## ***Step 2 of Group Project: Design Test and Execute Test Cases Using Input-Based Partitioning Technique***

1. Select a number of testable functions from the code of your application, or features from the application to be tested with the input domain modeling. Make sure to have a good reason for your selection.
2. List all of the input variables for the selected set of functions.
3. Define the characteristics of the input variables. Make sure you cover all input variables.
4. Partition the characteristics into blocks.
5. Define values for each block.
6. Select the coverage criteria.
7. Define a test set that satisfies the selected coverage criteria.
8. Execute the test cases on the application and report the results.

<https://observablehq.com/@spencermountain/spacetime-api>

Step 1 & 2? Function header:

add(num, unit)

Step 3: Define characteristics

num: This is the amount of time to add to provided time. Therefore, it is a number which can be characterized as negative, 0, or positive.

unit:

*This is a descriptor for unit of time to add to provided time. It is a string that can be of varying lengths. We characterized this variable by different lengths of time – Up to 1 day (exclusive), between 1 day and 1 month (exclusive), between 1 month and 1 year (exclusive), and a year or more.*

Step 4: Partition Characteristics into Blocks:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 |
| q1 = “Relation to 0” | Less than 0 | Equal to 0 | Greater than 0 |  |
| q2 = “Length of time” | 0 or more time and less than 1 day | 1 or more days and less than 1 month | 1 or more months and less than 1 year | 1 or more years |

Define values for each block:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 |
| q1 = “Relation to 0” | -2 | 0 | 3 |  |
| q2 = “Length of time” | hour | week | quarter | century |

Select the coverage criteria

BCC – Balances coverage with number of tests needed

Test

Base Choice: (3, week)

(3, hour)

(3, quarter)

(3, century)

(-2. Week)

(0, week)

--

1. Select the Testable Function:
   1. **s.timezone()** - return a bunch of meta-data about your current timezone
2. List Input Variables:
   1. Spacetime object
3. Define Characteristics of Input Variables
   1. Spacetime object: can be defined by its status as a date:
      1. A valid Spacetime object date with day light saving time (DST)
      2. A valid Spacetime object date without DST
      3. A valid Spacetime object date without DST and no timezone
      4. An invalid Spacetime date (date that does not exist)
4. Partition Characteristics into Blocks:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 |
| q1 = “Status of Spacetime date object” | Valid Spacetime object date without DST with a timezone | Valid date with DST with a timezone | Valid date with DST with no timezone | Date that does not exist |

1. Define Values for Each Block:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 |
| q1 = “State of Spacetime date object” | SpaceTime {  epoch: 1698710400000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: 1698724800000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: 1698710400000,  tz: null,  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } |

1. Select Coverage Criteria:
   1. ACoC – There is only one variable, so better to cover each possible choice of input variable
2. Define a Test Set:

b1.timezone()

b2.timezone()

b3.timezone()

b4.timezone()

1. Execute Test Cases:
2. Report the Results:

--

1. Select the Testable Function:
   1. **s.goto(tz)** - move to a new timezone, but at this same moment. Accepts an IANA code or abbreviation
2. List Input Variables:
   1. Spacetime Object
   2. tz: string
3. Define Characteristics of Input Variables:
   1. Spacetime object: can be defined by its status as a date:
      1. A valid Spacetime object date
      2. An Invalid Spacetime object date
   2. String: can be characterized by the format of timezone (tz) string
      1. null
      2. undefined
      3. IANA Code
      4. Abbreviation
      5. Invalid string of characters with length > 1
      6. Not of type string
      7. City name
      8. Time differential
4. Partition Characteristics into Blocks:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 | b5 | b6 | b7 | b8 |
| q1 = “Status of Spacetime date object” | Valid Spacetime object date | Invalid Spacetime object date |  |  |  |  |  |  |
| q2 = “Format of input string” | Null value | Undefined value | Tz IANA code | Tz Abbreviation | Invalid string of characters with length > 1 | Not of type string | City name | Time differential |

1. Define Values for Each Block:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 | b5 | b7 | b8 |
| q1 = “Status of Spacetime date object” | SpaceTime {  epoch: 1698710400000  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } |  |  |  |  |  |
| q2 = “Format of input string” | null | undefined | ‘America/Nassau’ | ‘gmt’ | ‘1234’ | ‘london’ | ‘-5h’ |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

The values of the blocks for q1 will be aliased by their block number (I.e. b1, b2, b3, etc...)

