Ontology-based Expert System for a Generic Drug Production of Pharmaceutical Dosage Forms

Software Design Document

By

**Mr. Narongrit Saisuwan 542115017**

**Mr. Panupak Wichaidit 542115047**

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

………………………………………..

**Dr. Chartchai Doungsa-ard**

# Document History

[[1]](#footnote-1)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Document Name** | **Version** | **Status** | **Date** | **Viewable** | **Reviewer** | **Responsible** |
| **Documents** | | | | | | |
| **OEGP –Software Design Document\_v.0.1.docx** | **0.1**   * Introduction * Class Diagram * Sequence Diagram * UI Design | Draft | Jun. 4, 14 | NS, PW, CD | NS, PW | NS, PW |
| **OEGP –Software Design Document\_v.0.2.docx** | **0.2**   * Acronyms * Class Diagram * Method description | Draft | July 6, 2014 | NS, PW, CD | NS, PW | NS, PW |
| **OEGP –Software Design Document\_v.1.0.docx** | **1.0**   * Update the figure number | Release | July 6, 2014 | NS, PW, CD | NS, PW | NS, PW |
| **OEGP-Software Design Document\_v1.1** | **1.1**   * Update substance, excipient and formulation class diagram and sequence diagram * Remove substance Properties class diagram and sequence diagram * Add class diagram and sequence at client part * Redesign UI | Draft | October 16, 2014 | NS, PW, CD | NS, PW | NS, PW |

Contents

[Document History ii](#_Toc401343944)

[Chapter 1 | Introduction 1](#_Toc401343945)

[1.1. Identification 1](#_Toc401343946)

[1.2. Acronyms 2](#_Toc401343947)

[Chapter 2 | Use Case and class diagram relationship 3](#_Toc401343948)

[Chapter 3 | Class Diagram 4](#_Toc401343949)

[3.1. Sub-Feature 5: Manage the drug substance 4](#_Toc401343950)

[3.1.1. CD-CI- 01: Substance Class Diagram (Client Side) 4](#_Toc401343951)

[3.1.2. CD-SV-01: Substance Class Diagram (Server Side) 9](#_Toc401343952)

[3.2. Sub-Feature 6: Manage the drug Excipient 15](#_Toc401343953)

[3.2.1. CD-CI- 02: Excipient Class Diagram (Client Side) 16](#_Toc401343954)

[3.2.2. CD-SV-02: Excipient Class Diagram (Server Side) 21](#_Toc401343955)

[3.3. Sub-Feature 7: Manage the drug formulation 25](#_Toc401343956)

[3.3.1. CD-CI-03: Formulation Class diagram (Client Side) 25](#_Toc401343957)

[3.3.2. CD-SV-03: Formulation Class Diagram (Server Side) 29](#_Toc401343958)

[3.4. Sub-Feature 2: Calculate the drug reformulation by using the inference engine. 33](#_Toc401343959)

[3.4.1. CD-CI- 04: Reformulation Class Diagram (Client Side) 33](#_Toc401343960)

[3.4.2. CD-SV-04: Reformulation Class Diagram (Server Side) 38](#_Toc401343961)

[3.5. Sub-Feature 3: View the drug reformulation history. 44](#_Toc401343962)

[3.5.1. CD-CI- 05: Reformulation History Class Diagram (Client Side) 45](#_Toc401343963)

[3.5.2. CD-SV-05: Reformulation History Class Diagram (Server Side) 48](#_Toc401343964)

[Chapter 4 | Sequence Diagram 52](#_Toc401343965)

[4.1. Sub-Feature 6: Manage the drug substance 53](#_Toc401343966)

[4.1.1. URS-09: The user adds a new substance to the system. 53](#_Toc401343967)

[4.1.2. URS-10: The user updates an existing substance in the system. 55](#_Toc401343968)

[4.1.3. URS-11: The user deletes an existing substance from the system. 57](#_Toc401343969)

[4.1.4. URS-12: The user views the substance in the system. 59](#_Toc401343970)

[4.2. Sub-Feature 7: Manage the drug excipient 61](#_Toc401343971)

[4.2.1. URS-13: The user adds a new excipient to the system. 61](#_Toc401343972)

[4.2.2. URS-14: The user updates an existing excipient in the system. 63](#_Toc401343973)

[4.2.3. URS-15: The user deletes an existing excipient from the system. 65](#_Toc401343974)

[4.2.4. URS-16: The user views the excipient in the system. 67](#_Toc401343975)

[4.3. Sub-Feature 8: Manage the drug formulation 69](#_Toc401343976)

[4.3.1. URS-17: The user adds a new drug’s formulation to the system. 69](#_Toc401343977)

[4.3.2. URS-18: The user updates an existing drug’s formulation in the system. 71](#_Toc401343978)

[4.3.3. URS-19: The user deletes an existing drug’s formulation from the system. 73](#_Toc401343979)

[4.3.4. URS-20: The user views the drug’s formulation in the system. 75](#_Toc401343980)

[4.4. Sub-Feature 2: Calculate the drug reformulation by using the inference engine. 77](#_Toc401343981)

[4.4.1. URS-06: The user calculates a drug reformulation by using an inference engine. 77](#_Toc401343982)

[4.5. Sub-Feature 3: View the drug reformulation history 79](#_Toc401343983)

[4.5.1. URS-07: The user views their drug reformulation history. 79](#_Toc401343984)

[Chapter 5 | User Interface 81](#_Toc401343985)

[5.1- Sub-Feature 5: Manage the drug substance property 82](#_Toc401343986)

[5.1.1 URS-09: The user adds a new substance property into the system. 82](#_Toc401343987)

[5.1.2 URS-10: The user updates an existing substance property into the system. 85](#_Toc401343988)

[5.1.3 URS-11: The user deletes an existing substance property from the system. 86](#_Toc401343989)

[5.2- Sub-Feature 6: Manage the drug substance 87](#_Toc401343990)

[5.2.1 URS-12: The user adds a new substance to the system. 87](#_Toc401343991)

[5.2.2 URS-13: The user updates an existing substance in the system. 89](#_Toc401343992)

[5.2.3 URS-14: The user deletes an existing substance from the system. 90](#_Toc401343993)

[5.2.4 URS-15: The user views the substance in the system 90](#_Toc401343994)

[5.3- Sub-Feature 7: Manage the drug excipient 91](#_Toc401343995)

[5.3.1 URS-16: The user adds a new excipient to the system. 91](#_Toc401343996)

[5.3.2 URS-17: The user updates an existing excipient in the system. 93](#_Toc401343997)

[5.2.3 URS-18: The user deletes an existing excipient from the system. 93](#_Toc401343998)

[5.2.4 URS-19: The user views the excipient in the system 93](#_Toc401343999)

[5.4- Sub-Feature 8: Manage the drug formulation 94](#_Toc401344000)

[5.4.1 URS-20: The user adds a new formulation to the system. 94](#_Toc401344001)

[5.4.2 URS-21: The user updates a tablet formulation in the system. 96](#_Toc401344002)

[5.4.3 URS-22: The user deletes an existing tablet formulation from the system. 97](#_Toc401344003)

[5.4.4 URS-23: The user views the tablet formulation in the system 97](#_Toc401344004)

# Introduction

## Identification

Software Design Document (SDD) is translated the requirement into the detail design. The SDD also explains the system architecture in detail. The purposes of the description are making a same understanding about the system. The software design consists of the list of Sub-Feature in a software requirement specification that is illustrated in Table 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Sub-Feature** | **URS No.** | **URS Name** | **Actor** |
| 2 | Calculate the drug reformulation by using the inference engine. | URS-06 | The user calculates a drug reformulation by using an inference engine. | General Pharmacists, Expert Pharmacists, Administrator |
| 3 | View the drug reformulation history | URS-07 | The user views their drug reformulation history. | General Pharmacists, Expert Pharmacists, Administrator |
| 5 | Manage the drug substance | URS-09 | The user adds a new substance into the system. | Expert Pharmacists, Administrator |
| URS-10 | The user updates an existing substance into the system. |
| URS-11 | The user deletes an existing substance from the system. |
| URS-12 | The user views the substance in the system. |
| 6 | Manage the drug excipient | URS-13 | The user adds a new excipient to the system. | Expert Pharmacists, Administrator |
| URS-14 | The user updates an existing drug excipient in the system. |
| URS-15 | The user delete an existing drug excipient in the system. |
| URS-16 | The user views all the drug excipient in the system. |
| 7 | Manage the drug formulation | URS-17 | The user adds a new drug formulation case into the system. | Expert Pharmacists, Administrator |
| URS-18 | The user updates an existing drug formulation case in the system. |
| URS-19 | The user deletes an existing drug formulation case in the system. |
| URS-20 | The user views all of the formulation in the system. |

**Table 1: The list of Sub-Feature in software requirement specification**

## Acronyms

|  |  |
| --- | --- |
| OEGP | Ontology-based Expert System for a Generic Drug Production of Pharmaceutical Dosage Forms |
| PDPO | Pharmaceutical Dosage Form Production Ontology |
| URS | User Requirement Specification |
| SRS | System Requirement Specification |
| CD | Class diagram |
| CD-CI | Class diagram For Client Side |
| CD-SV | Class diagram For Server Side. |
| SQD | Sequence Diagram |
| SQD-CI | Sequence Diagram For Client Side |
| SQD-SV | Sequence Diagram For Server Side |
| UI | User interface |
| SDD | Software Design Document |

# Use Case and class diagram relationship

In the 1st progress, the Sub-Feature in the table 1 is related with the class diagram that show on the list below this passage.

