Cosmosys
UNIVERSAL OMNITECH

Ice Cream Manager [ICM]

SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Version No. v1.0

Project Document Revision History

VERSION	DATE	REVISION AUTHOR	DESCRIPTION OF REVISION
0.1	2016-2-19	Cosmosys Team	Initial content generation.
0.2	2016-2-20	Marc King	Added Analysis Metrics section 8.3.
0.3	2016-3-3	Rodney Lewis	Added draft of sections 3.1, 3.1.2, 3.1.3
0.4	2016-3-6	Jacob Vacheresse	Added section seven validation draft.
0.5	2016-3-12	Rodney Lewis	Updated sections 3.1, 3.1.2, 3.1.3
0.6	2016-3-12	Aly Lakhani	Added sections 1.1 and 1.3
0.7	2016-3-12	Jacob Vacheresse	Added section 5.1
0.8	2016-3-13	Fan Zhang	Added section 2.1, 2.2, 2.3
0.9	2016-3-13	Marc King	Updated traceability matrix, analysis metrics, and product strategy.
0.10	2016-3-14	Marc King	Added UML class diagram, tables of figures.
0.11	2016-3-15	Aly Lakhani	Updated wording of section 1.1 and 1.3. Added sections 4.2, 4.2.1, 4.2.2, and 4.2.3
0.12	2016-3-15	Marc King	Added state transition diagrams to section 5.2
0.13	2016-3-17	Marc King	Added project manager-created metrics.
0.14	2016-3-17	Marc King	Copied traceability spreadsheet, added captions to sequence and communication diagrams.
1.0	2016-3-17	Cosmosys Team	Baselined.

Table of Contents

Table of Figures

1.0 Introduction

This Software Requirements Specification Document details the functionality, methodology, and estimates for the Ice Cream Manager software product. Ice Cream Manager is a product which allows the manager of an ice cream truck fleet to efficiently and easily manage their inventories, sales history, and truck routes.

1.1 Goals and Objectives

The goal of Ice Cream Manager is to provide a convenient one-stop solution for the majority of an ice cream manager's needs. This application is designed to supplement the job of the manager by adding inventory, tracking sales of each truck on their assigned route, as well as add trucks and/or routes. The manager will also be able to track freshness of the product and update the inventory in case of spoiled ice cream, to ensure quality.

1.2 Statement of Scope

Ice Cream Manager will be a graphical interface that will employ a system that will increase efficiency for the role of managing ice cream trucks. This application will allow the user to keep track of inventory levels, calculate sales though inventory levels, and manage their fleet of trucks. Inventory levels will be able to be displayed and modified for each truck. Each truck has an assigned route that may be manually edited by the user, or changed via a batch file. Additionally, each route may also have the zones which comprise it modified. If a truck is assigned to a route, the fuel efficiency of those trucks will be calculated and displayed. Trucks will have an assigned driver that will be able to report voting results of a given route. Shipments of ice cream will have an expiration date associated with it to prevent waste, older ice cream will be sold first or disposed of if the date passes. This software will support multiple languages.

1.3 Software Context

Ice Cream Manager strives to make the experience of the manager streamlined and centralized in a technology software, with a user-friendly GUI, in five different languages, with the ability to support languages read from right to left, along with the ability for the end-user to add their own languages.

1.4 Major Constraints

- End of the semester imposes a dead drop date for this application.
- Due to the team's responsibility to other classes, time is a limited resource.
- Change will inevitably occur, so we have to be mindful of how in-depth our own functionalities go.
- This software will be developed using C#, .NET, and WinForms.
- The Model-View-Controller paradigm will be used when designing this software package.

2.0 Usage Scenario

This section provides a usage scenario for the software. It organized information collected during requirements elicitation into use-cases.

2.1 User Profiles

- Manager: The manager of ice cream truck business. This type of user has the highest level of access to the system. Manager can directly manipulate modification and view functions for all the use cases. Manager also has responsibility to monitor the all functionalities run well in this software.
- Modifier: The system that edit information from the user. Manager updates information in system through modifier. In some use cases, modifier can directly access to view and edit information.
- View: The system that displays information to the user. For each use case, the view functionality will be associated with edit functionality and will always display the updated information to user.
- Control: The system that merges the batch file with the database. It will also process the batch files that need to be operated to reach manager's specific purpose. The system will take care of sales and waste record as well.

2.2 Use-Cases

- UC01 – Modify Inventory
- UC02 – Manage Sales
- UC03 – Modify Route
- UC04 – Modify Truck
- UC05 – Process Batch File
- UC06 – Modify Item
- UC07 – Modify Driver
- UC08 – Modify Settings
- UC09 – Modify Voting
- UC10 – Modify Preset
- UC11 – View Fuel Usage

2.3 Special Usage Considerations

- This software builds for Windows 7 and above operating system, so make sure your operating system meets the requirement before installing the software.
- The user of this software has to know one of the setting language provided to run the whole system properly.

3.0 Data Model and Description

3.1 Data Description

DATA OBJECT	DESCRIPTION
Item	A unit of ice cream
Sale	The result of an item being sold
Route	A collection or ordered zones
Zone	A location consisting of a city and a subset of the city that is mapped to a number
Truck	Object that holds items and travels routes
Poll	A collection of votes that are used to make changes to future inventory
Driver	Object that uses truck and drives routes
Preset	A pre-determined inventory created by the user

Table 1: Data Objects

3.1.1 Data Objects

- Item
  - Type: The price and name of a unit of ice cream
  - Item number: Unique reference number to an item
  - Expiration date: Date used to determine the freshness of a unit of ice cream
- Sale
  - Sale number: Unique reference number of a sale
  - Sale date: Date that the sale was made
- Route
  - Route number: Unique reference number of a route
- Zone
  - Zone number: A subset of a city that is mapped to a number
  - City: Name of a location
  - Distance: How long a zone is in miles
- Truck
  - Truck number: Unique reference number to a truck
  - Item capacity: The total number of items that a truck can have in its inventory
  - Fuel capacity: The total amount of fuel, in gallons, that a truck can hold
  - Fuel amount: The remaining amount of fuel, in gallons, that a truck has
  - Fuel consumption: Fuel efficiency in miles per gallon
- Poll
  - Poll number: Unique reference number for a poll
  - Poll name: User defined name for a poll
  - Total votes: The number of votes that were processed for a poll
- Driver
  - Driver number: Unique reference number for a driver
  - Driver name: Name of a driver

- Preset
  - Preset number: Unique reference number for a preset
  - Preset name: User defined name for a preset
  - Start date: The date that a preset will take effect
  - End date: The date that a preset will no longer be in effect

3.1.2 Relationships

Item

RESPONSIBILITY	COLLABORATOR
Add items	
Delete items	
Modify existing items	
Display item information	

Sale

RESPONSIBILITY	COLLABORATOR
Track item sales	Item
Calculate profit	Item
Calculate profit margin (Item sale price – Item production cost)	Item
Display sales by route	Route
Display sales by truck	Truck
Display sales by driver	Driver

Route

RESPONSIBILITY	COLLABORATOR
Add route	
Delete route	
Set zones	Zone
Set truck	Truck
Set fuel cost	
Apply inventory presets	Preset
Display assigned truck	Truck
Display route composition	Zone
Display inventory preset	
Display fuel cost	

Zone

RESPONSIBILITY	COLLABORATOR
Add zone	
Delete zone	
Modify existing area	
Display area	

Truck

RESPONSIBILITY	COLLABORATOR
Add truck	
Delete truck	
Set truck inventory	
Set route	Route
Set driver	Driver
Set fuel consumption	
Set fuel level	
Apply inventory preset	Preset
Calculate fuel usage	
Calculate fuel efficiency	Route
Display route	Route
Display driver	Driver
Display truck inventory	Item
Display fuel	

Poll

RESPONSIBILITY	COLLABORATOR
Create poll	Zone
Display results of poll	Zone

Driver

RESPONSIBILITY	COLLABORATOR
Add driver	
Remove driver	
Set number	
Set name	
Set salary	
Set truck	Truck
Calculate labor costs	
Display number	
Display name	
Display salary	
Display truck	

Preset

RESPONSIBILITY	COLLABORATOR
Create preset	Item
Delete preset	
Apply preset	Truck, Route
Modify preset	Item

3.1.3 Complete Data Model

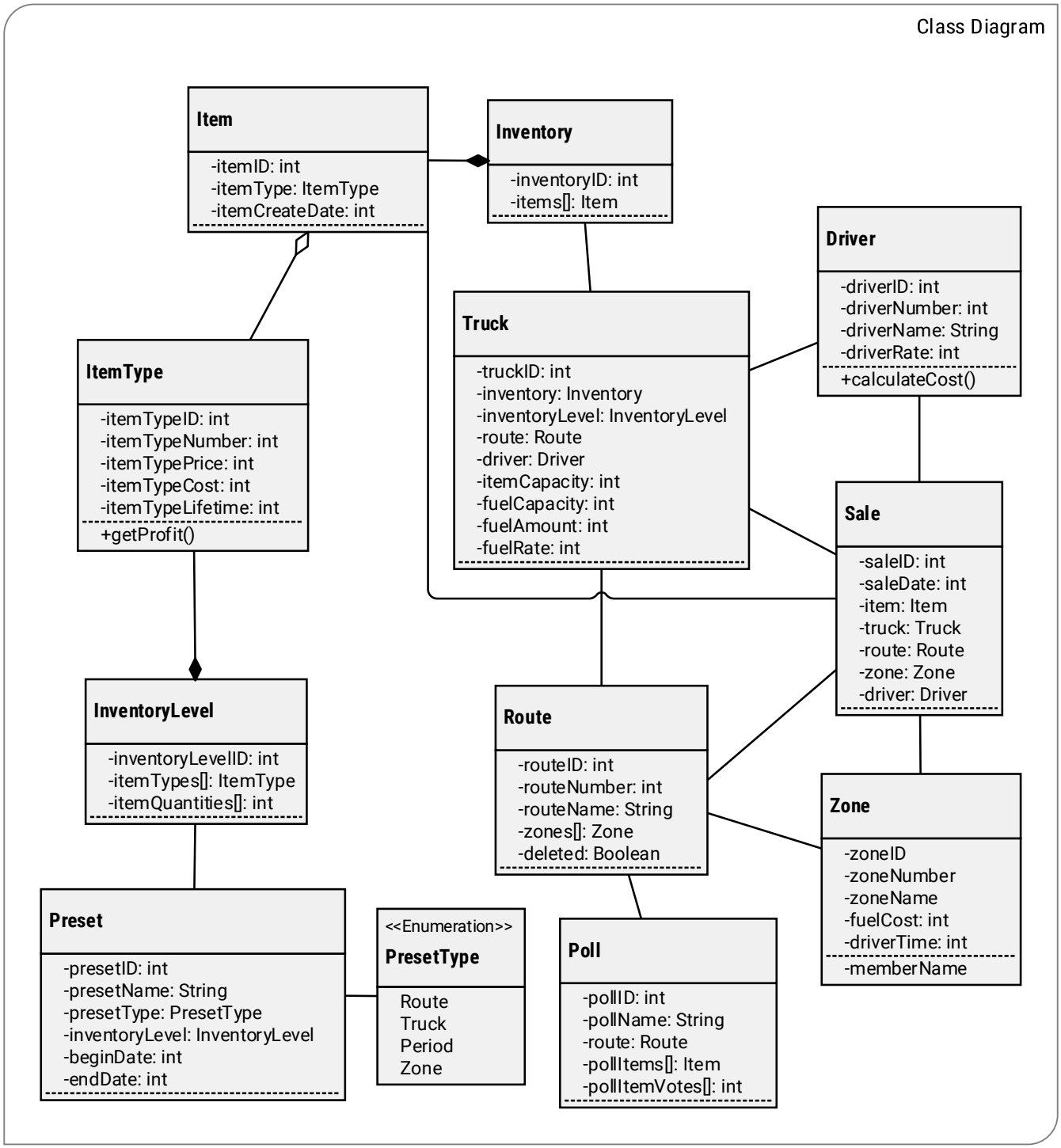

Diagram 1: UML Class Diagram

3.1.4 Data Dictionary

The Data Dictionary for the Ice Cream Manager can be found in the Microsoft Excel spreadsheet file named: *Data Dictionary.xlsx*

4.0 Functional Model and Description

Description of major software functions along with UML Use Case, sequence, and communication diagrams.

4.1 Use Cases

ID	NAME	DOCUMENT FILE
UCSD	**Use Case Summary Document**	`Use Case Summary Document.pdf`
UC01	**Modify Inventory**	`Use Case – Modify Inventory.pdf`
UC02	**Manage Sales**	`Use Case – Manage Sales.pdf`
UC03	**Modify Route**	`Use Case – Modify Route.pdf`
UC04	**Modify Truck**	`Use Case – Modify Truck.pdf`
UC05	**Process Batch File**	`Use Case – Process Batch File.pdf`
UC06	**Modify Item**	`Use Case – Modify Item.pdf`
UC07	**Modify Driver**	`Use Case – Modify Driver.pdf`
UC08	**Modify Settings**	`Use Case – Modify Settings.pdf`
UC09	**Modify Voting**	`Use Case – Modify Voting.pdf`
UC10	**Modify Preset**	`Use Case – Modify Preset.pdf`
UC11	**View Fuel Usage**	`Use Case – View Fuel Usage.pdf`

Table 2: Use Case Document Filenames

4.2 Software Interface Description

Ice Cream Manager will be a graphical user interface with an initial start-up screen from which the modify can navigate to tabs that will keep track of, and modify overall inventory, inventory levels for each truck, as well as the freshness of the inventories, add new items, calculate sales through inventory levels, assign routes to trucks and view the fuel efficiency of the trucks assigned to that route, update truck driver data and view polling results of a given route. The user will also be able to add or modify inventories in three ways: manually, by file upload, or by creating and using user presets, which will have its own tab, from where the user can also schedule the start and stop dates of the given preset. Ice Cream Manager will also interface with SQLite for database usage. The user will also be able to modify settings such as language or currency.

