3.3 Use Case Specification and Realisation

Use case 1: *Cleanse and Validate data*

- Goal in context:

The goal of this use case is to validate and ensure data entered into the system is cleansed and validated according to the requirements of the system if the data can be converted into valid form by cleansing.

- Primary Actors: Routed System(Primary)

- Pre-conditions:
  1. Routed System should be configured with right attribute name list and data formats.
  2. Check should be performed to decide whether data is in required form or not. (Use case: *Check if data contains required attributes and data format*)

- Post-conditions:
  1. System converts Data into valid format for further use.

- Basic Flow of Events:
  1. Data is forwarded to the module from Use case: Check if data contains required attributes and data format.
  2. Data is validated against the attribute name list and data formats.
  3. The data is checked for all attributes to see whether it has values within the permissible range.
  4. If Data passes step 2 and step 3, the data is sent back to Use case: Check if data contains required attributes and data format.

- Alternative Flows:

2a. Data is not validated.

2a1. Display error message and notify about the data formats violated and attributes missing.

3a. Data values are not within permissible range.

3a1. System normalizes data to fit within permissible range or displays error message if data format does not match with allowed formats.

4a. Data does not pass step 2 and/or step 3

4a1. Exit the system.

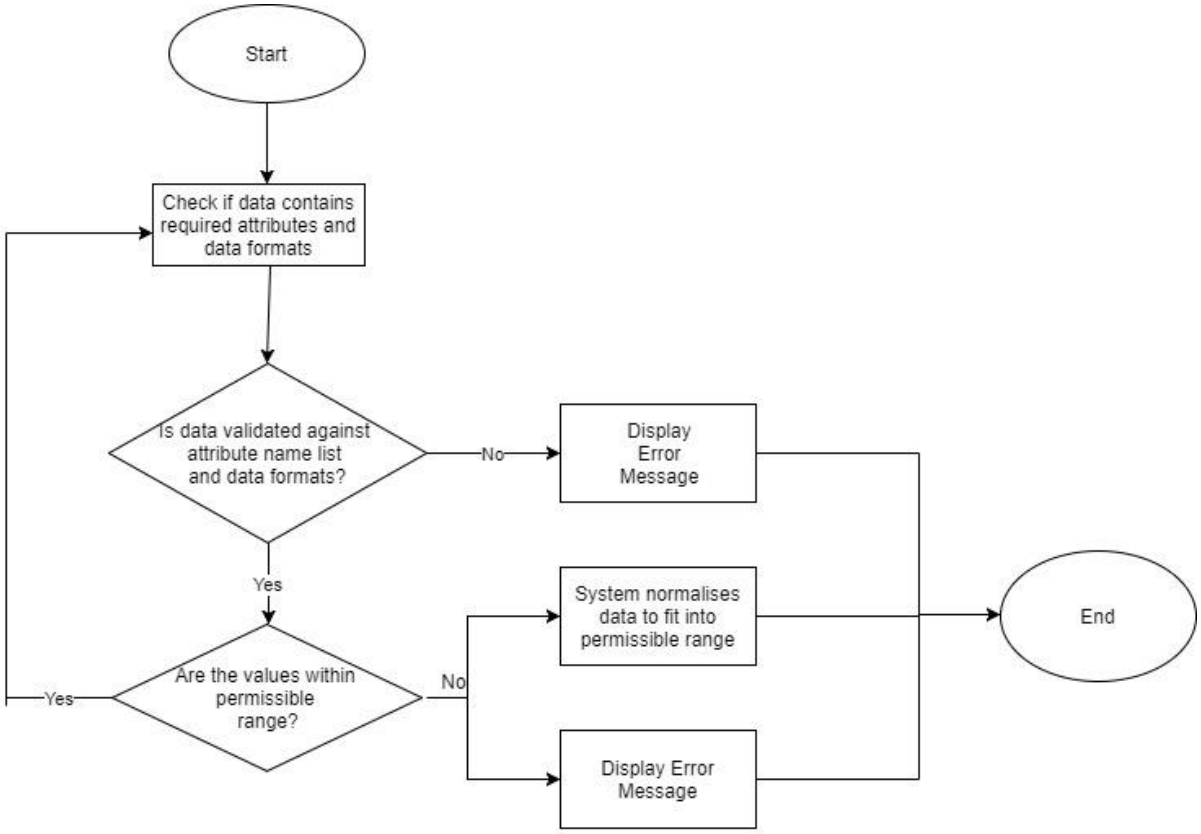


Fig 3.4 Activity Diagram - Cleanse and Validate Data

The Fig 3.4 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 1 – Cleanse and Validate Data. Refer Use Case 1 for specification.

The Fig 3.5 shows the sequence diagram corresponding to the activity diagram (Fig 3.4) and Use Case 1.