**Feature 5: PDPO System**

**Sub-Feature 6: Manage the drug substance**

* **Client Side**
* CD–CI-01 : Substance Class Diagram (Client Side)
* **Server Side**
* CD-SV-01 : Substance Class Diagram (Server Side)

**Sub-Feature 7: Manage the drug excipient**

* **Client Side**
* CD–CI-02 : Excipient Class Diagram (Client Side)
* **Server Side**
* CD-SV-02 : Excipient Class Diagram (Server Side)

**Sub-Feature 8: Manage the drug formulation**

* **Client Side**
* CD–CI-03 : Formulation Class Diagram (Client Side)
* **Server Side**
* CD-SV-03 : Formulation Class Diagram (Server Side)

**Feature 2: Rule Base System and Feature 4: Case Base Reasoning System**

**Sub-Feature 2: Calculate the drug reformulation by using the inference engine.**

* **Client Side**
* CD–CI-04 : Reformulation Class Diagram (Client Side)
* **Server Side**
* CD-SV-04 : Reformulation Class Diagram (Server Side)

**Sub-Feature 3: View the drug reformulation history.**

* **Client Side**
* CD–CI-05 : Reformulation History Class Diagram (Client Side)
* **Server Side**
* CD-SV-05 : Reformulation History Class Diagram (Server Side)

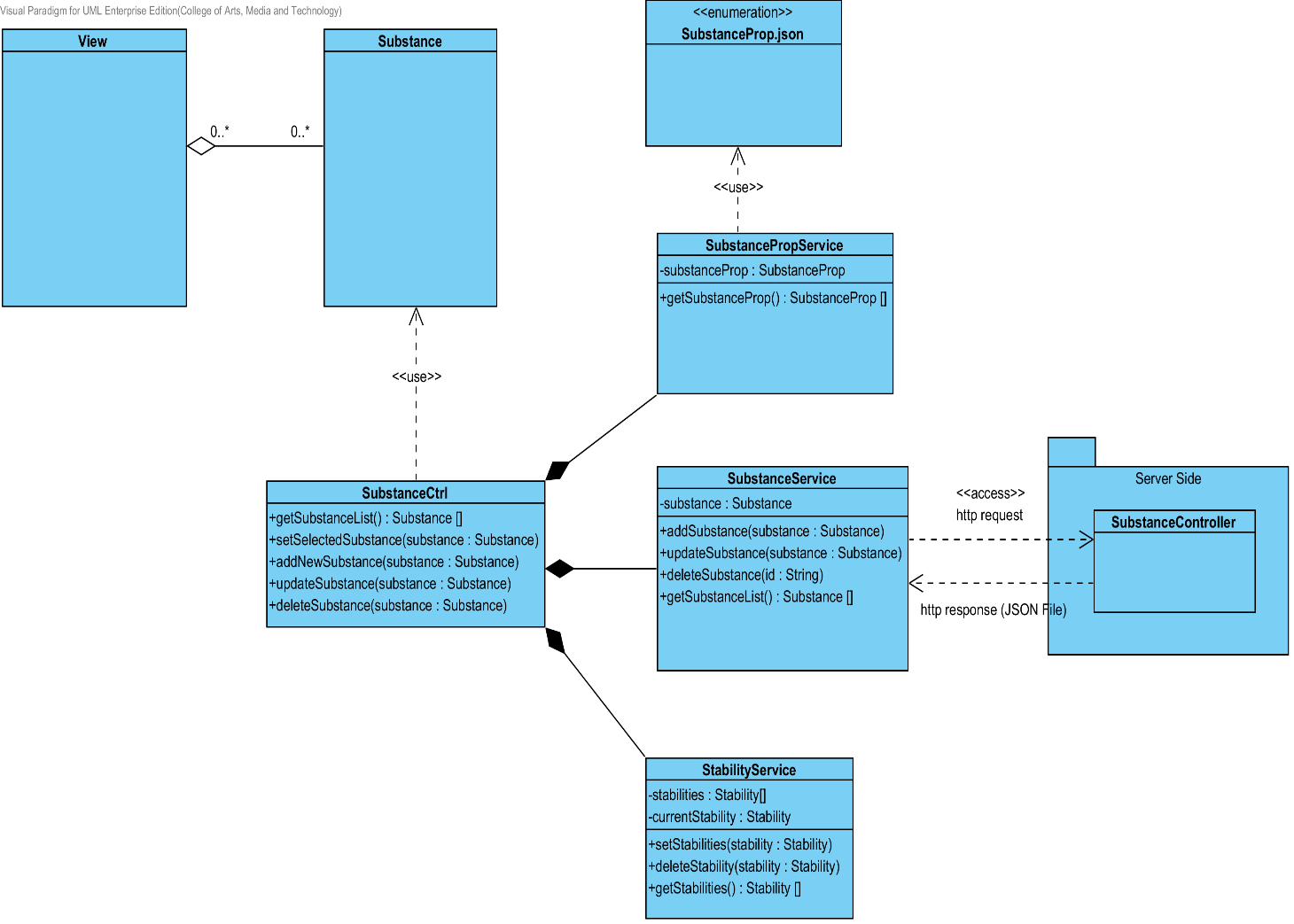
# Class Diagram

## Sub-Feature 5: Manage the drug substance

The Sub-Feature 5 is the substance management. The user can add, update, delete and view the substance. The substance is a part of excipient and formulation.

### CD-CI- 01: Substance Class Diagram (Client Side)

##### Class diagram

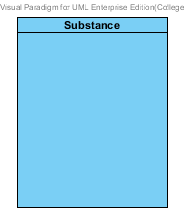


**Figure 1: CD-CI-01 – Substanec Class Diagram (Client Side)**

##### Class description

From the Figure number 1.It can divide into 5 Classes .The detail of each class is described on the next paragraph

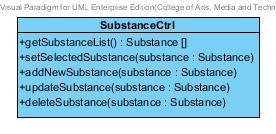
###### Substance Class



**Figure 2: Substance Class (Client Side)**

The Substance Class in Client Side is the model class that used for receiving substance data from the user and showing a substance data to the user. This model is controlled by SubstanceCtrl.

###### SubstanceCtrl



**Figure 3 : SubstanceCtrl (Client Side)**

The SubstanceCtrl is the substance controller that used for controlling the substance data that receive from the user or server. The SubstanceCtrl consists of 5 method. The detail of each method are shown on next paragraph.

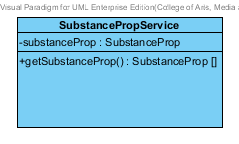
Attribute Description

N/A

Method Description

* + - **getSubstanceList() : Substance[]** - This method is used for getting the list of substance from the service. The method will return the list of substance object. On the other hand, if there are not any substance object in the database. The SubstanceCtrl will return null.
    - **setSelectedSubstance(Substance substance) –** This method is used for setting the substance data that selected by the user. The selected substance can use for updating or deleting substance object.
    - **addNewSubstance (Substance substance) –** This method is used for adding a new substance data to the database. The method will receive data from the user.
    - **UpdateSubstance(Substance substance) –** This method is used for updating the existing substance that user selected.
    - **DeleteSubstance(Substance substance) –** This method is used for deleting the existing substance from the database.

###### SubstancePropService



**Figure 4 : SubstancePropService (Client Side)**

The SubstancePropService used for getting the substance properties data from the JSON file. The SubstanceCtrl use this class for setting the properties of substance object.

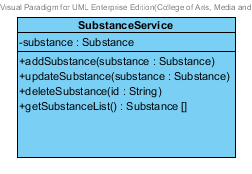
Attribute Description

N/A

Method Description

* **getSubstanceProp() : SubstanceProp[]** : The getSubstanceProp method is used for getting substance data from the JSON file.

###### SubstanceService



**Figure 5 : Substance Service (Client Side)**

The Substance Service is the service that use for adding, updating, deleting and getting data from the server. The SubstanceService can do its business by using http request such as POST, PUT, GET and DELELTE.

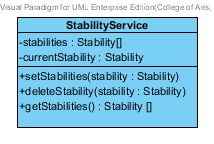
Attribute Description

* **Substance** – The Substance object that receive from SubstanceCtrl and get from the server.

Method Description

* **addSubstance (substance: Substance)** – This method is used for adding a new substance to database. The addSubstance method send the data that user input to the server by POST request.
* **updateSubstance (substance : Substance) –** This method is used for updating the existing substance object. The updateSubstance method send the edited data that user input to the server by PUT request.
* **deleteSubstance (id: String) –** This method is used for deleting the existing substance object. The deleteSubstance method send the id of the substance object to the server by DELETE request.
* **getSubstanceList() : Substance [] –** This method is used for getting the substance object from the database. This method send the GET request to server for getting the data.

###### StabilityService



**Figure 6 : StabilityService (Client Side)**

The StabilityService is the service that use for setting, deleting and getting the stability object to the list.

Attribute Description

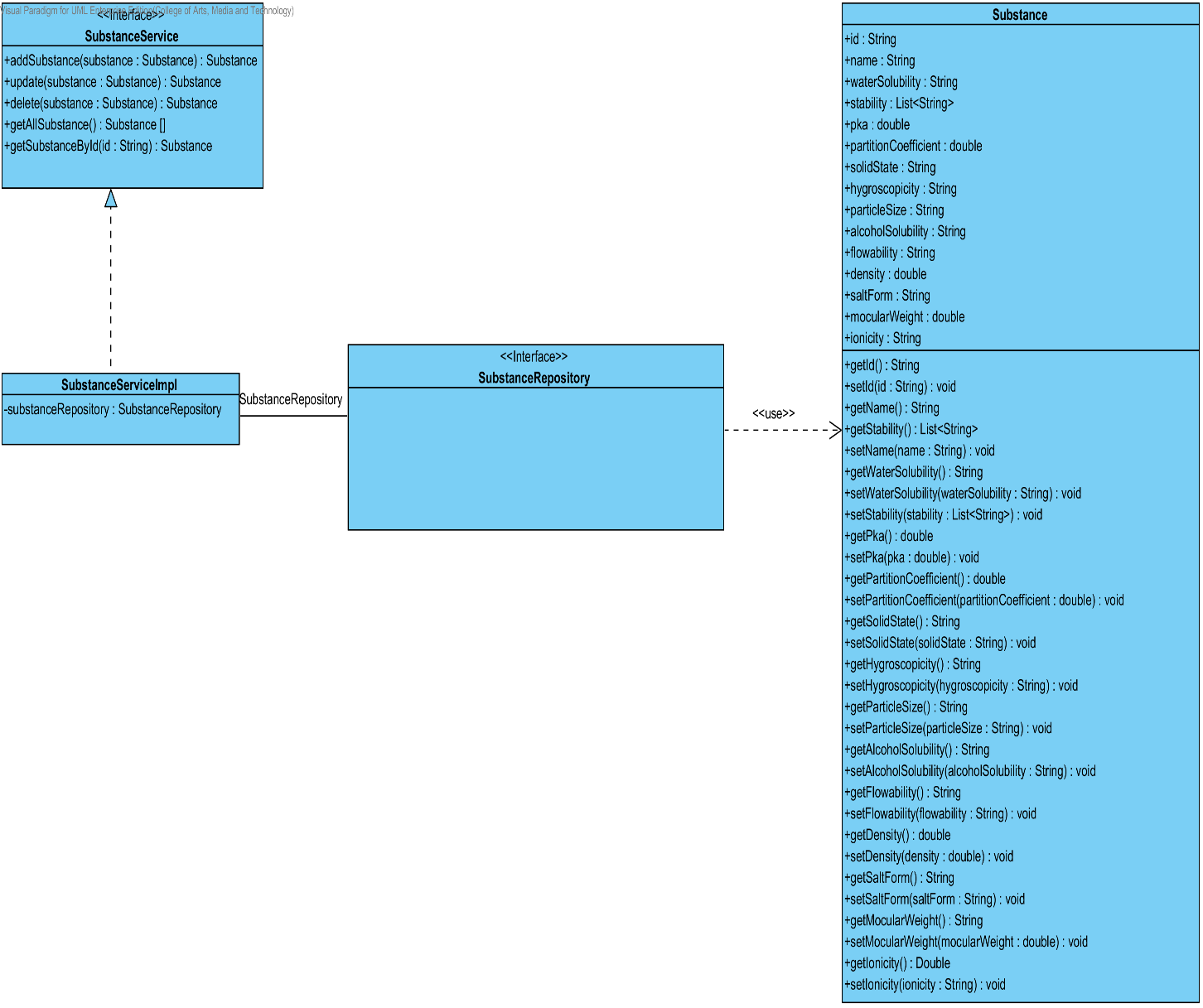
* **Stabilities** – The list of stability object that receive from the server.
* **currentStability** – The stability that user want to add to the substance object

Method Description

* **setStabilities(stability: Stability)** - This method use for setting the stability that user input.
* **deleteStability(stability : Stability)** - This method user for deleting the stability that user input.
* **getStabilities () : Stabilities []** – This method is used when getting the stability data that user input.

### CD-SV-01: Substance Class Diagram (Server Side)

##### Class diagram

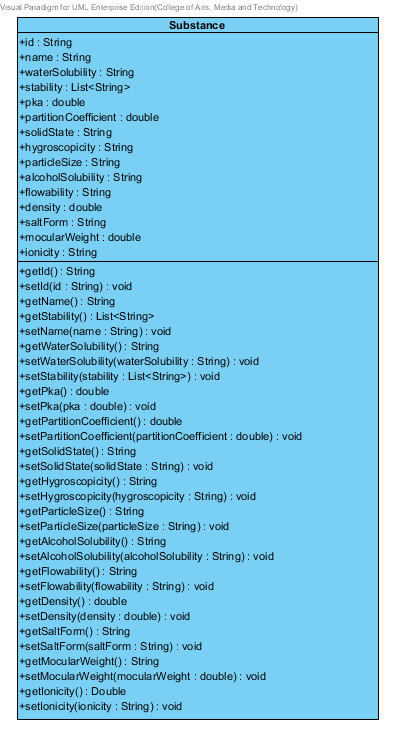


**Figure 7 : CD-SV-01 – Substance Class Diagram (Server Side).**

##### Class description

From the figure 55, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

###### Substance class



**Figure 8 : Substance Class (Server Side)**

Substance is a part of drug’s excipient. Substance class is an entity class that will be saved to the system. The substance class consists of 15 attributes follow the list below this passage.

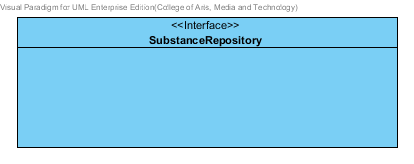
Attribute description

* + - **Id** – the identity of the substance class. Id attribute is a String.
    - **Name** – the name of the substance class. The substance name is String.
    - **waterSolubility** – the soluble of water in each substance. The water solubility is String.
    - **Stability –** the stability of substance. The stability attribute is the list of String.
    - **pKa –** the dissociation constant of the drug. It used for estimating the result of drug. The pKa is a double number.
    - **PartitionCoefficient** – the partition coefficient is the distribution between substances. The partition coefficient is double number.
    - **Solid State**- the statement that change to solid. The solid state is String.
    - **Hygroscopicity** – the ability to absorb the moisture from the air and water. The hygroscopicity is the String
    - **ParticleSize** – the size of particle in each substance. The particle size is String.
    - **AlcoholSolubility**- the soluble of alcohol in each substance. The alcohol solubility is String.
    - **FlowAbility –** the format of flowing that estimate from the medium. The flow ability is the String.
    - **Density** – the ration between mass and volume. The density is the double number.
    - **SaltForm –** the form of salt that used in each substance. The salt form is String.
    - **MolecularWeight** – the weight of each molecular in each substance**.** The molecular weight is double number.
    - **Ionicity –** the ionicity of substance. The ionic is String.

Method description

* + - **Getter and Setter method** – It used when the system set value and get value.

###### SubstanceRepository Interface



**Figure 9 : SubstanceRepository (Server Side)**

SubstanceRepository Interface is an interface that use for CRUD with entity classes in the system. All of SubstanceRepository interface’s method is generated from Spring Data MongoDB framework. SubstanceRepository interface consists of 4 methods is shown below this passage.

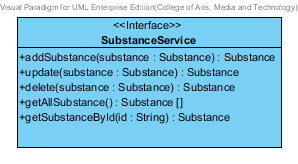
Attribute description

N/A

Method description

* + - **Save (substance: Substance)** - The save method is generated from Spring Data MongoDB framework. This method is used when the user wants to add a new substance or update existing substance to the system. The input variable is a substance object.
    - **Delete (substance: Substance)** – The delete method is generated from Spring Data MongoDB framework. This method is used when the user wants to delete the substance from the system. The input variable is substance object.
    - **findAll (): Substance []** – The findAll method is generated from Spring Data MongoDB framework. This method is used when the user wants to retrieve all of substance data from the system. The result of this method is a list of substance object.
    - **findOne (id: String): Substance** – The findOne method is used when the user wants to retrieve the substance data from the system. The system gets a substance object by the id of substance.

###### SubstanceService



**Figure 10 : SubstanceService (Server Side)**

SubstanceService is business processing logic for substance entity. SubstanceService manages the substance data through the SubstanceRepository interface. SubstanceService consists of 6 methods follow the list below this passage.

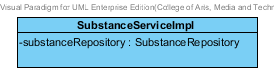
Attribute description

N/A

Method description

* + - **addSubstance (substance: Substance) : Substance** – The adding substance method is used, when the user wants to add a new substance to the database. This method adds a new substance by input variable of substance object. If the substance object that input by the user is not contained in the database, this method will add a new substance to the database and return the substance object from the database to the user after the adding substance is successful. On the other hand, when the substance object that input by the user is contained in the database. This method will return a null value to the user.
    - **updateSubstance (substance: Substance): Substance** - The updating substance method is used, when the user wants to update an existing substance on in the database. This method update the existing substance by input variable of substance object. If the substance object that input by the user is contained in the database, this method will update an existing substance in the database and return the substance object from the database to the user after the updating substance is successful. On the other hand, when the substance object that input by the user is not contained in the database. This method will return a null value to the user.
    - **deleteSubstance (substance: Substance) : Substance** – The deleting substance method is used when the user wants to deletes the existing substance from the database. This method delete the substance by input variable of substance object. If the substance object that input by the user is contained in the database, this method will delete an existing substance from the database and return the substance object to the user after the deleting substance is successful. On the other hand, when the substance object that input by the user is not contained in the database. This method will return a null value to the user.
    - **getAllSubstance() : Substance []** – The getAllSubstance method is used, when the user wants to get all substance data in the database. This method is return as a list of substance object database.
    - **getSubstanceById(id String) : Substance** – The getSubstanceById method is used, when the user wants to get the substance data in the system. This method gets substance object from the database by id that input by the user. On the other hand, if the id that input by user is not contained in the database. This method will return null value to the user.

###### SubstanceServiceImpl



**Figure 11 : SubstanceServiceImpl (Server Side)**

SubstanceServiceImpl is the substance service class that implements the method from SubstanceService. So, the method of SubstanceServiceImpl is same as SubstanceService.

Attribute description

* + - **SubstanceRepository** – the repository of substance. This attribute is used for substance data management.