4.2.1 External Machine Interfaces

Ice Cream Manager will need a system running Windows 7 or higher to run.

4.2.2 External System Interfaces

Ice Cream Manager interfaces with SQLite for its database

4.2.3 Human Interface

Ice Cream Manager will have a graphical user interfaces, with tabs that will keep track of, and modify overall inventory, inventory levels for each truck, as well as the freshness of the inventories, add new items, calculate sales through inventory levels, assign routes to trucks and view the fuel efficiency of the trucks assigned to that route, update truck driver data and view polling results of a given route. The user will also be able to add or modify inventories in three ways: manually, by file upload, or by creating and using user presets, which will have its own tab, from where the user can also schedule the start and stop dates of the preset.

4.3 Sequence Diagrams

1. Modify Inventory

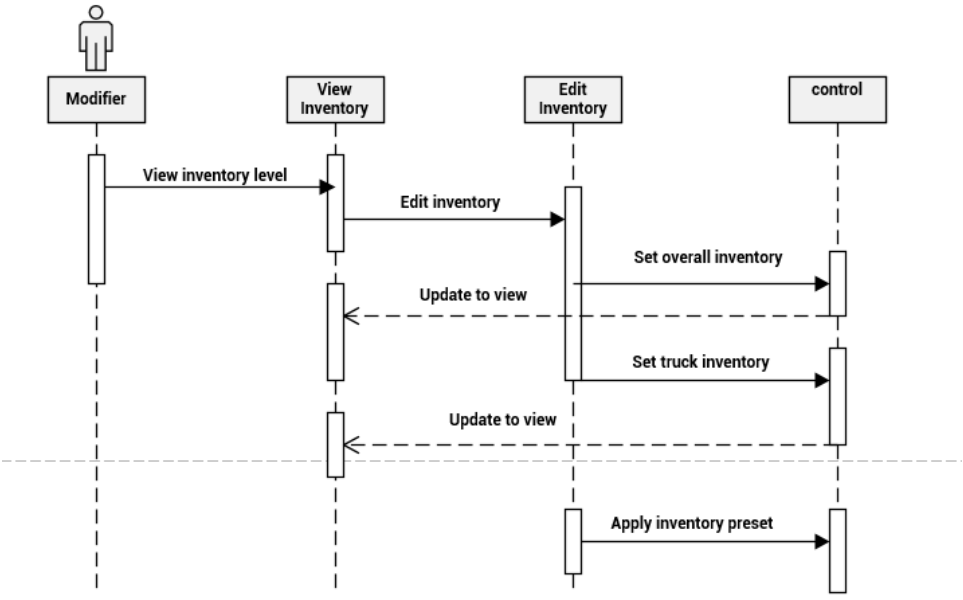

Diagram 2: Modify Inventory Sequence Diagram

2. View Sales

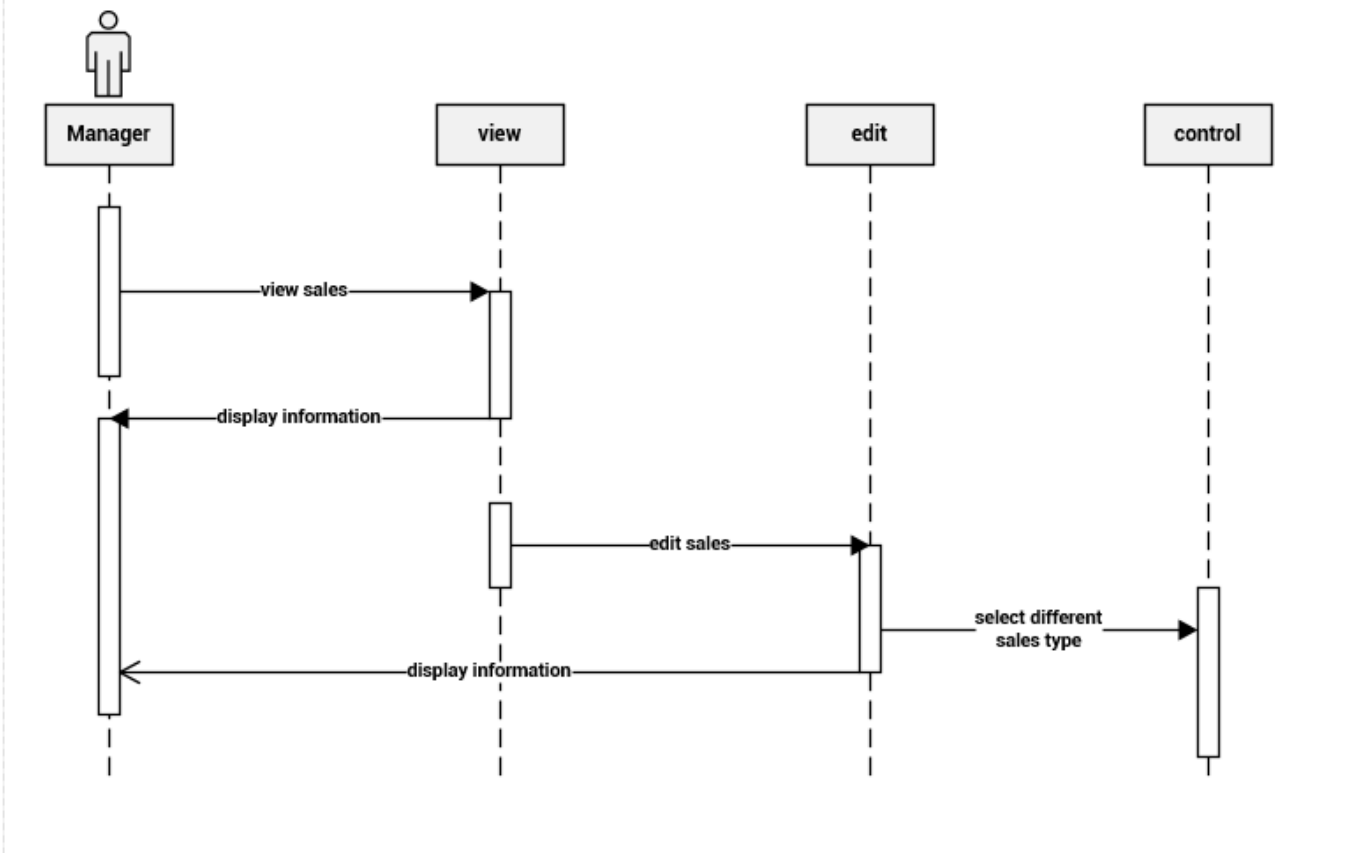

Diagram 3: Manage Sales Sequence Diagram

3. Modify Route

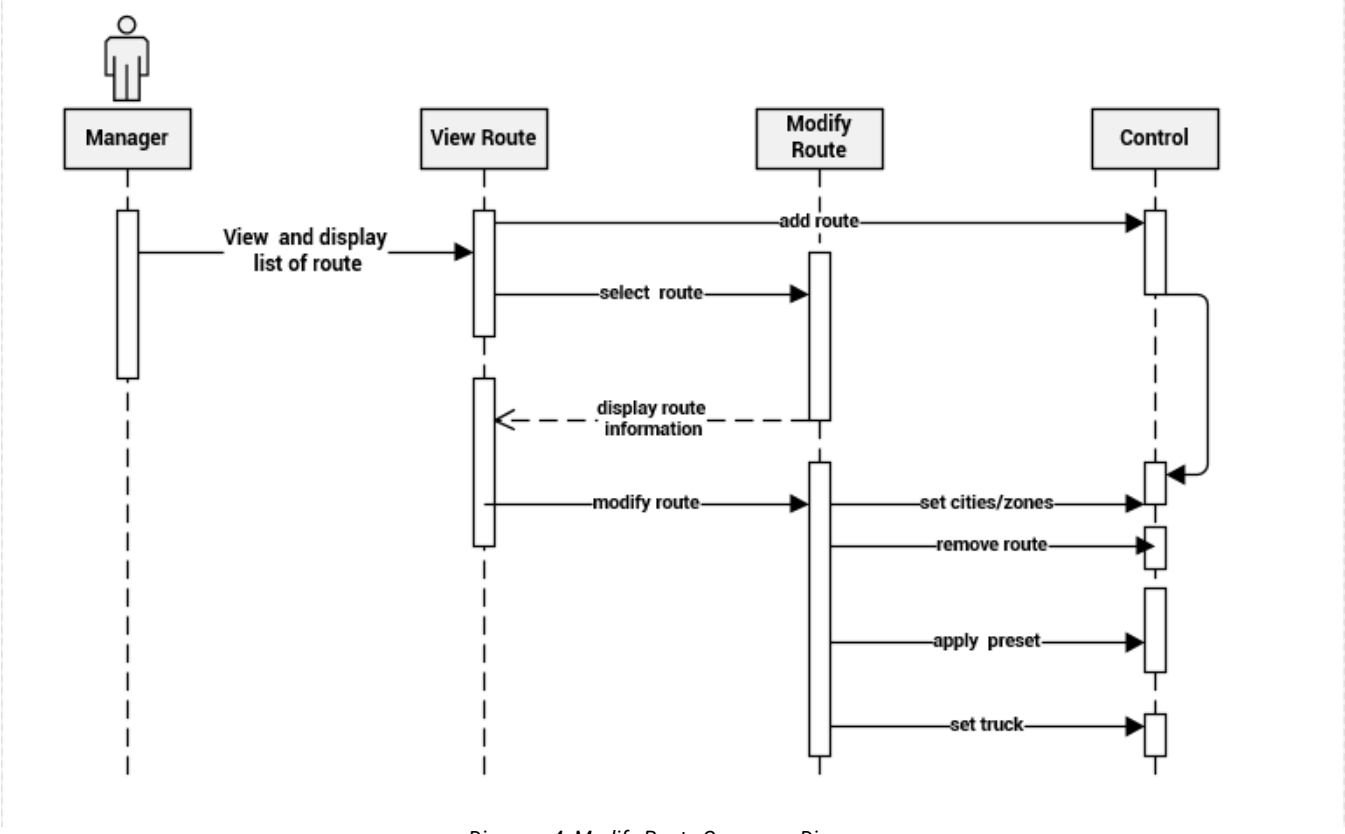

Diagram 4: Modify Route Sequence Diagram

4. Modify Truck

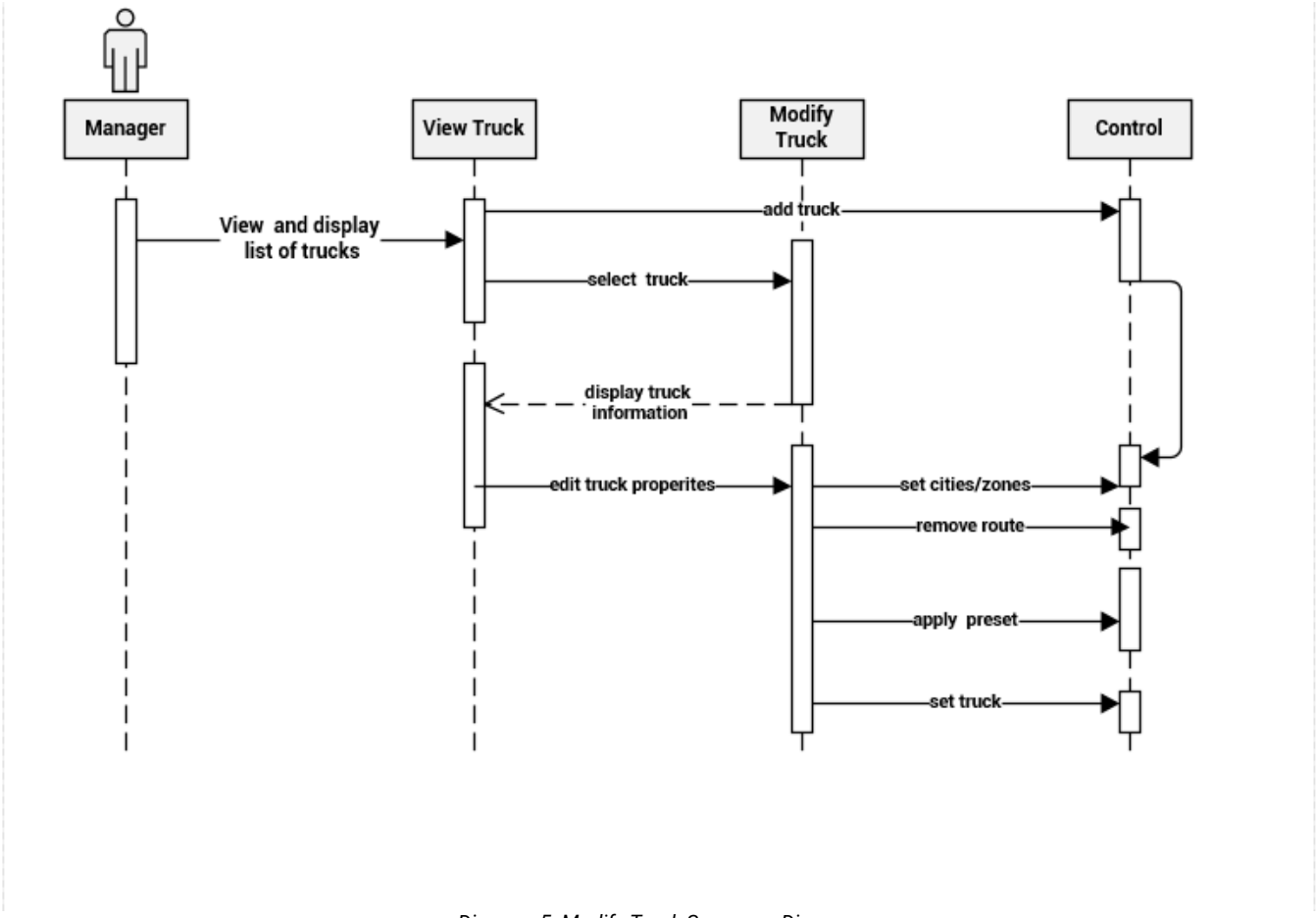

Diagram 5: Modify Truck Sequence Diagram

5. Process Batch File

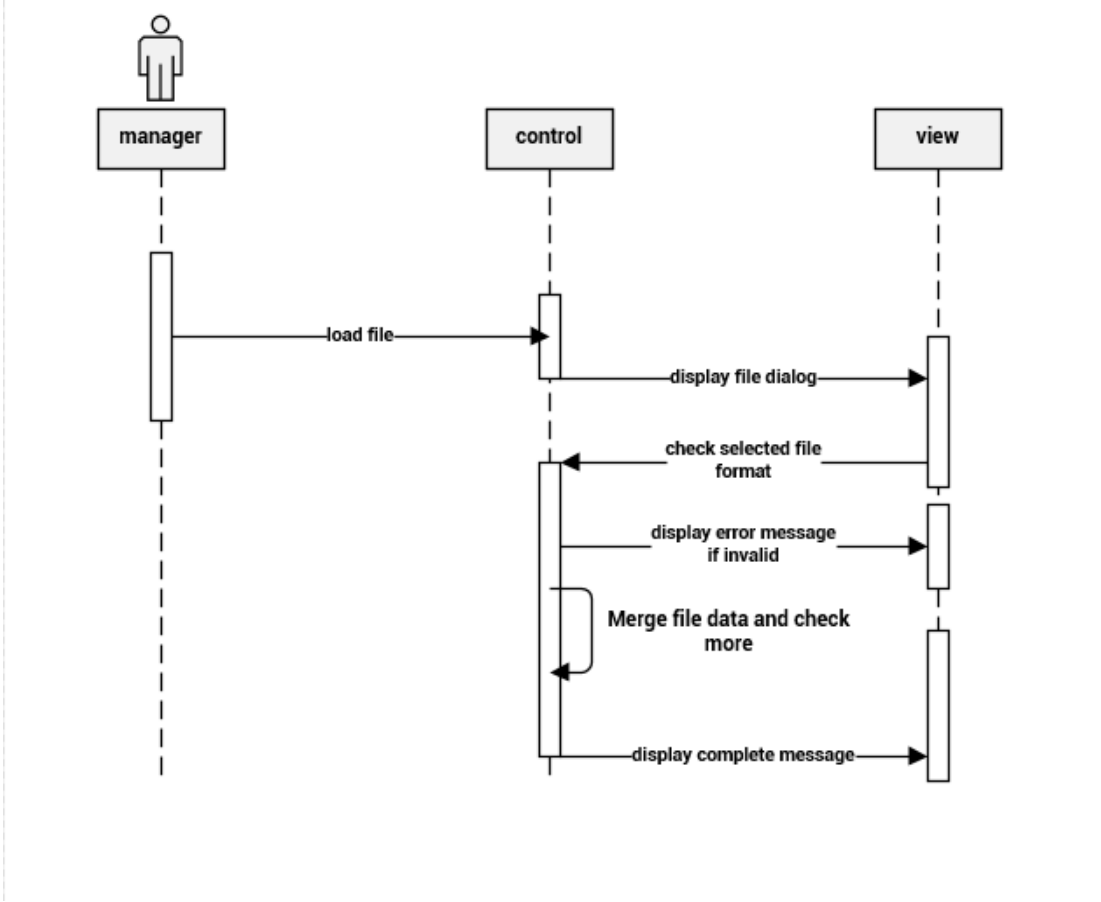

Diagram 6: Process Batch File Sequence Diagram

6. Modify Item

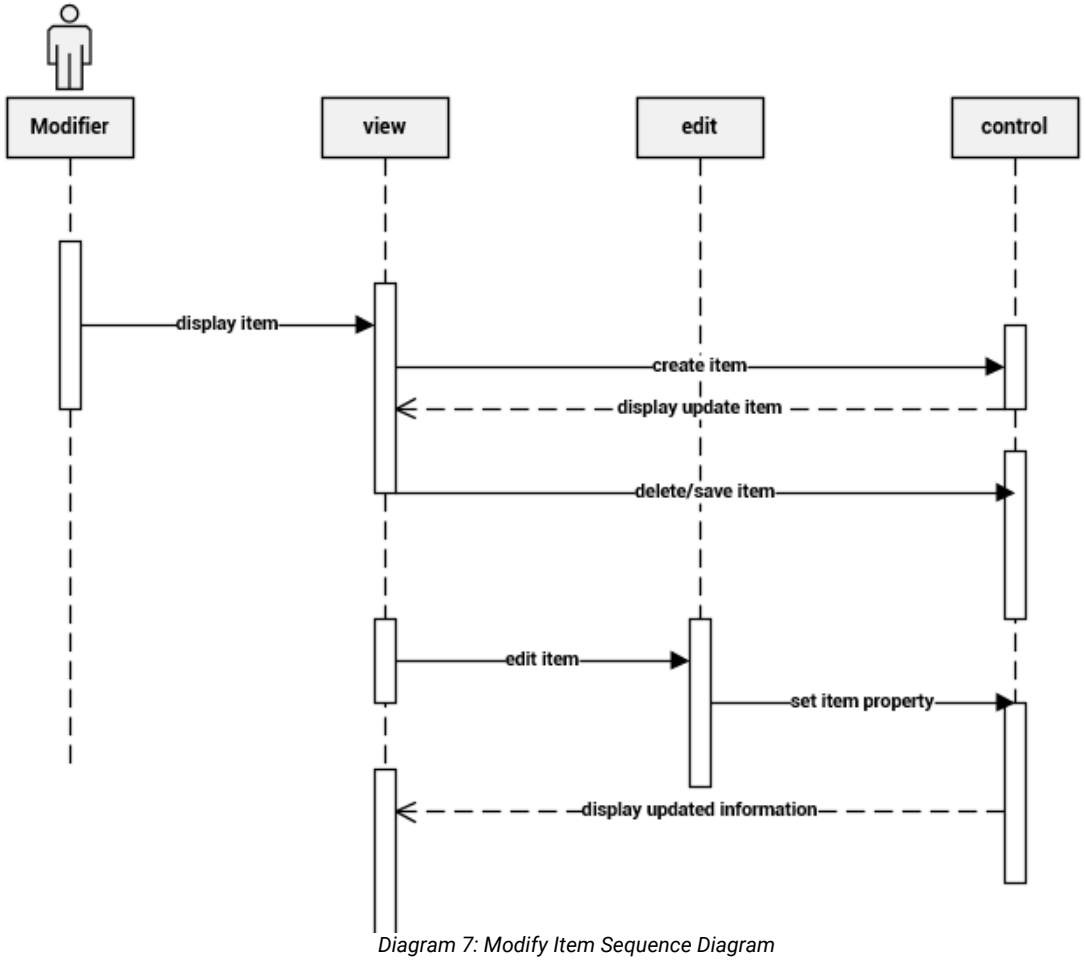

Diagram 7: Modify Item Sequence Diagram

7. Modify Driver

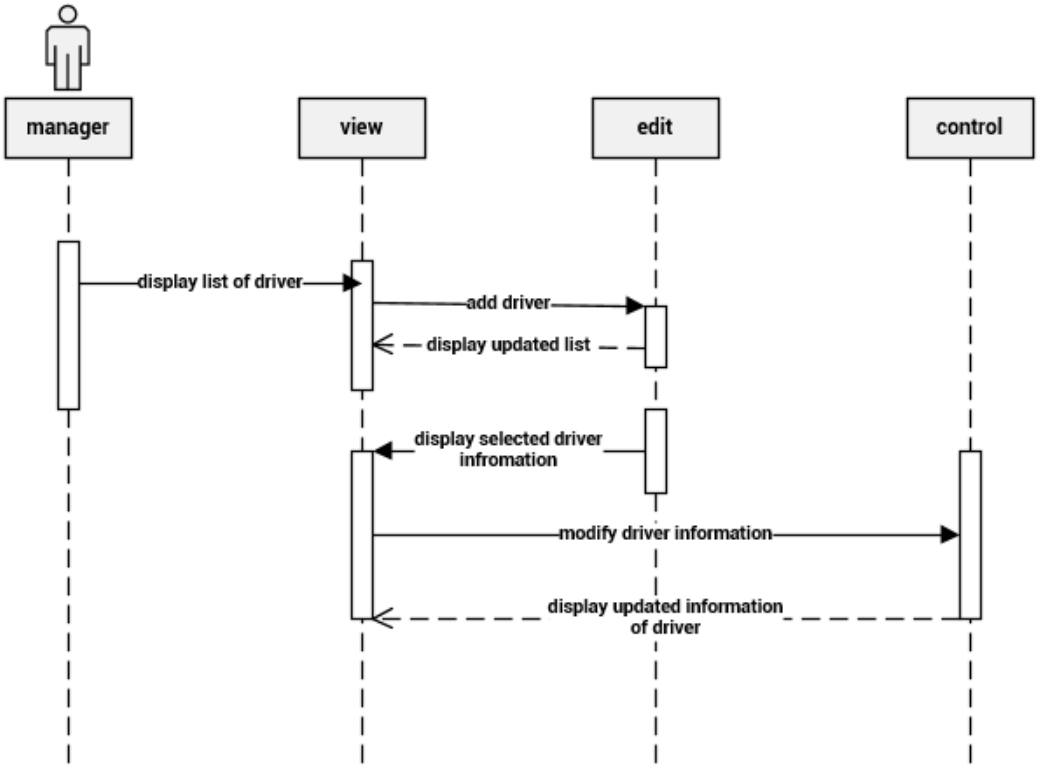

Diagram 8: Modify Driver Sequence Diagram

8. Modify Setting

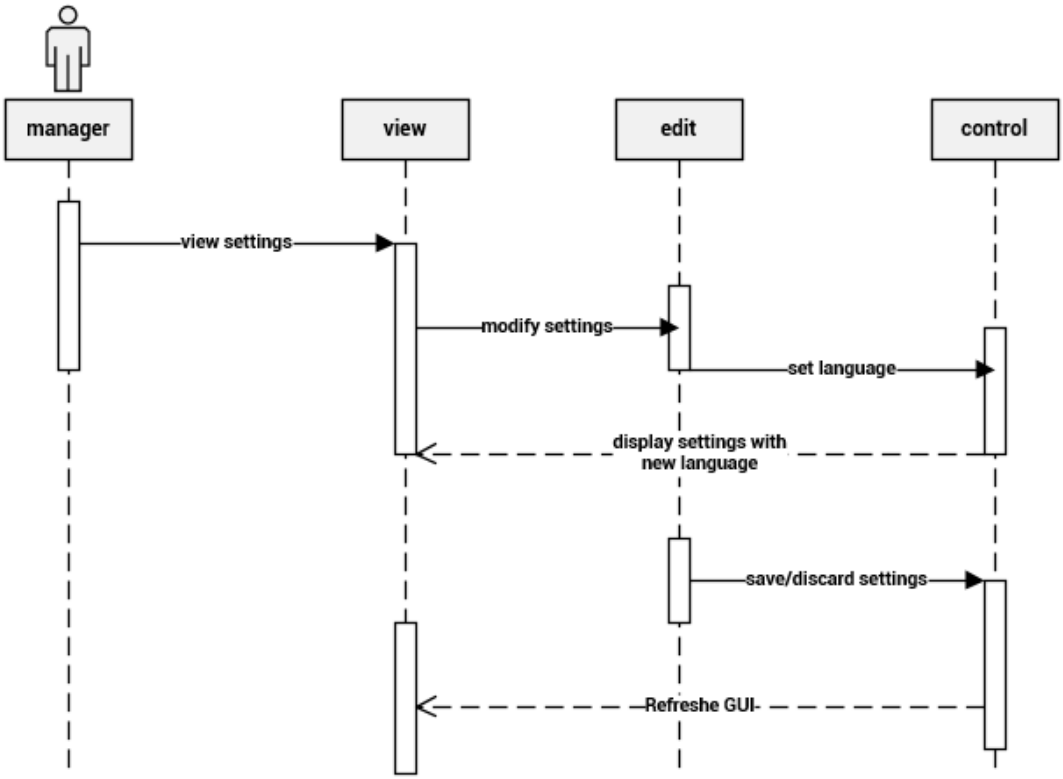

Diagram 9: Modify Settings Sequence Diagram

9. Modify Voting

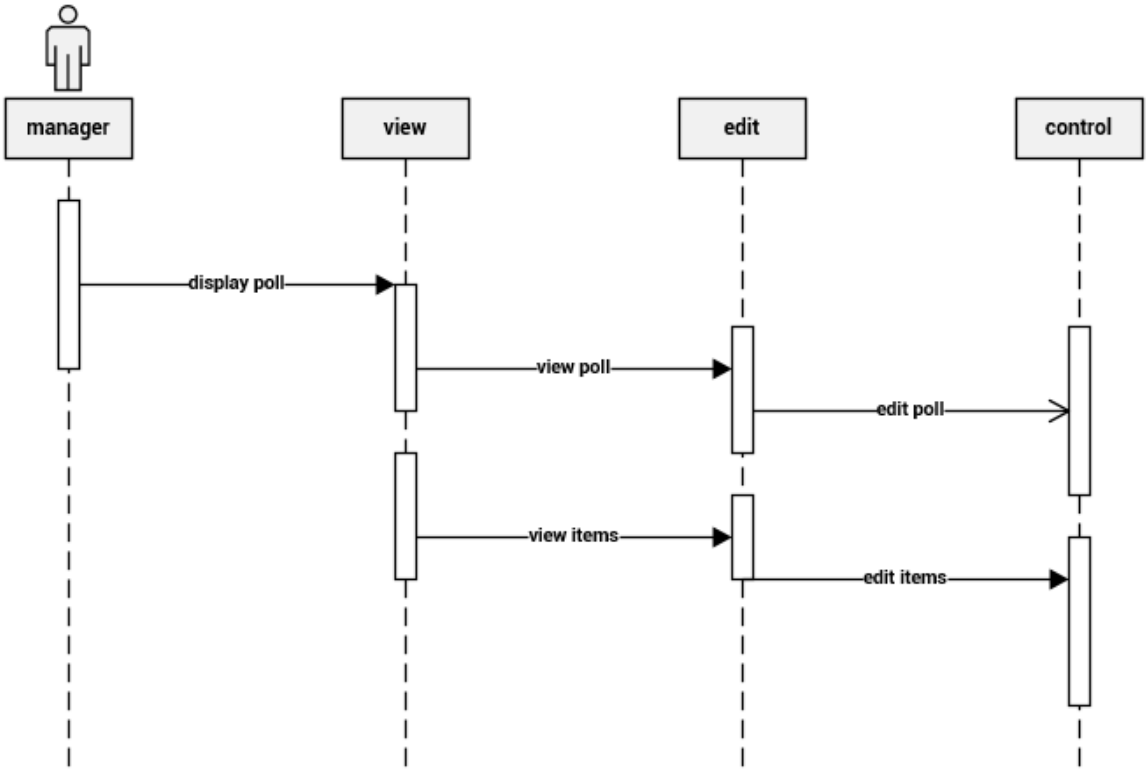

Diagram 10: Modify Voting Sequence Diagram

10. Modify Preset

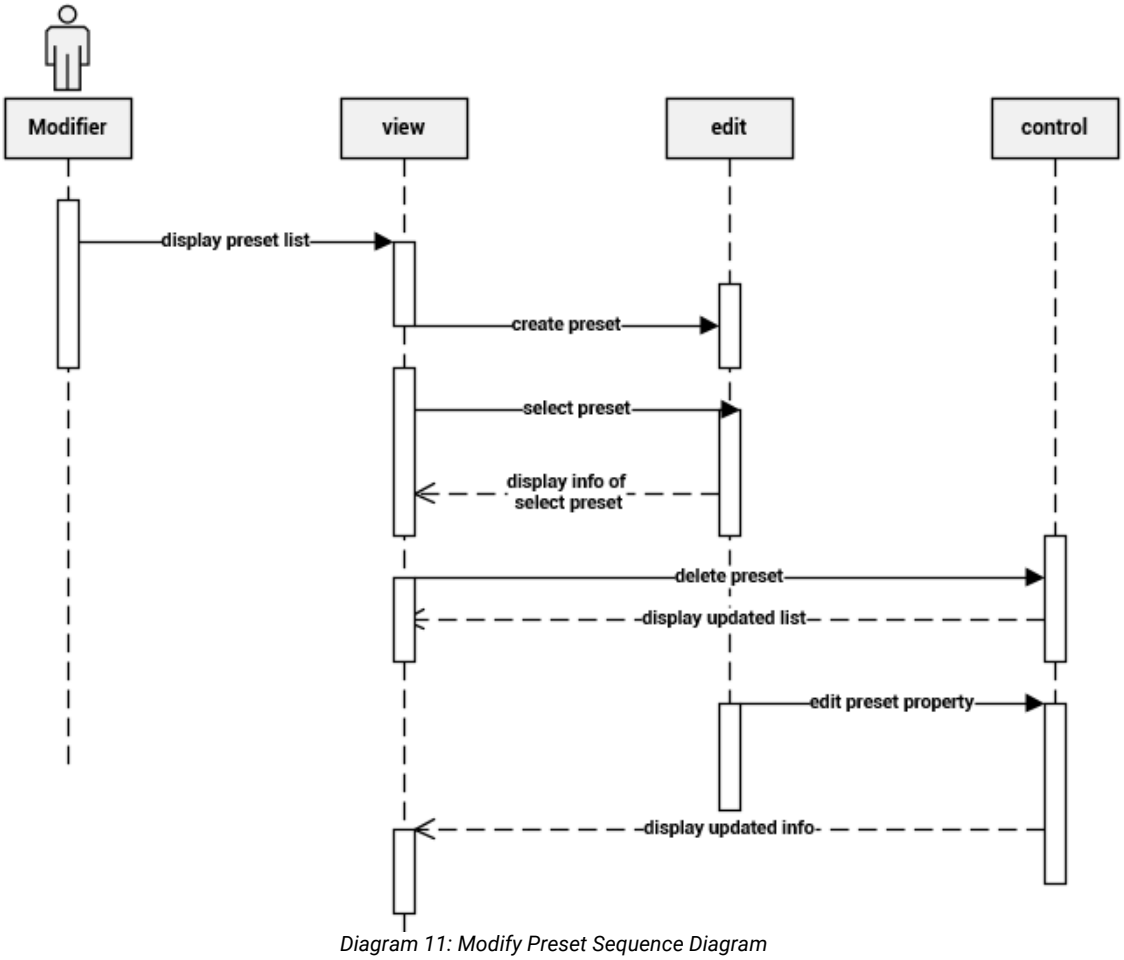

Diagram 11: Modify Preset Sequence Diagram

11. View Fuel Usage

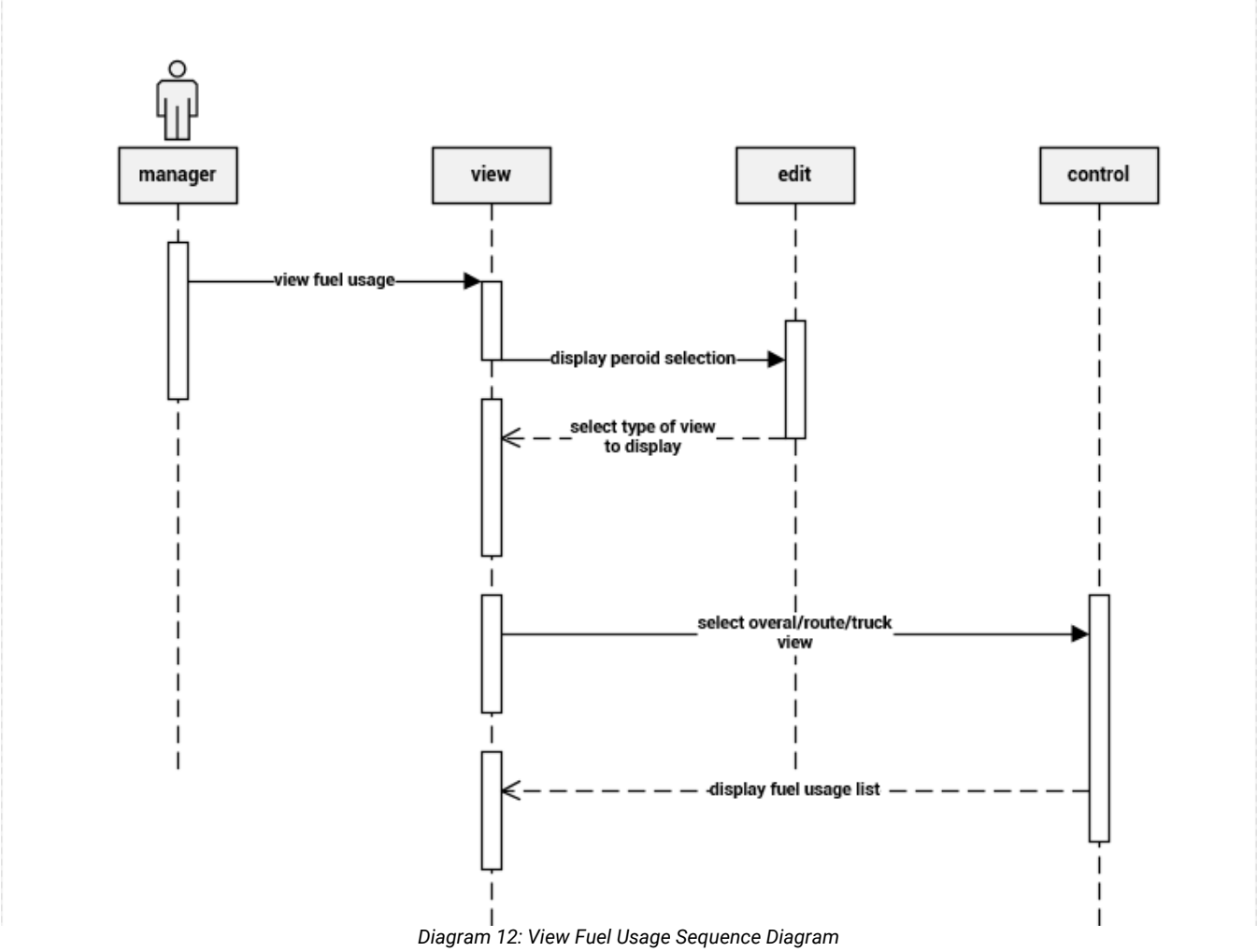

Diagram 12: View Fuel Usage Sequence Diagram

4.4 Communication Diagrams

1. Process Batch File

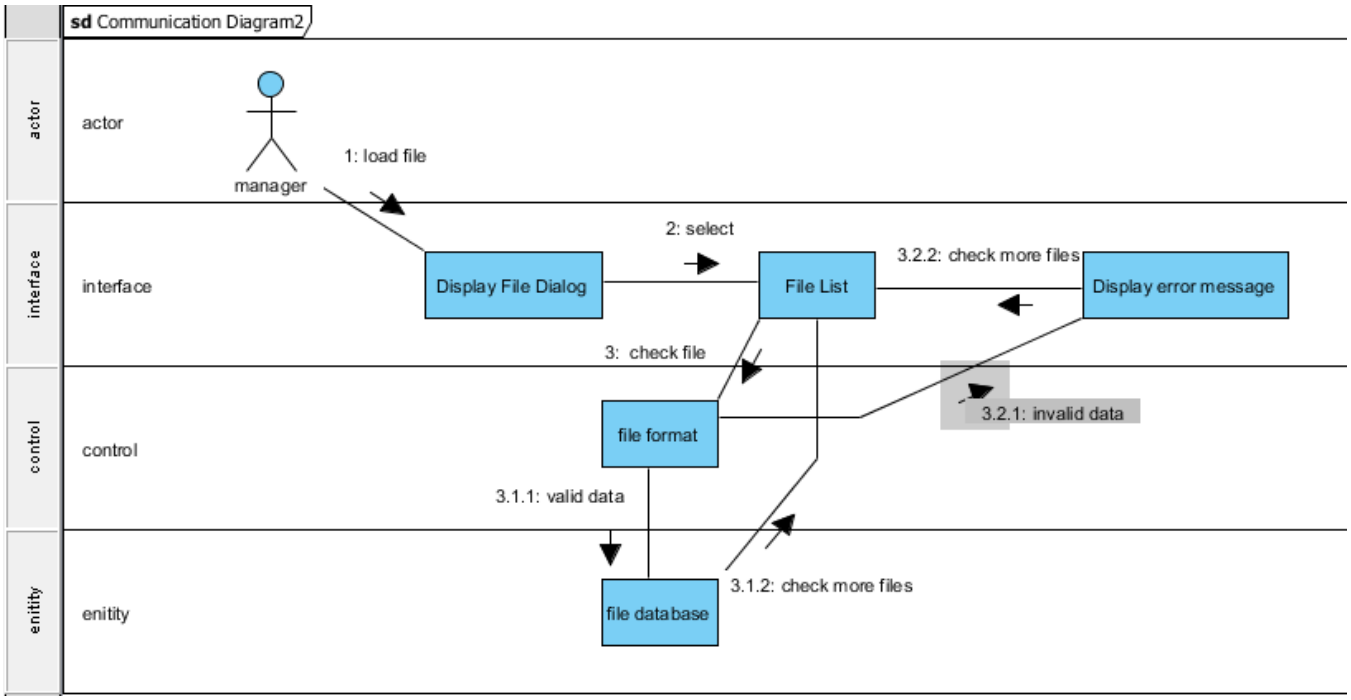

Diagram 13: Process Batch File Communication Diagram

2. Modify Inventory

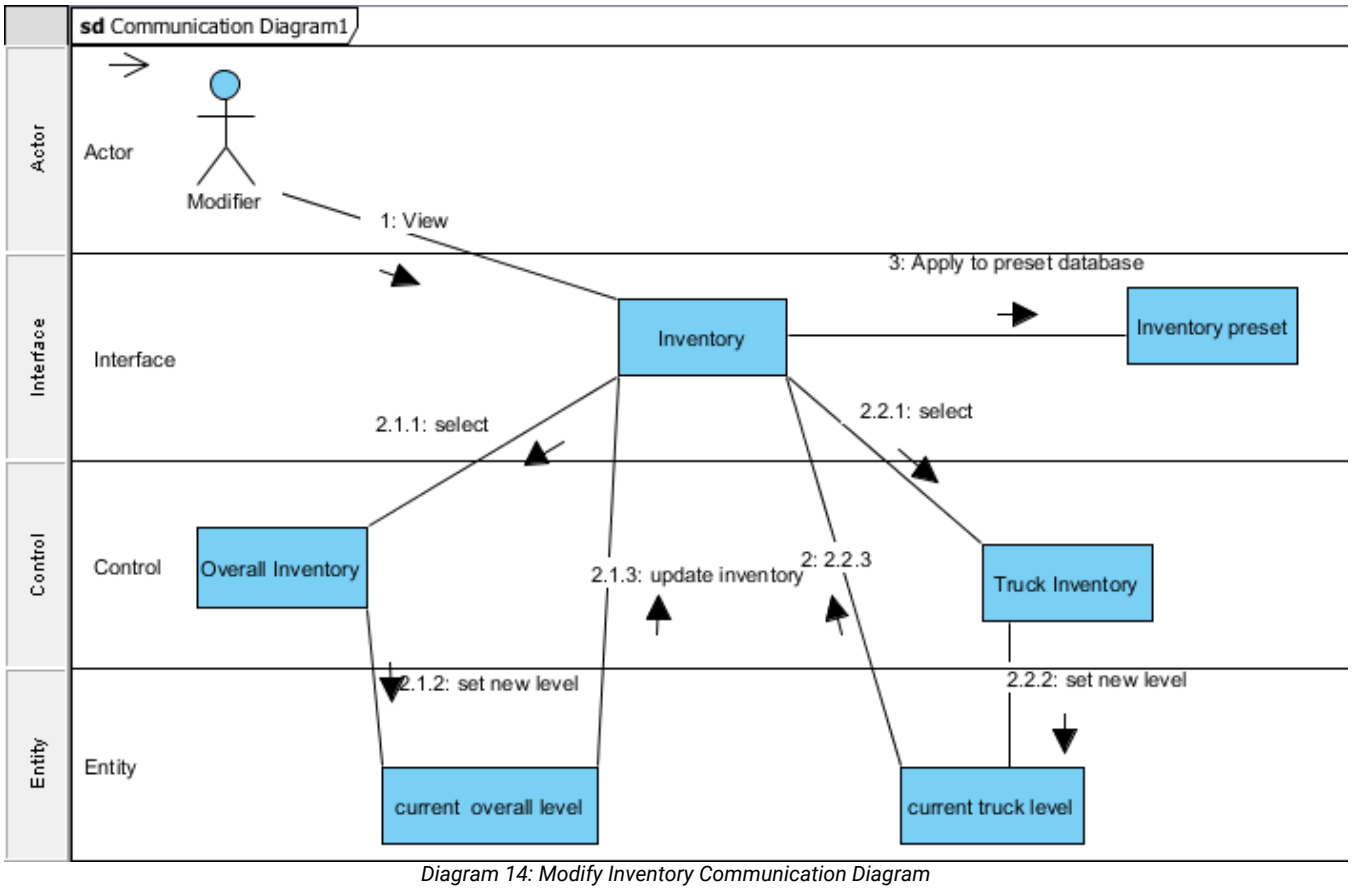

Diagram 14: Modify Inventory Communication Diagram

3. Modify Settings

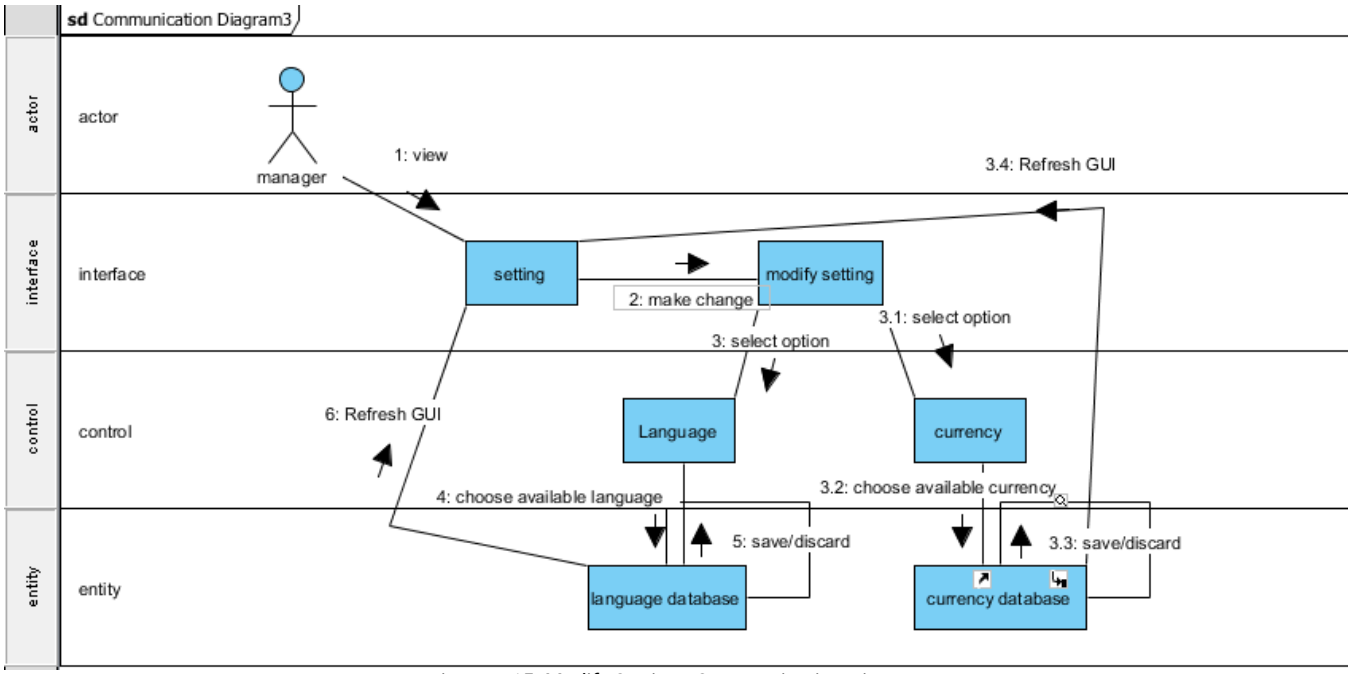

Diagram 15: Modify Settings Communication Diagram

4. View Fuel Usage

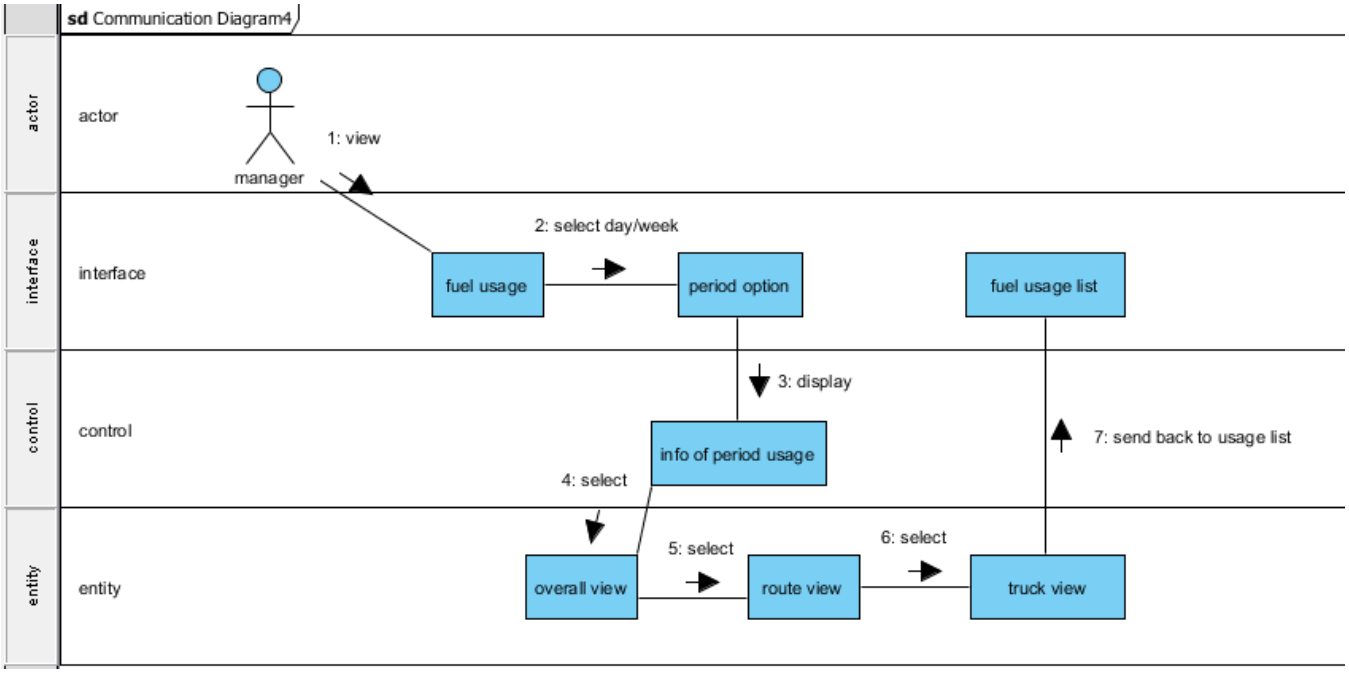

Diagram 16: View Fuel Usage Communication Diagram

5. View Driver & Truck

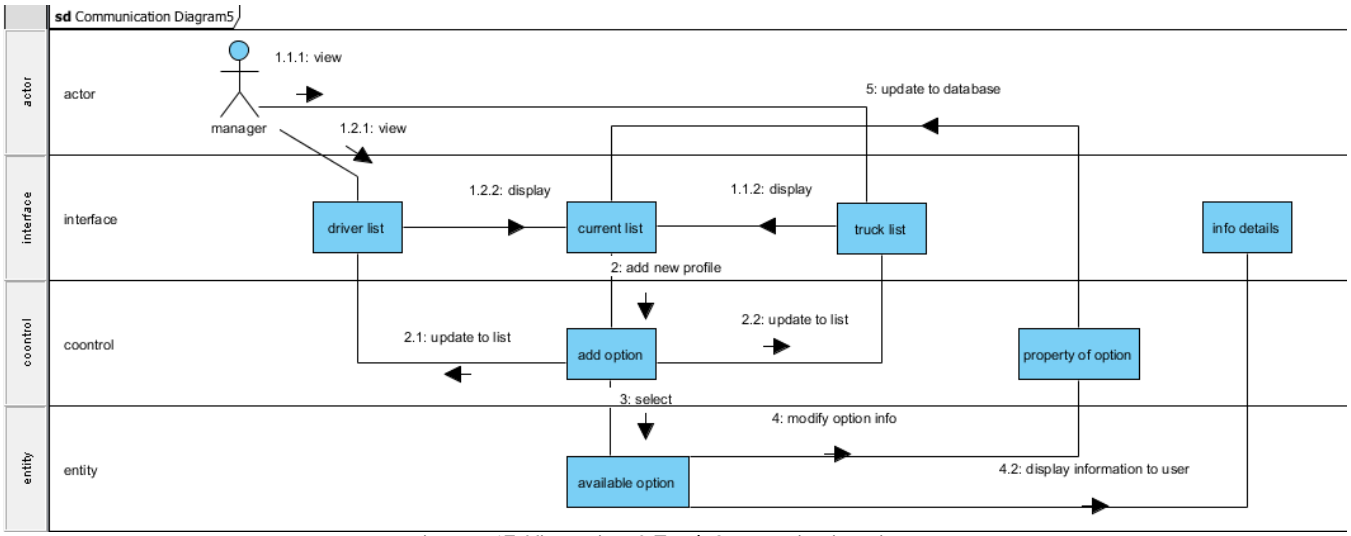

Diagram 17: View Driver & Truck Communication Diagram

6. Modify Route

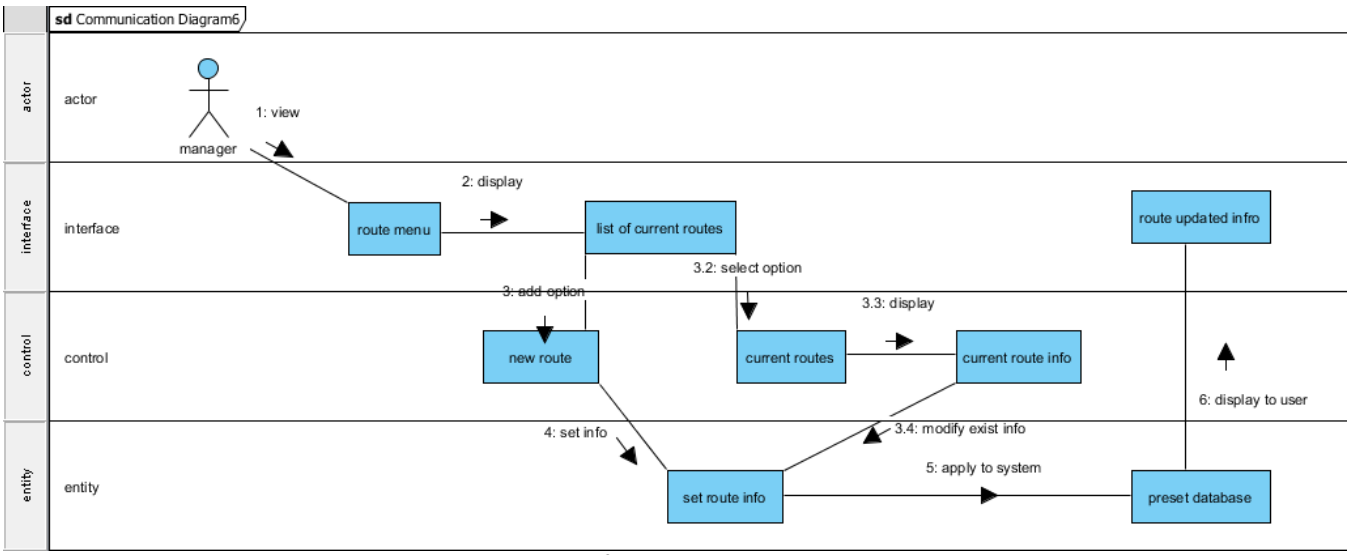

Diagram 18: Modify Route Communication Diagram

5.0 Behavioral Model and Description

This section details the behavior of the software. User and system events that cause software transitions are specified along with the states transitioned between.