Base Case: b1.goto(‘America/Nassau’)

b1.goto(null)

b1.goto(undefined)

b1.goto(‘gmt’)

b1.goto(‘1234’)

b1.goto(‘london’)

b1.goto('-5h’)

b2.goto(‘America/Nassau’)

1. Execute Test Cases:
2. Report the Results:

--

1. Select the Testable Function:
   1. **s.startOf(unit)** - move to the first millisecond of the day, week, month, year, etc
2. List Input Variables:
   1. Spacetime object
3. Define Characteristics of Input Variables:
   1. Spacetime object: can be defined by its status as a date:
      1. A valid Spacetime object date
      2. An Invalid Spacetime object date
   2. Unit of time:
      1. Enum of the following values:
         1. second
         2. minute
         3. quarterhour
         4. hour
         5. day
         6. week
         7. month
         8. quarter
         9. season
         10. year
         11. decade
         12. century
4. Partition Characteristics into Blocks:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 | b5 | b6 | b7 | b8 | b9 | b10 | b11 | b12 |
| q1 = “Status of Spacetime date object” | Valid Spacetime object date | Invalid Spacetime object date |  |  |  |  |  |  |  |  |  |  |
| q2= “Enum of units of time” | second | minute | quaterhour | hour | day | week | month | quarter | season | year | decade | century |

1. Define Values for Each Block:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Partition | b1 | b2 | b3 | b4 | b5 | b6 | b7 | b8 | b9 | b10 | b11 | b12 |
| q1 = “State of Spacetime date object” | SpaceTime { epoch: 1698710400000  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } |  |  |  |  |  |  |  |  |  |  |
| q2= “Enum of units of time” | second | minute | quaterhour | hour | day | week | month | quarter | season | year | decade | century |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

Base Case: b1.startOf(‘day’)

b1.startOf(‘second’)

b1.startOf(‘minute’)

b1.startOf(‘quaterhour’)

b1.startOf(‘hour’)

b1.startOf(‘week’)

b1.startOf(‘month’)

b1.startOf(‘quarter’)

b1.startOf(‘season’)

b1.startOf(‘year’)

b1.startOf(‘decade’)

b1.startOf(‘century’)

b2.startOf(‘day’)

1. Execute Test Cases:
2. Report the Results:

--

1. Select the Testable Function:
   1. **s1.isSame(s2, unit) -** detect if two date/times are the same day, week, or year, etc
2. List Input Variables:
   1. s1: SpaceTime object used for comparison
   2. s2: SpaceTime object used for comparison (against s1)
   3. Unit: Check if a given unit of time is the same between the two SpaceTime objects
3. Define Characteristics of Input Variables:
   1. s1: can be defined by its status as a date:
      1. A valid Spacetime object date with day light saving time (DST)
      2. A valid Spacetime object date without DST
      3. A valid Spacetime object date without DST and no timezone
      4. An invalid Spacetime date (date that does not exist)
   2. s2: can be defined by its status as a date:
      1. A valid Spacetime object date with day light saving time (DST)
      2. A valid Spacetime object date without DST
      3. A valid Spacetime object date without DST and no timezone
      4. An invalid Spacetime date (date that does not exist)
   3. Unit: Any one of the valid enumeration values of units
      1. second
      2. minute
      3. quarterhour
      4. hour
      5. day
      6. week
      7. month
      8. quarter
      9. season
      10. year
      11. decade
      12. century
4. Partition Characteristics into Blocks:
   1. Functionality-based approach – the input of unit is an enumeration so this parameter will have a known value. The SpaceTime Objects are only used for comparison which means regardless of its value, the output will be either true or false. By checking for a true value and a false value, we are able to verify the completeness of the function.

|  |  |  |
| --- | --- | --- |
| **Partition** | **b1** | **b2** |
| q1: “Relation of time unit of s1 to s2” | true | false |