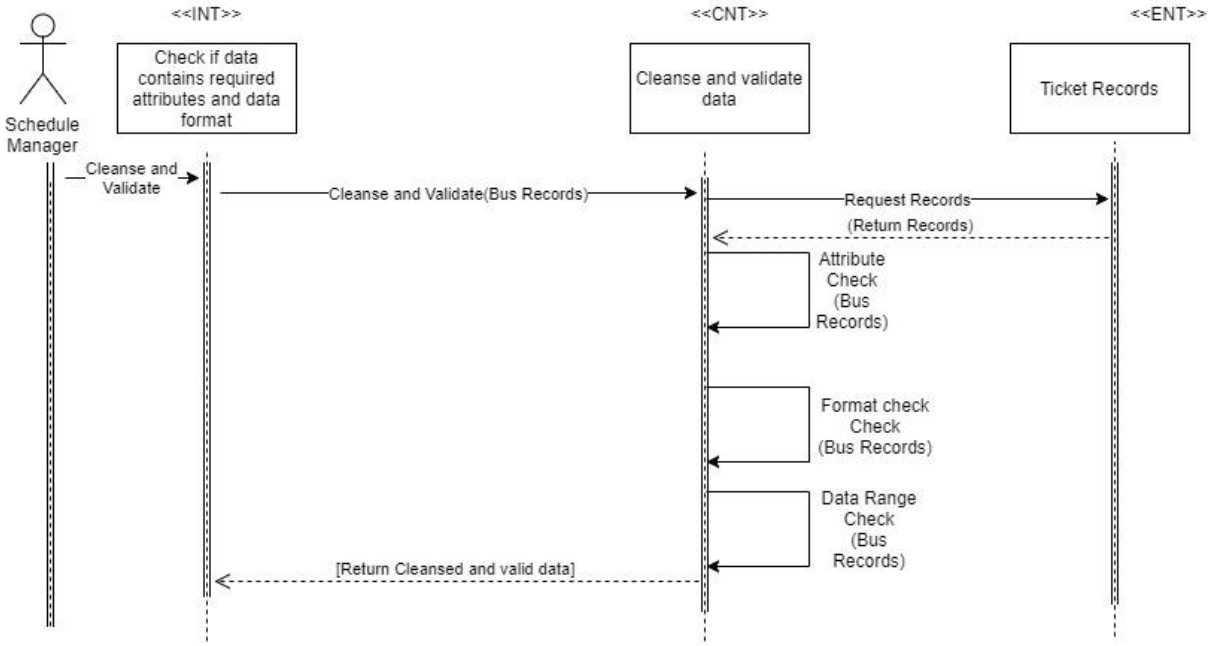


Fig 3.5 Sequence Diagram - Cleanse and Validate data

Use case 2: *Cluster data into groups by time slots*

- Goal in context:

The goal of this use case is to use a favourable clustering algorithm to form a multi-level/model based clusters, first according to the days of the week and then, according to the pre-defined 3 hour time slots.

- Primary Actors: Routed System(Primary)

- Pre-conditions:
  1. Routed System should be configured with pre-defined time slots and the favourable algorithm.
  2. Data should be checked to ensure it is in the required valid format. (Use case: Check if data contains required attributes and data format)

- Post-conditions:
  1. System groups data into useful clusters for further processing.

- Basic Flow of Events:
  1. Data is forwarded to the module from Use case: Check if data contains required attributes and data format.
  2. Data is clustered into groups based on the day of the week.
  3. Data is further clustered within the groups according to the pre-defined time slots they lie in.
  4. If Data passes step 2 and step 3, the data is sent back to Use case: Check if data contains required attributes and data format.

- Alternative Flows:

2a. Data could not be clustered according to the days.

2a1. Display error message that data cannot be used by the system.

3a. Data could not be clustered according to the time slots.

3a1. Display error message that data cannot be grouped by time slots.

4a. Data does not pass step 2 and/or step 3

4a1. Exit the system.