5.1 Description for Software Behavior

5.1.1 Events

Event Flow 1: User Views Sales

1. User selects Sales from menu
2. User selects to log sales
    a. Data is input and saved
3. User selects to log expiration
    a. Data is input and saved
4. User selects to view sales
    a. Period is selected
    b. Type is selected.
    c. Sales are calculated

Event Flow 2: User Modifies Entity

1. User selects Route, Truck, Driver, Poll or Item from menu
2. User selects specific entity
    a. User modifies entities properties
    b. System updates database
3. User adds entity
    a. System updates database

Event Flow 3: User Modifies Preset

1. User selects Preset from menu
2. User selects specific preset
    a. User modifies presets properties
    b. User schedules preset
    c. System updates database

Event Flow 4: User Processes Batch File

1. User selects Batch File Processing from menu
2. User inputs file information
3. System checks file
4. System updates database

Event Flow 5: User Modifies Settings

1. User selects settings from menu
2. User selects languages
    a. User selects specific language
    b. Graphical interface updates

3. User selects Currencies
    a. User selects specific currency
    b. Graphical interface updates

Event Flow 6: View Fuel Usage
1. User selects Fuel from menu
2. User selects fuel period
3. User selects overall, route or truck
4. System calculates fuel information

5.1.2 States

Main
1. Menu

Event Flow 1: User Views Sales
1. Sales Selection
2. Sale Input
3. Expiration Input
4. Sale Period Select
5. Sale Entity Select
6. Sale Display

Event Flow 2: User Modifies Entity
1. Entity List
2. Entity Addition
3. Entity Modification

Event Flow 3: User Modifies Preset
1. Preset List
2. Preset Addition
3. Preset Modification

Event Flow 4: User Processes Batch File
1. File Information Input
2. Loading/Error Checking
3. File Valid
4. File Invalid

Event Flow 5: User Modifies Settings
1. Setting View
2. Language Selection
3. Currency Selection