1. Define Values for Each Block:

|  |  |  |
| --- | --- | --- |
| **Partition** | **b1** | **b2** |
| Comparison result | (new SpaceTime(SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  }), new SpaceTime(SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  }), hour) | (SpaceTime {  epoch: 1698710400000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  }, SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  }, ‘hour`) |

1. Select Coverage Criteria:
   1. ACoC – We only need to test that both outcomes are achievable
2. Define a Test Set:

a. (new SpaceTime(SpaceTime {

epoch: 1698714000000,

tz: 'america/new\_york',

silent: true,

british: undefined,

\_weekStart: 1,

\_today: {}

})).isSame(new SpaceTime(SpaceTime {

epoch: 1698710400000,

tz: 'america/new\_york',

silent: true,

british: undefined,

\_weekStart: 1,

\_today: {}

}), ‘hour’)

b. (new SpaceTime(SpaceTime {

epoch: 1698714000000,

tz: 'america/new\_york',

silent: true,

british: undefined,

\_weekStart: 1,

\_today: {}

})).isSame(new SpaceTime(SpaceTime {

epoch: 1698714000000,

tz: 'america/new\_york',

silent: true,

british: undefined,

\_weekStart: 1,

\_today: {}

}), ‘hour’)

1. Execute Test Cases:
2. Report the Results:

**Getters/Setters**

1. Select the Testable Function:

**s.time(str) -** set or return a formatted, 12-hour time, like '11:30pm'

1. List Input Variables:
   1. s: SpaceTime object
   2. str: string to set time to (optional)
2. Define Characteristics of Input Variables:
   1. s: Validity of Spacetime Object value (real datetime)
   2. str: format of string input
3. Partition Characteristics into Blocks:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** |
| q1=”Validity of SpaceTime object value” | Valid SpaceTime value | Invalid SpaceTime value |  |  |  |  |
| q2=”format of string value” | null | 12hr am, just hour | 12hr pm, just hour | 12hr am, with hour, colon, and minutes | 12hr pm, with hour, colon, and minutes | 24hr |

1. Define Values for Each Block:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** |
| q1=”Validity of SpaceTime object value” | SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined. \_weekStart: 1,  \_today: {}  } |  |  |  |  |
| q2=”format of string value” | null | ‘1am’ | ‘3pm’ | ‘1:43am’ | ‘8:13pm’ | ‘16:00’ |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

Aliased the SpaceTime value to their block id for easier readability

Base Case: b1.time(‘1am’)

b1.time()

b1.time(‘3pm’)

b1.time(‘1:43am’)

b1.time(‘8:13pm’)

b1.time(‘16:00’)

b2.time(‘1am’)

1. Execute Test Cases:
2. Report the Results:
3. Select the Testable Function:

**.week(num) -** set or return the week-number of the year (1-52).

1. List Input Variables:
2. Define Characteristics of Input Variables:
   1. s: SpaceTime Object
   2. num: Week of year represented as an integer
3. Partition Characteristics into Blocks:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** |
| q1=”Validity of SpaceTime object value” | Valid SpaceTime value | Invalid SpaceTime value |  |  |  |  |
| q2=”Relation to weeks of year represented as integers” | null | Negative number less than -52 | Negative number between [-52, 0) | zero | Positive number between (0, 52] | Positive number greater than 52 |

1. Define Values for Each Block:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** |
| q1=”Validity of SpaceTime object value” | SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined. \_weekStart: 1,  \_today: {}  } |  |  |  |  |
| q2=”Relation to weeks of year represented as integers” | null | -53 | -3 | 0 | 10 | 104 |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

Base Case: b1.week()

b1.week(-53)

b1.week(-3)

b1.week(0)

b1.week(10)

b1.week(104)

b2.week()

1. Execute Test Cases:
2. Report the Results:
3. Select the Testable Function:

**s.quarter(num) -**  set or return the fiscal-quarter (1-4)

1. List Input Variables:
   1. s: SpaceTime object
   2. num: number representing the fiscal year quarter
2. Define Characteristics of Input Variables:
   1. Validity of SpaceTime object
   2. Relation of the value to the integer representation of a quarter
3. Partition Characteristics into Blocks:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** |
| q1=”Validity of SpaceTime object value” | Valid SpaceTime value | Invalid SpaceTime value |  |  |  |  |
| q2=”Relation of quarter represented as integers” | null | Negative number less than -4 | Negative number between [-4, 0) | zero | Positive number between (0, 4] | Positive number greater than 4 |

