 < digits < 5), Student card code (Numeric - 14 < digits < 14)
  4. **Invalid Output:** program doesn’t proceed, Program crashing

1. Equivalent Classes
   1. EC1 - Student Number [100000001, 999999999]
   2. EC2 - ISBN-10 [1000000001, 9999999999]
   3. EC3 - Employee Number [10001, 99999]
   4. EC4 - Student card code [10000000000001, 99999999999999]
2. Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| Student # | 100000000 | 167934082 | 1000000000 |
| ISBN-10 | 1000000000 | 4672895719 | 10000000000 |
| Employee # | 10000 | 15561 | 100000 |
| Student card code | 10000000000000 | 10034004500670 | 100000000000000 |

1. Steps for testing
   1. Precondition: System (order placement software) is open/logged on to the main page
   2. Input: Book ISBN-10, Student Number, Employee Number, Student Number Code
   3. Expected Output: Receipt
   4. Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 | TC5 |
| Input 1: 167934082 | Input 1: 5 | Input 1: 5 | Input 1: 167937082 | Input 1: 167934082 |
| Input 2: 4672895719 | Input 2: 2 | Input 2: 467985719 | Input 2: 4672235719 | Input 2: 4672895719 |
| Input 3: 15561 | Input 3: 5A7 | Input 3: 15781 | Input 3: 12181 | Input 3: 15561 |
| Input 4: 12345678910111 | Input 4:  12345565677 | Input 4:  12345678922611 | Input 4: 12345678998761 | Input 4: 12345565677 |
| Output: “order placed, Order# 56690” | Output: INVALID | Output: INVALID | Output: “Student Num Not Found” | Output: “Card code Invalid” |

1. Story #9: As an employee, I want to be able to view the working hours of my fellow employees so that I can coordinate with them.
2. Valid/Invalid Input/output
   1. **Valid Input:** none
   2. **Valid Output:** table of employees working that day or “no one is working today”
   3. **Invalid Input**: none
   4. **Invalid Output:** program doesn’t proceed, Program crashing
3. Steps for testing
   1. Precondition: System (order placement software) is open/logged on to the main employee page
   2. Input: press “view employee working hours”
   3. Expected Output: table of employees working that day or “no one is working today”
   4. Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |
| --- | --- |
| TC1 | TC2 |
| Output: “table of hours” | Output: No one is working today |

1. Story #8: As an administrator, I want to be able to set the working hours of employees so that I can manage them.
2. Valid/Invalid Input/output
   1. **Valid Input:** Employee Number (Numeric-5 digits), Date, Starting hour, Ending hour
   2. **Valid Output:** “Hours set”
   3. **Invalid Input**: Characters a-z,A-Z, Special characters, Employee Number (Numeric- 5 < digits < 5), Date(yyyy-MM-dd), Hour(0:00-23:59)
   4. **Invalid Output:** program doesn’t proceed, Program crashing
3. Equivalent Classes
   1. EC1 - Employee Number [10001, 99999]
   2. EC2 - Date [2022-01-01, 2022-12-31]
   3. EC3 – Hour [8:00, 17:00]
4. Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| Employee # | 10000 | 15561 | 100000 |
| Date | 2021-11-15 | 2022-03-12 | 2023-01-10 |
| Hour | 01:00 | 12:30 | 23:00 |

1. Steps for testing
   1. Precondition: System (order placement software) is open/logged on to the main page
   2. Input: Employee Number, Date, Starting hour, Ending hour
   3. Expected Output: Message that hours were set
   4. Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 | TC5 |
| Input 1: 15561 | Input 1: 5A7 | Input 1: 15781 | Input 1: 12181 | Input 1: 12345 |
| Input 2: 2022-10-01 | Input 2: 2022-11-27 | Input 2: 2023-04-01 | Input 2: 2022-07-05 | Input 2: 2022-05-03 |
| Input 3: 11:00 | Input 3: 12:00 | Input 3: 08:00 | Input 3: 07:00 | Input 3: 13:00 |
| Input 4: 15:00 | Input 4: 14:00 | Input 4: 10:00 | Input 4: 12:00 | Input 4: 18:00 |
| Output: “Hours were set” | Output: INVALID | Output: INVALID | Output: INVALID | Output: INVALID |

1. Story #3: As an employee in the bookstore, I want to be able to search the inventory so I can know what items are currently in stock.

* Valid/Invalid Input/output
* **Valid Input:** ISBN-10 (Numeric-10 digits)
* **Valid Output:** Alphanumeric (Eg: “book in stock”, “ISBN code not found” etc)
* **Invalid Input**: Characters a-z,A-Z, Special characters, ISBN-10 (Numeric- 10 < digits < 10)
* **Invalid Output:** program doesn’t proceed, Program crashing
* Equivalent Classes
* EC1 - ISBN-10 [1000000001, 9999999999]
* Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| ISBN-10 | 1000000000 | 4672895719 | 10000000000 |

* Steps for testing
* Precondition: System (order placement software) is open/logged on to the main page
* Input: Book ISBN-10, Student Number, Employee Number, Student Number Code
* Expected Output: Receipt
* Postcondition: System will go back to its original state – main page.
* Test Cases:

|  |  |  |
| --- | --- | --- |
| TC1 | TC2 | TC3 |
| Input : 4672895719 | Input : 2 | Input: 1234567890 |
| Output: “Book Details” | Output: INVALID | Output: “ISBN code not found” |

1. Story #5: As an administrator I want to order more books, so I can update my inventory.

* Valid/Invalid Input/output
* **Valid Input:** ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits)
* **Valid Output:** Alphanumeric (Eg: “book ordered, order# 56690”, “Error: couldn’t place order”, “employee num not found” etc)
* **Invalid Input**:  Special characters other than @, ISBN-10 (Numeric- 10 < digits < 10), Employee Number (Numeric- 5 < digits < 5).
* **Invalid Output:** program doesn’t proceed, Program crashing
* Equivalent Classes
* EC1 - ISBN-10 [1000000001, 9999999999]
* EC2 - Employee Number [10001, 99999]
* Boundary Value Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| Input Type | InValid | Valid | InValid |
| ISBN-10 | 1000000000 | 4672895719 | 10000000000 |
| Employee # | 10000 | 15561 | 100000 |

* Steps for testing
* Precondition: System (order placement software) is open/logged on to the main page
* Input: Book ISBN-10, Student Number, Employee Number, e-mail address
* Expected Output: Reservation number, e-mail sent!
* Postcondition: System will go back to its original state – main page.
* Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 | TC5 |
| Input 1: 4672895719 | Input 1: 2 | Input 1: 467985719 | Input 1: 1672235345 | Input 1: 167223534567 |
| Input 2: 15561 | Input 2: 5A7 | Input 2: 15781 | Input 2: 155615 | Input 2: 15561 |
| Output: “Order made, Order# 56690” | Output: INVALID | Output: “Employee Number not found” | Output:  “INVALID” | Output: “INVALID” |