Method description

Same as SubstanceService

## Sub-Feature 6: Manage the drug Excipient

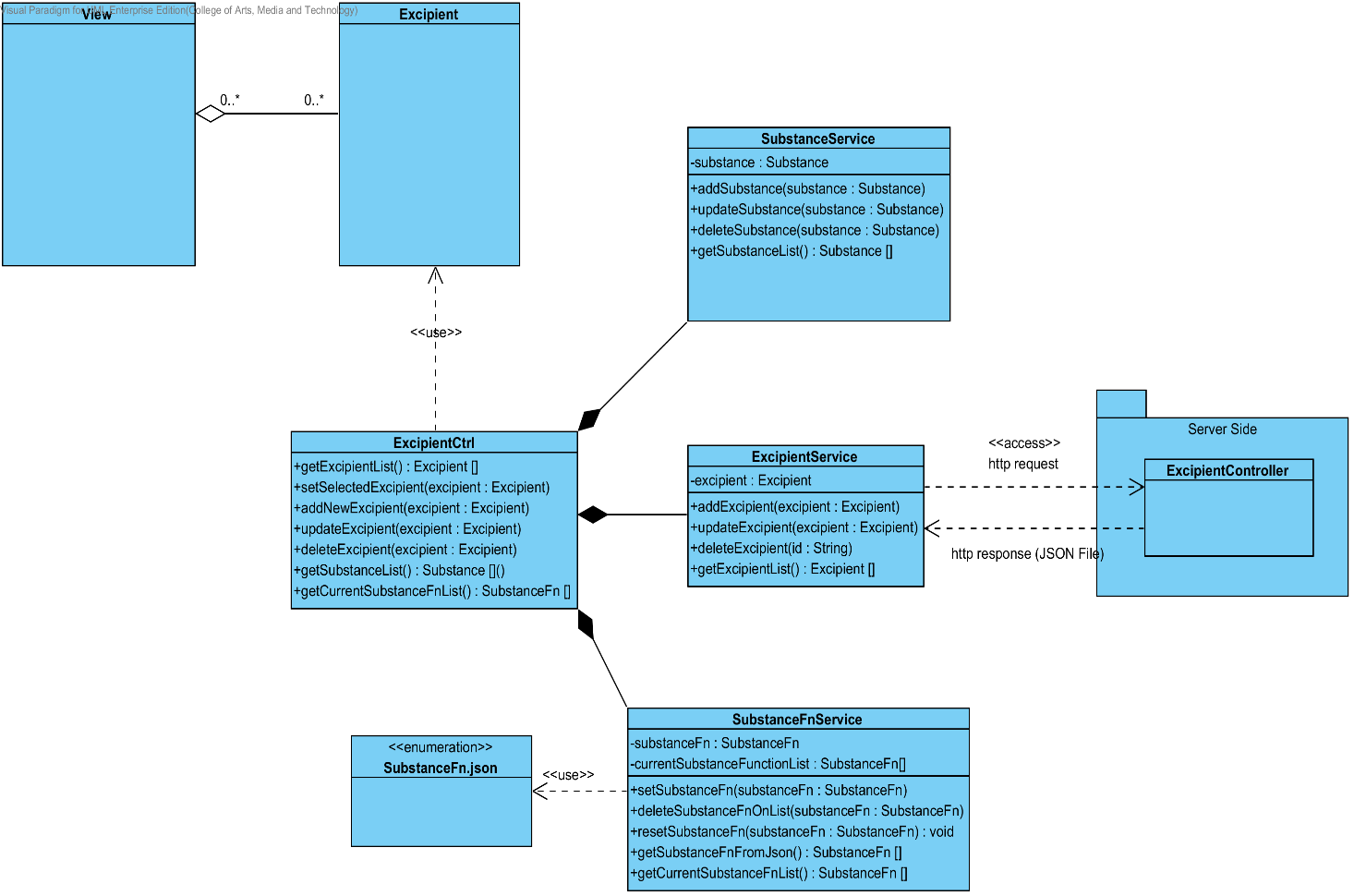
The Sub-Feature 6 is the excipient management. The user can add, update, delete and view the excipient. Each excipient is created from the substance and substanceFunction. The relationship between excipient entity, substance entity, and substanceFunction is illustrated in the Figure 60 below on this passage.



**Figure 12: The Entity Relationship between Excipient, Substance Function and Substance**

### CD-CI- 02: Excipient Class Diagram (Client Side)

##### Class diagram

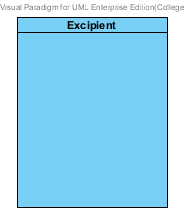
****

**Figure 13: CD-CI-02 – Excipient Class Diagram (Client Side).**

##### Class description

From the Figure number 13 .It can divide into 5 Classes .The detail of each class is described on the next paragraph

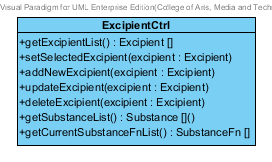
###### Excipient Class



**Figure 14: Excipient Class (Client Side)**

The Excipient Class in Client Side is the model class that used for receiving excipient data from the user and showing an excipient data to the user. This model is controlled by ExcipientCtrl.

###### ExcipientCtrl



**Figure 15 : ExcipientCtrl (Client Side)**

The ExcipientCtrl is the excipient controller that used for controlling the excipient data that receive from the user or server. The ExcipientCtrl consists of 5 method. The detail of each method are shown on next paragraph.

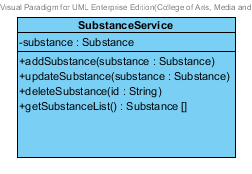
Attribute Description

N/A

Method Description

* + - **getExcipientList() : Excipient[]** - This method is used for getting the list of excipient from the server. The method will return the list of excipient object. On the other hand, if there are not any excipient object in the database. The ExcipientCtrl will return null.
    - **setSelectedExcipient(excipient : Excipient) –** This method is used for setting the excipient data that selected by the user. The selected excipient can use for updating or deleting excipient object.
    - **addNewExcipient (excipient : Excipient) –** This method is used for adding a new excipient data to the database. The method will receive data from the user.
    - **UpdateExcipient(excipient : Excipient) –** This method is used for updating the existing excipient that user selected.
    - **DeleteExcipient(excipient : Excipient) –** This method is used for deleting the existing excipient from the database.
    - **getExcipientList() : Excipient[]** - This method is used for getting the list of excipient from the service. The method will return the list of excipient object. On the other hand, if there are not any excipient object in the database. The getExcipientList will return null.
    - **getCurrentSubstanceFnList() : SubstanceFn []** - This method is used for getting substance function object from the substanceFnService.

###### SubstanceService



**Figure 16 : Substance Service (Client Side)**

The Substance Service is used for adding, updating, deleting and getting data from the server. The ExcipientCtrl will use this service for creating excipient object.

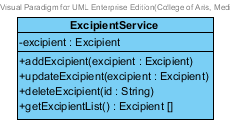
Attribute Description

* **Substance** – The Substance object that receive from SubstanceCtrl and get from the server.

Method Description

* **addSubstance (substance: Substance)** – This method is used for adding a new substance to database. The addSubstance method send the data that user input to the server by POST request.
* **updateSubstance (substance : Substance) –** This method is used for updating the existing substance object. The updateSubstance method send the edited data that user input to the server by PUT request.
* **deleteSubstance (id: String) –** This method is used for deleting the existing substance object. The deleteSubstance method send the id of the substance object to the server by DELETE request.
* **getSubstanceList() : Substance [] –** This method is used for getting the substance object from the database. This method send the GET request to server for getting the data.

###### ExcipientService



**Figure 17 : Excipient Service (Client Side)**

The Excipient Service is the service that use for adding, updating, deleting and getting data from the server. The ExcipientService can do its business by using http request such as POST, PUT, GET and DELELTE.

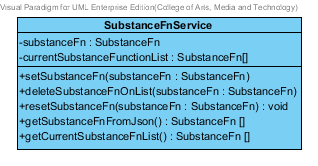
Attribute Description

* **Excipient** – The Excipient object that receive from ExcipientCtrl and get from the user.

Method Description

* **addExcipient (excipient: Excipient)** – This method is used for adding a new excipient to database. The addExcipient method send the data that user input to the server by POST request.
* **updateExcipient (excipient : Excipient) –** This method is used for updating the existing excipient object. The updateExcipient method send the edited data that user input to the server by PUT request.
* **deleteExcipient (id: String) –** This method is used for deleting the existing excipient object. The deleteExcipient method send the id of the excipient object to the server by DELETE request.
* **getExcipientList() : Excipient [] –** This method is used for getting the excipient object from the database. This method send the GET request to server for getting the data.

###### SubstanceFnService



**Figure 18 : SubstanceFnService (Client Side)**

The SubstanceFnService used for getting the substance function data from the JSON file. The ExcipientCtrl use this class for setting the substance function of excipient object.

Attribute Description

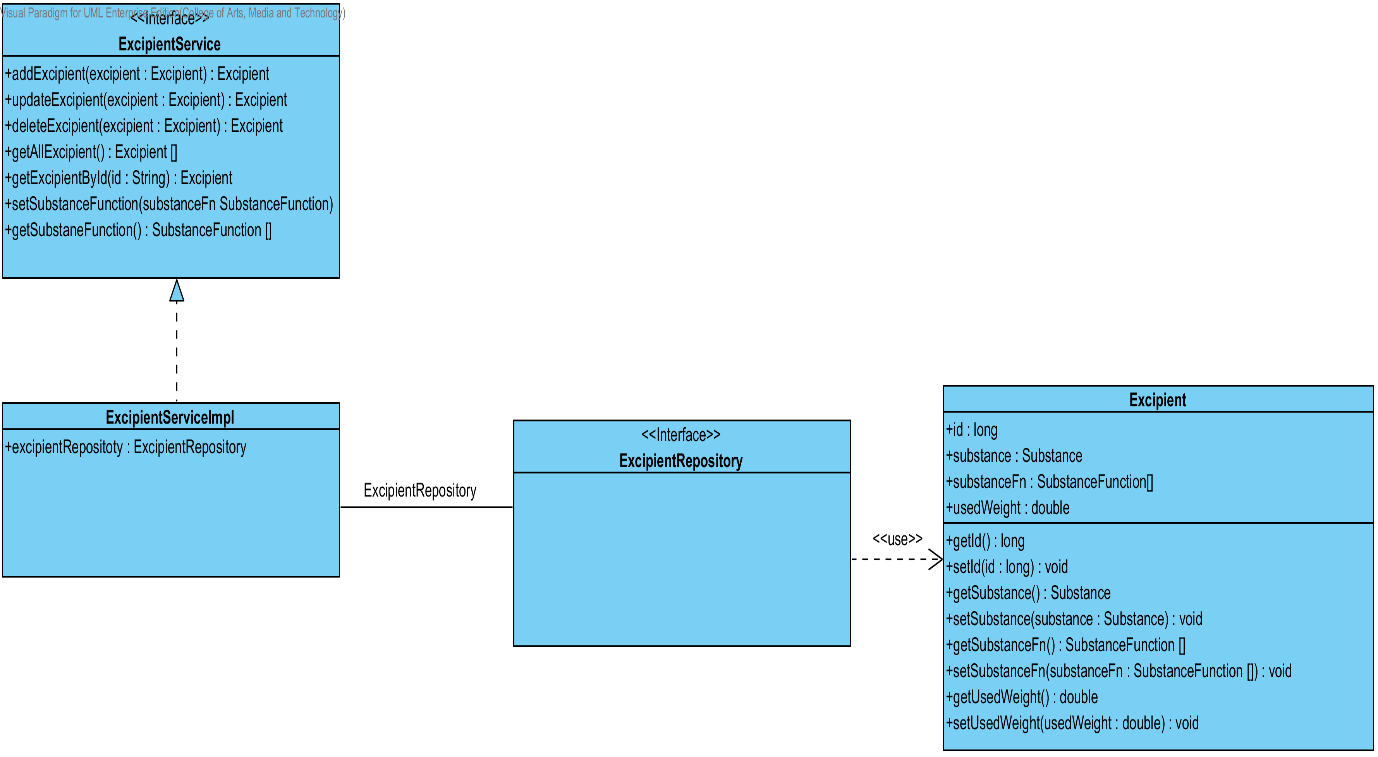
* **SubstanceFn** – Substance Function data that get from the JSON file.
* **CurrentSubstanceFunctionList** – Substance Function that user select to use in excipient object.

Method Description

* **setSubstanceFn (substanceFn : SubstanceFn)** – This method is used for setting the substance function object that user selected.
* **deleteSubstanceFnOnList (substanceFn : SubstanceFn)** – This method is used for deleting the substance function object from the current substance function list.
* **resetSubstanceFn(substanceFn : SubstanceFn)** – This method is used for reset the substance function in the list. Which mean the substance function list will be a null value.
* **getSubstanceFnFromJson() : SubstanceFn[]** – This method is used for getting the substance function from the JSON file

### CD-SV-02: Excipient Class Diagram (Server Side)

##### Class diagram

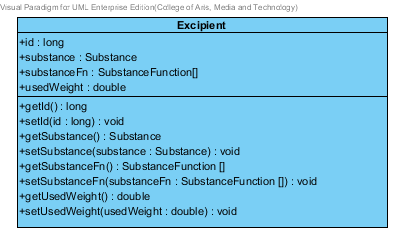


**Figure 19 : CD-SV-02 – Excipient Class Diagram (Server Side).**

##### Class description

From the figure 19, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

###### Excipient class



**Figure 20 : Excipient Class (Server Side)**

Excipient is a part of drug’s formulation. Excipient class is an entity class that will be saved to the system. The excipient class consists of 5 attributes follow the list below this passage.

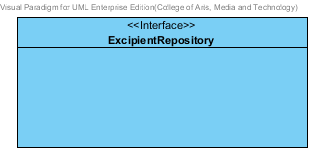
Attribute description

* + - **Id** – the identity of the excipient class. Id attribute is a long number.
    - **Substance** – the substance of the excipient class. The substance attribute is a substance object.
    - **SubstanceFn–**the substance Function of the excipient class .The user can set the substance Function in each excipient .The substance Function is a substance function object.
    - **usedWeight** – the weight of substance function that will be use in each excipient.

Method description

* + - **Getter and Setter method** – It used when the system set value and get value.

###### ExcipientRepository Interface



**Figure 21: Excipient Repository (Server Side)**

ExcipientRepository Interface is an interface that use for CRUD with entity classes in the system. The most of ExcipientRepository interface’s method is generated from Spring data MongoDB framework. ExcipientRepository interface consists of 4 methods is shown below this passage.

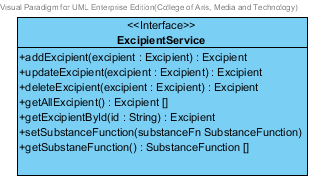
Attribute description

N/A

Method description

* + - **Save (excipient: Excipient)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new excipient or update existing excipient to the system. The input variable is an excipient object.
    - **Delete (excipient: Excipient)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the excipient from the system. The input variable is excipient object.
    - **findAll ():Excipient []** – The findAll method is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of excipient data from the system. The result of this method is a list of excipient object.
    - **findOne (id: String): Excipient** – The findOne method is used when the user wants to retrieve the excipient data from the system. The system gets an excipient object by the id of excipient.

###### ExcipientService



**Figure 22: Excipient Service (Server Side)**

ExcipientService is business processing logic for excipient entity. ExcipientService manages the excipient data through the ExcipientRepository interface. ExcipientService consists of 7 methods follow the list below this passage.

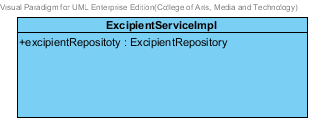
3.2.1.2.3.1: Attribute description

N/A

3.2.1.2.3.2: Method description

* + - **addExcipient (excipient: Excipient)** – The adding excipient method is used, when the user wants to add a new excipient to the database. This method adds a new excipient by input variable of excipient object. If the excipient object that input by the user is not contained in the database, this method will add a new excipient to the database and return the excipient object from the database to the user after the adding excipient is successful. On the other hand, when the excipient object that input by the user is contained in the database. This method will return a null value to the user.
    - **updateExcipient (excipient: Excipient)** - The updating excipient method is used, when the user wants to update an existing excipient on in the database. This method update the existing excipient by input variable of excipient object. If the excipient object that input by the user is contained in the database, this method will update an existing excipient in the database and return the excipient object from the database to the user after the updating excipient is successful. On the other hand, when the excipient object that input by the user is not contained in the database. This method will return a null value to the user.
    - **deleteExcipient (excipient: Excipient)** – The deleting excipient method is used when the user wants to deletes the existing excipient from the database. This method delete the excipient by input variable of excipient object. If the excipient object that input by the user is contained in the database, this method will delete an existing excipient from the database and return the excipient object to the user after the deleting excipient is successful. On the other hand, when the excipient object that input by the user is not contained in the database. This method will return a null value to the user.
    - **getAllExcipient() : Excipient []** – The getAllExcipient method is used, when the user wants to get all excipient data in the database. This method is return as a list of excipient object from the database.
    - **getExcipientById(id : String) :** Excipient – The getExcipientById method is used, when the user wants to get the excipient data in the system. This method gets excipient object from the database by id that input by the user. On the other hand, if the id that input by user is not contained in the database. This method will return null value to the user.
    - **setSubstanceFunction(substanceFn : SubstanceFn) –** This method is used for setting the substance function to the substance object.
    - **getSubstanceFunction() : SubstanceFunction[] –** This method is used for getting the substance function that set at setSubstanceFunction method.

###### ExcipientServiceImpl



**Figure 23: Excipient Service Implementation (Server Side)**

ExcipientServiceImpl is the excipient service class that implements the method from ExcipientService. So, the method of ExcipientServiceImpl is same as ExcipientService.

Attribute description

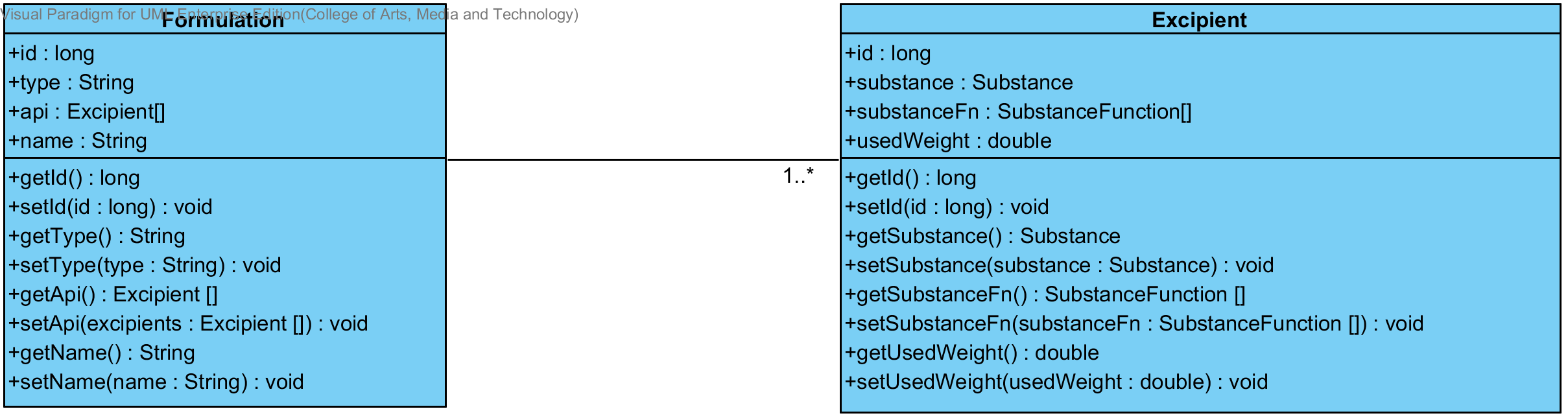
* + - **ExcipientRepository** – the repository of excipient. This attribute is used for excipient data management.

Method description

Same as ExcipientService

## Sub-Feature 7: Manage the drug formulation

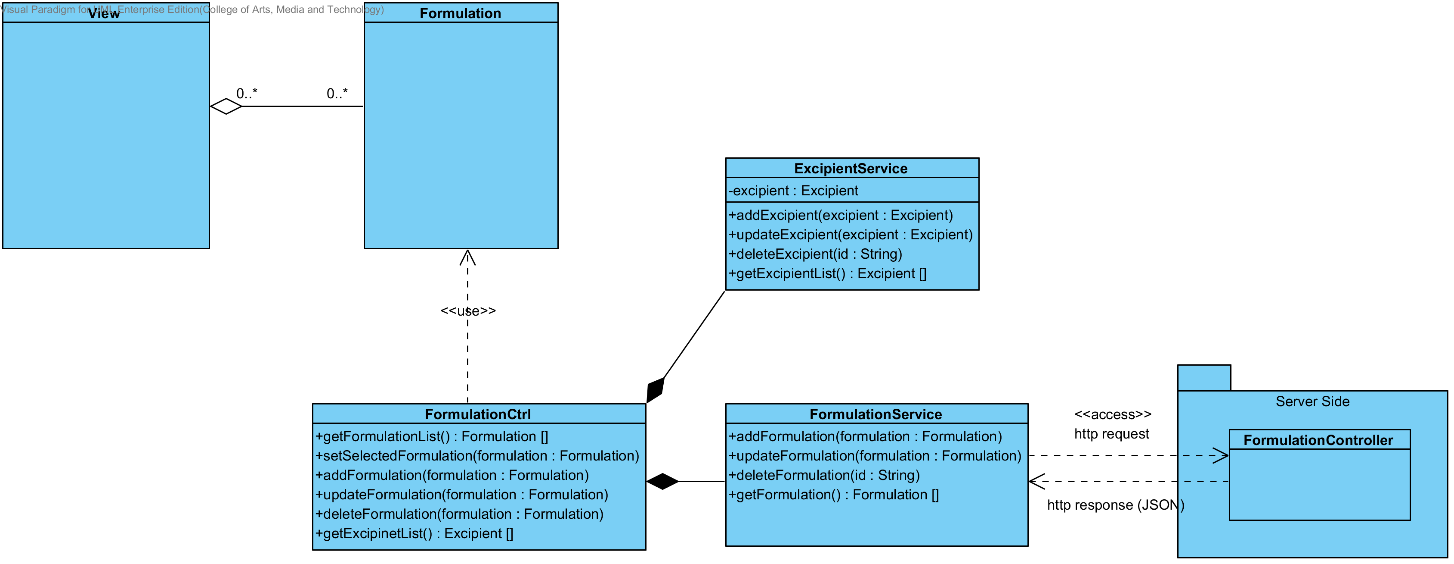
The Sub-Feature 7 is the drug’s formulation management. The users can add, update, delete and view the formulation. Each drug’s formulation is created from the excipient. The relationship between drug’s formulation and the excipient is illustrated in the Figure 24 below on this passage.



**Figure 24: Entity Relationship between Formulation and Excipient**

### CD-CI-03: Formulation Class diagram (Client Side)

##### Class diagram

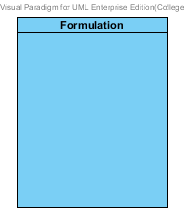


**Figure 25: CD-CI-03 – Formulation Class Diagram (Client Side).**

##### Class description

From the Figure number 25 .It can divide into 4 Classes .The detail of each class is described on the next paragraph

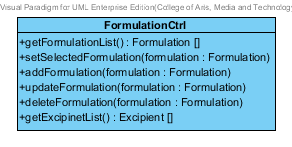
###### Formulation Class



**Figure 26: Formulation Class (Client Side)**

The Formulation Class in Client Side is the model class that used for receiving the formulation data from the user and showing the formulation data to the user. This model is controlled by FormulationCtrl.

###### FormulationCtrl



**Figure 27 : FormulationCtrl (Client Side)**

The FormulationCtrl is the formulation controller that used for controlling the formulation data that receive from the user or server. The FormulationCtrl consists of 5 method. The detail of each method are shown on next paragraph.

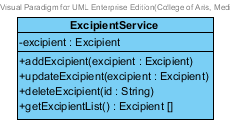
Attribute Description

N/A

Method Description

* + - **getFormulationList () : Formulation[]** - This method is used for getting the list of formulation from the server. The method will return the list of formulation object. On the other hand, if there are not any formulation object in the database. The getFormulationList method will return null.
    - **setSelectedFormulation (formulation : Formulation) –** This method is used for setting the formulation data that selected by the user. The selected formulation can use for updating or deleting formulation object.
    - **addFormulation (formulation : Formulation) –** This method is used for adding a new formulation data to the database. The method will receive data from the user.
    - **updateFormulation (formulation : Formulation) –** This method is used for updating the existing formulation that user selected.
    - **deleteFormulation (formulation : Formulation) –** This method is used for deleting the existing formulation from the database.
    - **getExcipientList() : Excipient []** - This method is used for getting the list of excipient from the service. The method will return the list of excipient object. On the other hand, if there are not any excipient object in the database. The getExcipientList will return null.

###### ExcipientService



**Figure 28 : Excipient Service (Client Side)**

The Excipient Service is the service that use for adding, updating, deleting and getting data from the server. The ExcipientService can do its business by using http request such as POST, PUT, GET and DELELTE.

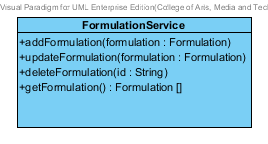
Attribute Description

* **Excipient** – The Excipient object that receive from ExcipientCtrl and get from the user.

Method Description

* **addExcipient (excipient: Excipient)** – This method is used for adding a new excipient to database. The addExcipient method send the data that user input to the server by POST request.
* **updateExcipient (excipient : Excipient) –** This method is used for updating the existing excipient object. The updateExcipient method send the edited data that user input to the server by PUT request.
* **deleteExcipient (id: String) –** This method is used for deleting the existing excipient object. The deleteExcipient method send the id of the excipient object to the server by DELETE request.
* **getExcipientList() : Excipient [] –** This method is used for getting the excipient object from the database. This method send the GET request to server for getting the data.

###### FormulationService



**Figure 29 : Formulation Service (Client Side)**

The Formulation Service is the service that use for adding, updating, deleting and getting data from the server. The FormulationService can do its business by using http request such as POST, PUT, GET and DELELTE.

Attribute Description

* N/A

Method Description

* **addFormulation (formulation : Formulation)** – This method is used for adding a new formulation to database. The addFormulation method send the data that user input to the server by POST request.
* **updateFormulation (formulation : Formulation)** – This method is used for updating the existing formulation object. The updateFormulation method send the edited data that user input to the server by PUT request.
* **deleteFormulation( formulation: Formulation)** – This method is used for deleting the existing formulation object. The deleteFormulation method send the id of the excipient object to the server by DELETE request.
* **getFormulation() : Formulation[] –** This method is used for getting the formulation object from the server by GET request.

### CD-SV-03: Formulation Class Diagram (Server Side)

##### Class diagram

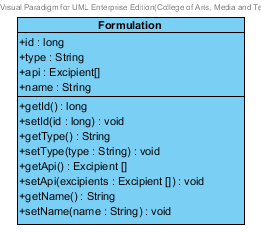


**Figure 30 : CD-SV-03 – Formulation Class Diagram (Server Side).**

##### Class description

From the figure 30, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

###### Formulation class



**Figure 31: Formulation Class (Server Side)**

Formulation class is an entity class that will be saved to the system. The Formulation class consists of 3 attributes follow the list below this passage.

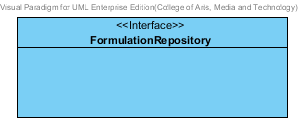
Attribute description

* + - **Id** – the identity of the Formulation class. Id attribute is a long number.
    - **Name** – the name of the Formulation class. The Formulation attribute is a Formulation object.
    - **Excipient –**the excipient of the Formulation. The user can set the excipient more than 1 excipient in each formulation.
    - **Type –** the type of each formulation. There will be “Solution” or “Tablet”.

Method description

* + - **Getter and Setter method** – It used when the system set value and get value.

###### FormulationRepository Interface



**Figure 32 : Formulation Repository (Server Side)**

FormulationRepository Interface is an interface that use for CRUD with entity classes in the system. The most of FormulationRepository interface’s method is generated from Spring data MongoDB framework. FormulationRepository interface consists of 4 methods is shown below this passage.

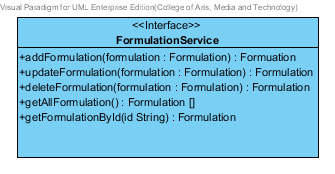
Attribute description

N/A

Method description

* + - **Save (formulation: Formulation)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new Formulation or update existing Formulation to the system. The input variable is the Formulation object.
    - **Delete (formulation: Formulation)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the Formulation from the system. The input variable is Formulation object.
    - **findAll ():Formulation []** – The findAll method is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of Formulation data from the system. The result of this method is a list of Formulation object.
    - **findOne (id: String): Formulation** – The findOne method is used when the user wants to retrieve the Formulation data from the system. The system gets a Formulation object by the id of Formulation.

###### FormulationService



**Figure 33: Formulation Service (Server Side)**

FormulationService is business processing logic for Formulation entity. FormulationService manages the Formulation data through the FormulationRepository interface. FormulationService consists of 6 methods follow the list below this passage.

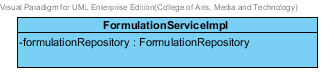
3.4.1.2.3.1: Attribute description

N/A

3.4.1.2.3.2: Method description

* + - **addFormulation (formulation: Formulation)** – The adding Formulation method is used, when the user wants to add a new Formulation to the database. This method adds a new Formulation by input variable of Formulation object. If the Formulation object that input by the user is not contained in the database, this method will add a new Formulation to the database and return the Formulation object from the database to the user after the adding Formulation is successful. On the other hand, when the Formulation object that input by the user is contained in the database. This method will return a null value to the user.
    - **updateFormulation (formulation: Formulation)** - The updating Formulation method is used, when the user wants to update an existing Formulation on in the database. This method update the existing Formulation by input variable of Formulation object. If the Formulation object that input by the user is contained in the database, this method will update an existing Formulation in the database and return the Formulation object from the database to the user after the updating Formulation is successful. On the other hand, when the Formulation object that input by the user is not contained in the database. This method will return a null value to the user.
    - **deleteFormulation (formulation:Formulation) -** The deleting Formulation method is used when the user wants to deletes the existing Formulation from the database. This method delete the Formulation by input variable of Formulation object. If the Formulation object that input by the user is contained in the database, this method will delete an existing Formulation from the database and return the Formulation object to the user after the deleting Formulation is successful. On the other hand, when the formulation object that input by the user is not contained in the database. This method will return a null value to the user.
    - **getAllFormulation() : Formulation []** – The getAllFormulation method is used, when the user wants to get all Formulation data in the database. This method is return as a list of Formulation object from the database.
    - **getFormulationById(id : String) : Formulation**  – The getFormulationById method is used, when the user wants to get the Formulation data in the system. This method gets Formulation object from the database by id that input by the user. On the other hand, if the id that input by user is not contained in the database. This method will return null value to the user.

###### FormulationServiceImpl



**Figure 34 : Formulation Service Implementation (Server Side)**

FormulationServiceImpl is the formulation service class that implements the method from FormulationService. So, the method of FormulationServiceImpl is same as FormulationService.

Attribute description

* + - **FormulationRepository** – the repository of Formulation. This attribute is used for Formulation data management.

Method description

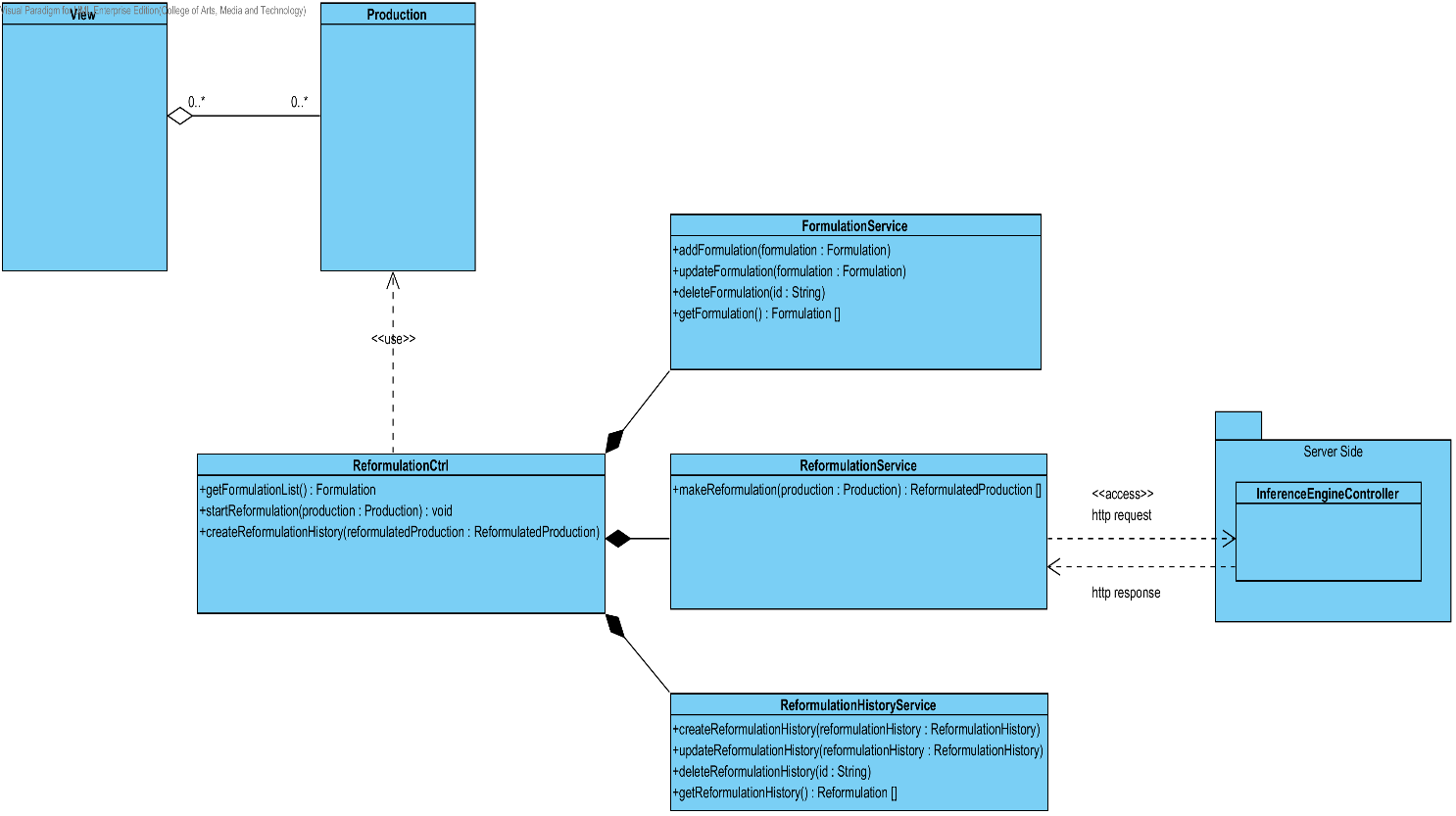
Same as FormulationService

## Sub-Feature 2: Calculate the drug reformulation by using the inference engine.

The Sub-Feature 2 is the making reformulation. The user can input the formulation and value for making reformulation by use the Rule base, Case Base Reasoning and Hybrid Reasoning. The user can save their reformulation to the history for viewing it after.

### CD-CI- 04: Reformulation Class Diagram (Client Side)

##### Class diagram

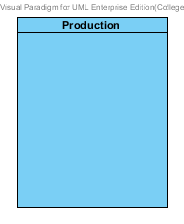
****

**Figure 35: CD-CI-04 – Reformulation Class Diagram (Client Side).**

##### Class description

From the Figure number 35 .It can divide into 5 Classes .The detail of each class is described on the next paragraph

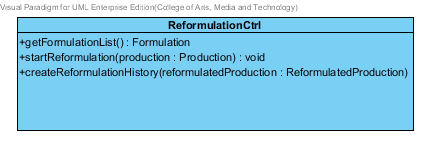
###### Production Class



**Figure 36: Production Class (Client Side)**

The Production Class in Client Side is the model class that used for receiving production data from the user and showing the production data to the user. This model is controlled by ReformulationCtrl.

###### ReformulationCtrl



**Figure 37 : ReformulationCtrl (Client Side)**

The Reformulation Ctrl is the controller that used for managing the production data when the user making reformulation. The ReformulationCtrl also get the production after reformulate process for showing result to the user. The ReformulationCtrl consists of 3 method. The detail of each method are shown on next paragraph.

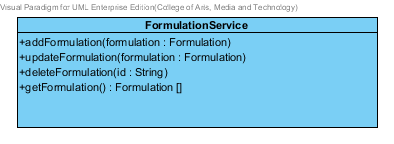
Attribute Description

N/A

Method Description

* + - **getFormulationList() : Formulation[]** – This method is used for getting the formulation list for making reformulation. On the other hand if there are not any formulation object it will return null.
    - **startReformulation(production : Production) –** This method is used for reformulating production. This method will get the reformulated production after reformulation.
    - **createReformulationHistory –** This method is used for creating reformulation history after the user get reformulation result.

###### FormulationService



**Figure 38 : Formulation Service (Client Side)**

The Formulation Service is the service that use for adding, updating, deleting and getting data from the server. The FormulationService can do its business by using http request such as POST, PUT, GET and DELELTE.

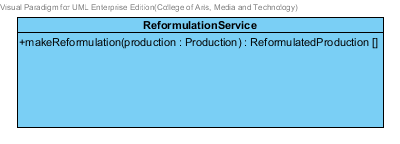
Attribute Description

* N/A

Method Description

* **addFormulation (formulation : Formulation)** – This method is used for adding a new formulation to database. The addFormulation method send the data that user input to the server by POST request.
* **updateFormulation (formulation : Formulation)** – This method is used for updating the existing formulation object. The updateFormulation method send the edited data that user input to the server by PUT request.
* **deleteFormulation( formulation: Formulation)** – This method is used for deleting the existing formulation object. The deleteFormulation method send the id of the excipient object to the server by DELETE request.
* **getFormulation() : Formulation[] –** This method is used for getting the formulation object from the server by GET request.

###### ReformulationService



**Figure 39 : Reformulation Service (Client Side)**

The Reformulation Service is the service that use for making drug reformulation with the server.

