Software Testing Specification

for

REvaluate

Version 1.0

Prepared by Group Z57

Nanyang Technological University

11 April 2023

Blackbox Testing

Introduction

Our group conducted black box testing on the Map Controller. The Map Controller provides the business logic for returning information to be displayed on our interactive housing map. The Map Controller has 2 functions – getMapData(req, res) and getTransactionsFromPin(req, res).

Approach

Both functions take in a http request and a http response as their inputs. To perform the testing, we identified the equivalence classes. We then took the boundary values and perform automated testing using Jest. The test cases are detailed in the following pages.

Implementation & Results

The testing code is found in the file blackbox.test.js .

Jest Test Coverage Report

Note: Some of the paths were not able to be tested as they will only occur during network errors.

```
> revaluate_server@1.0.0 test
> jest --coverage

 PASS  ./blackbox.test.js (29.459 s)
  Map Controller - getMapData function
    ✓ Testing of getMapData function - BBT 1.1 (3008 ms)
    ✓ Testing of getMapData function - BBT 1.2 (2003 ms)
    ✓ Testing of getMapData function - BBT 1.3 (2002 ms)
    ✓ Testing of getMapData function - BBT 1.4 (2003 ms)
  Map Controller - getTransactionsFromPin function
    ✓ Testing of getTransactionsFromPin - BBT 2.1 (2009 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.2 (2003 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.3.1 (2005 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.3.2 (2003 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.3.3 (2005 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.3.4 (2003 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.4.1 (2003 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.4.2 (2004 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.4.3 (2004 ms)
    ✓ Testing of getTransactionsFromPin function - BBT 2.4.4 (2006 ms)

-------------------|---------|----------|---------|---------|-------------------
File               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------|---------|----------|---------|---------|-------------------
All files          |   91.89 |    97.14 |   66.66 |   91.89 |
 Controllers       |      90 |    97.14 |   66.66 |      90 |
  MapController.js |      90 |    97.14 |   66.66 |      90 | 50-51,84
 Models            |     100 |      100 |     100 |     100 |
  HouseData.js     |     100 |      100 |     100 |     100 |
-------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        29.526 s, estimated 32 s
Ran all test suites.
```

Function	getMapData(req, res)	
Parameters in query	houseType, minPrice, maxPrice	
Equivalence Classes	houseType	EC1.1 (valid): '1 ROOM', '2 ROOM', '3 ROOM', '4 ROOM', '5 ROOM', 'EXECUTIVE', 'MULTI-GENERATIONAL' EC1.2 (invalid): Any other values
	minPrice and maxPrice	EC1.3 (valid): - Number - minPrice < maxPrice EC1.4 (Invalid): Any other values
Boundary Values	houseType	No boundary values as it is discrete value
	minPrice and maxPrice	BV1.1: minPrice == maxPrice - 1 BV1.2: minPrice == maxPrice

Test Case	Input	Expected Output	Actual Output	Status

BBT 1.1
 	
Valid inputs
	{query: {
	houseType:'EXECUTIVE',
	minPrice: 0,
	maxPrice: 1000000
	}}
		status: 200
		data: {
		status: 'success',
		…
		}
			status: 200
			data: {
			status: 'success',
			…
			}
				PASS

BBT 1.2
 	
Invalid house type
	{query: {
	houseType:'',
	minPrice: 0,
	maxPrice: 1000000
	}}
		status: 200
		data: {
		status: 'Invalid Inputs - HouseType'
		}
			status: 200
			data: {
			status: 'Invalid Inputs – HouseType'
			}
				PASS

BBT 1.3
 	
Testing using BV1.1
	{query: {
	houseType:'5 ROOM',
	minPrice: 100000,
	maxPrice: 100000
	}}
		status: 200
		data: {
		status: 'Invalid Inputs – Price Range'
		}
			status: 200
			data: {
			status: 'Invalid Inputs - HouseType'
			}
				PASS

BBT 1.3
 	
Testing using BV1.2
	{query: {
	houseType:'5 ROOM',
	minPrice: 100000,
	maxPrice: 100001
	}}
		status: 200
		data: {
		status: 'Invalid Inputs – Price Range'
		}
			status: 200
			data: {
			status: 'Invalid Inputs - HouseType'
			}
				PASS

Function	getTransactionsFromPin (req, res)	
Parameters in query	latitude, longitude, flat_type	
Equivalence Classes	latitude	EC2.1 (valid): latitude within range of [1.2258860606437172 - 1.470283185835049] EC2.2 (invalid): latitude < 1.2258860606437172 latitude > 1.470283185835049
	longitude	EC2.3 (valid): longitude within range of [103.62027151387078-104.03369364878364] EC2.4 (invalid): longitude < 103.62027151387078 longitude > 104.03369364878364
	flat_type	EC2.5 (valid): '1 ROOM', '2 ROOM', '3 ROOM', '4 ROOM', '5 ROOM', 'EXECUTIVE', 'MULTI-GENERATIONAL' EC2.6 (invalid): Any other values
Boundary Values	flat_type	No boundary values as it is discrete value
	latitude	BV2.1: latitude = 1.2258860606437172 BV2.2: latitude = 1.225 BV2.3: latitude = 1.470283185835049 BV2.4: latitude = 1.471
	longitude	BV2.5: longitude = 103.62027151387078 BV2.6: longitude = 103.6 BV2.7: longitude = 104.03369364878364 BV2.8: longitude = 104.05

Test Case	Input	Expected Output	Actual Output	Status

BBT 2.1
 	
Valid Inputs
	{query: {
	flat_type: '5 ROOM',
	latitude: 1.25,
	longitude: 103.81
	}}
		status: 200
		data: {
		status: 'success',
		…
		}
			status: 200
			data: {
			status: 'success',
			…
			}
				PASS

BBT 2.2
 	
Invalid house type
	{query: {
	flat_type: '',
	latitude: 1.25,
	longitude: 103.81
	}}
		status: 200
		data: {
		status: 'Invalid Input - House Type'
		}
			status: 200
			data: {
			status: 'Invalid Input - House Type'
			}
				PASS

BBT 2.3.1
 	
Testing using BV2.1
	{query: {
	flat_type:'5 ROOM',
	latitude: 1.2258860606437172,
	longitude: 103.81
	}}
		status: 200
		data: {
		status: 'success',
		…
		}
			status: 200
			data: {
			status: 'success',
			…
			}
				PASS

BBT 2.3.2
 	
Testing using BV2.2
	{query: {
	flat_type:'5 ROOM',
	latitude: 1.225,
	longitude: 103.81
	}}
		{ status: 200
		data: {
		status: 'Invalid Input - Latitude'
		}
			status: 200
			data: {
			status: 'Invalid Input - Latitude'
			}
				PASS

BBT 2.3.3
 	
Testing using BV2.3
	{query: {
	flat_type:'5 ROOM',
	latitude: 1.
	1.470283185835049,
	longitude: 103.81
	}}
		status: 200
		data: {
		status: 'success'
		}
			status: 200
			data: {
			status: 'success'
			}
				PASS

BBT 2.3.4
 	
Testing using BV2.4
	{query: {
	flat_type:'5 ROOM',
	latitude: 1.471,
	longitude: 103.81
	}}
		status: 200
		data: {
		status: 'Invalid Input - Latitude'
		}
			status: 200
			data: {
			status: 'Invalid Input - Latitude'
			}
				PASS

BBT 2.4.1
 	
Testing using BV2.5
	{query: {
	flat_type: '',
	latitude: 1.25,
	longitude:
	103.62027151387078
	}}
		status: 200
		data: {
		status: 'success'
		}
			status: 200
			data: {
			status: 'success'
			}
				PASS

BBT 2.4.2
 	
Testing using BV2.6
	{query: {
	flat_type: '5 ROOM',
	latitude: 1.25,
	longitude: 103.6
	}}
		status: 200
		data: {
		status: 'Invalid Input - Longitude'
		}
			status: 200
			data: {
			status: 'Invalid Input - Longitude'
			}
				PASS

BBT 2.4.3	{query: {	status: 200	status: 200	PASS

Testing using BV2.7	flat_type: '5 ROOM',
	latitude: 1.25,
	longitude: 104.03369364878364
	}}
		data: {
		status: 'success',
		…
		}
			data: {
			status: 'success',
			…
			}	

BBT 2.4.4
 	
Testing using BV2.8
	{query: {
	flat_type: '5 ROOM',
	latitude: 1.25,
	longitude: 104.05
	}}
		status: 200
		data: {
		status: 'Invalid Input – Longitude'
		}
			status: 200
			data: {
			status: 'Invalid Input – Longitude'
			}
				PASS

Whitebox Testing

Introduction

Our group conducted white box testing on 2 functions that implement complex business logic – getNeighbourhoodPriceComparisonChart(req, res) in ChartController and getAdvice(req, res) in DetailsController. The getNeighbourhoodPriceComparisonChart(req, res) takes user input and returns a time-series data of average house pricing. Due to different user types, the way of searching and aggregating the data is different across the different user types. The getAdvice(req, res) function acts as a logic-based agent, generating a suitable house type and neighbourhood according to the user selection.

Approach

We drew the Control Flow Graphs (CFG) for both functions and sought to perform Level 2 coverage (100% branch coverage). We then perform automated testing using Jest. The CFG and test cases are detailed in the following pages.

Implementation & Results

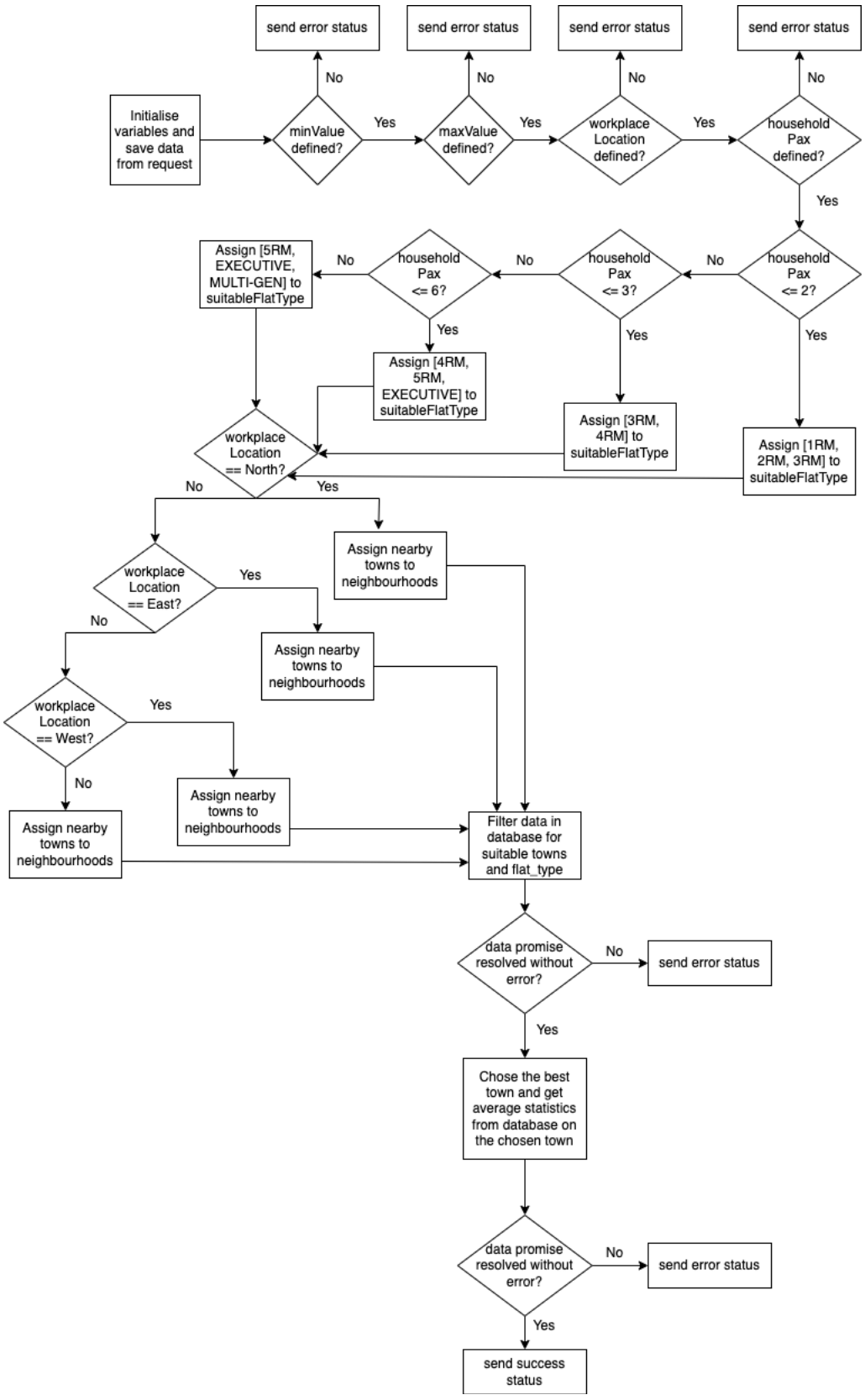
The testing code is found in the file whitebox.test.js .

Jest Test Coverage Report

Note: The test coverage is low as the whitebox testing is only for the chosen functions instead of the whole controller. Some of the paths were not able to be tested as they will only occur during network errors.

