Black-box test design

1. Utility > utility.function.js > getDateInSqlFormat

1.Equivalence Partitioning

Input

For date type provided – return value:

I partition(valid): Date object – returns string representation of Date;
II part.(invalid): number – returns null;
III part.(invalid):string – returns null;
IV partition (invalid): object that is not Date() – returns null;
V partition (invalid) – null – returns null;
VI partition (invalid) – undefiled – returns null;
VII partition (invalid) – nothing passed – returns null;

Output:

I partition (valid) – string representation of Date object with format
"YYYY-MM-dd hh:mm:ss";
II partition (valid) – null;
III partition (invalid) – string that is not a representation of Date;
IV partition (invalid) – object;
V partition (invalid) number;

2.Boundary Value Analysis

I partition

1. valid date value: any valid Date obj – new Date("2021-12-34T22:51:52.000Z")
2. invalid date value: year part wrong – new Date("202100-12-24T22:51:52.000Z");
3. invalid date value: month part – new Date("2021-14-24T22:51:52.000Z");
4. invalid date value: day part – new Date("2021-12-50T22:51:52.000Z");
5. invalid date value: hour part – new Date("2021-12-24T30:25:52.000Z");
6. invalid date value: minute part – new Date("2021-12-24T10:90:52.000Z");
7. invalid date value: seconds part – new Date("2021-12-24T10:25:90.000Z");

II partition

1. any number value is invalid

III partition

1. any string value is invalid

IV partition

1. any object that is not Date(), is invalid

V partition

1. null is invalid

VI partition

1. undefined is invalid

VII partition

1. no value is invalid

Utility > utility.generators.js> characterGenerator

```
 * Generates a string with the given length with characters or numbers
from `[a-z],[A-Z],[0-9]
 * @param {Number}   length of the string that we want returned.
 * @returns {String} A semi-randomly generated string
```

1.Equivalence Partitioning

Input Partitioning:

1. partition valid - number (1 - 11) | returns string with same length as number
2. partition invalid – null/undefined | returns null
3. partition invalid - string | returns null
4. partition invalid - 0 | returns null
5. partition invalid - non-integer number | returns null
6. partition invalid - negative number | returns null

Output:

1. partition valid - characters from [a-z],[A-Z],[0-9],(length to input is 4,most used length)
2. partition invalid - empty string (e.g. input is -2)
3. partition valid - output is string

2.Boundary Value Analysis

1. invalid value: 0 input | returns null
2. valid value: 1 input | returns 1 character
3. valid value: 2 input | returns 2 characters
4. invalid value: 12 input | returns null
5. valid value: 11 input | returns 11 characters
6. valid value: 10 input | returns 10 characters

Utility > utility.generators.js> numberGenerator

```
/**
 * Generates a characters between 0 and 9,the string length is equal to
length input.
 * @param {Number}   length of the string that we want returned. Based
upon length.
 * @returns {String}  Returns a semi-randomly generated number-only
string using numbers between 0 and 9
 */
```

1.Equivalence Partitioning

Input

1. partition invalid length - 0
2. partition valid length - positive number
3. partition invalid length - negative number
4. partition invalid length - a string
5. partition invalid length - null
6. partition invalid length - undefined
7. partition invalid length - non-integer number
8. partition invalid length - object

Output

1. partition valid - string contains only numbers
2. partition valid - output is string type
3. partition valid - output length same as input length
4. partition valid - null
5. partition invalid - characters not a number
6. partition invalid - type number
7. partition invalid - empty string

2.Boundary Value Analysis

1. invalid value: 0 length | returns null
2. valid value: 1 length | returns 1 string character
3. valid value: 2 length | returns 2 string characters
4. valid value: 4 length | returns 4 string characters
5. valid value: 5 length | returns 5 string characters
6. invalid value: 6 length | returns null

Utility > utility.generators.js> transactionDateGenerator

```
/**
 *
 * Generates the end portion of the transaction id string based on
today's date'
 * @returns {String}  Returns a date string with the format ddmmyy
 */
```

1.Equivalence Partitioning

Input:

1. partition valid no value | returns today’s date in format “ddmmyy”

Output:

1. partition valid - string contains only numbers | returns today's date in format “ddmmyy”
2. partition valid - output is of string type | returns today's date in format “ddmmyy”
3. partition valid - output length is 6 | returns today's date in format “ddmmyy”
4. partition invalid - output length is 0 | returns today's date in format “ddmmyy”
5. partition invalid - output is null | returns today's date in format “ddmmyy”
6. partition invalid - output is undefined | returns today's date in format “ddmmyy”

Utility > utility.calculations.js> calculateAmount

```
/**
 * The function calculates the amount that has to be payed for the
delivery of the package based on a couple of factors.
 * The base amount to pay is 35DKK for a delivery.
 * @param {Number} volume Volume of the package,retrieved from a
`package` object
 * @param {Number} weight Weight of the package,retrieved from a
`package` object
 * @param {Boolean} international Will the package be
internationalized,retrieved from a `delivery` object
 * @param {Boolean} electronics Does the package contain
electronics,retrieved from a `package` object
 * @param {Boolean} oddsized Is the form of the package
oddsized,retrieved from a `package` object
 * @param {Boolean} fragile Are there any fragile things inside the
package,retrieved from a `package` object
 * @returns {Number} The price after calculations for how much to pay.
 */
```

1.Equivalence Partitioning

Input:

1. partition valid 1 - 800 volume - e.g. 400 value | returns 35
2. partition valid 800.1-12500 volume - e.g. 6650 value | returns 52,5
3. partition valid 12500.1 and above volume - e.g. 100000 value | returns 70
4. partition invalid - volume is 0 | returns null
5. partition invalid - volume is -1 | returns null
6. partition invalid - volume is null | returns null
7. partition invalid - volume is string | returns null
8. partition valid - 1-3 weight - 2 partition value | returns 35
9. partition valid - 3.1-5 weight - 4 partition value | returns 52,5
10. partition valid - 5.1 and above - 7.5 partition value | returns 70
11. partition invalid - weight is 0 | returns null
12. partition invalid - weight is -1 | returns null
13. partition invalid - weight is null | returns null
14. partition invalid - weight is string | returns null
15. partition invalid - international is string | returns null
16. partition valid - international is number(1) | returns 70
17. partition valid - international is number(0) | returns 35
18. partition invalid - international is number(not 0 or 1) | returns null
19. partition invalid - international is negative number | returns null
20. partition invalid international is non-integer number | returns null
21. partition invalid international is null/undefined | returns null
22. partition invalid electronics is string | returns null
23. partition valid - electronics is number(1) | returns 70
24. partition valid - electronics is number(0) | returns 35
25. partition invalid electronics is number(not 0 or 1)| returns null
26. partition invalid - electronics is negative number | returns null
27. partition invalid - electronics is non-integer number | returns null
28. partition invalid- electronics is null/undefined | returns null
29. partition invalid oddsized is string | returns null
30. partition valid - oddsized is number(1) | returns 70
31. partition valid - oddsized is number(0) | returns 35
32. partition invalid oddsized is number(not 0 or 1) | returns null
33. partition invalid - oddsized is negative number | returns null
34. partition invalid - oddsized is non-integer number | returns null
35. partition invalid oddsized is null/undefined | returns null
36. partition invalid fragile is string | returns null
37. partition valid - fragile is number(1) | returns 70
38. partition valid - fragile is number(0) | returns 35

39. partition invalid fragile is number(not 0 or 1) | returns null
40. partition invalid - fragile is negative number | returns null
41. partition invalid - fragile is non-integer number | returns null
42. partition invalid fragile is null/undefined | returns null

Output:

1. valid is 35 (all values are 0)
2. valid is 175 (all values are 1)
3. valid if value of type number
4. valid is null
5. invalid is a string

2. Boundary Value Analysis

1. valid lower boundary for volume – 1
2. valid lower boundary for volume – 2
3. invalid lower boundary for volume - 0
4. valid upper boundary for volume- 9 999 999
5. valid upper boundary for volume- 10 000 000
6. invalid upper boundary for volume - 10 000 001
7. valid lower boundary for weight– 1
8. valid lower boundary for weight– 2
9. invalid lower boundary for weight- 0
10. valid upper boundary for weight- 49
11. valid upper boundary for weight- 50
12. invalid upper boundary for weight- 51
13. valid international - 0
14. valid international -1
15. invalid international - -1
16. invalid international - 2
17. valid electronics - 0
18. valid electronics -1
19. invalid electronics - -1
20. invalid electronics - 2
21. valid oddsized - 0
22. valid oddsized -1
23. invalid oddsized - -1
24. invalid oddsized - 2
25. valid fragile - 0
26. valid fragile -1
27. invalid fragile - -1
28. invalid fragile - 2

3.Decision Table Testing

Weight = 1–3,3-5,5>= Volume =1-800,800-12500,12500>=

Rules	R 1	R 2	R 3	R 4	R 5	R 6	R 7	Actions
Conditions								BasePrice = 35dkk
Simple delivery	Y	N	N	N	N	N	N	BasePrice
Simple delivery and min volume	Y	N	N	N	N	N	N	BasePrice
Simple delivery and med volume	Y	Y	N	N	N	N	n	BasePrice +(BasePrice * 0.5)
Simple delivery and max volume	Y	Y	N	N	N	N	N	BasePrice +(BasePrice * 1)
Simple delivery and min weight	Y	N	N	N	N	N	N	BasePrice
Simple delivery and med weight	Y	N	Y	N	N	N	N	BasePrice +(BasePrice * 0.5)
Simple delivery and max weight	Y	N	Y	N	N	N	N	BasePrice +(BasePrice * 1)
Simple delivery and International	Y	N	N	Y	N	N	N	BasePrice +(BasePrice * 1)
Simple delivery and Oddsized	Y	N	N	N	Y	N	N	BasePrice +(BasePrice * 1)
Simple delivery and electronics	Y	N	N	N	N	Y	N	BasePrice +(BasePrice * 1)
Simple delivery and fragile	Y	N	N	N	N	N	Y	BasePrice +(BasePrice * 1)
Simple delivery, min volume, min weight,international,electronics,oddsized,fragile	Y	N	N	Y	Y	Y	Y	BasePrice +(BasePrice * 1) +(BasePrice * 1)+(BasePrice * 1)+(BasePrice * 1)

Rules
Simple delivery: min weight, min volume, no international, no fragile, no oddsized,no electronics
R1: Base delivery price apply
R2: Extra Volume means extra price
R3: Extra Weight means extra price
R4: International means extra price
R5: Electronicsmeans extra price
R6: Oddsized means extra price
R7: Fragile means extra price

The Decision table diagram provides us with information and a review of how our function behaves under different inputs.
In this particular case, we describe the creation of the price for one delivery based on some fixed constants for specific situations.
One simple delivery's price is considered our minimum price for a delivery to exist. Once other options come in , such as oddSized or Fragile, the price changes by 1 base unit(simple delivery) for each additional property. The weight and volume are divided into 3 categories each: min, med, max, and depend on an integer rather than a true or false.

Utility> utility.calculations.js > calculateVolume

```
 */
const calculateVolume = (height, width, depth) => { // values is in cm
(10,10,1) = 100cm3 =
    if (checkIfPosNumber(height) && checkIfPosNumber(width) &&
checkIfPosNumber(depth)){
        const result = height * depth * width;
        return (result < 1 ? 1 : result)  // 0.01 m3
    }else{
        return null
    }
}
/**
```

1.Equivalence Partitioning

Input:

1. partition valid height 1
2. partition invalid - height null/undefined | returns null
3. partition invalid - height 0 | returns null
4. partition invalid - height string | returns null
5. partition invalid - height -1 | returns null
6. partition valid - width 1
7. partition invalid - width null/undefined | returns null
8. partition invalid - width 0 | returns null
9. partition invalid - width string | returns null
10. partition invalid - width -1 | returns null
11. partition valid - depth 1
12. partition invalid - depth null/undefined | returns null
13. partition invalid - depth string | returns null
14. partition invalid - depth -1| returns null

Output:

1. partition invalid null
2. partition invalid 10 000 000
3. partition invalid string
4. partition invalid -1
5. partition invalid 0
6. partition valid 1

2.Boundary Value Analysis

1. invalid height 0
2. valid height 1
3. valid height 1.1
4. invalid width 0
5. valid width 1
6. valid width 1.1
7. invalid depth 0
8. valid depth 1
9. depth depth 1.1

Model >delivery.model.js > generateUUID

```
 /**
     * Generates a unique string identifier with the format
`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`
     * The `x` can be a character or number from `[a-z],[A-Z],[0-9]`
     * @returns {String}  Returns a 36 character long semi-unique
identifier
     */
```

1.Equivalence Partitioning

Input:

1. partition valid - no value | returns string in format `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, where x is a character from [a-z],[A-Z],[0-9]

Output

1. valid length of string is 36
2. valid string format --`xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`?
3. invalid null/undefined
4. invalid 0 length string
5. invalid number
6. invalid string format different from the intended ?

Model > payment.model.js > generateTransactionId

```
 /**
     * Generates a unique string id with the  format `xxxxxxxxNNNNN -
ddmmyy`
     *  The `x` represents a character or number from
`[a-z],[A-Z],[0-9]` that has been generated randomly.
     * The `N` represents a number from `[0-9]` that has been generated
randomly.
```

```
     * The `ddmmyy` is based on the current date. `yy` is the last two
numbers of the year.
     * @returns {String}  Returns a 20 character long semi-unique
identifier.
     */
```

1.Equivalence Partitioning

Input:

1. partition valid no value | returns string in format `xxxxxxxxNNNNN - ddmmyy`, x is characters from [a-z],[A-Z],[0-9], N is character from [0-9],d

Output:

1. valid length 20
2. valid string format `xxxxxxxxNNNNN - ddmmyy`
3. invalid null/undefined
4. invalid 0 length string
5. invalid number
6. invalid string format different from the intended