Event Flow 6: View Fuel Usage
1. Fuel Period Selection

2. Fuel Entity Selection
3. Fuel Usage Display

5.2 State Transition Diagrams

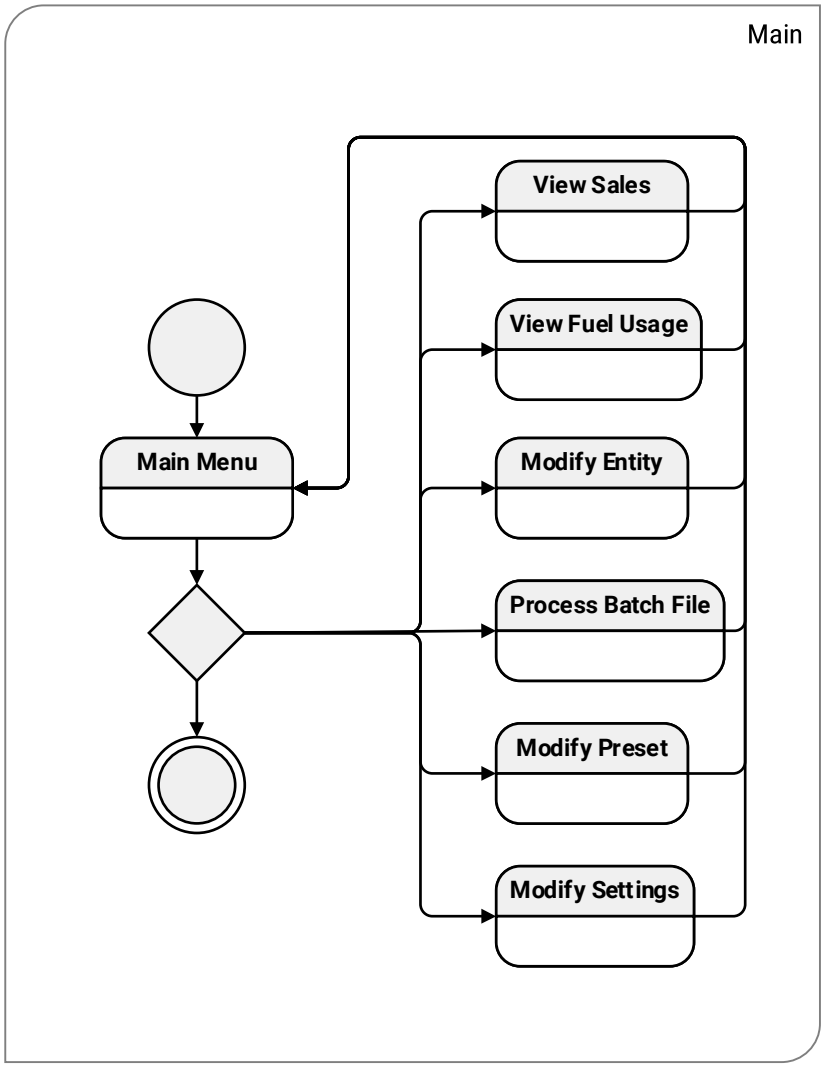

Diagram 19: Main State Transition Diagram

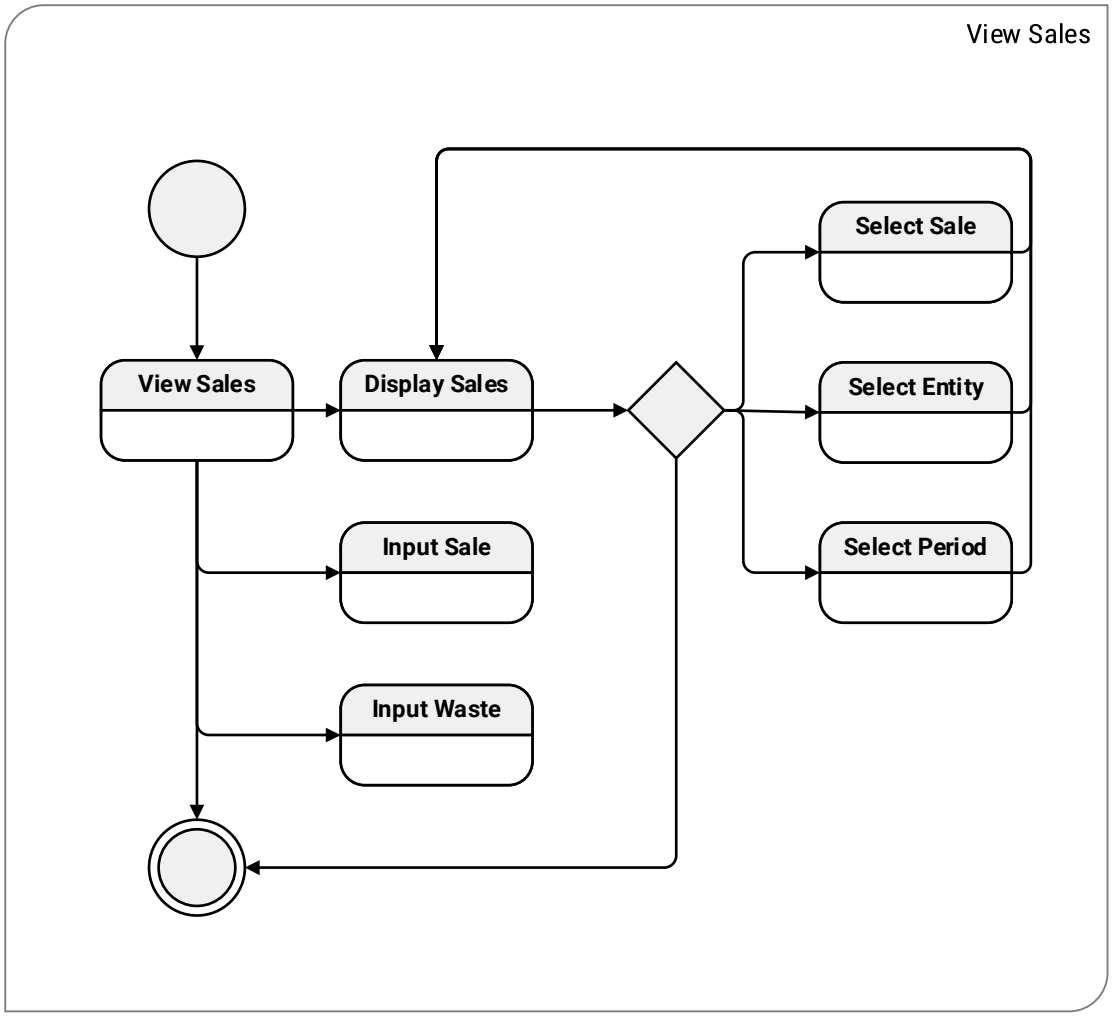

Diagram 20: View Sales State Transition Diagram

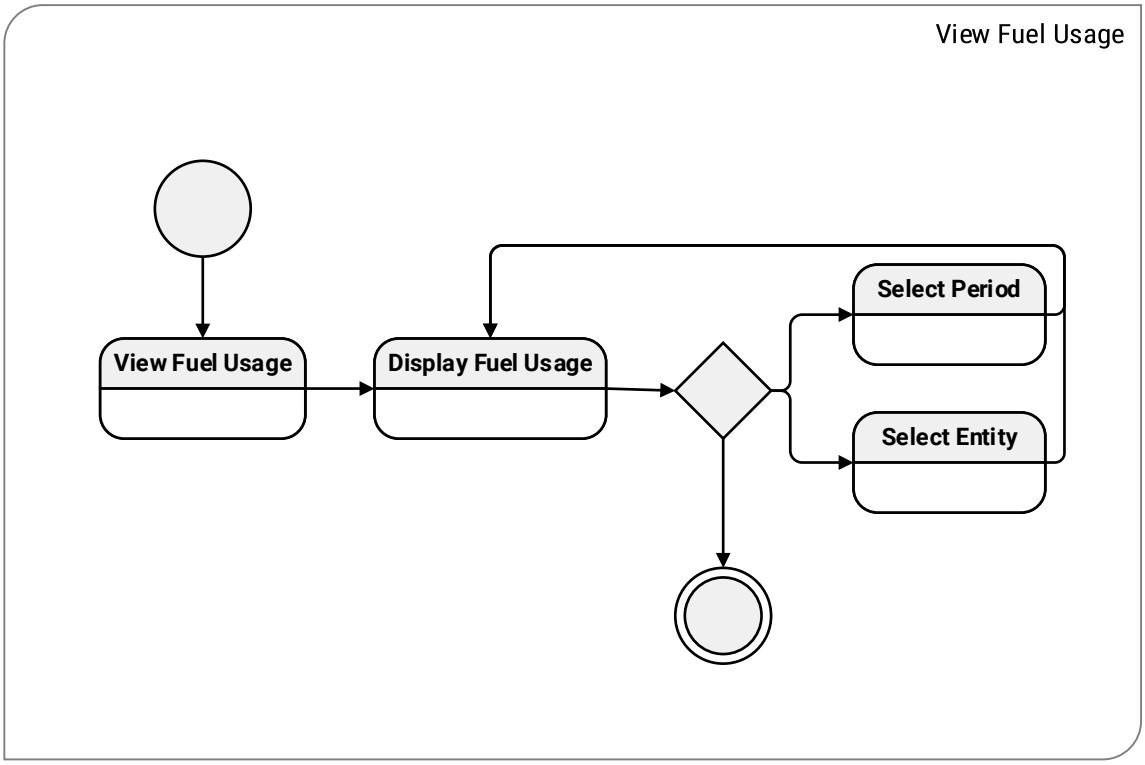

Diagram 21: View Fuel Usage State Transition Diagram

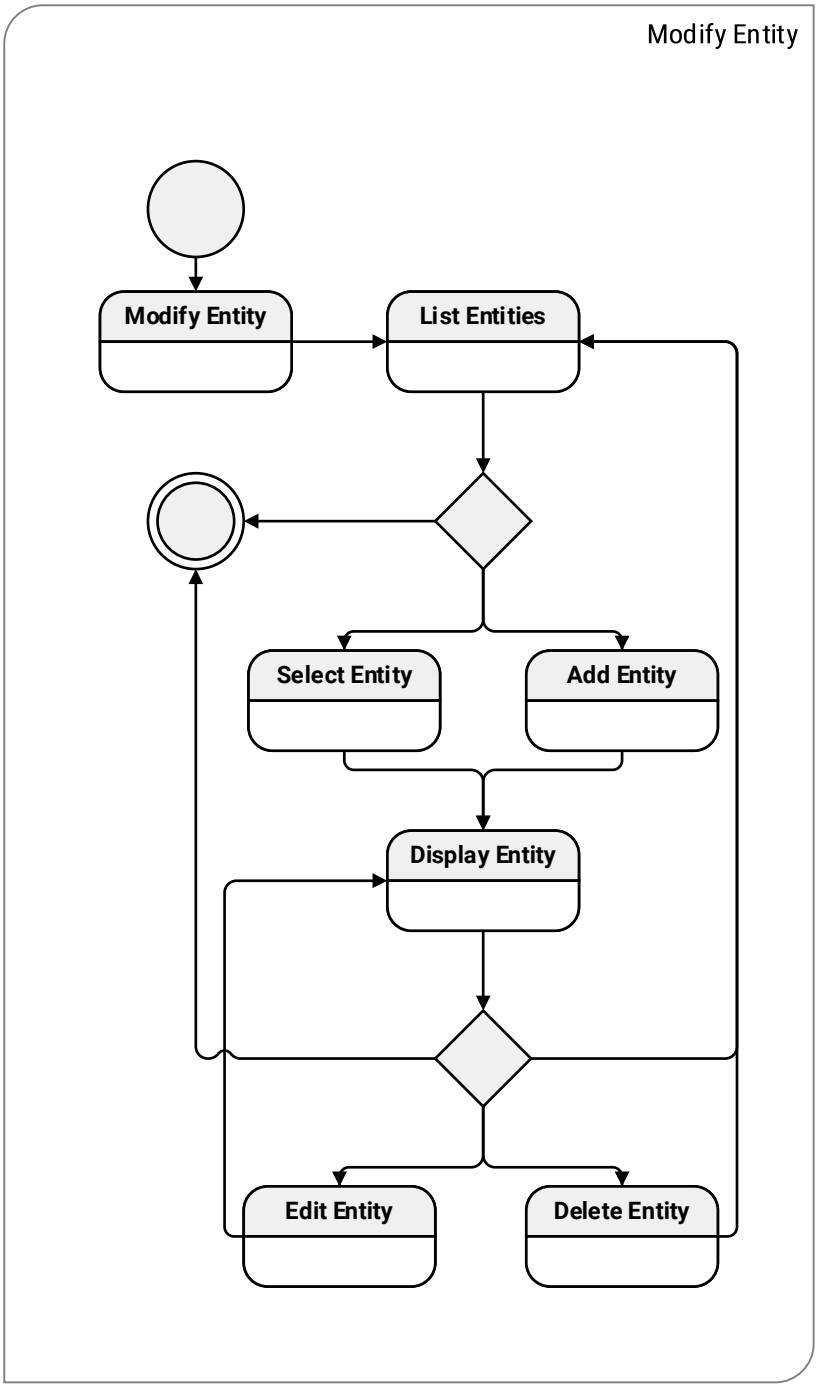

Diagram 22: Modify Entity State Transition Diagram

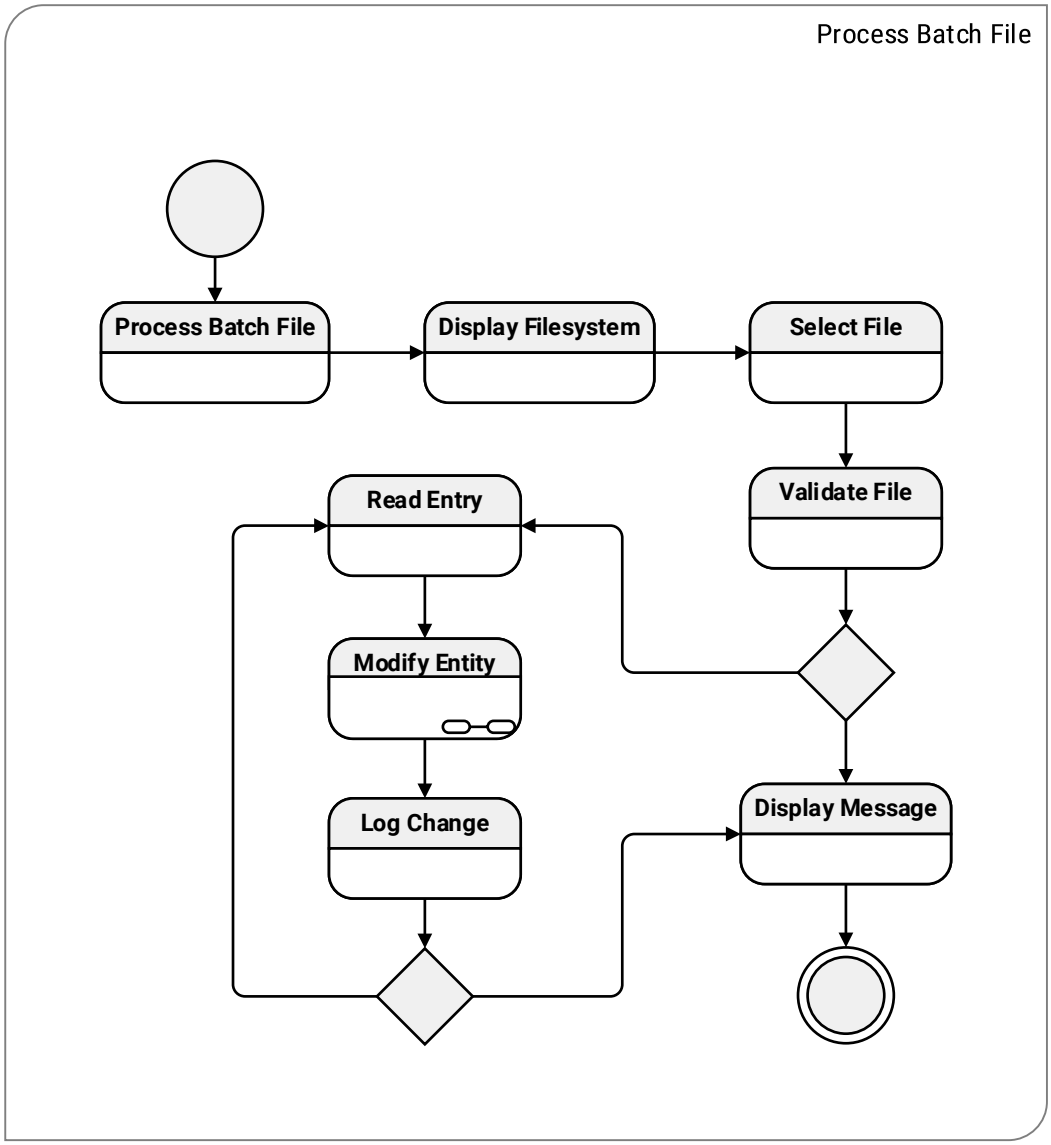

Diagram 23: Process Batch File State Transition Diagram

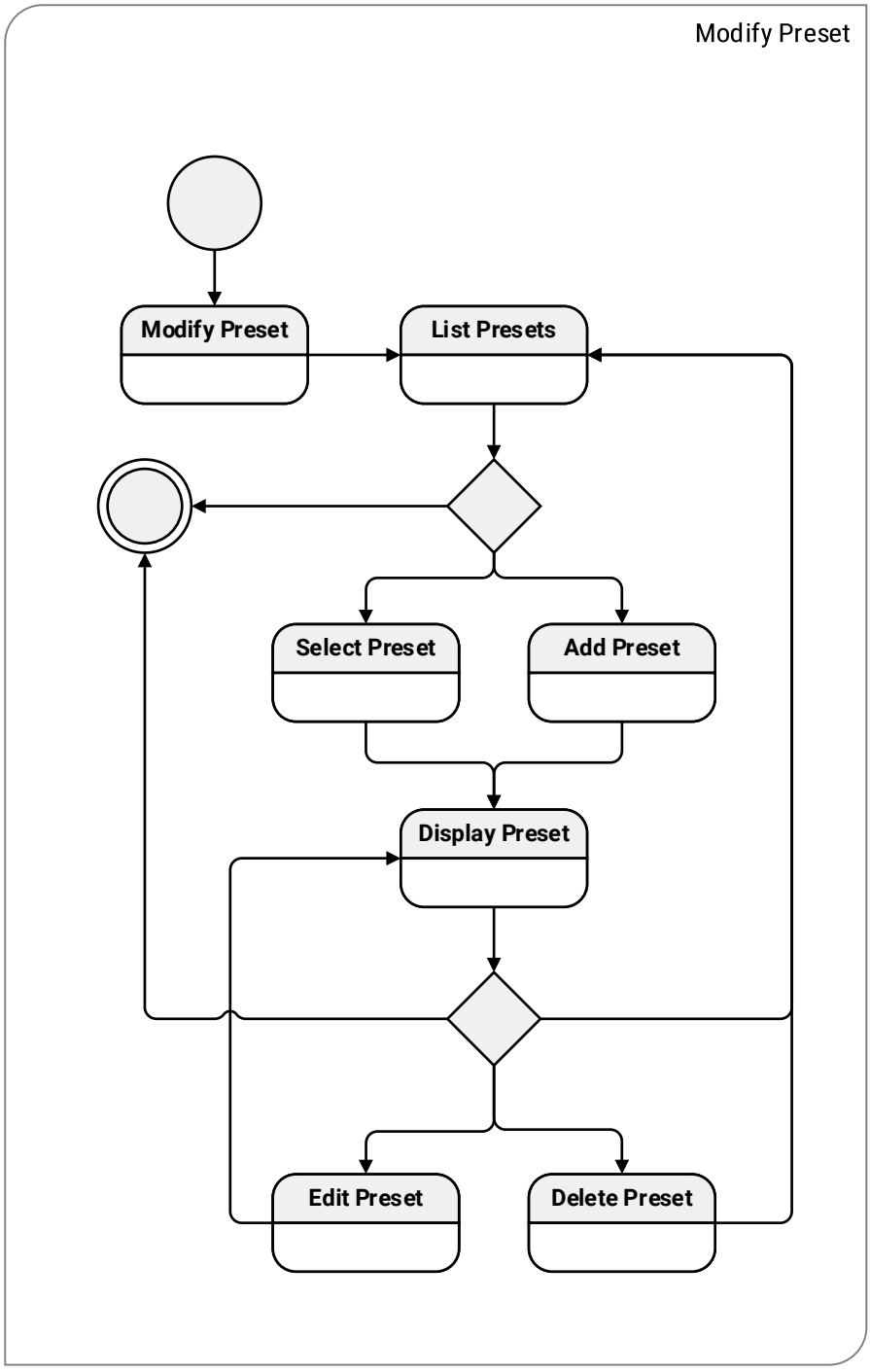

Diagram 24: Modify Preset State Transition Diagram

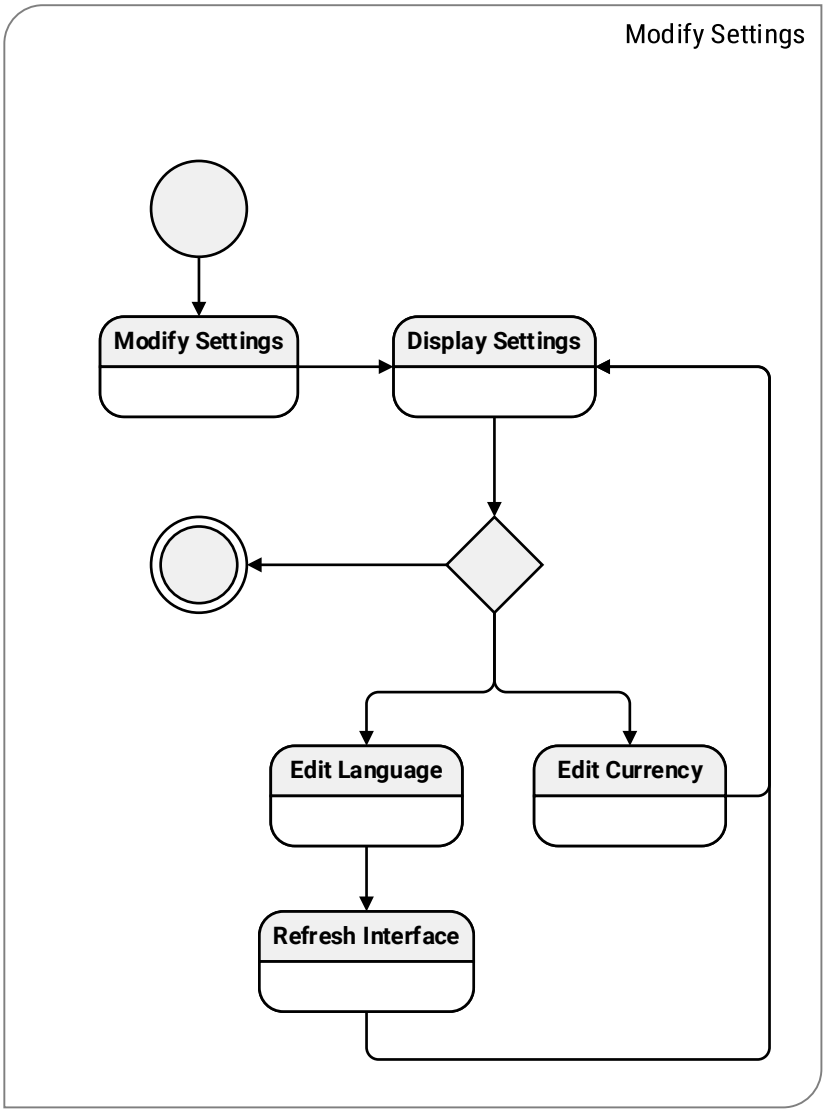

Diagram 25: Modify Settings State Transition Diagram

6.0 Restrictions, Limitations, and Constraints

Special issues which impact the specification, design, or implementation of the software are noted here.

7.0 Validation Criteria

Our validation process has been planned to ensure nothing is added to our software that doesn't work. Before we begin development for a requirement the first thing we will do is plan unit tests that the addition must pass before integration. After the developer completes a software addition that passes all the unit tests the team lead will give the addition a technical review to suggest changes and ensure that the addition is readable, understandable and that it can be easily edited. Before each integration developers who were not involved with the addition will plan and perform black box integration tests to make sure additions work with the completed parts of the project. After the project is considered complete we will conduct validation tests within the team but also have people outside Cosmosys use our software to make sure the software meets the requirements and solves our customer's problems.

7.1 Classes of Tests

- **Unit Testing**
  Before each addition we will plan unit tests that must be passed before an addition is considered for integration. During development developers will also unit test as they go to be more time efficient.
- **Integration Testing**
  Before integration we will review additions for readability, and make sure additions are factorable. After review we will do integration tests on additions before merging them with the software.
  - **Top-Down Approach**
    We plan to use a top-down approach in development. We will develop the base parts of the software first and each developer will work on a branch of the project with each addition going through integration testing.
  - **Regression**
    With each new addition we will retest the entire project at key points to make sure new additions do not cause problems higher up in the project.
  - **White-Box**
    During white box testing we will make sure every part of our program is tested based on its logical structure. These tests will be planned after development because they are based on the internal structure of the software and will be done before validation testing.
- **Validation Testing**
  After the project is functionally complete we will test it to validate that it meets our business requirements.
  - **Configuration Review**
    In this step we will review the software and all associated design documentation for consistency and completion. We will also review to see if all requirements have been met.
  - **Customer Acceptance Testing**
    Before project completion, if time permits, we will informally present the software to the customer to get a confirmation that our software meets the requirements.
  - **Black-Box Testing**
    After completion we will do internal black box testing within our team with members

testing parts they haven't worked on without looking at the code. After that we will have people outside the team use our software to make sure functional requirements are met.

- **Performance Testing**
  We have set performance bounds (described in section 7.3) for speed of execution that we will test for after the project is complete.
    - **Refactoring**
      If our software fails performance tests we will refactor it to minimize time issues before deployment.
- **Deployment Testing**
  Our software needs to work for our customer so we will test it on computers at University of Michigan -Dearborn to make sure it can work during our presentation and grading.

7.2 Expected Software Response

High level software requirement validation tests are described in the table below. These software responses will be verified in black-box testing. Many functions are done in two spots so the associated use cases listed will each be tested separately.

TESTED USE CASES	TEST DESCRIPTION	EXPECTED RESULT
UC01, UC04	View of inventory on trucks after batch processing.	Inventory is displayed properly and clearly. Values match batch processing files. Values are consistent in UC01 and UC04
UC01, UC04	Set inventory levels	Inventory changes are reflected in the database and can be viewed in the program and are consistent in UC01 and UC04
UC01, UC03	Apply inventory presets	Inventory presets change truck inventory appropriately.
UC02	View of sales reports for drivers, routes and trucks over a day and a week.	Sales reports are all consistent with planned sales data by week and day, and are clear and understandable.
UC03	Display cities and zones	Zones and cities match batch processing file.
UC03	Add cities to a route	Cities are added to the route in the database and are reflected in the UC03 view.
UC03, UC04	Display trucks assigned to routes	Trucks are displayed clearly and match batch processing files in UC03 and UC04 views.
UC03, UC04	Assign truck to route	Truck assignments are reflected in the database and are consistent in views in UC03 and UC04.
UC04, UC07	View of driver to truck assignments	Driver and truck assignments match batch processing and are consistent in UC04 and UC07 views.
UC04	View of truck fuel level and usage.	Truck fuel level and usage are consistent with batch files (if used) in UC04 view.
UC04, UC07	Assign driver to truck	Truck to driver assignments are updated in database and are consistent in UC04 and UC07 views.
UC04	Set truck fuel level and usage	Changes to truck fuel level and usage are consistent in the database and in the UC04 view.
UC05	Batch file processing	Batch file data is reflected by database and in all data views.
UC06	Item viewing	Item expirations match the day they were added. All other values match batch processing.
UC06	Item modification	Item modifications are consistent in the database and view in UC06.
UC07	View driver properties (number, name, salary)	Driver values are consistent with batch processing.
UC07	Driver property changes (number, name, salary)	Driver properties are changed in database and UC07 view.
UC08	View settings (currency, language)	Settings are clearly displayed and readable by anyone.
UC08	Change settings (currency, language)	Language and currency changes are reflected and accurate in every use case.
UC09	Poll viewing	Item, vote count and zone are clearly displayed and reflect the database.
UC09	Poll creation	New poll values are visible in database and uc09 view.
UC10	Preset view	Preset name, type and item levels are clear and reflected in database.
UC10	Change presets	Preset changes are reflected in UC10 view.
UC10	Preset scheduling	Presets automatically apply when scheduled.
UC11	Fuel usage views	Fuel usage is calculated correctly and displayed clearly for routes and trucks and displayed clearly.

Table 3: Requirements Validation Tests

7.3 Performance Bounds

To set performance standards we've come up with two speed limitations for the types of user actions in the software. Every user action that doesn't display sales or fuel history or data for multiple entities must be complete within 5 seconds. Every user input that involves displaying sales or fuel data over a day or week or displaying information about multiple entities must complete in 10 seconds.

8.0 Appendices

Presents information that supplements the Requirements Specification

8.1 System Traceability Matrix

Document		SPMP	SRSD	UCSD	DIAG	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11	DFDS	DATD
Software Project Management Plan	SPMP	X	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Software Requirements Specification Document	SRSD		X	■	■													
Use Case Summary Document	UCSD			X	■	■	■	■	■	■	■	■	■	■	■	■	■	
Diagrams Visio Document	DIAG				X	■	■	■	■	■	■	■	■	■	■	■	■	
Use Case - Modify Inventory	UC01					X	■			■	■				■		■	
Use Case - Manage Sales	UC02						X			■			■			■	■	
Use Case - Modify Route	UC03							X		■							■	
Use Case - Modify Truck	UC04								X	■							■	
Use Case - Process Batch File	UC05									X	■	■					■	
Use Case - Modify Item	UC06										X						■	
Use Case - Modify Driver	UC07											X					■	
Use Case - Modify Settings	UC08												X				■	
Use Case - Modify Voting	UC09													X			■	
Use Case - Modify Presets	UC10														X	■	■	■
Use Case - View Fuel Usage	UC11															X	■	
Data Flow Diagrams Document	DFDS																X	
Data Dictionary	DATD																	X

Table 4: Traceability Matrix

Document	Section	Document	Section
DATD	REF	SRSD	3.1.4
DIAG	CM	SRSD	3.1.3
	STDS	SRSD	5.2
	UC01	UC01	6.0
		UCSD	4.0
	UC02	UC02	6.0
		UCSD	4.0
	UC03	UC03	6.0
		UCSD	4.0
	UC04	UC04	6.0
		UCSD	4.0
	UC05	UC05	6.0
		UCSD	4.0
	UC06	UC06	6.0
		UCSD	4.0
	UC07	UC07	6.0
		UCSD	4.0
	UC08	UC08	6.0
		UCSD	4.0
	UC09	UC09	6.0
		UCSD	4.0
	UC10	UC10	6.0
		UCSD	4.0
	UC11	UC11	6.0
		UCSD	4.0
SPMP	1.1	SRSD	1.2
	1.4	SRSD	1.4
	1.0	SRSD	1.0
SRSD	2.2	UCSD	3.0
	1.0	SPMP	1.0
	1.2	SPMP	1.1
	1.4	SPMP	1.4
	3.1.3	DIAG	CM
	3.1.4	DATD	REF
	4.1	UC01	REF
		UC02	REF
		UC03	REF
		UC04	REF
		UC05	REF
		UC06	REF
		UC07	REF
		UC08	REF

Document	Section	Document	Section
SRSD	4.1	UC09	REF
		UC10	REF
		UC11	REF
		UCSD	REF
	5.2	DIAG	STDS
UC01	2.1	UCSD	5.0
	6.0	DIAG	UC01
	REF	SRSD	4.1
		UCSD	3.0
UC02	2.1	UCSD	5.0
	6.0	DIAG	UC02
	REF	SRSD	4.1
		UCSD	3.0
UC03	2.1	UCSD	5.0
	6.0	DIAG	UC03
	REF	SRSD	4.1
		UCSD	3.0
UC04	2.1	UCSD	5.0
	6.0	DIAG	UC04
	REF	SRSD	4.1
		UCSD	3.0
UC05	2.1	UCSD	5.0
	6.0	DIAG	UC05
	REF	SRSD	4.1
		UCSD	3.0
UC06	2.1	UCSD	5.0
	6.0	DIAG	UC06
	REF	SRSD	4.1
		UCSD	3.0
UC07	2.1	UCSD	5.0
	6.0	DIAG	UC07
	REF	SRSD	4.1
		UCSD	3.0
UC08	2.1	UCSD	5.0
	6.0	DIAG	UC08
	REF	SRSD	4.1
		UCSD	3.0
UC09	2.1	UCSD	5.0
	6.0	DIAG	UC09
	REF	SRSD	4.1
		UCSD	3.0
UC10	2.1	UCSD	5.0

Document	Section	Document	Section
UC10	6.0	DIAG	UC10
	REF	SRSD	4.1
		UCSD	3.0
UC11	2.1	UCSD	5.0
	6.0	DIAG	UC11
	REF	SRSD	4.1
		UCSD	3.0
UCSD	3.0	SRSD	2.2
		UC01	REF
		UC02	REF
		UC03	REF
		UC04	REF
		UC05	REF
		UC06	REF
		UC07	REF
		UC08	REF
		UC09	REF
		UC10	REF
		UC11	REF
	4.0	DIAG	UC01
		DIAG	UC02
		DIAG	UC03
		DIAG	UC04
		DIAG	UC05
		DIAG	UC06
		DIAG	UC07
		DIAG	UC08
		DIAG	UC09
		DIAG	UC10
		DIAG	UC11
	5.0	UC01	2.1
		UC02	2.1
		UC03	2.1
		UC04	2.1
		UC05	2.1
		UC06	2.1
		UC07	2.1
		UC08	2.1
		UC09	2.1
		UC10	2.1
		UC11	2.1
	REF	SRSD	4.1

Table 5: Traceability Lookup Table

8.2 Product Strategies

This product will be electronically delivered to the customer. No marketing or distribution will be executed.

8.3 Analysis Metrics to be Used

The following is a description of the goal of the metrics used during this project, definitions of the various metrics used, and their applicability to the different aspects of the project.

8.3.1 Goal of the Metrics

Analyze the *Ice Cream Manager* architecture for the purpose of evaluating architectural components with respect to the ability to make *Ice Cream Manager* more extensible from the viewpoint of the software engineers performing the work in the context of product enhancement until deployment to the customer.

Towards that goal, the metrics that are collected will be used to answer the following questions:

Q1: Was the project planned thoroughly enough to promote efficient, extensible, and reusable source code?
Q2: Was the source code written in an efficient, extensible, and reusable way?

8.3.2 Team-Created Requirements Model Metrics

The quality of the Requirements Model for this project will be determined using two different metrics; specificity and completeness.

Specificity

Specificity is the measure of the precision of language and lack of ambiguity in the Requirements Model. Measurement of specificity occurs through the interpretation of requirements by reviewers. The frequency in which reviewers agree regarding how a requirement is interpreted improves the specificity score. The specificity score is calculated using the following equation:

$$S = \frac{n_{ui}}{n_r}$$

Where S is the specificity score, $\mathbf{n_{ui}}$ is the number of requirements for which all the reviewers had identical interpretations, and n_r is the total number of functional and non-functional requirements. When interpreting the specificity score the closer it is to $\mathbf{1}$ the more specific—or less ambiguous—the requirements.

Completeness

Completeness is the measure of how complete the Requirements Model is. Measurement of completeness occurs through examining the ratio between the number of functional requirements and the number of the inputs and states in the model. The completeness score is calculated using the following equation:

$$C = \frac{n_u}{n_i \times n_s}$$

Where C is the completeness score, n_u is the number of unique functional requirements, n_i is the number of inputs defined or implied by the specification, and n_s is the number of states defined by the specification. When interpreting the completeness score the closer it is to $\mathbf{1}$ the higher the percentage of necessary functions that have been specified.

8.3.3 Team-Created Object-Oriented Model Metrics

The quality of the Object-Oriented Model for this project will be determined using two different metrics; responsibility and linking.

Responsibility

Responsibility is the measure of how much a single class is responsible for performing. Measurement of the responsibility score for a single class occurs though examining the number of methods within or inherited by the class, and the number of attributes within or inherited by the class. The responsibility score is calculated using the following equations:

$$\boldsymbol{R = R_m + R_a}$$

$$\boldsymbol{R_m = n_m + n_{im}}$$

$$\boldsymbol{R_a = n_a + n_{ia}}$$

Where $\boldsymbol{R}$ is the overall responsibility score, $\boldsymbol{R_m}$ is the method responsibility score, $\boldsymbol{R_a}$ is the attribute responsibility score, $\boldsymbol{n_m}$ is the number of methods within the class, $\boldsymbol{n_{im}}$ is the number of methods inherited by the class, $\boldsymbol{n_a}$ is the number of attributes within the class, and $\boldsymbol{n_{ia}}$ is the number of attributes inherited by the class. Properties—attribute getters/setters in *C#*—are not included when counting any methods or attributes for this calculation. When interpreting the method or attribute responsibility scores any values higher than $\boldsymbol{10}$ is likely indicative of a need to spread the responsibilities to a new or existing class. Likewise, an overall responsibility score with a value higher than $\boldsymbol{15}$ is also indicative of a class that needs its responsibilities lightened.

Linking

Linking is the measure of how many external classes are required for a class to perform its function. Measurement of the linking score for a single class occurs through examining the number of non-static objects that are initiated within the class.

$$\boldsymbol{L = 2n_{eo} + n_{io}}$$

Where $\boldsymbol{L}$ is the overall linking score, $\boldsymbol{n_{eo}}$ is the number of objects external to the project that are instantiated, and $\boldsymbol{n_{io}}$ is the number of objects internal to the project that are instantiated. When interpreting the linking score, a value of $\boldsymbol{5}$ to $\boldsymbol{7}$ is considered normal, while a value of $\boldsymbol{10}$ or higher is warning sign that the class may rely upon too many other classes to perform its function.

8.3.4 Team-Created Source Code Quality Metrics

The quality of the source code for this project will be determined using two different metrics; operations and parameters.

Operations

Operations is the measure of how many methods contain more than the specified number of source code operations. An operation is any line of source code that performs a function (i.e. everything except lines of comments or lines that only contain brackets).

$$O = n_{mx}$$

Where O is the operations score, and n_{mx} is the number of methods that have an excess of 25 operations. There are reasons a method may contain excess operations, but they should be rare instances. For that reason, the operations score is expected to remain as low as possible. A operations score of 3 to 5 is expected, if the operations score approaches or exceeds 10 it could be indicative of overburdened methods that should be refactored.

Parameters

Parameters is the measure of how many methods require more than the specified number of parameters. Only parameters that are required for the successful operation of the method are counted towards the recommended maximum.

$$P = n_{px}$$

Where P is the parameters score, and n_{px} is the number of methods that require more than 3 parameters. There are reasons a method may contain excess parameters, but they should be rare instances. For that reason, the parameters score is expected to remain as low as possible. A parameters score of 5 to 7 is expected. If the operations score approaches or exceeds 15, it is indicative of overly complex methods that need to be refactored.

8.3.2 Project Manager-Created Metrics List

The following is a list of metrics that the project manager came up with for the analysis and design category.

Function-based Metric

This metric measures the functionality delivered by a system. The data that will be collected are the function points that we estimated in our Software Project Management Plan. The function points are processes by counting the number of user inputs, outputs, inquires, files, and eternal interfaces. The results would show if our estimation was close and if we need to add or delete function points.

Architectural Design Metrics

This metric focus on the features of the design with stress on the structure and effectiveness of the components. The following are software design complexity measures that are defined in this metrics:

Structural Complexity

S(j)= 〖f^2〗_out (j)

Where fout(j) is the number of modules that are subordinating module j.

Data Complexity

D(j)=(V(j))/[f_out (j)+1]

Where V(j) is the number of input and output variables passed to and from module ‘j’

System Complexity

C(j)=S(j)+D(j)

The results would show the quality of the design.

Requirements Volatility Metrics

This metric track the change over time of requirements. The data would be the new requirements and this would be process to see if there is a change. The results will help us keep track on the changes and be able to comply with those changes.

9.0 Software Requirements Specification Review and Signoff

Review and Signoff of the Software Requirements Specification Document.

NAME	PROJECT TEAM ROLE	SIGNATURE	DATE
Camille Williams	Project Manager		2016-03-17
Marc King	Team Lead		2016-03-17
Aly Lakhani	Developer		2016-03-17
Rodney Lewis	Developer		2016-03-17
Jacob Vacheresse	Developer		2016-03-17
Fan Zhang	Developer		2016-03-17