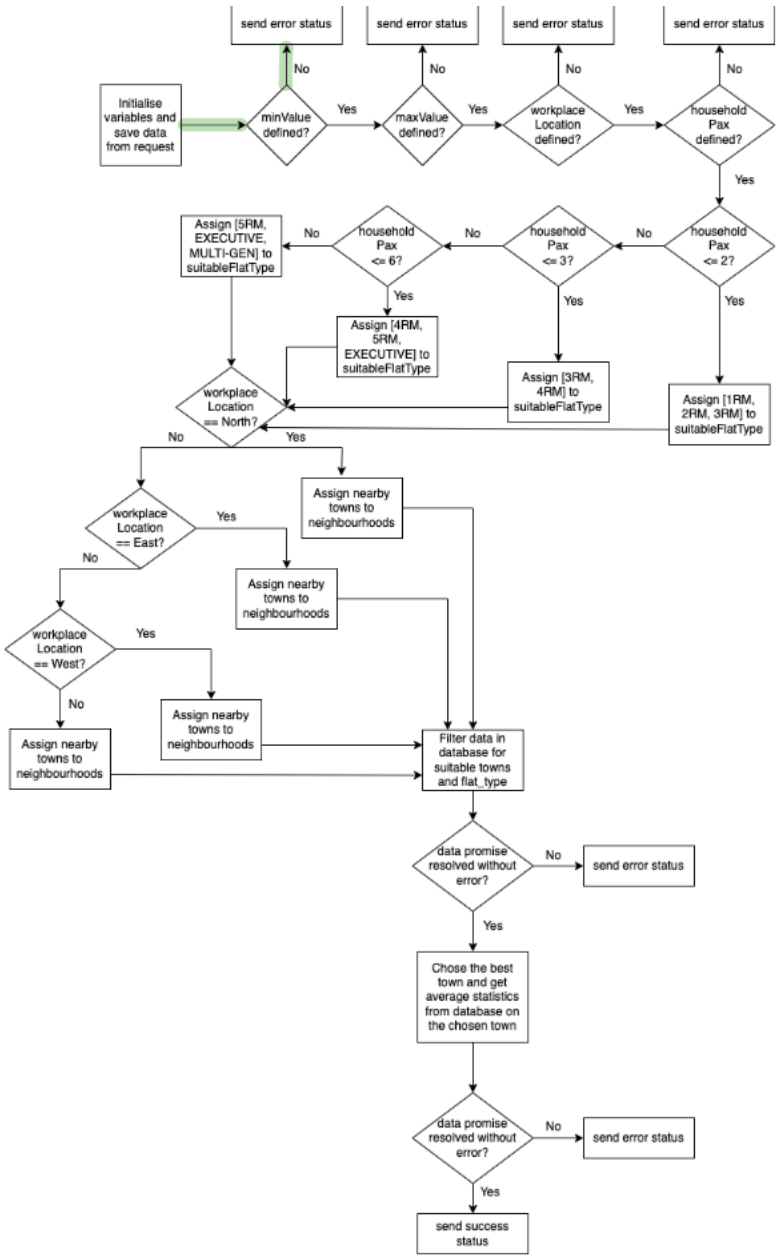
```
> revaluate_server@1.0.0 test
> jest --coverage

 PASS  ./whitebox.test.js (78.627 s)
  Details Controller - getAdvice function
    ✓ Testing of getAdvice function - WBT 1A (3007 ms)
    ✓ Testing of getAdvice function - WBT 1B (3003 ms)
    ✓ Testing of getAdvice function - WBT 1C (3003 ms)
    ✓ Testing of getAdvice function - WBT 1D (3003 ms)
    ✓ Testing of getAdvice function - WBT 1E (3004 ms)
    ✓ Testing of getAdvice function - WBT 1F (3005 ms)
    ✓ Testing of getAdvice function - WBT 1G (3003 ms)
    ✓ Testing of getAdvice function - WBT 1H (3003 ms)
    ✓ Testing of getAdvice function - WBT 1I (3004 ms)
    ✓ Testing of getAdvice function - WBT 1I (3003 ms)
    ✓ Testing of getAdvice function - WBT 1J (3004 ms)
    ✓ Testing of getAdvice function - WBT 1K (3003 ms)
    ✓ Testing of getAdvice function - WBT 1L (3004 ms)
    ✓ Testing of getAdvice function - WBT 1M (3003 ms)
    ✓ Testing of getAdvice function - WBT 1N (3005 ms)
    ✓ Testing of getAdvice function - WBT 1O (3003 ms)
    ✓ Testing of getAdvice function - WBT 1P (3003 ms)
    ✓ Testing of getAdvice function - WBT 1R (3003 ms)
    ✓ Testing of getAdvice function - WBT 1S (3003 ms)
    ✓ Testing of getAdvice function - WBT 1T (3003 ms)
    ✓ Testing of getAdvice function - WBT 1U (3003 ms)
  Chart Controller - getNeighbourhoodPriceComparisonChart function
    ✓ Testing of getNeighbourhoodPriceComparisonChart function - WBT 2A (3006 ms)
    ✓ Testing of getNeighbourhoodPriceComparisonChart function - WBT 2B (3002 ms)
    ✓ Testing of getNeighbourhoodPriceComparisonChart function - WBT 2C (3004 ms)
    ✓ Testing of getNeighbourhoodPriceComparisonChart function - WBT 2D (3002 ms)
    ✓ Testing of getNeighbourhoodPriceComparisonChart function - WBT 2E (3002 ms)

-----------------------|---------|----------|---------|---------|-------------------------------------------
File                   | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------|---------|----------|---------|---------|-------------------------------------------
All files              |    31.5 |    43.75 |   16.27 |   33.53 |
 Controllers           |   29.01 |    43.75 |   16.27 |   30.62 |
  ChartController.js   |   24.21 |    26.92 |   11.11 |   24.67 | 16-40,70,88-112,114-143,146-173,177-256
  DetailsController.js |   33.67 |    55.26 |      25 |   36.14 | 15-144,224-226
 Models                |     100 |      100 |     100 |     100 |
  HouseData.js         |     100 |      100 |     100 |     100 |
-----------------------|---------|----------|---------|---------|-------------------------------------------
Test Suites: 1 passed, 1 total
Tests:       26 passed, 26 total
Snapshots:   0 total
Time:        78.715 s
Ran all test suites.
```

Control Flow Graph for getAdvice (req, res)



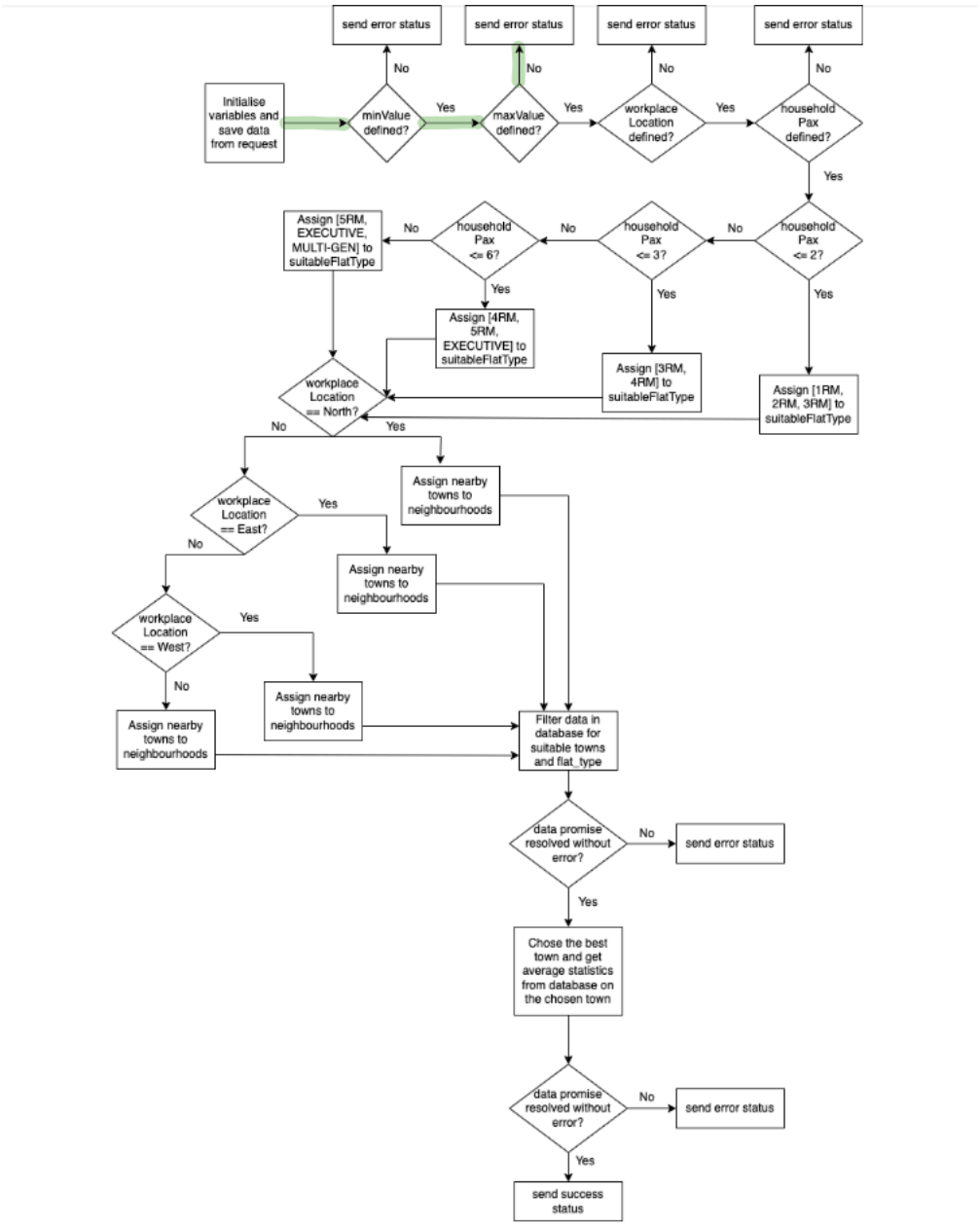
Test Code	Input	Expected Output	Actual Output	Status

WBT 1A	{query: {
	maxValue: 1000000,
	workplaceLocation: 'North',
	householdPax: 5
	}}
		Status: 200
		Data: {
		Status: 'error'
		}
			Status: 200
			Data: {
			Status: 'error'
			}	



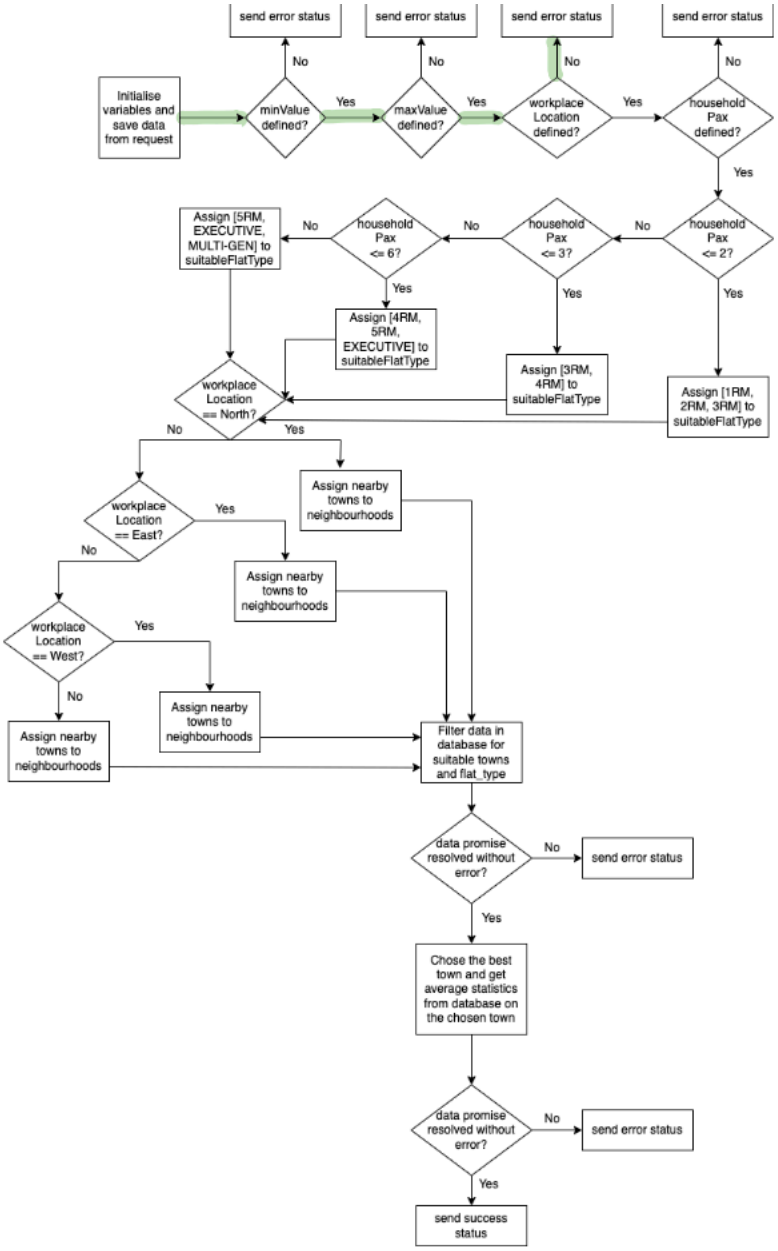
Test Code	Input	Expected Output	Actual Output	Status

WBT 1B	{query: {
	minValue: 100000,
	workplaceLocation: 'North',
	householdPax: 5
	}}
		Status: 200
		Data: {
		Status: 'error'
		}
			Status: 200
			Data: {
			Status: 'error'
			}	



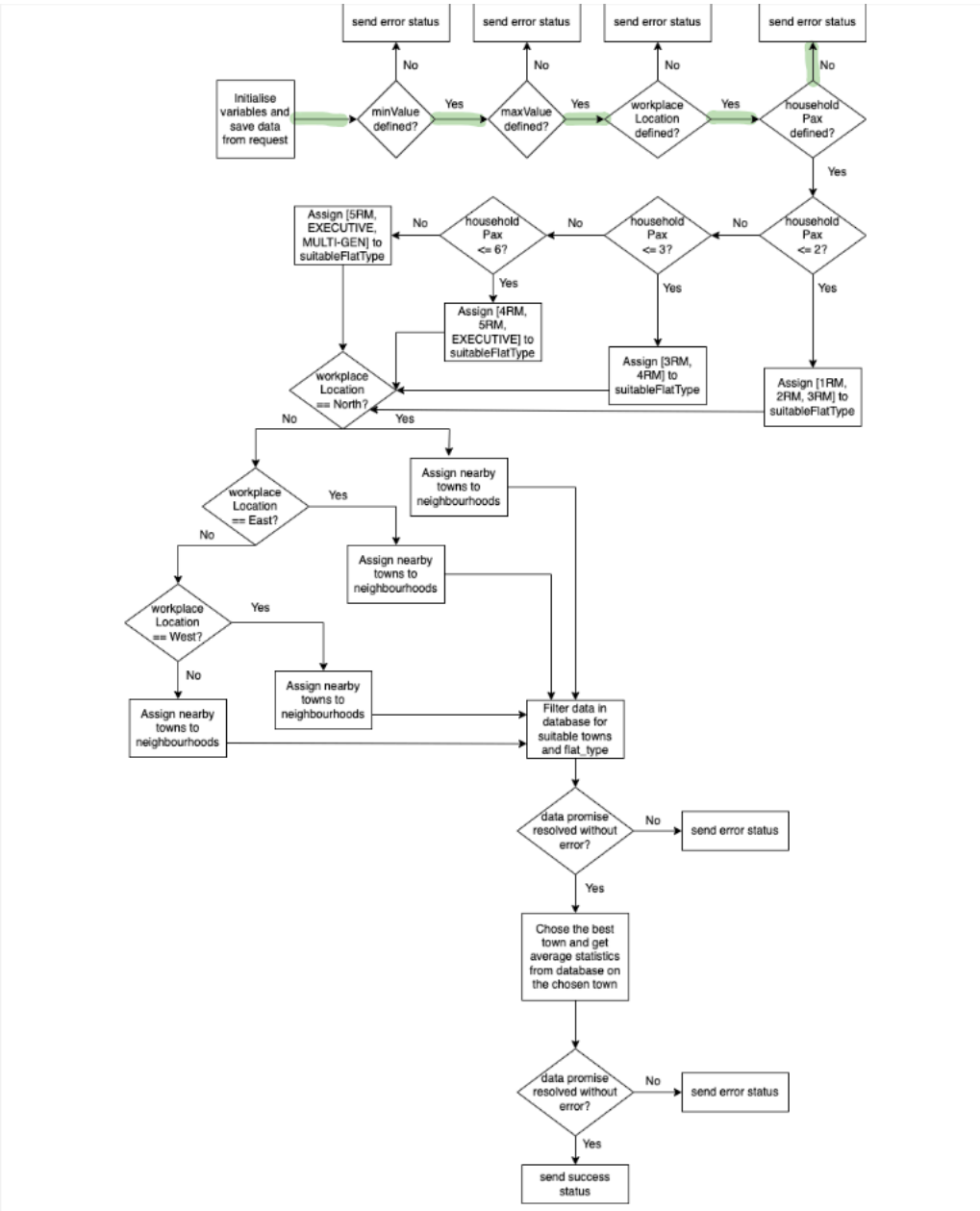
Test Code	Input	Expected Output	Actual Output	Status

WBT 1C	{query: {
	minValue: 100000,
	maxValue: 1000000,
	householdPax: 5
	}}
		Status: 200
		Data: {
		Status: 'error'
		}
			Status: 200
			Data: {
			Status: 'error'
			}	



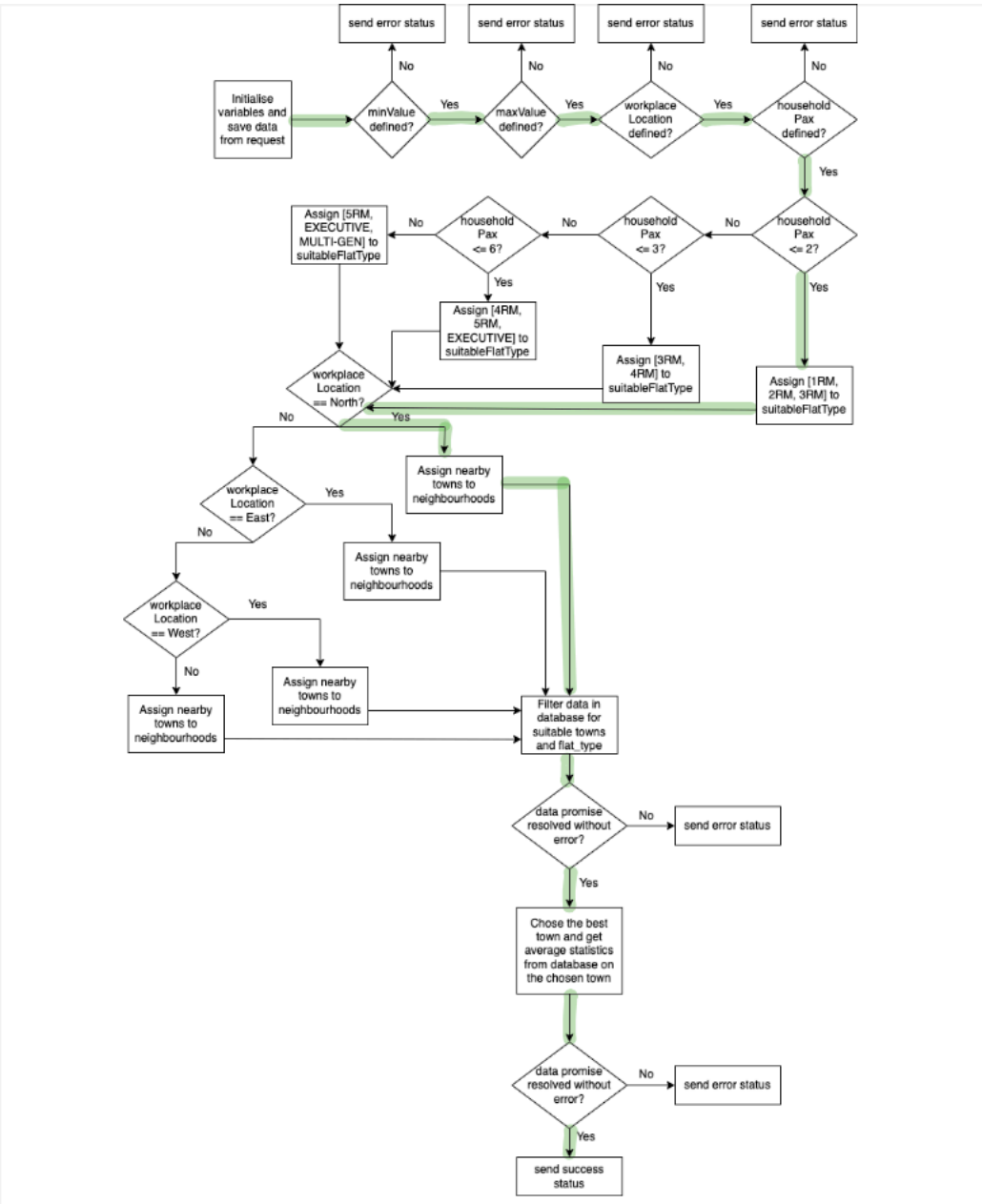
Test Code	Input	Expected Output	Actual Output	Status

WBT 1D	{query: {
	minValue: 0,
	maxValue: 1000000,
	workplaceLocation: 'North'
	}}
		Status: 200
		Data: {
		Status: 'error'
		}
			Status: 200
			Data: {
			Status: 'error'
			}	



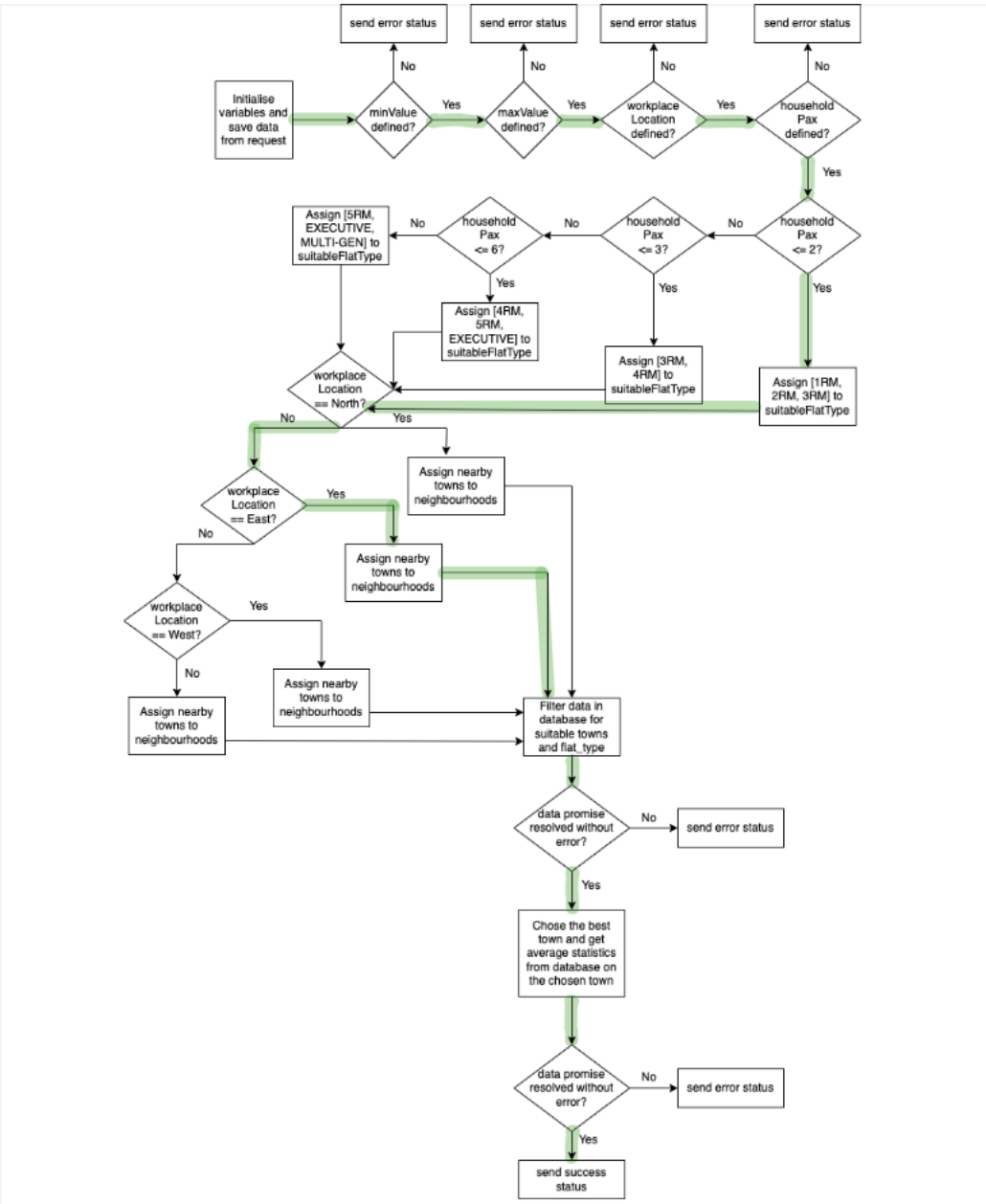
Test Code	Input	Expected Output	Actual Output	Status

WBT 1E	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'North',
	householdPax: 2
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



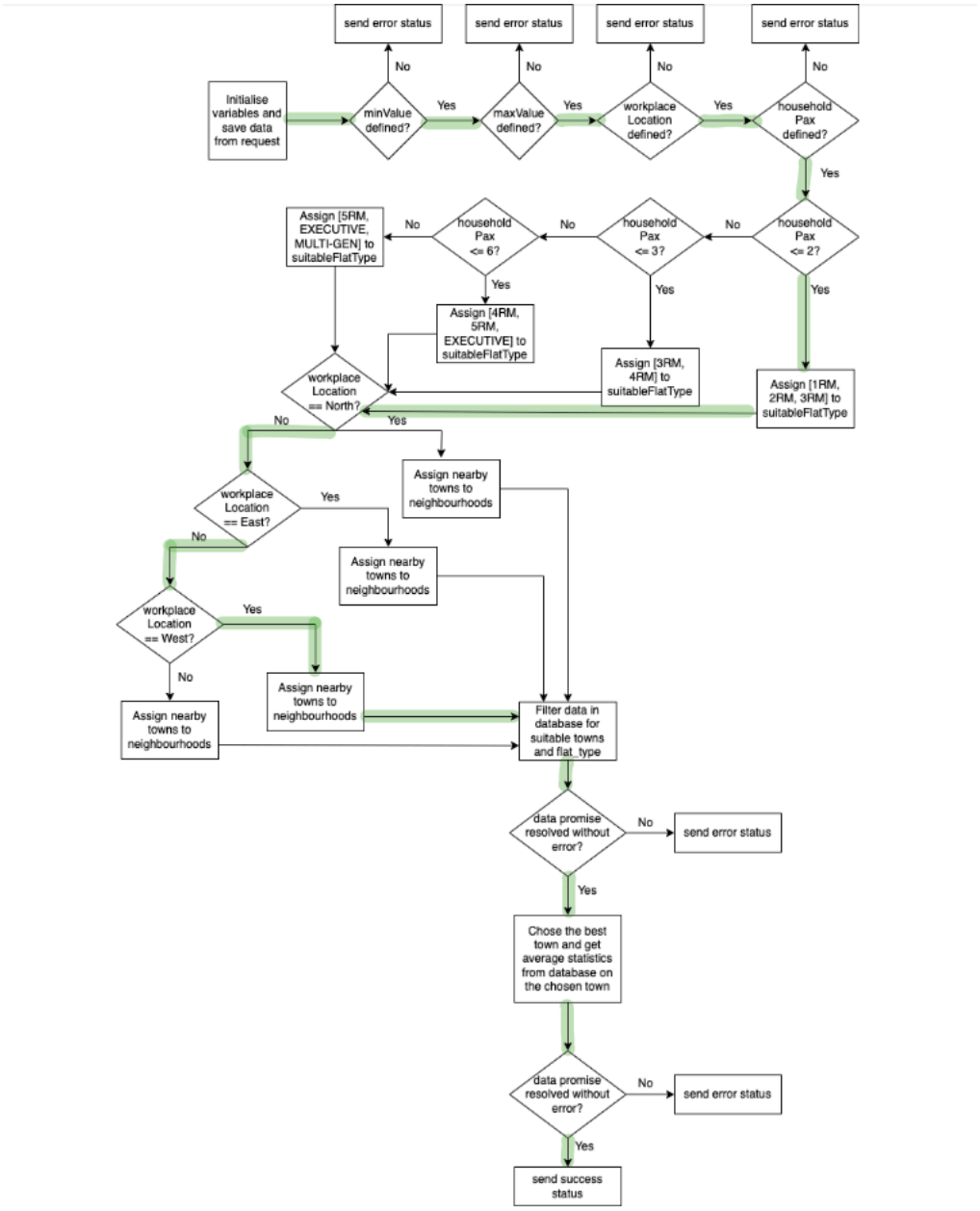
Test Code	Input	Expected Output	Actual Output	Status

WBT 1F	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'East',
	householdPax: 2
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



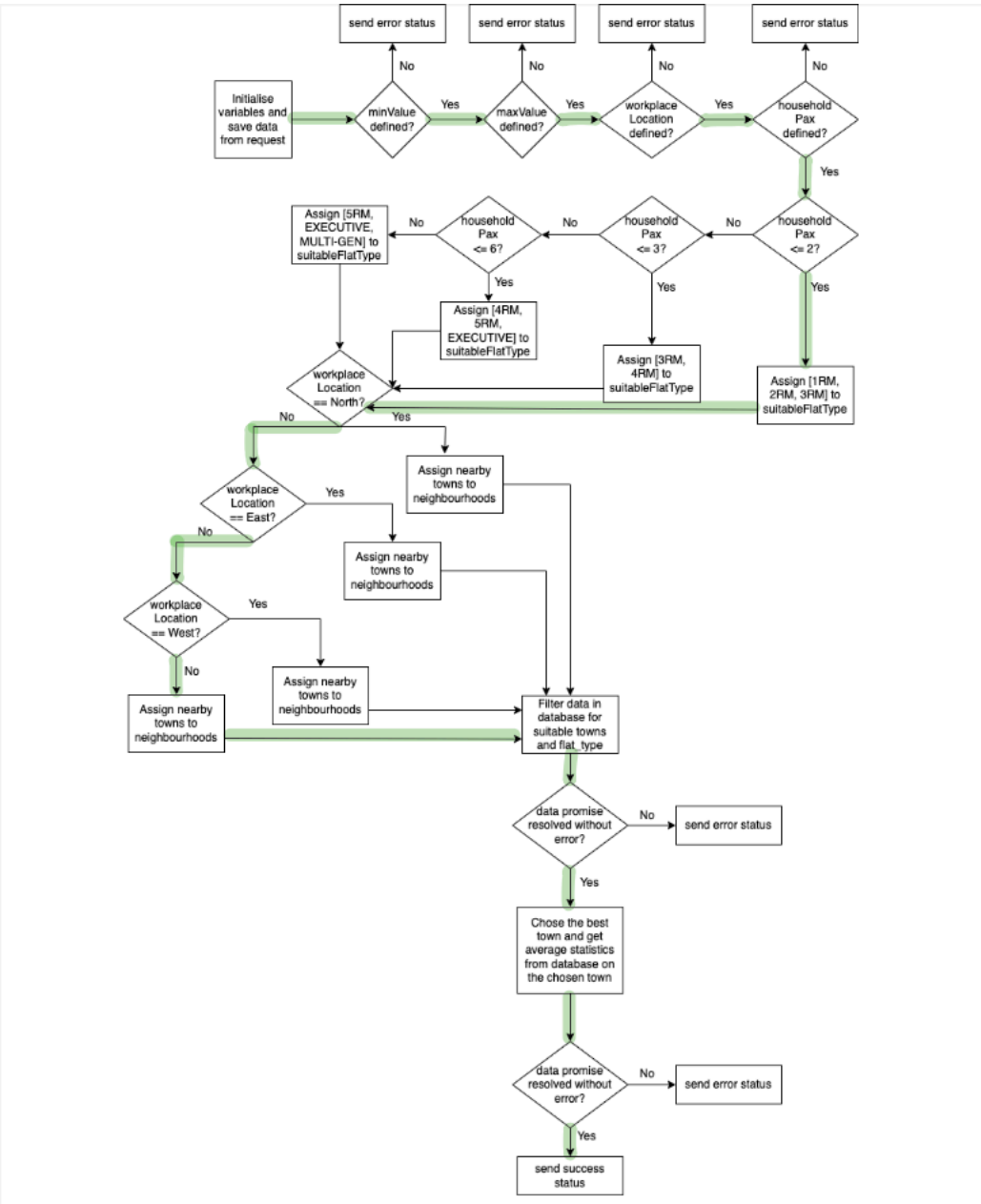
Test Code	Input	Expected Output	Actual Output	Status

WBT 1G	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation:
	'West',
	householdPax: 2
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



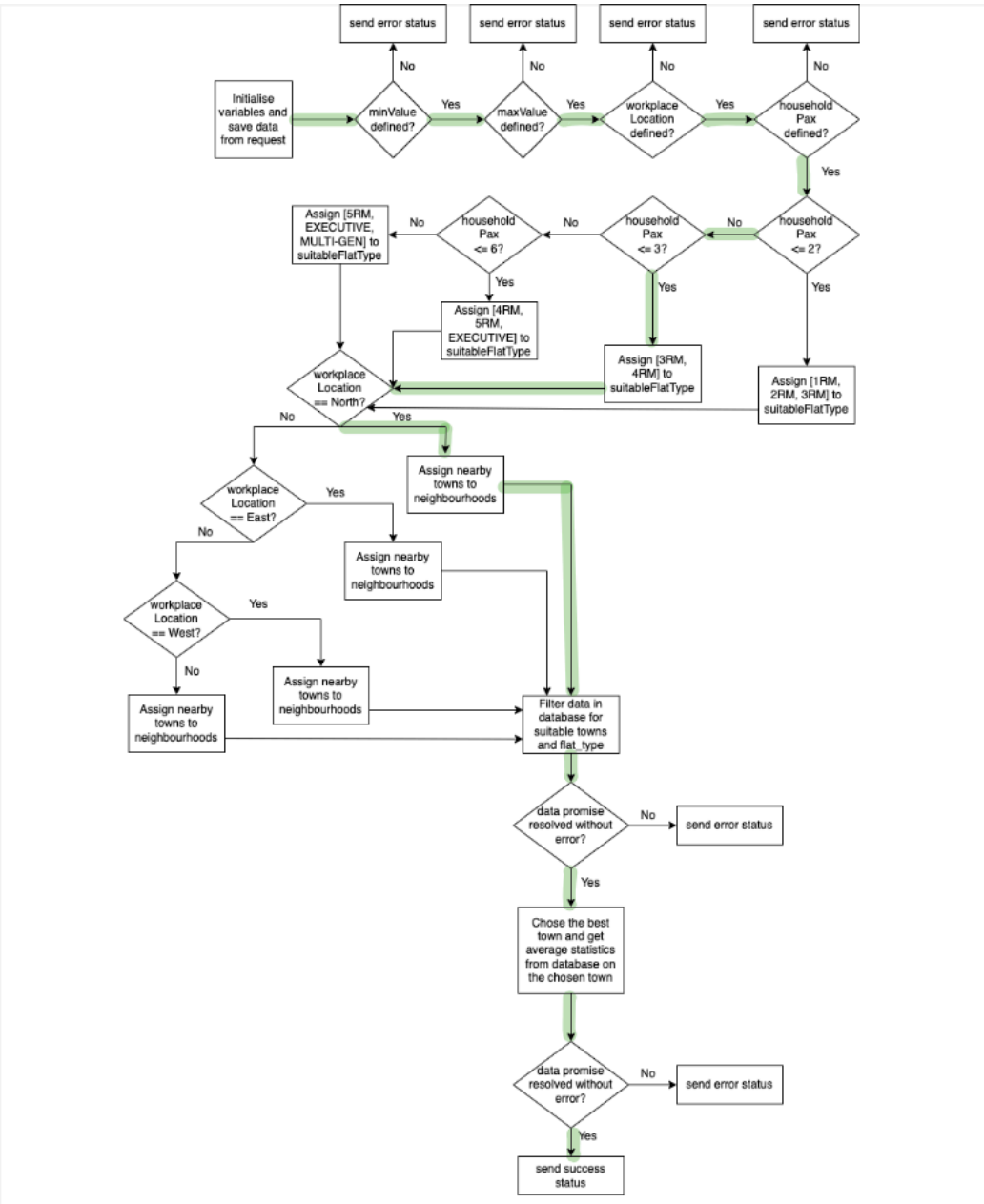
Test Code	Input	Expected Output	Actual Output	Status

WBT 1H	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation:
	'South',
	householdPax: 2
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



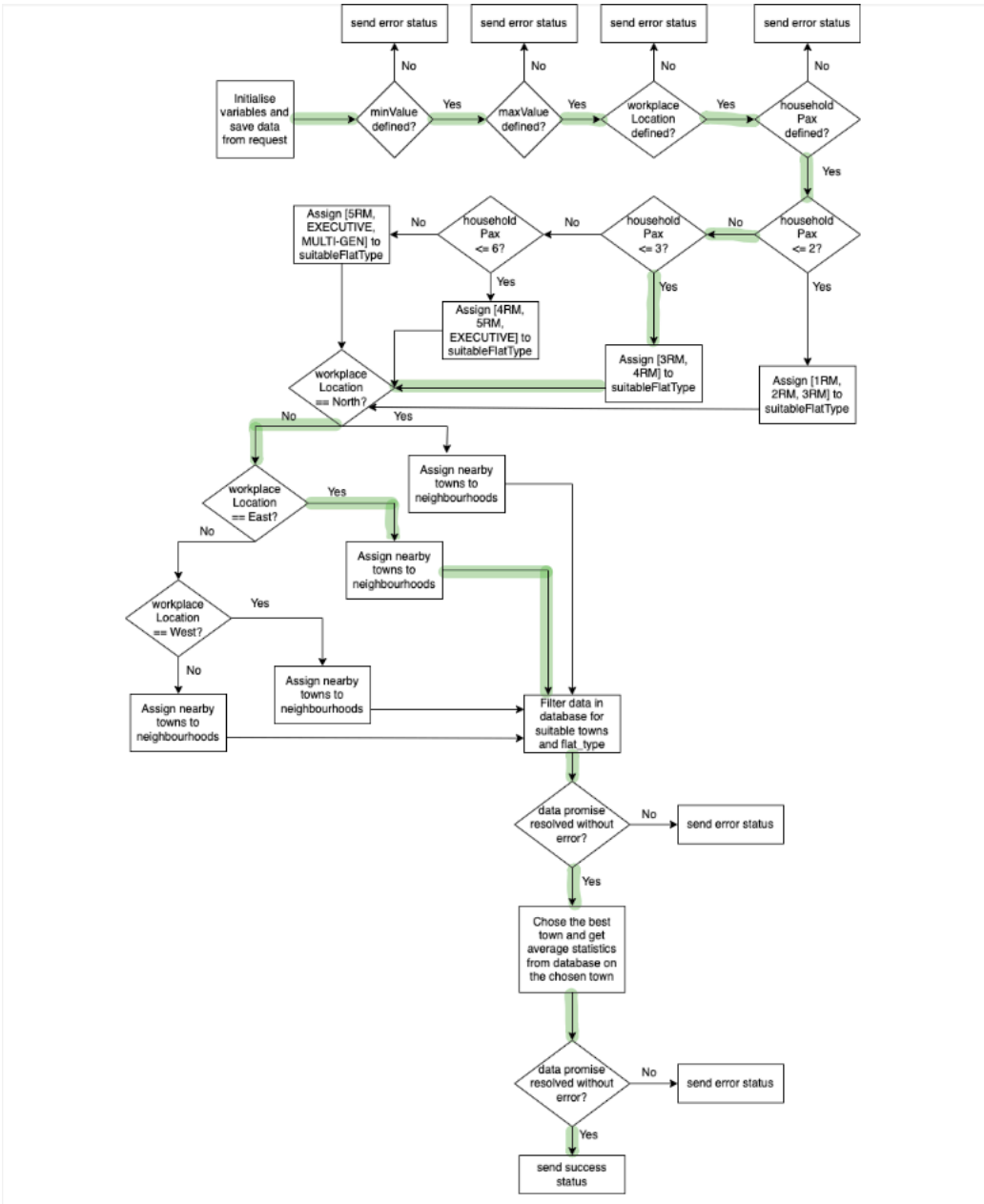
Test Code	Input	Expected Output	Actual Output	Status

WBT 1I	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'North',
	householdPax: 3
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



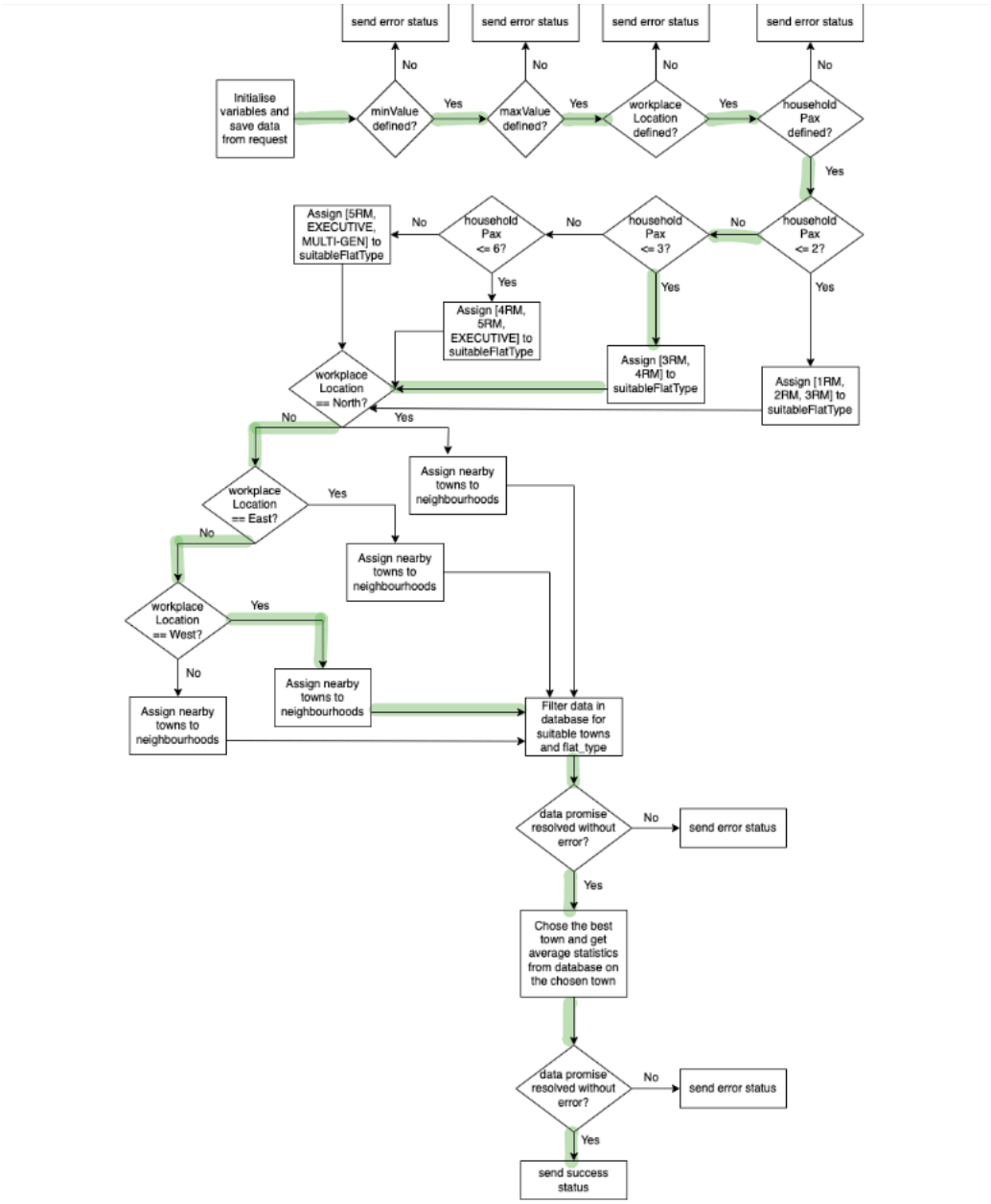
Test Code	Input	Expected Output	Actual Output	Status

WBT 1J	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'East',
	householdPax: 3
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



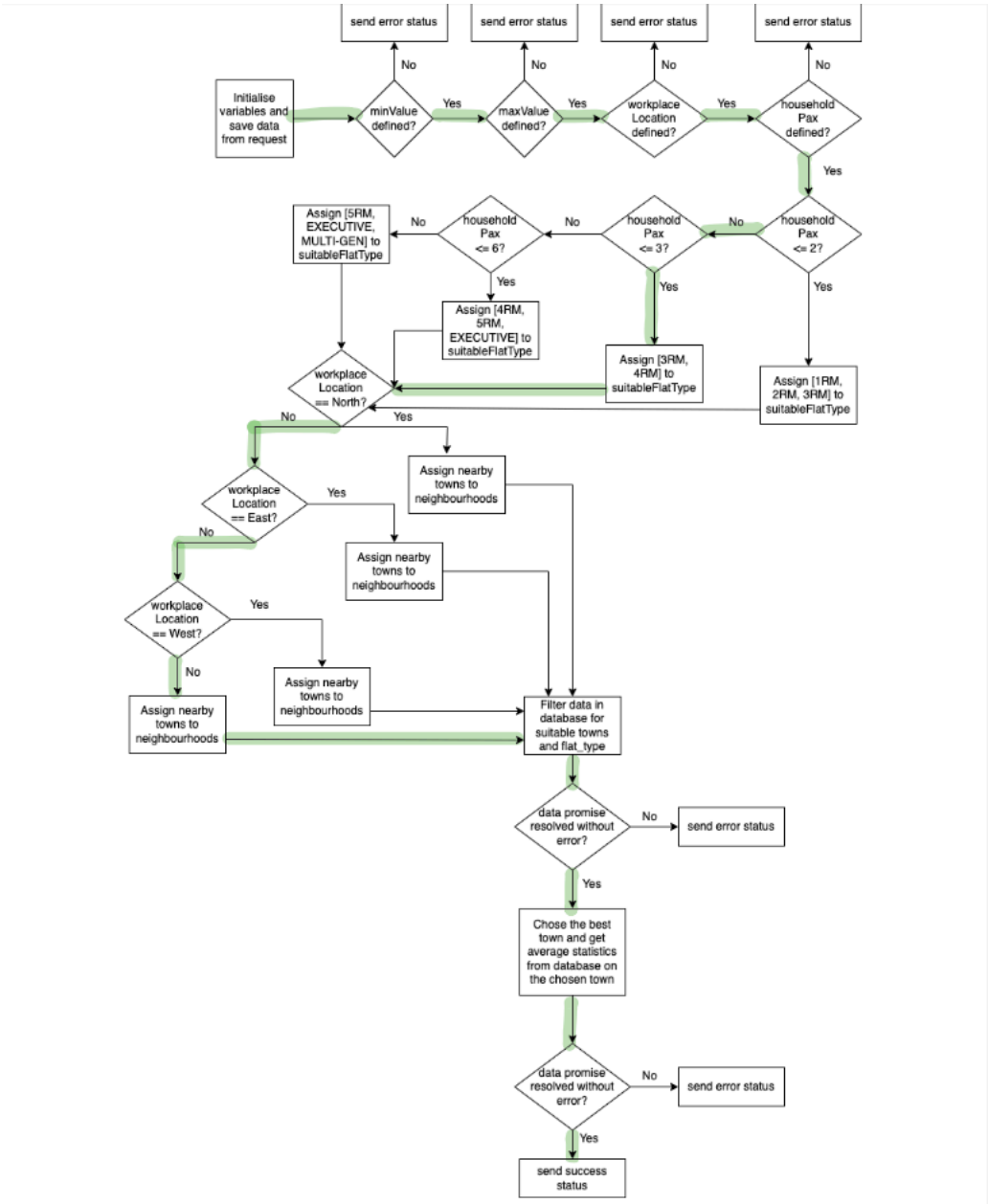
Test Code	Input	Expected Output	Actual Output	Status

WBT 1K	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'West',
	householdPax: 3
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