1. Define Values for Each Block:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** |
| q1=”Validity of SpaceTime object value” | SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined. \_weekStart: 1,  \_today: {}  } |  |  |  |  |
| q2=”Relation to weeks of year represented as integers” | null | -5 | -1 | 0 | 3 | 6 |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

Base Case: b1.quarter(null)

b1.quarter(-5)

b1.quarter(-1)

b1.quarter(0)

b1.quarter(3)

b1.quarter(6)

b2.quarter()

1. Execute Test Cases:
2. Report the Results:
3. Select the Testable Function:

**s.season(str) -** set or return the name of the season, spring/summer/fall/autumn/winter

1. List Input Variables:
   1. s: SpaceTime object
   2. str: gets the season of the date if null or sets the season based on a string on enumerated values
2. Define Characteristics of Input Variables:
   1. S: Validity of SpaceTime object
   2. Str: null and Enum of 'spring', 'summer', 'fall', 'autumn', 'winter'
3. Partition Characteristics into Blocks:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** |
| q1=”Validity of SpaceTime object value” | Valid SpaceTime value | Invalid SpaceTime value |  |  |  |
| q2=”enum value” | null | 'spring’ | 'summer’ | 'winter’ | ‘fall’ |

1. Define Values for Each Block:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** |
| q1=”Validity of SpaceTime object value” | SpaceTime {  epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined. \_weekStart: 1,  \_today: {}  } |  |  |  |
| q2=”enum value” | null | 'spring’ | 'summer’ | 'winter’ | ‘fall’ |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

Base Case: b1.season(null)

b1.season(‘spring’)

b1.season(‘summer’)

b1.season(‘winter’)

b1.season(‘fall’)

b2.season(null)

1. Execute Test Cases:
2. Report the Results:
3. Select the Testable Function:

**s. monthName** **(month) -** set or return the current month as a string, like 'april'

1. List Input Variables:
   1. s: SpaceTime Object
   2. month: null to get month and enum of the 12 months
2. Define Characteristics of Input Variables:
   1. s: Validity of the SpaceTime object
   2. Month: Enum of months
      1. January
      2. February
      3. March
      4. April
      5. May
      6. June
      7. July
      8. August
      9. September
      10. October
      11. November
      12. December
3. Partition Characteristics into Blocks:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** | **b7** | **b8** | **b9** | **b10** | **b11** | **b12** |
| q1=”Validity of SpaceTime object value” | Valid SpaceTime value | Invalid SpaceTime value |  |  |  |  |  |  |  |  |  |  |
| q2=”Relation to weeks of year represented as integers” | January | February | March | April | May | Jun | July | August | September | October | November | December |

1. Define Values for Each Block:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Partition** | **b1** | **b2** | **b3** | **b4** | **b5** | **b6** | **b7** | **b8** | **b9** | **b10** | **b11** | **b12** | **b13** |
| q1=”Validity of SpaceTime object value” | SpaceTime { epoch: 1698714000000,  tz: 'america/new\_york',  silent: true,  british: undefined,  \_weekStart: 1,  \_today: {}  } | SpaceTime {  epoch: null,  tz: 'america/new\_york',  silent: true,  british: undefined. \_weekStart: 1,  \_today: {}  } |  |  |  |  |  |  |  |  |  |  |  |
| q2=”Relation to weeks of year represented as integers” | January | February | March | April | May | Jun | July | August | September | October | November | December | null |

1. Select Coverage Criteria:
   1. BCC – Balances coverage with number of tests needed
2. Define a Test Set:

Base Case: b1.monthName(null)

b1.monthName(‘january’)

b1.monthName(‘february’)

b1.monthName(‘march’)

b1.monthName(‘april’)

b1.monthName(‘may’)

b1.monthName(‘june’)

b1.monthName(‘july’)

b1.monthName(‘august’)

b1.monthName(‘september’)

b1.monthName(‘october’)

b1.monthName(‘’november)

b1.monthName(‘december’)

b2.monthName(null)

1. Execute Test Cases:
2. Report the Results: