Black Box test cases:

* 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.

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

* 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
  + **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits), E-mail (Alphanumeric – username@uwindsor.ca)
  + **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.
  + **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)
  + **Invalid Output:** program doesn’t proceed, Program crashing

1. Equivalent Classes
   * EC1 - Student Number [100000001, 999999999]
   * EC2 - ISBN-10 [1000000001, 9999999999]
   * 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
   * 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: 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 |

* 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
  + **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits), E-mail (Alphanumeric – username@uwindsor.ca)
  + **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.
  + **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)
  + **Invalid Output:** program doesn’t proceed, Program crashing

1. Equivalent Classes
   * EC1 - Student Number [100000001, 999999999]
   * EC2 - ISBN-10 [1000000001, 9999999999]
   * 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
   * 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: 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 |

* 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
  + **Valid Input:** Student Number (Numeric- 9 digits), ISBN-10 (Numeric-10 digits), Employee Number (Numeric-5 digits), Student card code (Numeric-14 digits)
  + **Valid Output:** Alphanumeric (Eg: “order placed, receipt”, “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), Student card code (Numeric - 14 < digits < 14)
  + **Invalid Output:** program doesn’t proceed, Program crashing

1. Equivalent Classes
   * EC1 - Student Number [100000001, 999999999]
   * EC2 - ISBN-10 [1000000001, 9999999999]
   * EC3 - Employee Number [10001, 99999]
   * 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
   * 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 | 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” |

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.

1. Valid/Invalid Input/output
   * **Valid Input:** Employee Number (Numeric-5 digits), Date
   * **Valid Output:** 2 Timestamps of working hours or “Employee not working this day”
   * **Invalid Input**: Characters a-z,A-Z, Special characters, Employee Number (Numeric- 5 < digits < 5), Date(yyyy-MM-dd)
   * **Invalid Output:** program doesn’t proceed, Program crashing
2. Equivalent Classes
   * EC1 - Employee Number [10001, 99999]
   * EC2 - Date [2022-01-01, 2022-12-31]
3. Boundary Value Analysis

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

1. Steps for testing
   * Precondition: System (order placement software) is open/logged on to the main page
   * Input: Employee Number, Date
   * Expected Output: Hours a fellow employee is working, or message saying employee isn’t working that day
   * Postcondition: System will go back to its original state – main page.

Test Cases:

|  |  |  |  |
| --- | --- | --- | --- |
| TC1 | TC2 | TC3 | TC4 |
| Input 1: 15561 | Input 1: 5A7 | Input 1: 15781 | Input 1: 12181 |
| Input 2: 2022-10-01 | Input 2: 2022-11-27 | Input 2: 2023-04-01 | Input 2: 2022-07-05 |
| Output: “Employee isn’t working today” | Output: INVALID | Output: INVALID | Output: “10am – 2pm” |

Story #8: As an administrator, I want to be able to set the working hours of employees so that I can manage them.

1. Valid/Invalid Input/output
   * **Valid Input:** Employee Number (Numeric-5 digits), Date, Starting hour, Ending hour
   * **Valid Output:** “Hours set”
   * **Invalid Input**: Characters a-z,A-Z, Special characters, Employee Number (Numeric- 5 < digits < 5), Date(yyyy-MM-dd), Hour(0:00-23:59)
   * **Invalid Output:** program doesn’t proceed, Program crashing
2. Equivalent Classes
   * EC1 - Employee Number [10001, 99999]
   * EC2 - Date [2022-01-01, 2022-12-31]
   * EC3 – Hour [8:00, 17:00]
3. 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
   * Precondition: System (order placement software) is open/logged on to the main page
   * Input: Employee Number, Date, Starting hour, Ending hour
   * Expected Output: Message that hours were set
   * 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 |