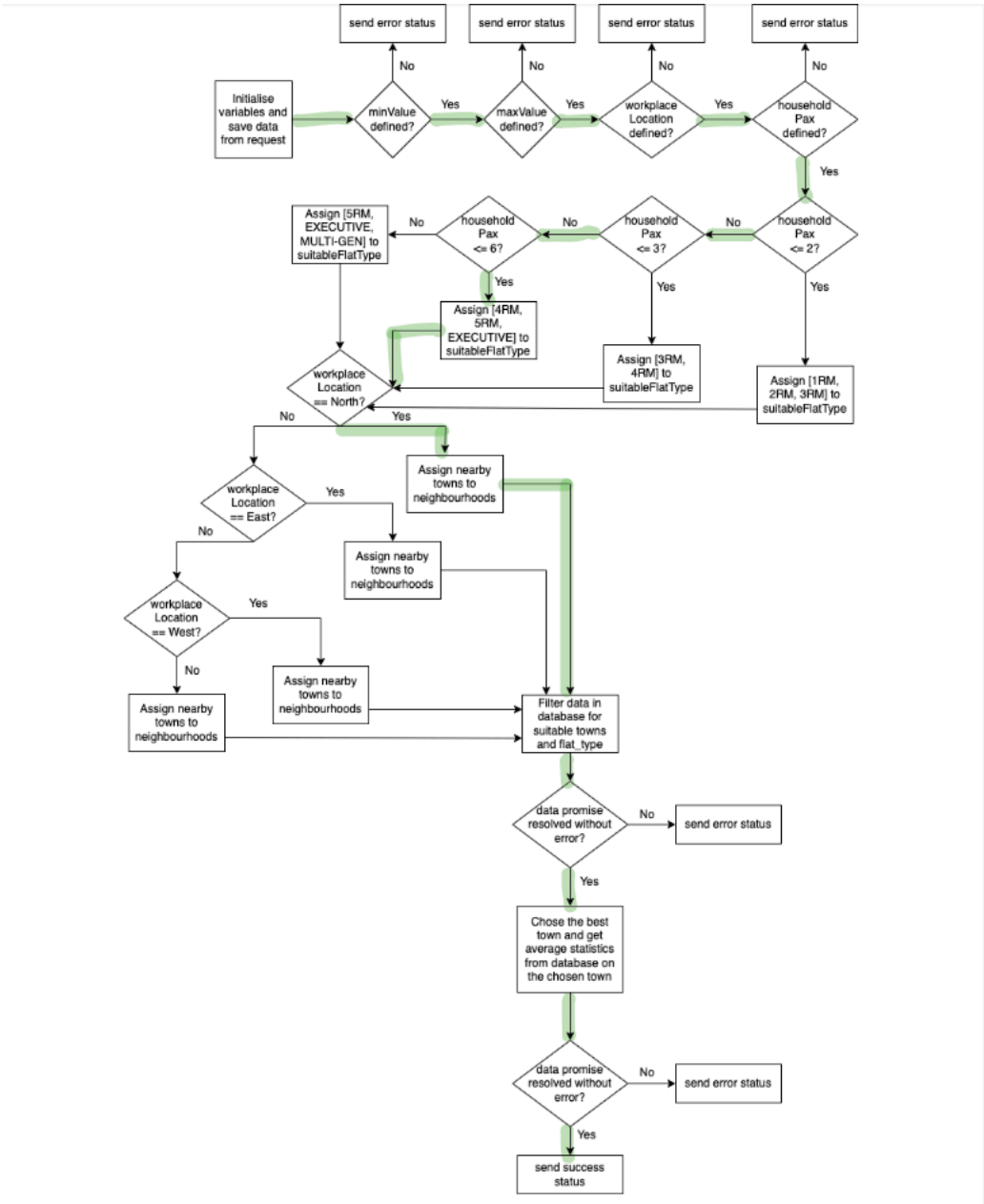
Test Code	Input	Expected Output	Actual Output	Status

WBT 1L	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation:
	'South',
	householdPax: 3
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



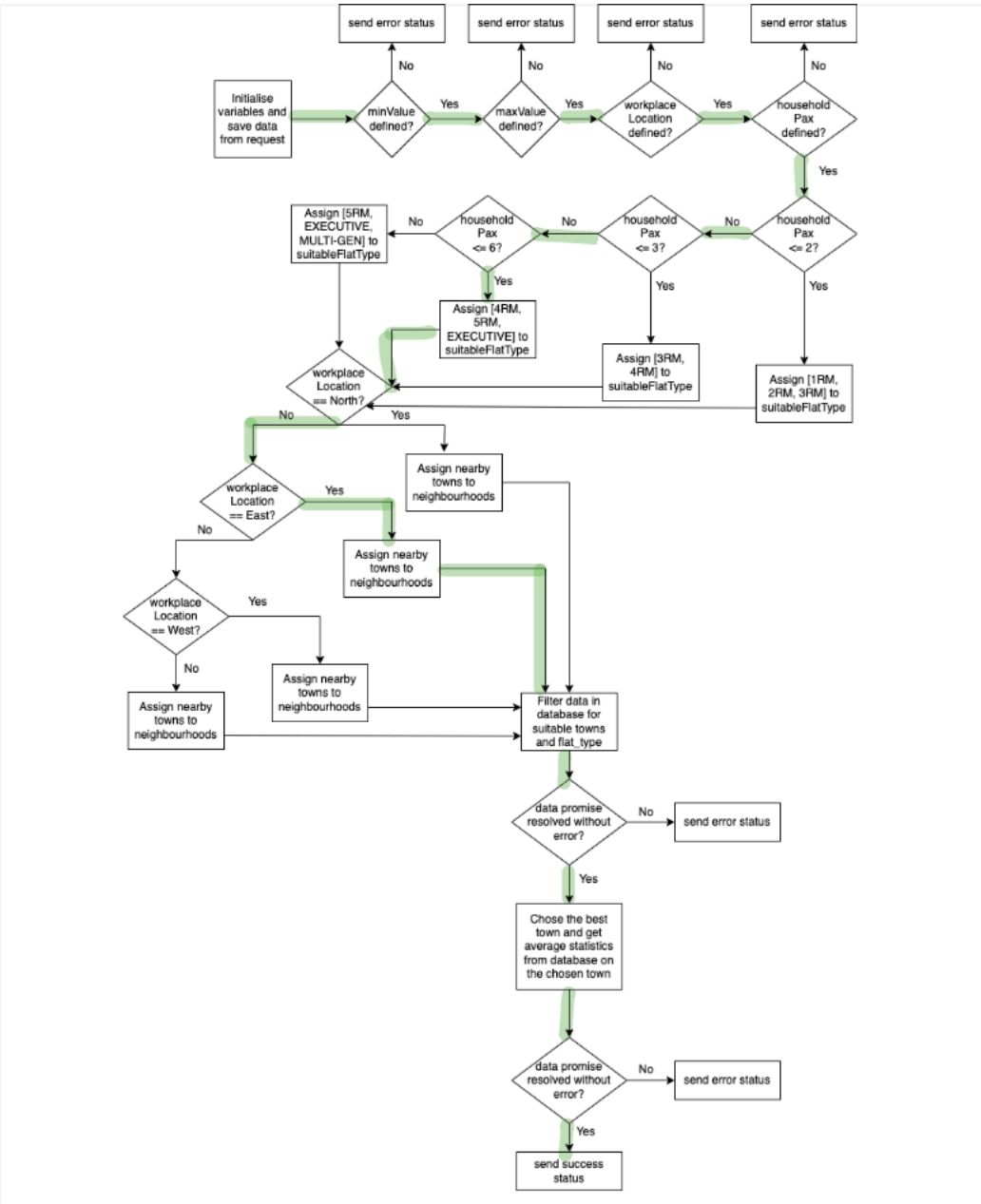
Test Code	Input	Expected Output	Actual Output	Status

WBT 1M	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'North',
	householdPax: 4
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



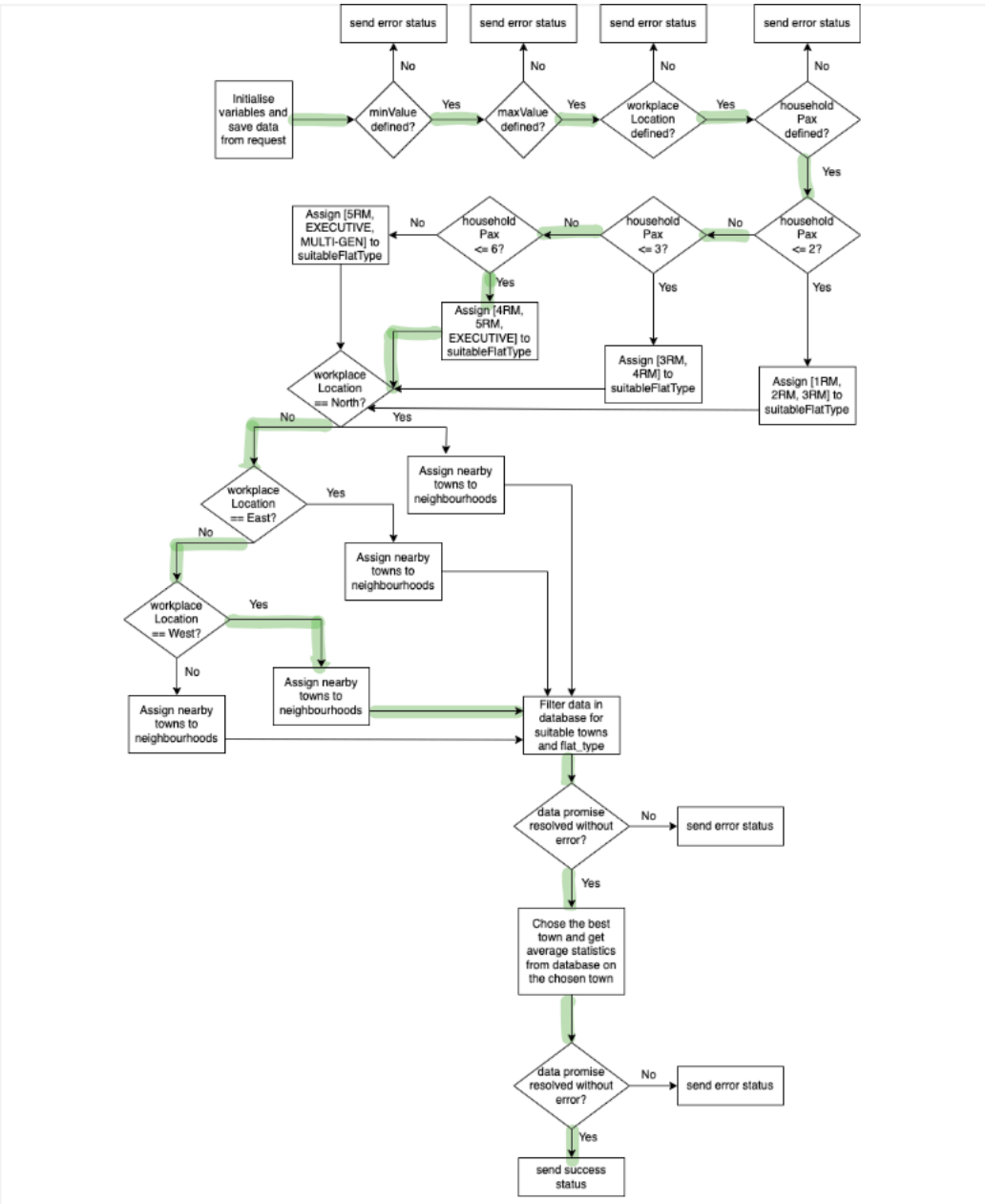
Test Code	Input	Expected Output	Actual Output	Status

WBT 1N	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'East',
	householdPax: 4
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



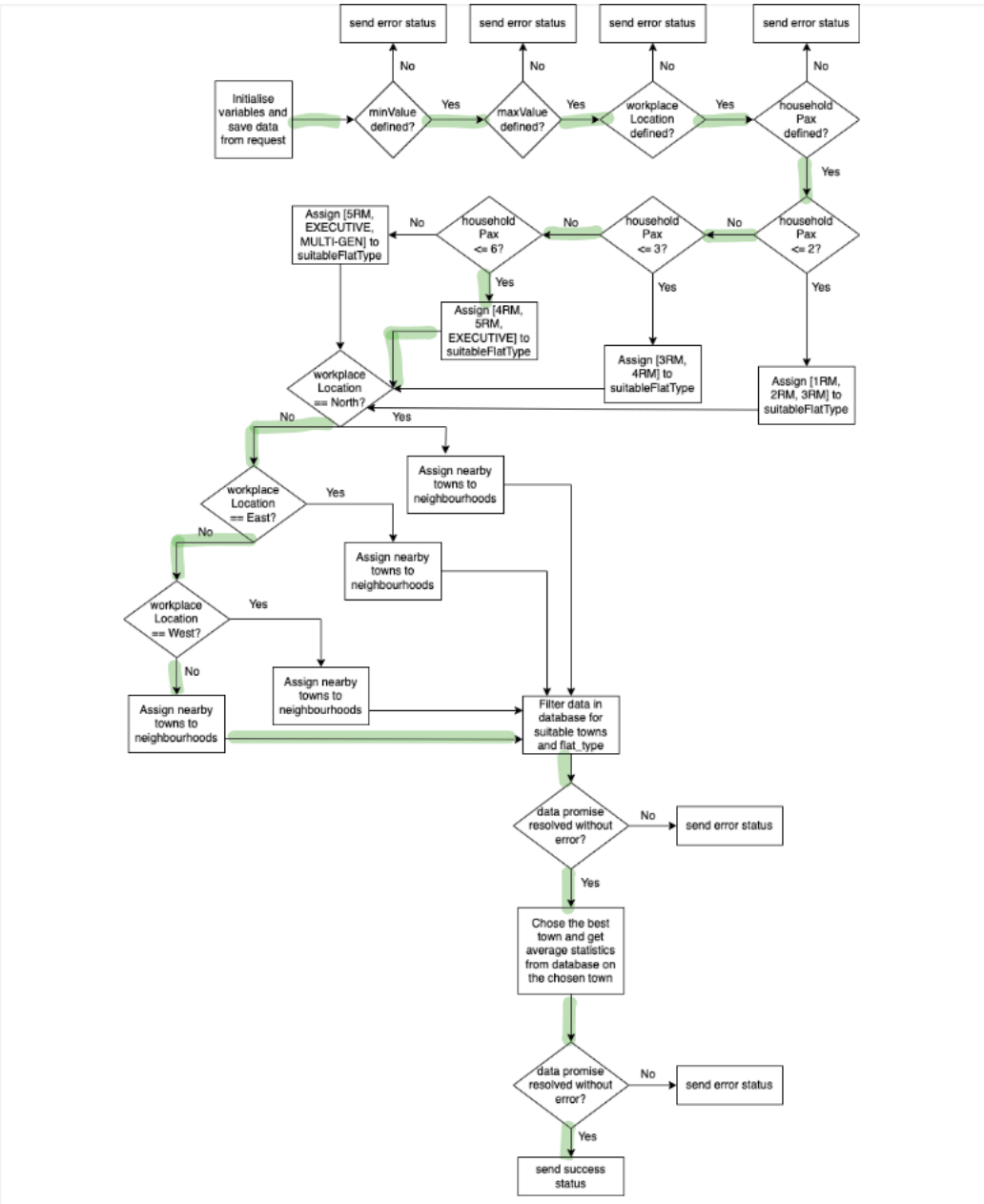
Test Code	Input	Expected Output	Actual Output	Status

WBT 1O	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation:
	'West',
	householdPax: 4
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