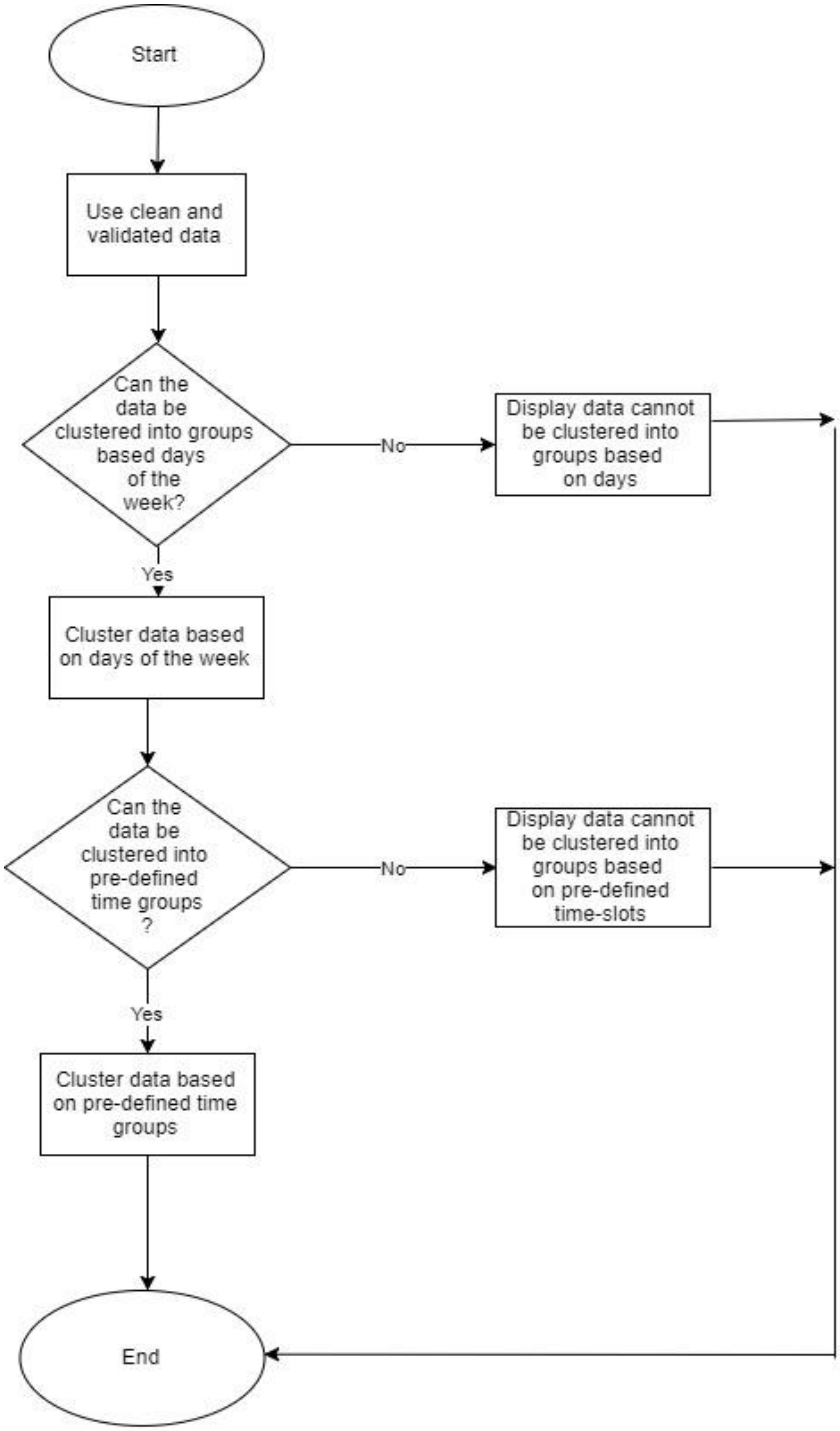


Fig 3.6 Activity Diagram - Cluster data into groups by time slots

The Fig 3.6 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 2 – Cluster data into groups by time slots. Refer Use Case 2 for specification.

The Fig 3.7 shows the sequence diagram corresponding to the activity diagram (Fig 3.6) and Use Case 2.

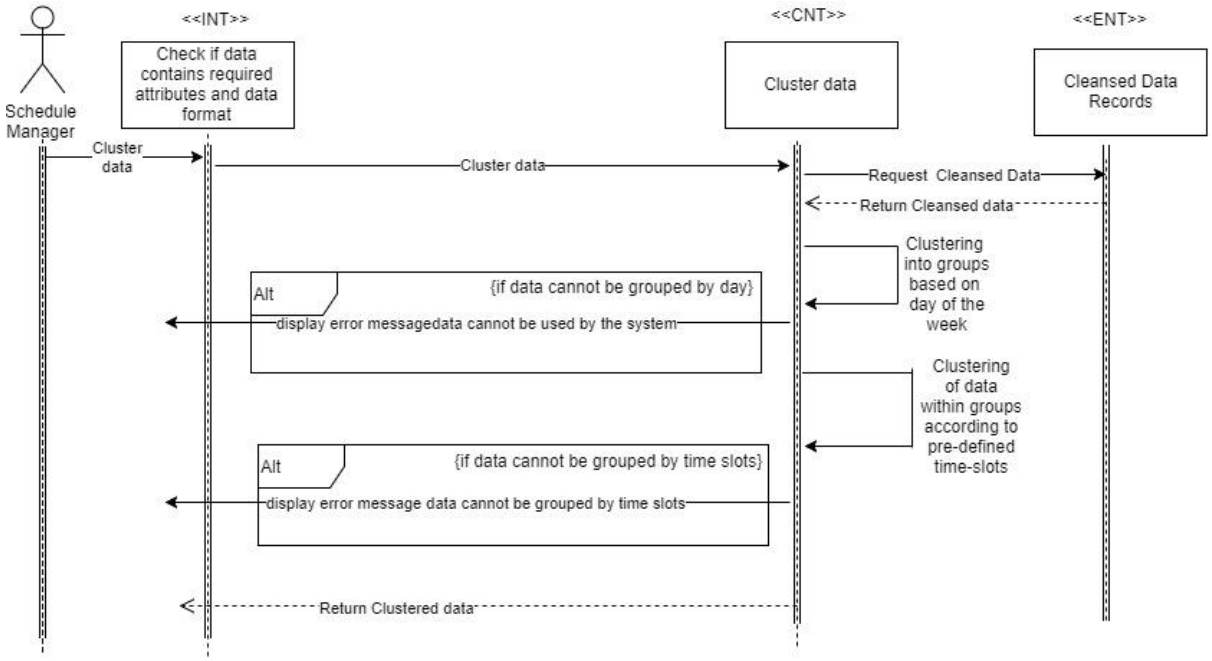


Fig 3.7 Sequence Diagram - Cluster data into groups by time slots

Use case 3: *Check if data contains required attributes and data format.*

- Goal in context:

  The goal of this use case is to perform data validation and ensure data is converted in the format required by the system if possible.

- Primary Actors: Schedule Manager(Primary), Routed System(Secondary)

- Pre-conditions:
  1. Routed System should be fed the data by the Schedule Manager.
  2. System should be configured with the required metadata and policies.

- Post-conditions:
  1. The data returned by the system should be in the form required by the system for further processing (ie. in useful form)

- Basic Flow of Events:
  1. Schedule Manager enters the Bus usage statistics data into the system as a file input.
  2. System checks the data whether it passes all the data policies and formats required by the system for further processing.
  3. If data passes step 2, do Use Case: Cluster into groups by time slots.
  4. If Use Case: Cluster into groups by time slots returns data, output data to Schedule Manager as a file.

- Alternative Flows:

2a. Data does not pass the data policies and formats required by the system.

2a1. Perform Use Case: Cleanse and validate data.

2a2.If Use Case: Cleanse and validate data returns data, resume from step 3.

2a3. If Use Case: Cleanse and validate data does not return data, notify Schedule manager to submit appropriate data.

4a. Use Case: Cluster into groups by time slots does not return data.

4a1. Notify Schedule Manager about the error and exit the system.

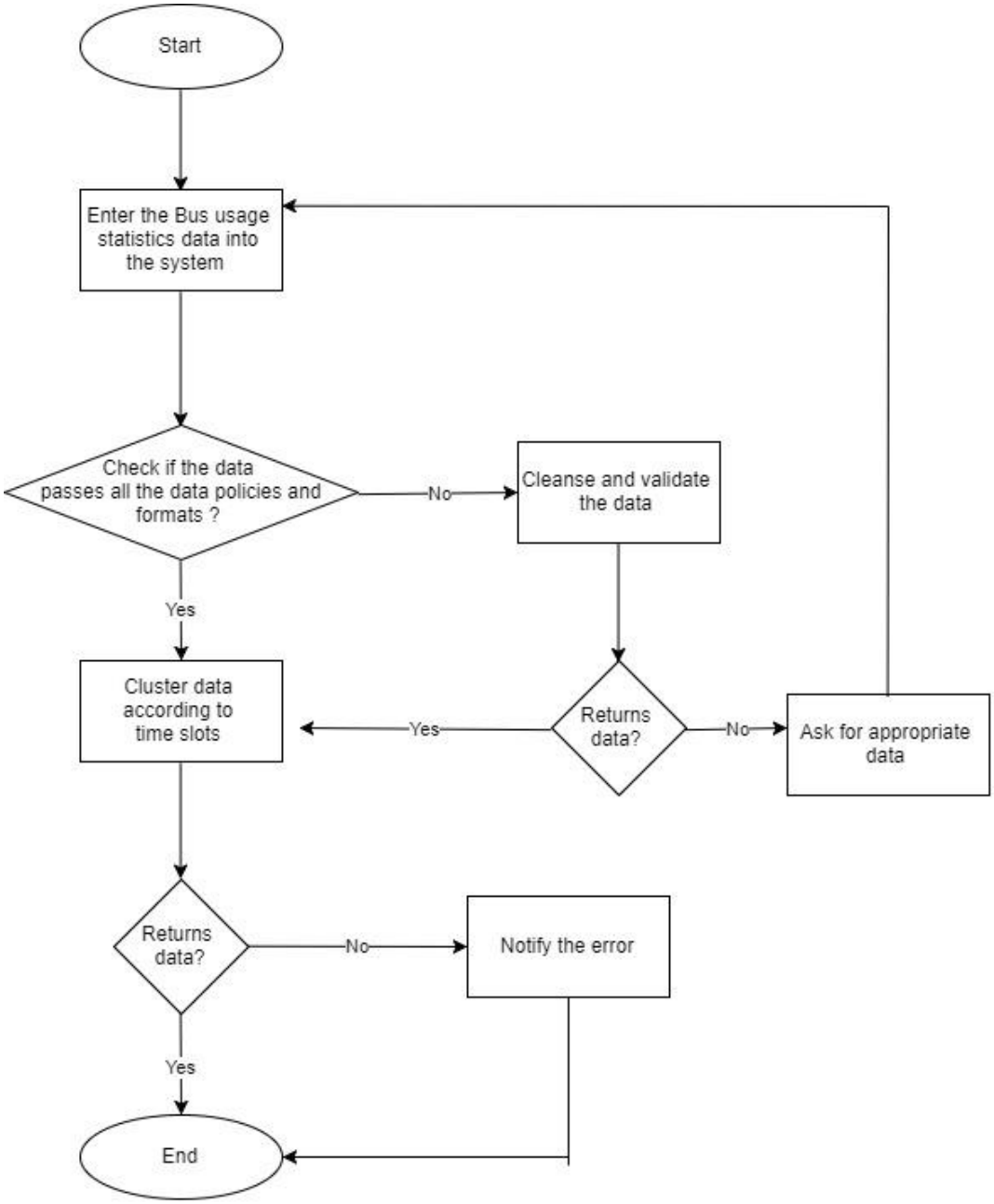


Fig 3.8 Activity Diagram - Check if data contains required attributes and data formats

The Fig 3.8 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 3 – Check if data contains required attributes and data format. Refer Use Case 3 for specification.

The Fig 3.9 shows the sequence diagram corresponding to the activity diagram (Fig 3.8) and Use Case 3.

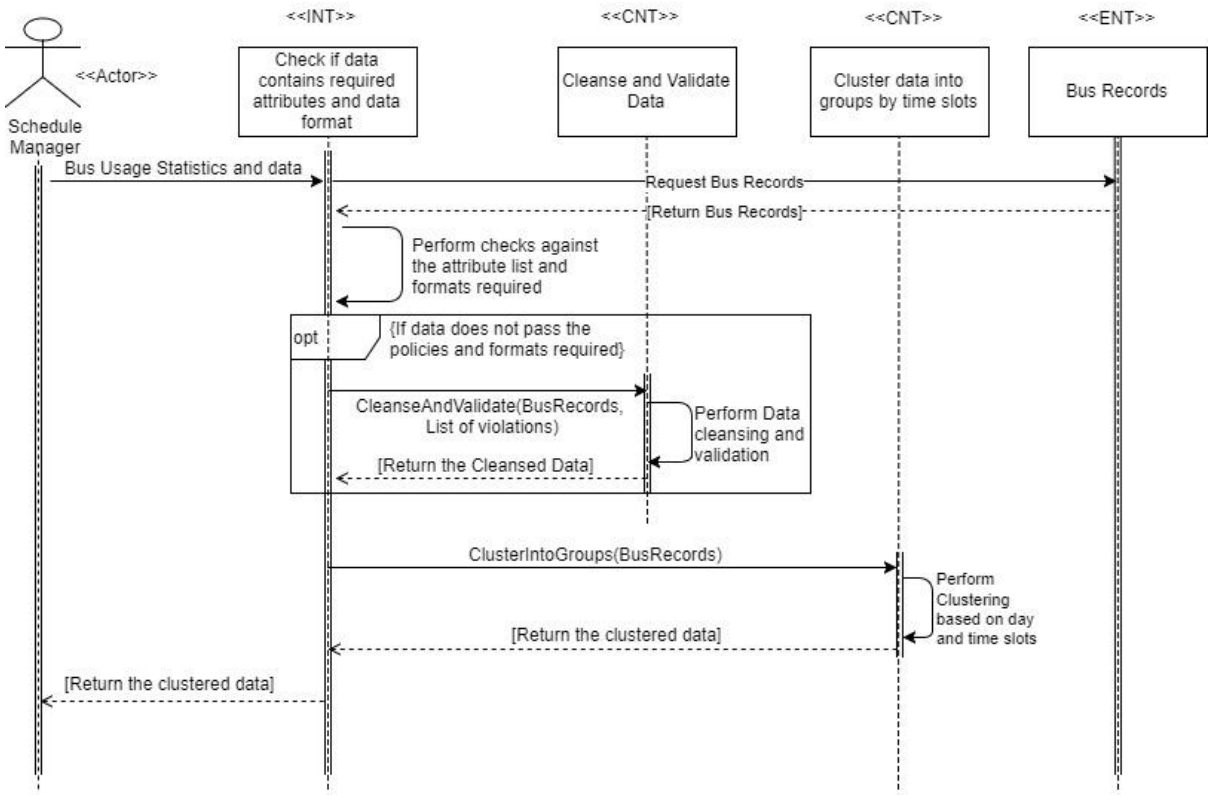


Fig 3.9 Sequence Diagram - Check if data contains required attributes and data format

Use case 4: *Allocate buses to routes*

- Goal in context:

The goal of this use case is to use a favourable predictive/fleet size allocation algorithm to allocate optimal number of buses to each route.

- Primary Actors: Routed System(Primary)

- Pre-conditions:
  1. Routed System should be fed data already cleansed and clustered into groups. (Use case: Check if data contains required attributes and data format)

- Post-conditions:
  1. System generates optimal Bus allocations for each route.

- Basic Flow of Events:
  1. Data is forwarded to the module from Use case: Generate optimal bus utilization schedule.
  2. Apply the bus allocation algorithmby using the forwarded data as input.
  3. Send the bus allocations for each route to the Use case: Generate optimal bus utilization schedule.

- Alternative Flows:

2a. Bus allocation algorithm encounters an error.

2a1. Retry the bus allocation algorithm 3 times.

2a2. Display error message and exit the system if retries fail.

3a. Data could not be sent back to the Use case: Generate optimal bus utilization schedule.

3a1. Retry to send data 3 times.

3a2. Display error message and exit the system if retries fail.

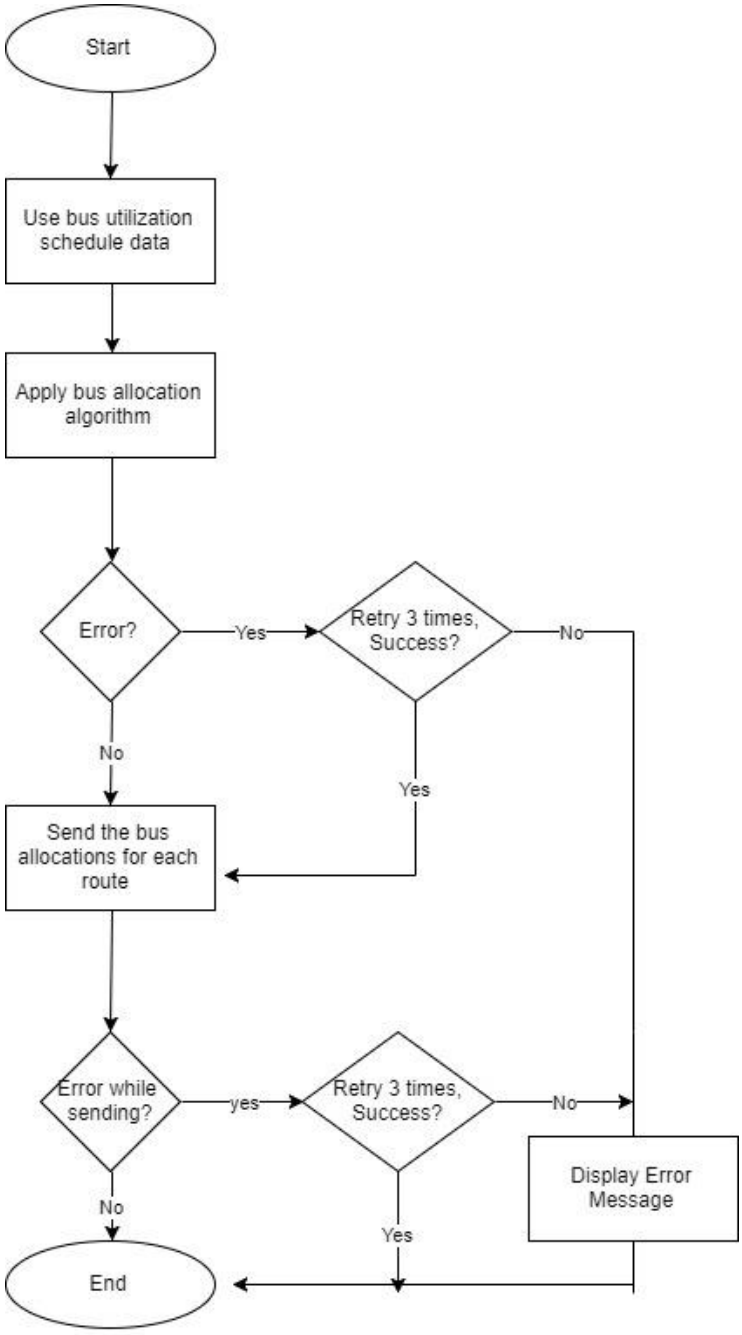


Fig 3.10 Activity Diagram - Allocate Buses to Routes

The Fig 3.10 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 4 – Allocate buses to routes. Refer Use Case 4 for specification.

The Fig 3.11 shows the sequence diagram corresponding to the activity diagram (Fig 3.10) and Use Case 4.

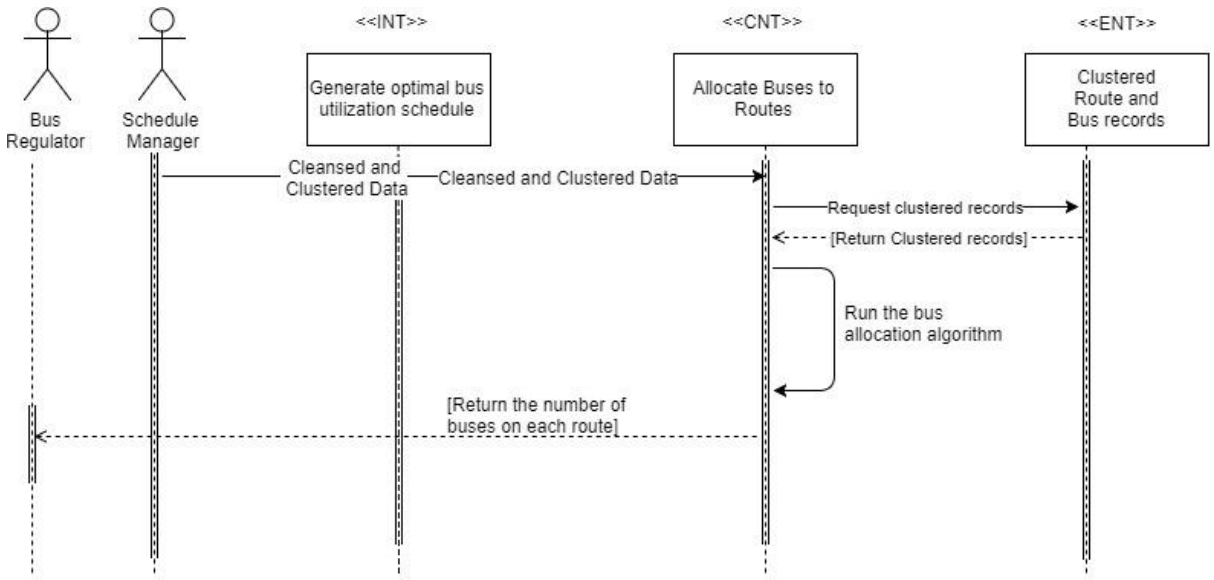


Fig 3.11 Sequence Diagram - Allocate Buses to Routes

Use case 5: *Predict bus schedules for each route*

- Goal in context:

The goal of this use case is to use a favourable predictive algorithm to generate bus schedules for each route.

- Primary Actors: Routed System(Primary)

- Pre-conditions:
  1. Routed System should be fed data already cleansed and clustered into groups. (Use case: Check if data contains required attributes and data format)
  2. Routed System should have information about the number of buses allocated per route. (Use case: Allocate buses to routes)

- Post-conditions:
  1. System generates optimal Bus schedules for each route.

- Basic Flow of Events:
  1. Cleansed and clustered data file and the bus allocation information is forwarded to the module from Use case: Generate optimal bus utilization schedule.
  2. Apply the predictive algorithm to generate schedule for each route by using the forwarded data as input.
  3. Send the bus schedules for each route generated in step 2 to the Use case: Generate optimal bus utilization schedule.

- Alternative Flows:

2a. Predictive algorithm encounters an error.

2a1. Retry the algorithm 3 times.

2a2. Display error message and exit the system if retries fail.

3a. Data could not be sent back to the Use case: Generate optimal bus utilization schedule.

3a1. Retry to send data 3 times.

3a2. Display error message and exit the system if retries fail.

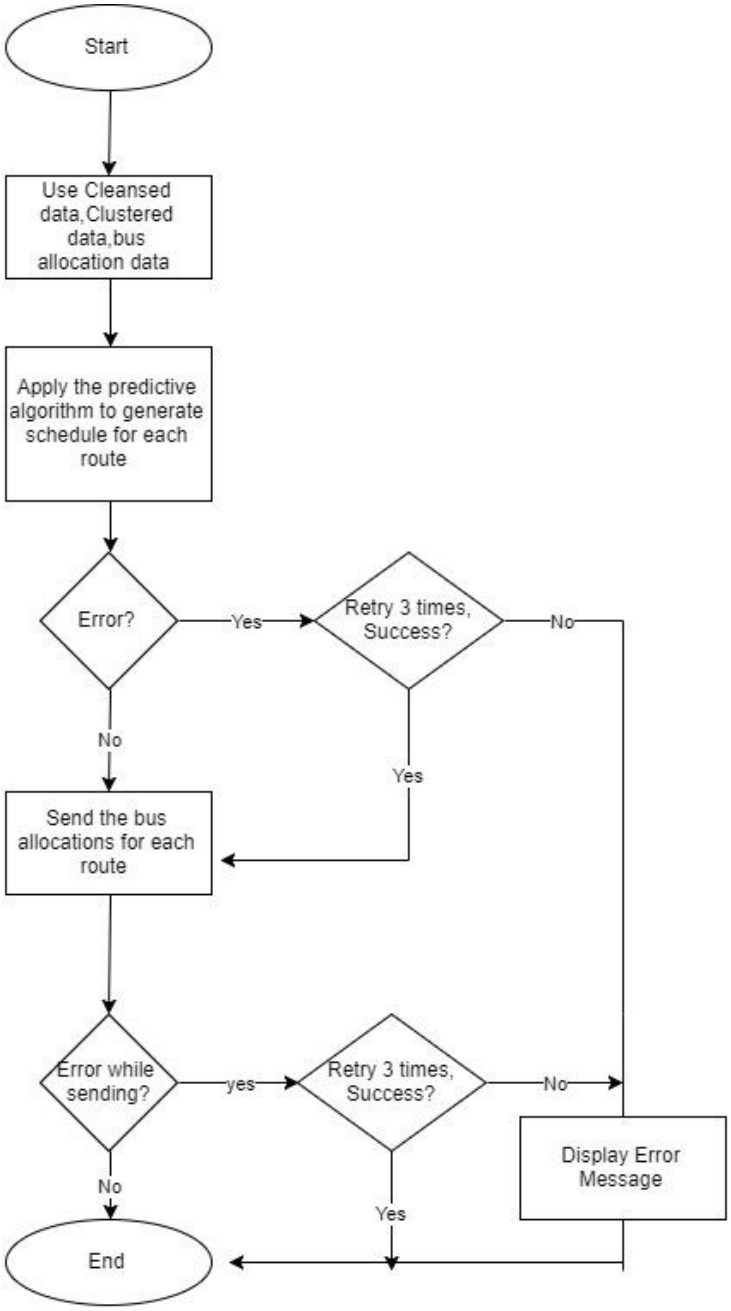
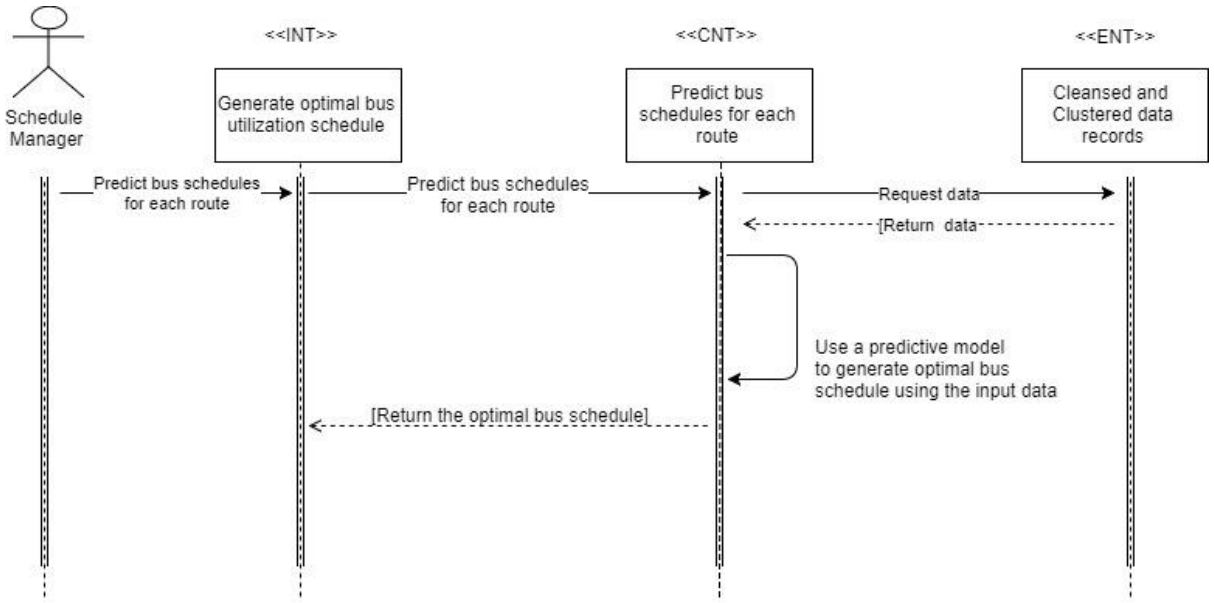


Fig 3.12 Activity Diagram - Predict Bus schedules for each route

The Fig 3.12 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 5 – Predict bus schedules for each route. Refer Use Case 5 for specification.

The Fig 3.13 shows the sequence diagram corresponding to the activity diagram (Fig 3.12) and Use Case 5.



Fif 3.13 Sequence Diagram - Predict bus schedules for each route

Use case 6: *Generate optimal bus utilization schedule*

- Goal in context:

The goal of this use case is act as an interface which co-ordinates between general allocation and predictive algorithms, and generate the bus utilization and scheduling tables.

- Primary Actors: Schedule Manager(Primary), Bus Regulator(Primary), Routed System(Secondary)

- Pre-conditions:
  1. Routed System should be fed data already cleansed and clustered into groups. (Use case: Check if data contains required attributes and data format)

- Post-conditions:
  1. System generates bus utilization and scheduling tables for each route.

- Basic Flow of Events:
  1. Cleansed and clustered data file is fed to the module by the Schedule Manager in the form of a file.
  2. To generate bus allocation per route, do Use case: Allocate buses to routes and send the data file as input.
  3. If Use case: Allocate buses to routes returns data, then output the bus allocation tables ie. Number of buses per route to the Bus regulator as a file and go to step 4.
  4. To generate bus schedules for each route, do Use case: Predict bus schedules for each route and send the clustered data along with the bus allocation information obtained in step 2 as input.
  5. If step 4 returns data, then send the bus schedules for each route generated in step 4 to the Schedule Manager as a file.

- Alternative Flows:

3a.Use case: Allocate buses to routes does not return data.

3a1. Retry Use case: Allocate buses to routes 3 times.

3a2. if step 3a1 returns data, go to step 4 else go to step 3a3.

3a3. Notify Schedule Manager and Bus regulator about the error and exit the system.

5a.Use case: Predict bus schedules for each route does not return data.

5a1. Retry Use case: Predict bus schedules for each route 3 times.

5a2.if step 5a1 returns data, go to step 5 else go to step 5a3.

5a3. Notify Schedule Manager about the error and exit the system.

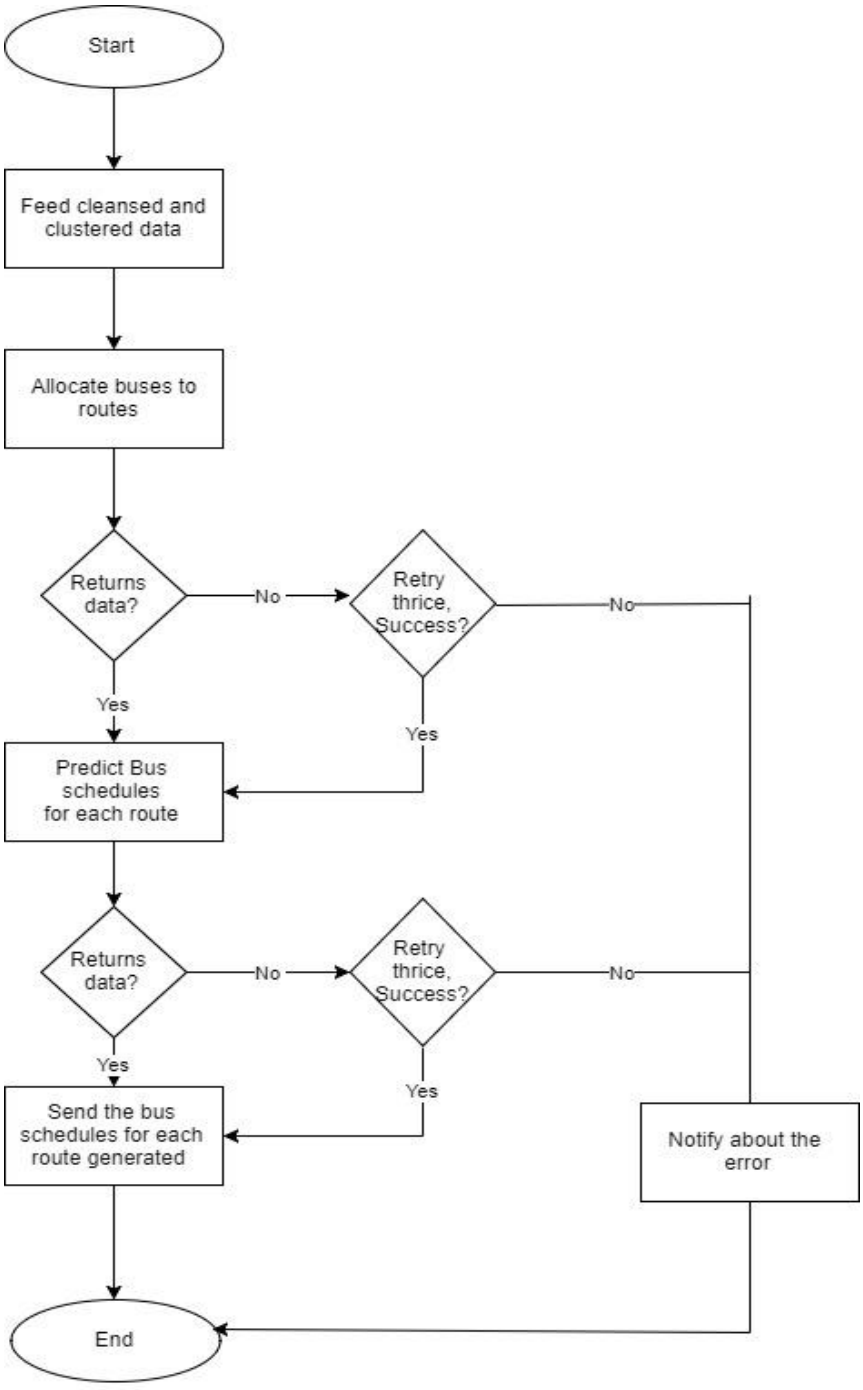


Fig 3.14 Activity Diagram - Generate Optimal Bus Utilization schedule

The Fig 3.14 shows the activity diagram with the basic flow as well as the alternate flow for Use Case 6 – Generate optimal bus utilization schedule. Refer Use Case 6 for specification.

The Fig 3.15 shows the sequence diagram corresponding to the activity diagram (Fig 3.14) and Use Case 6.

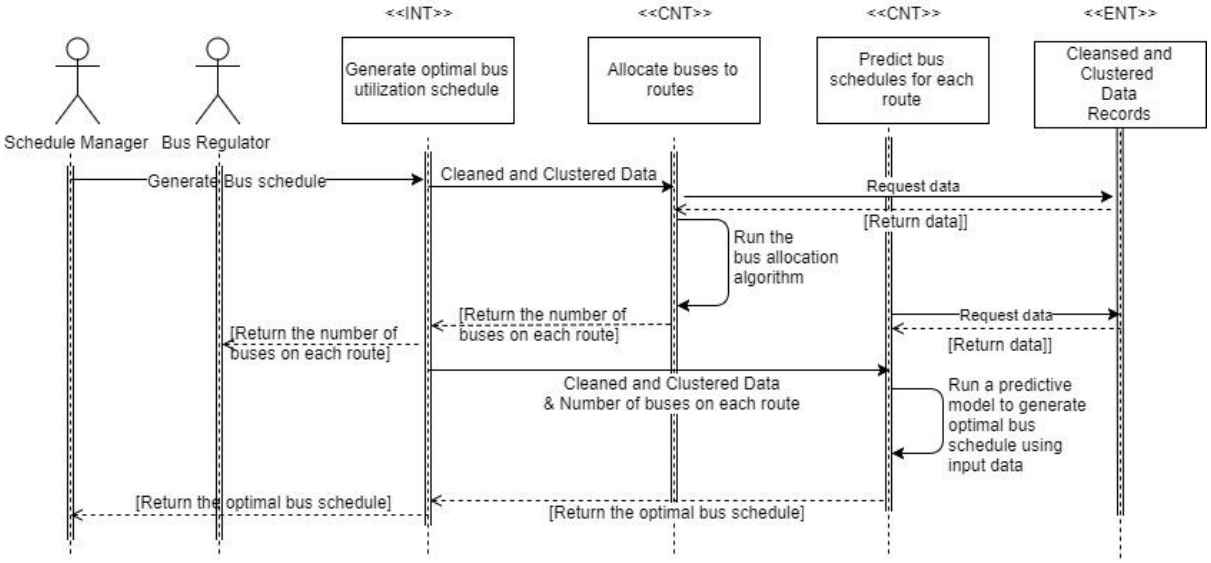


Fig 3.15 Sequence Diagram - Generate optimal bus utilization schedule