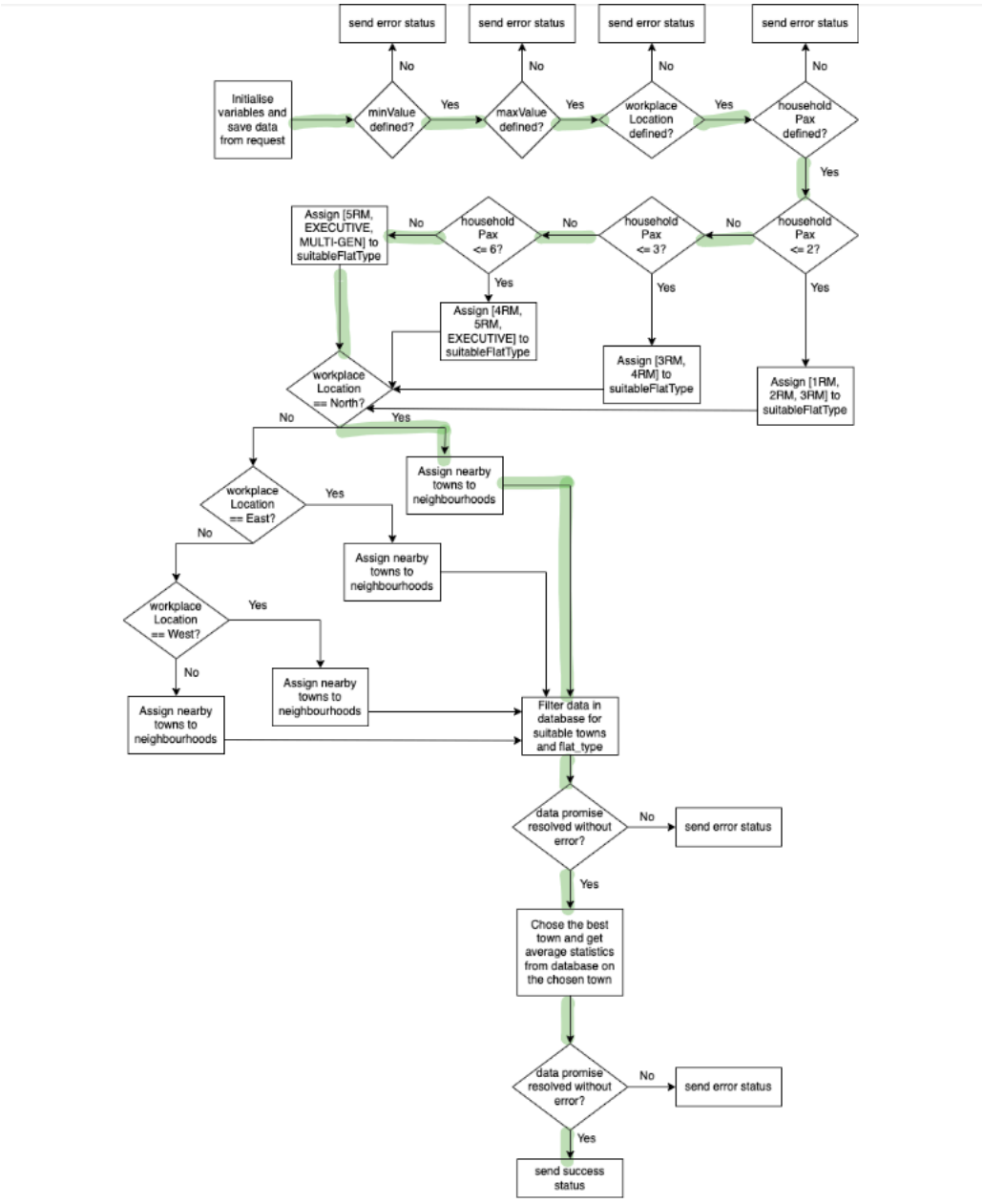
Test Code	Input	Expected Output	Actual Output	Status

WBT 1P	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'South',
	householdPax: 4
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



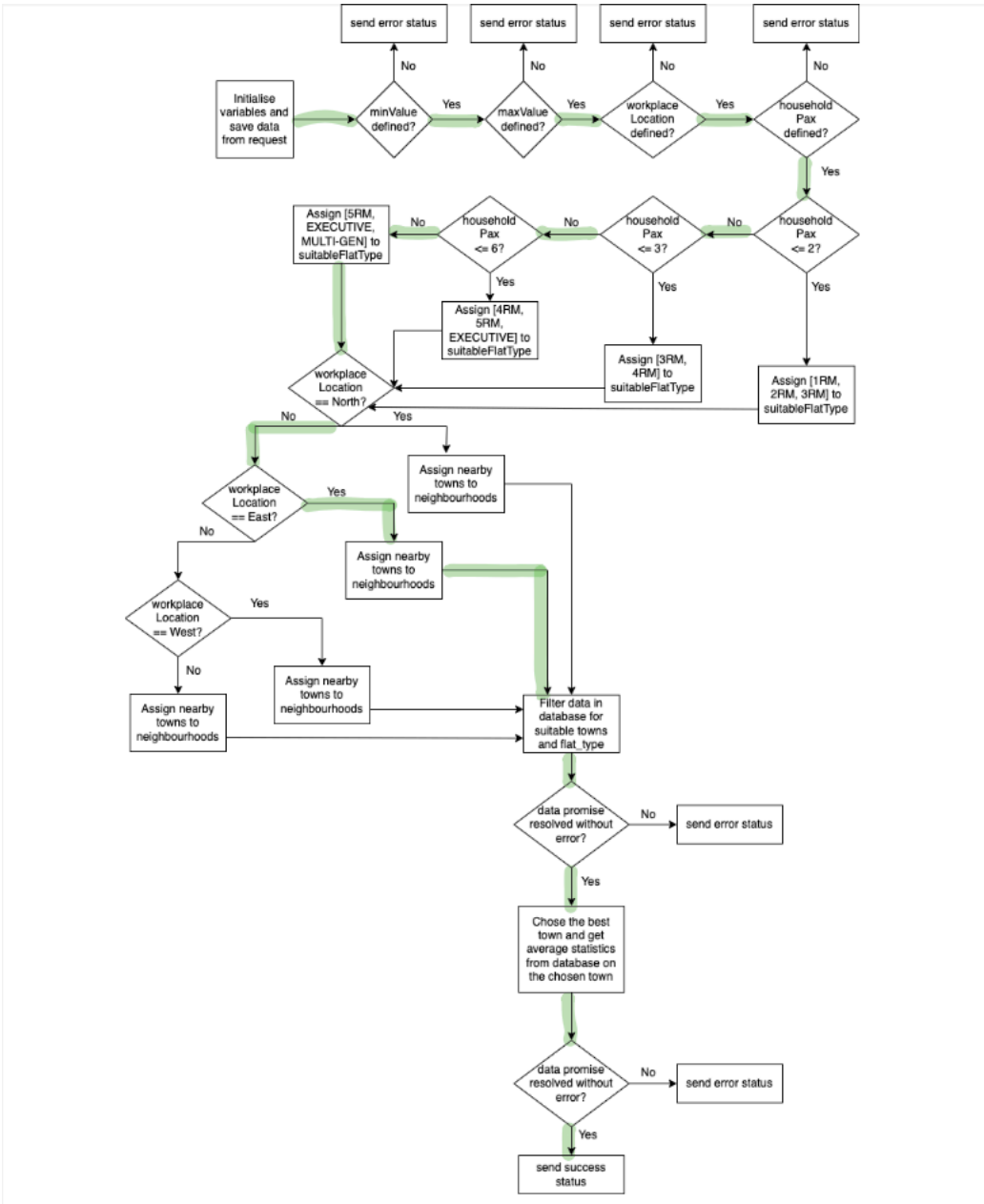
Test Code	Input	Expected Output	Actual Output	Status

WBT 1R	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'North',
	householdPax: 8
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



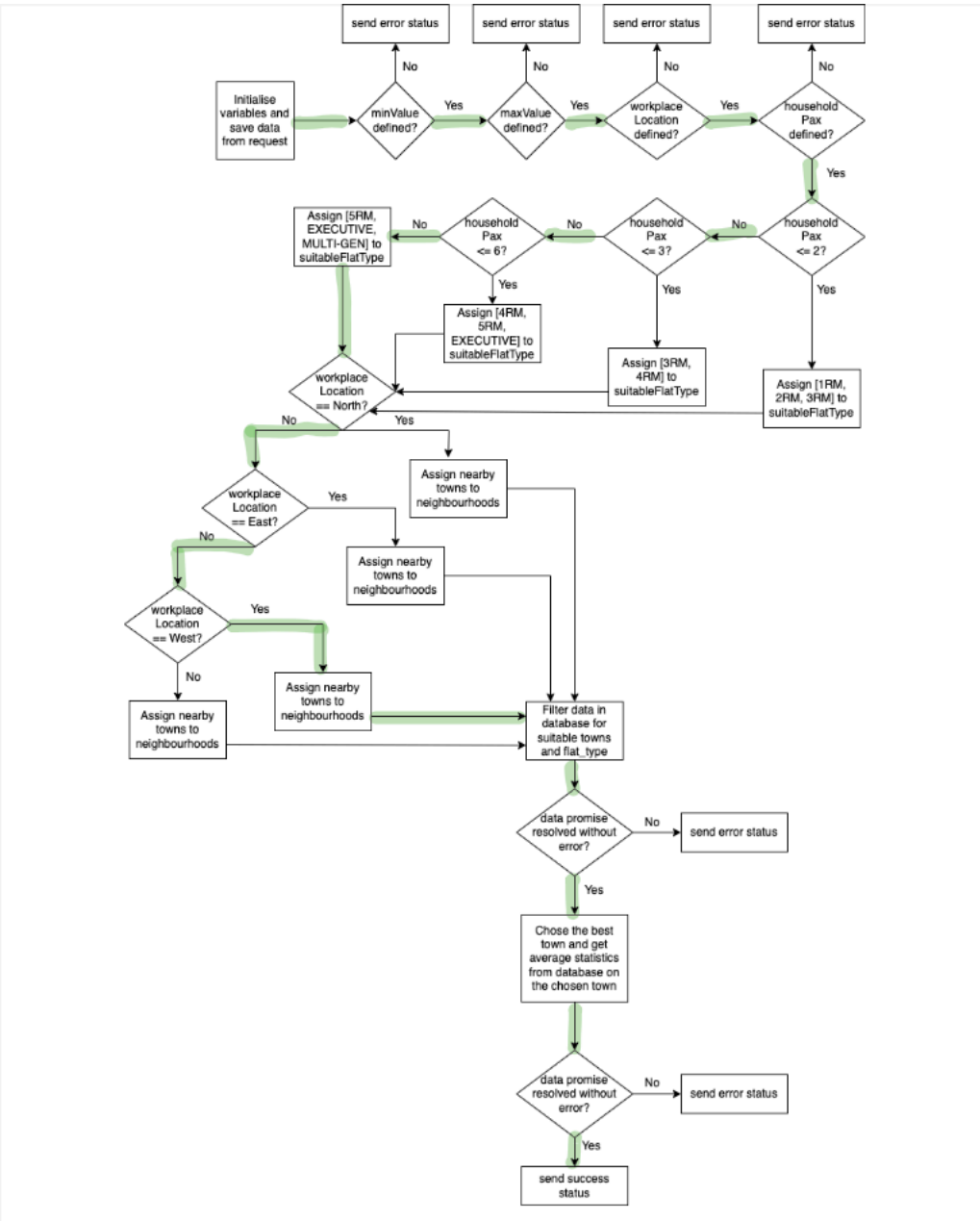
Test Code	Input	Expected Output	Actual Output	Status

WBT 1S	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'East',
	householdPax: 8
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



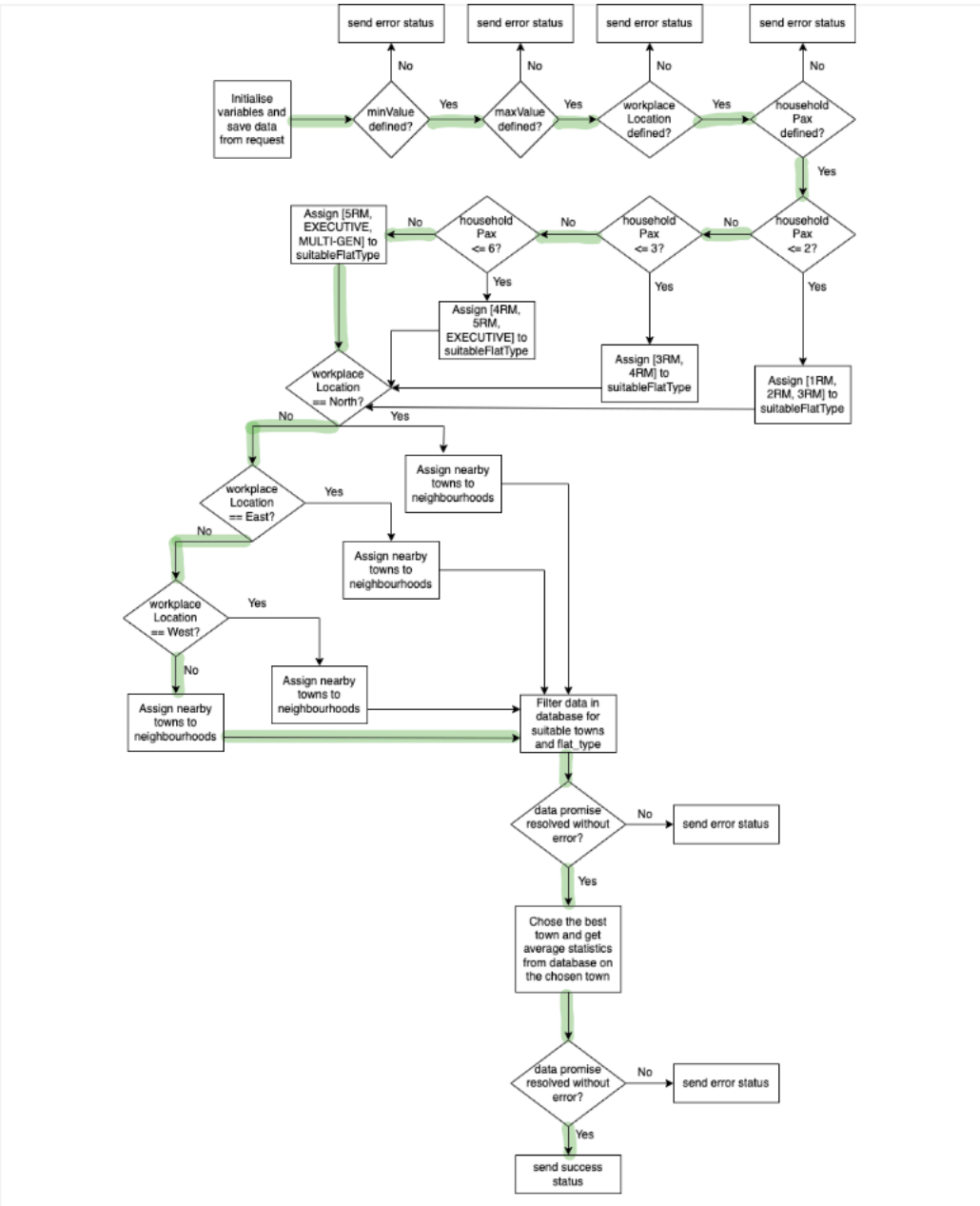
Test Code	Input	Expected Output	Actual Output	Status

WBT 1T	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation: 'West',
	householdPax: 8
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	

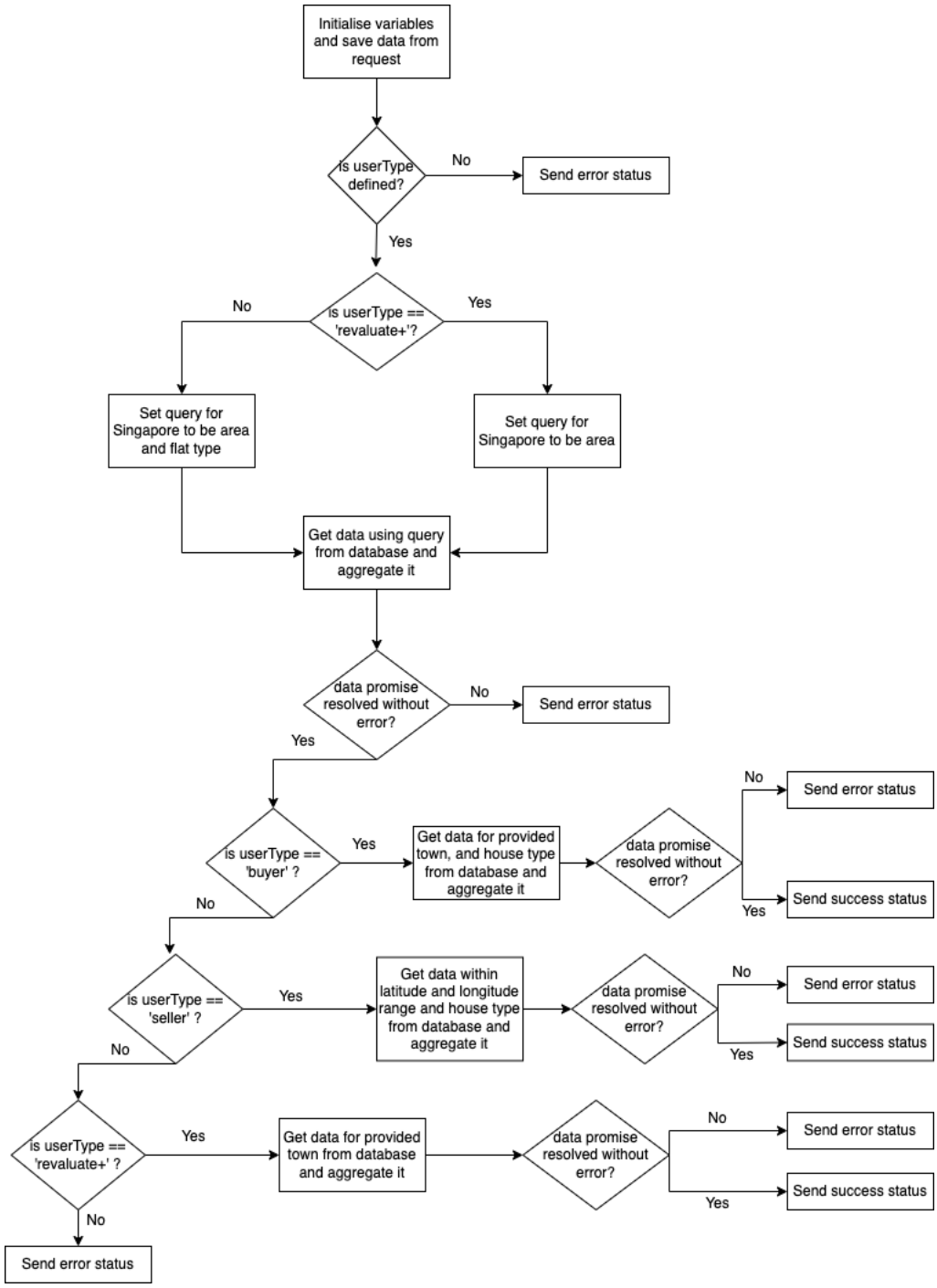


Test Code	Input	Expected Output	Actual Output	Status

WBT 1U	{query: {
	minValue: 100000,
	maxValue: 1000000,
	workplaceLocation:
	'South',
	householdPax: 8
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	

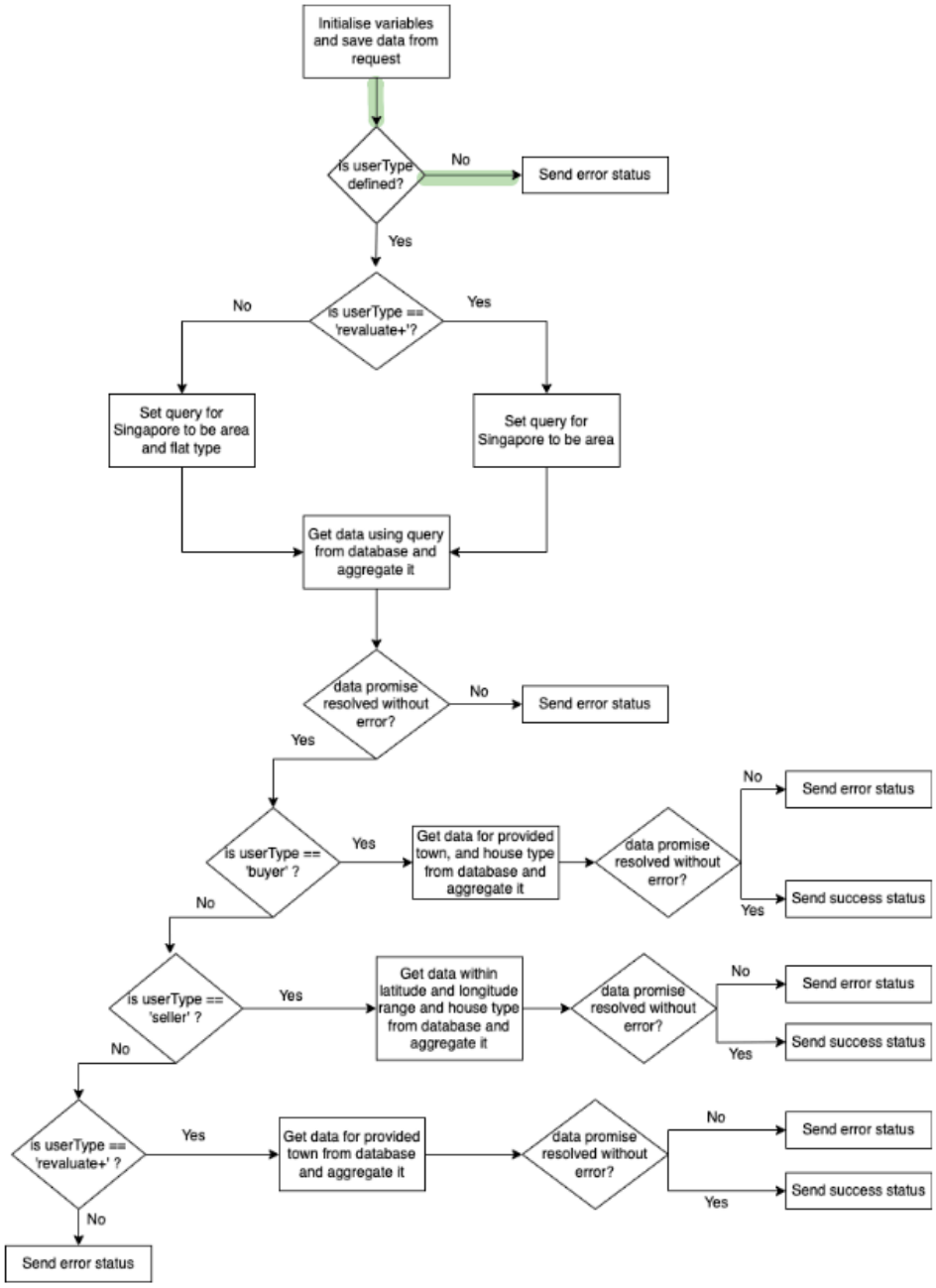


Control Flow Graph for getNeighbourhoodPriceComparisonChart(req, res)



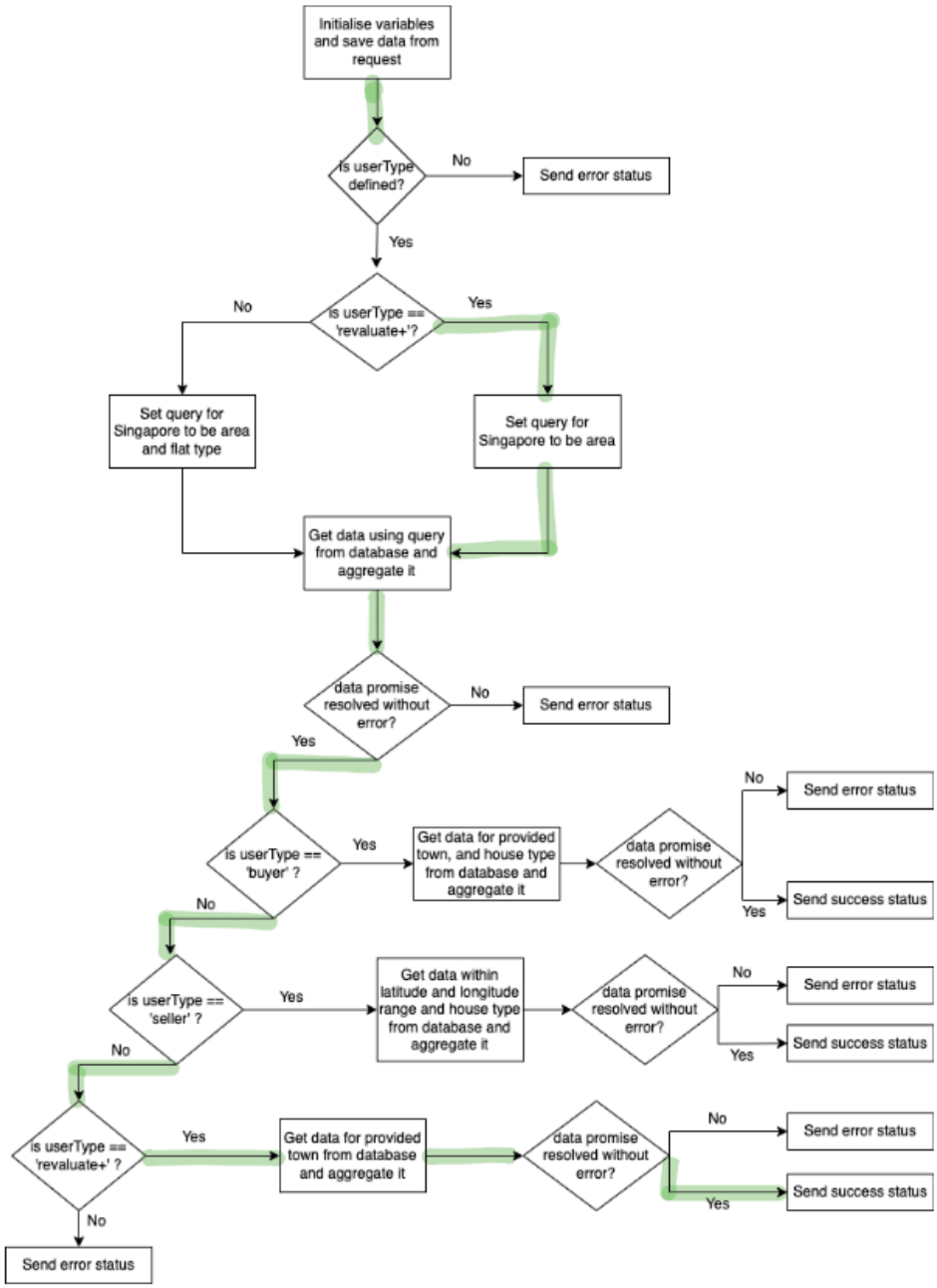
Test Code	Input	Expected Output	Actual Output	Status

WBT 2A	{query: {}}	Status: 200
		Data: {
		Status: ‘error’,
		…
		}
			Status: 200
			Data: {
			Status: ‘error’,
			…
			}	



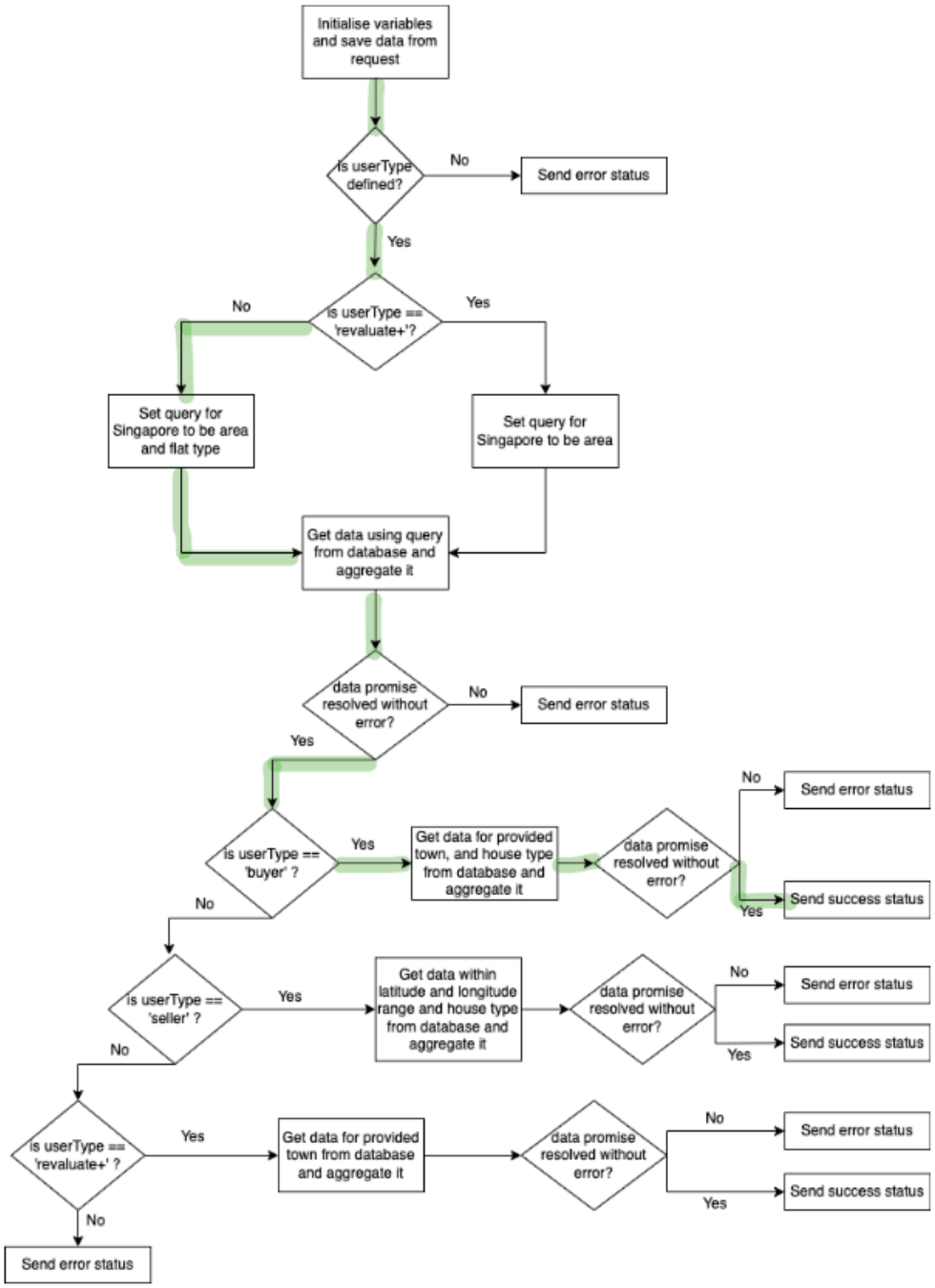
Test Code	Input	Expected Output	Actual Output	Status

WBT 2B	{query: {
	userType:
	‘revaluate+’
	}}
		Status: 200
		Data: {
		Status: ‘success’,
		…
		}
			Status: 200
			Data: {
			Status: ‘success’,
			…
			}	



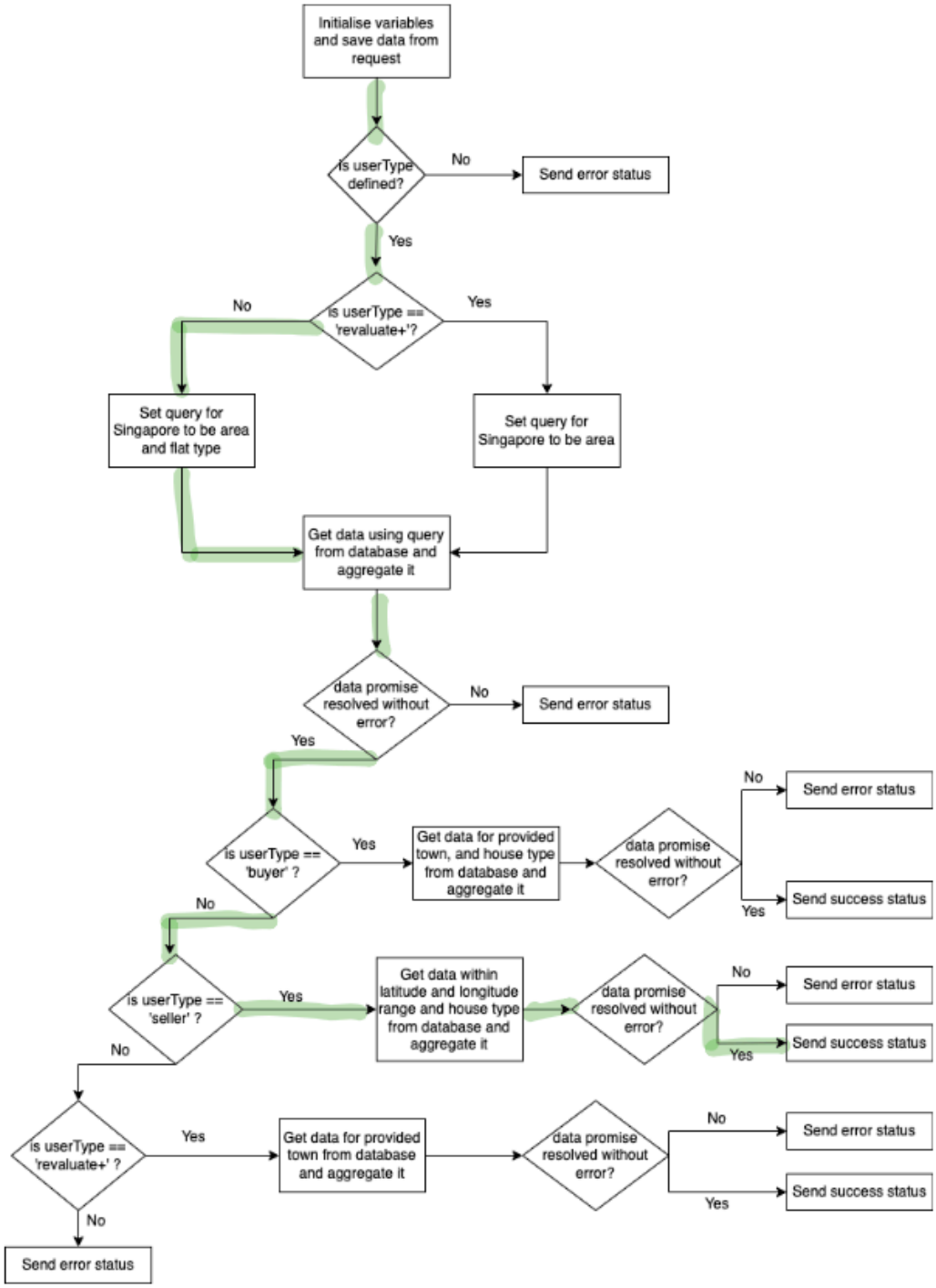
Test Code	Input	Expected Output	Actual Output	Status

WBT 2C	{query: {
	userType: 'buyer',
	flat_type: '5 ROOM',
	town: 'BEDOK'
	}}
		Status: 200
		Data: {
		Status: 'success',
		…
		}
			Status: 200
			Data: {
			Status: 'success',
			…
			}	



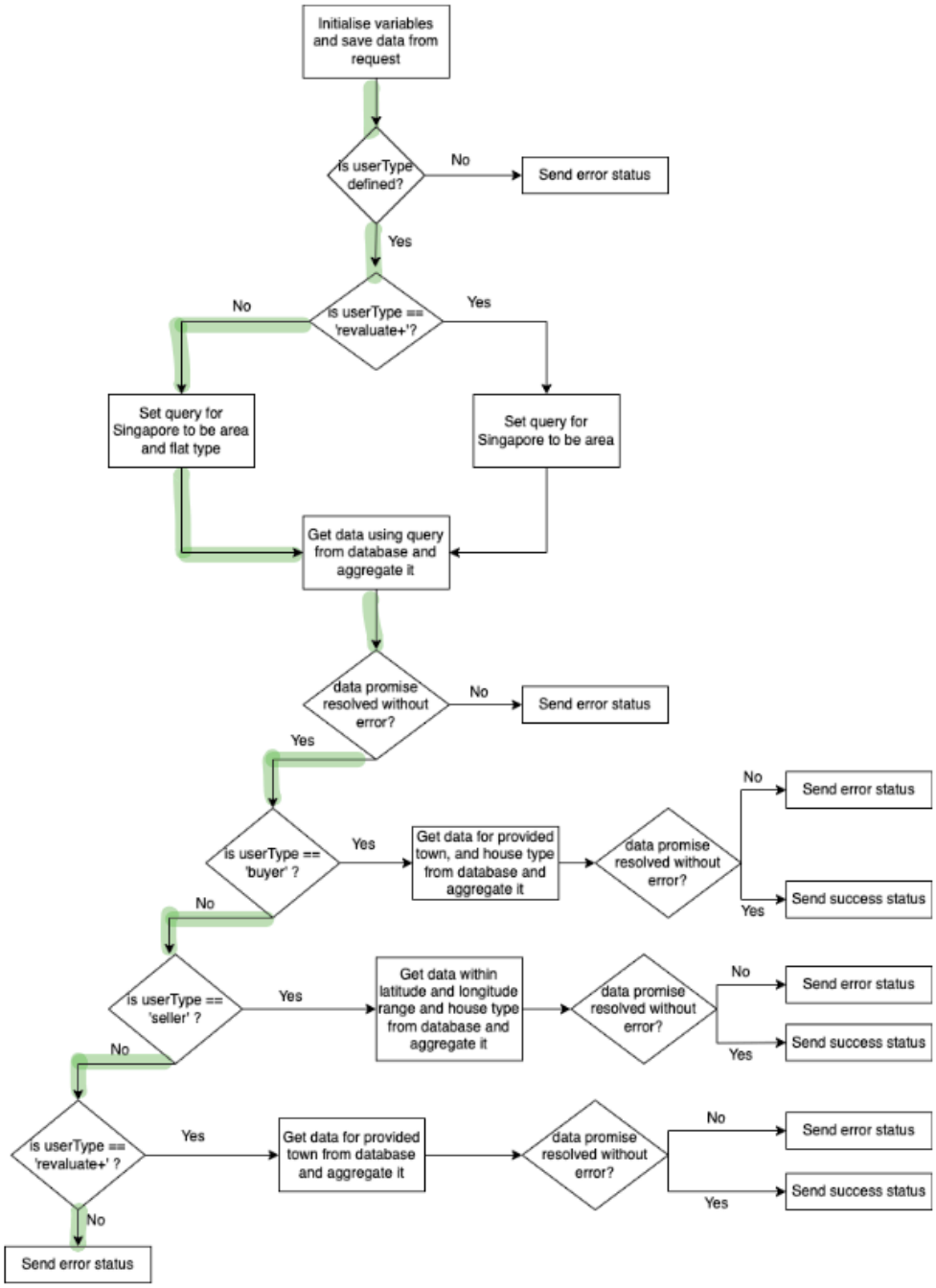
Test Code	Input	Expected Output	Actual Output	Status

WBT 2D	{query: {
	userType: ‘seller’,
	flat_type: ‘5 ROOM’,
	latitude: 1.25,
	longitude: 103.81
	}}
		Status: 200
		Data: {
		Status: ‘success’,
		…
		}
			Status: 200
			Data: {
			Status: ‘success’,
			…
			}	



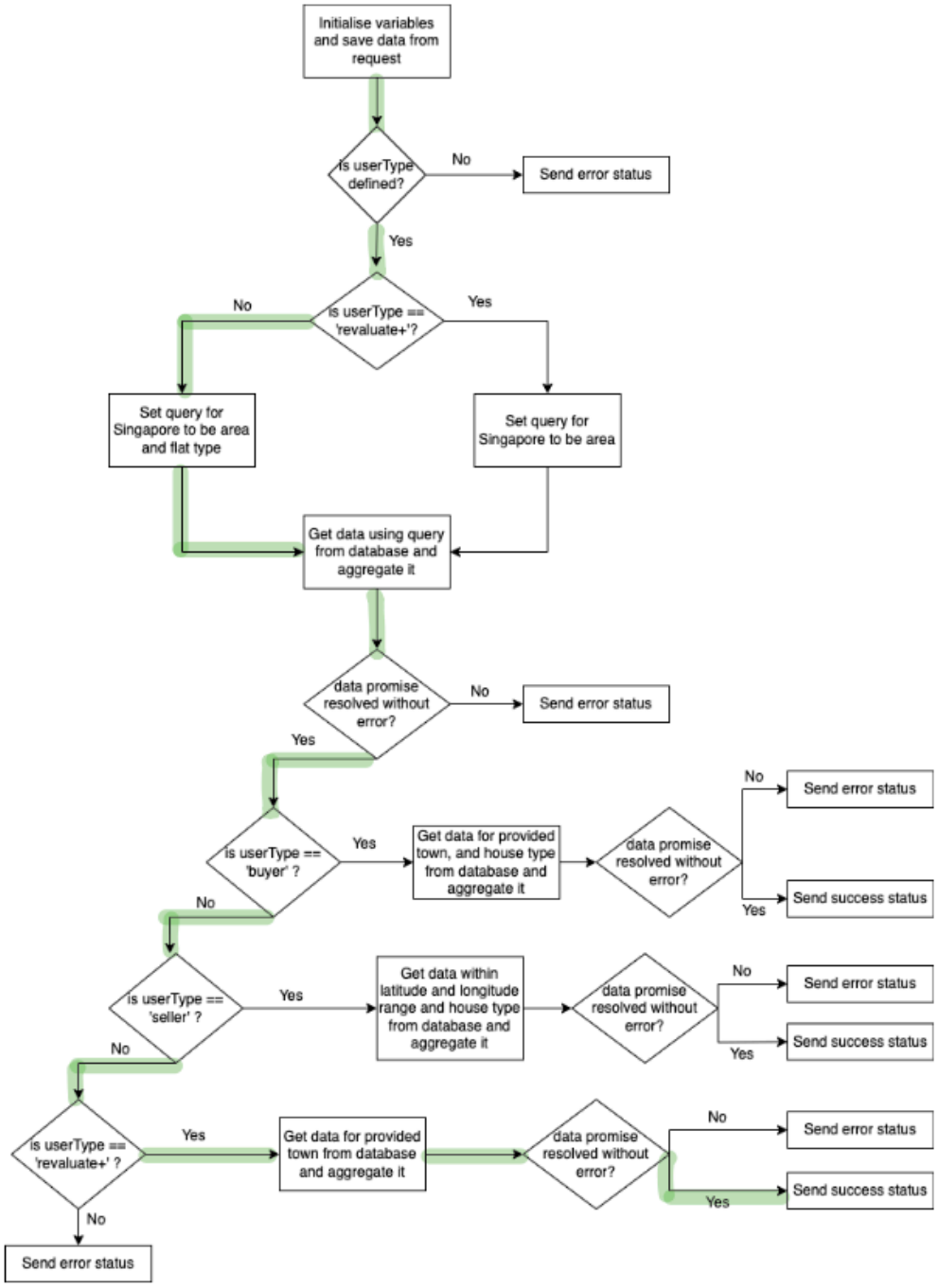
Test Code	Input	Expected Output	Actual Output	Status

WBT 2E	{query: {
	userType: ‘random’,
	flat_type: ‘5 ROOM’
	}}
		Status: 200
		Data: {
		Status: ‘error’,
		…
		}
			Status: 200
			Data: {
			Status: ‘error’,
			…
			}	

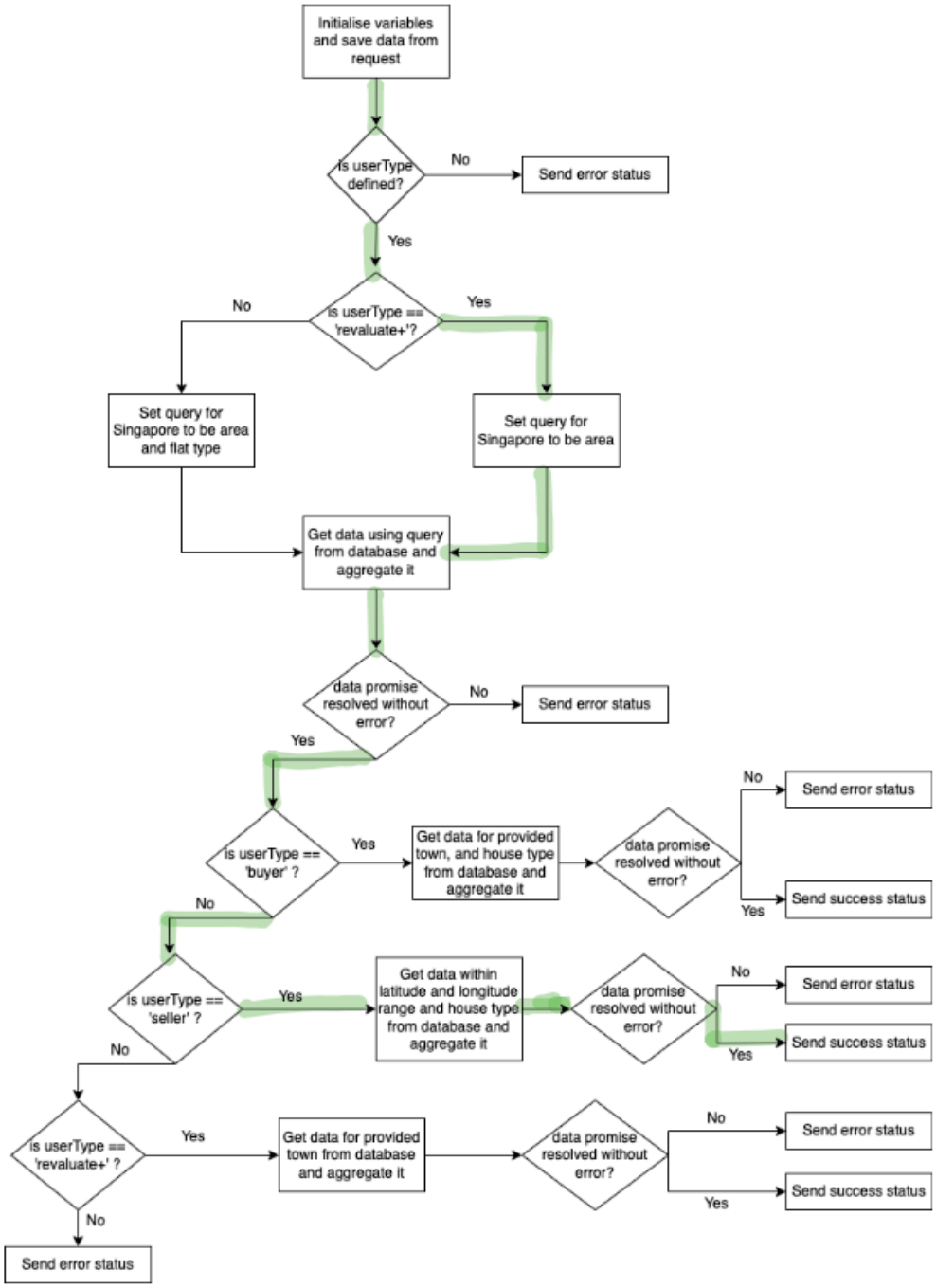


Unreachable paths:

User type cannot be both not ‘revaluate+’ and ‘revaluate+’



User type cannot be both ‘revaluate+’ and ‘seller’



User type cannot be both ‘revaluate+’ and ‘buyer’