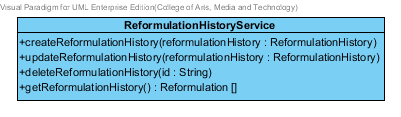
Attribute Description

* N/A

Method Description

* **makeReformulation (production : Production)** – This method is used for making drug reformulation. The method will send the production object to server by POST request. After that, the method will receive the reformulated production list from the server.

###### ReformulationHistoryService



**Figure 40 : ReformulationHistoryService (Client Side)**

The Reformulation History Service is the service that use for adding, updating, deleting and getting data from the server. The Reformulation History can do its business by using http request such as POST, PUT, GET and DELELTE.

Attribute Description

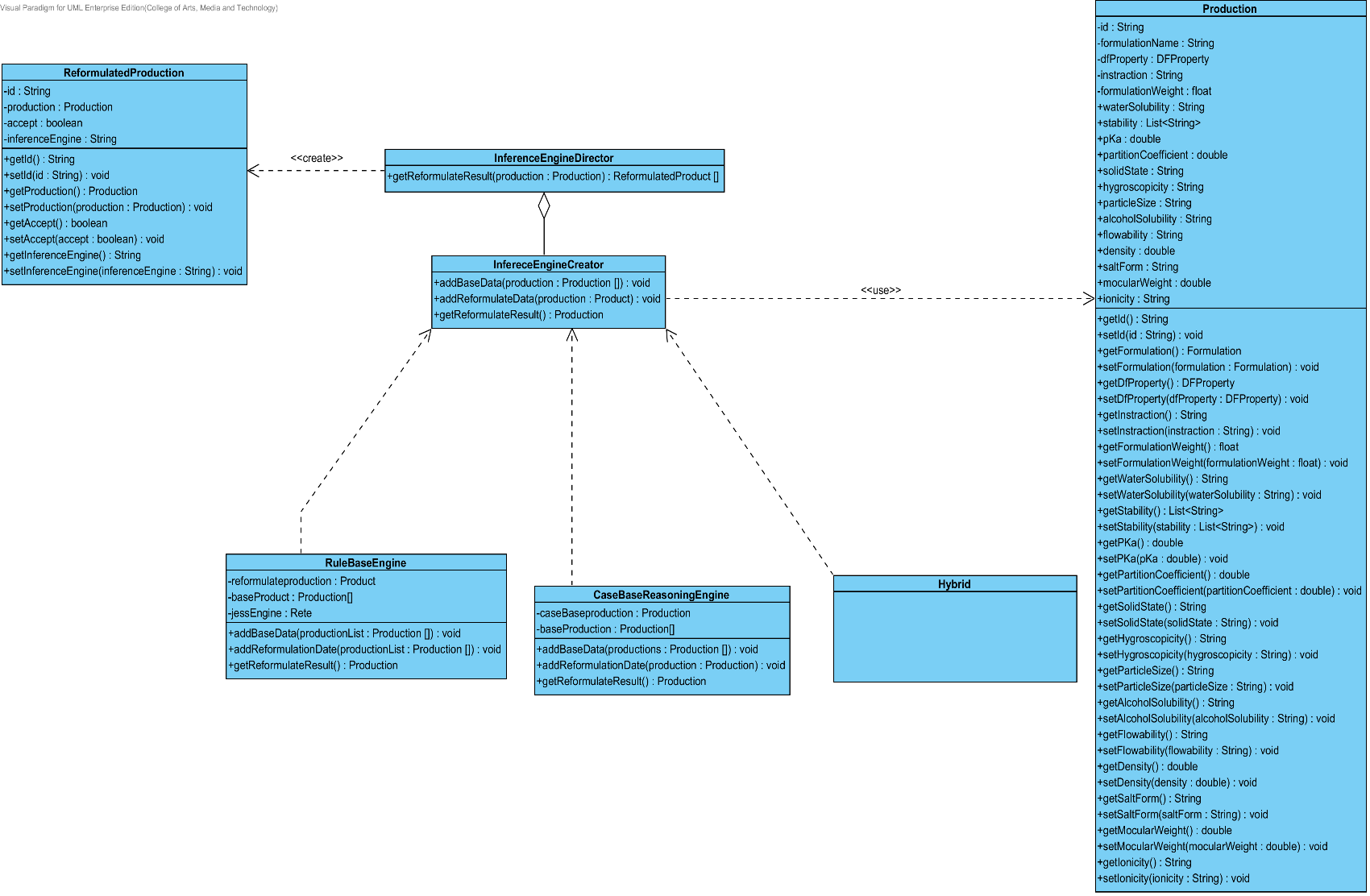
* N/A

Method Description

* **createReformulationHistory (reformulationHistory : ReformulationHistory)** – This method is used for adding a new reformulation history to database. The createReformulationHistory method sends the data that user input to the server by POST request.
* **updateReformulationHistory (substanceFn : SubstanceFn)** – This method is used for updating the existing reformulation history. The updateReformulationHistory method sends the edited data that user input to the server by PUT request.
* **deleteReformulationHistory (substanceFn : SubstanceFn)** This method is used for deleting the existing reformulation history. The deleteReformulationHistory method sends the id of the reformulation history id to the server by DELETE request.
* **getReformulationHistory : ReformulationHistory[]** – This method is used for getting the reformulation history from the server by GET request.

### CD-SV-04: Reformulation Class Diagram (Server Side)

##### Class diagram

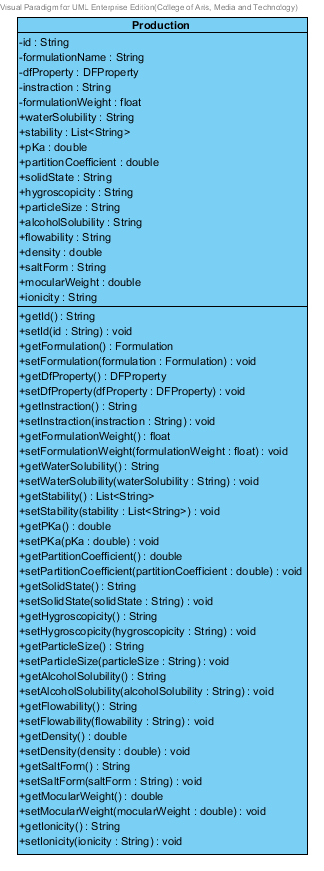


**Figure 41 : CD-SV-04 – Reformulation Class Diagram (Server Side).**

##### Class description

From the figure 41, it can divide into 7 important classes. The detail of each class is described on the next paragraph.

###### Production class



**Figure 42 : Production Class (Server Side)**

Production is a part of drug reformulation. Production object is created after reformulation successful. The production class consists of 17 attributes follow the list below this passage.

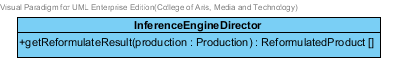
Attribute description

* + - **Id** – the identity of the production class. Id attribute is a long number.
    - **Formulation Name** – the name of formulation that use for creating a new production.
    - **DF Property -** The pharmaceutical value that come from the lab or a real drug.
    - **Instruction –** The instruction about how to create this production. The instruction is created after reformulation.
    - **FormulationWeight –** The weight of formulation in each production.
    - **Other** – the other are water solubility, stability, pka and etc. The detail of each attribute is show on Substance class.

Method description

* + - **Getter and Setter method** – It used when the system set value and get value.

###### InferenceEngineDirector



**Figure 43: Inference Engine Director (Server Side)**

The InferenceEnigneDirector is director for managing the inference engine such as Rule base, Case base reasoning and hybrid reasoning. This class will help the inference engine working correctly.

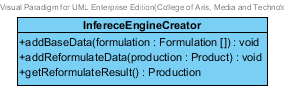
Attribute description

N/A

Method description

* + - **getReformulateResult(production: Production)**: ReformulatedProduct[] – This method will receive the production object from client side. After that, it will use the inference engine for reformulating the production. Finally, this method will send the reformulation result that call reformulatedProduct to the user.

###### InferenceEngineCreator



**Figure 44: Inference Engine Creator (Server Side)**

The inference engine creator will create the inference engine follow the InferenceEngineDirector command. This interface consists of 3 method that will be show on below this paragraph.

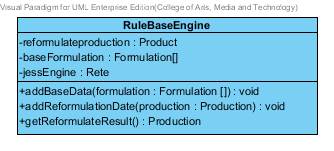
3.2.1.2.3.1: Attribute description

N/A

3.2.1.2.3.2: Method description

* + - **addBaseData(formulation : Formulation[] ) : void** – This method is used for adding the base data from the database. The base data is used by the inference engine for calculating reformulation.
    - **addReformulateData(production : Production) : void –** This method is used for adding the production data that user want to reformulation.The production will calculate with the base data.
    - **getReformulateResult() :** Production – After reformulation, this method will return the reformulation result to the user.

###### RuleBaseEngine



**Figure 45: Rule Base Engine (Server Side)**

Rule Base Engine, the rule base engine is the one of inference engine that use for reformulating the production. The Rule Base engine implement the method from inference engine creator.

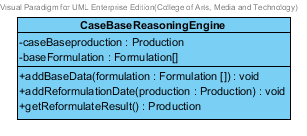
Attribute description

* + - **Reformulateproduction** – the production that use for reformulation.
    - **BaseFormulation –**the base formulation is get from the database.
    - **jessEngine –** the rule base engine is used for making reformulation.

Method description

Same as InferenceEnigneCreator

###### CaseBaseReasoningEngine



**Figure 46: Rule Base Engine (Server Side)**

Case Base Reasoning Engine, the Case Base Reasoning engine is the one of inference engine that use for reformulating the production. The Case Base Reasoning engine implement the method from inference engine creator.

Attribute description

* + - **Reformulateproduction** – the production that use for reformulation.
    - **BaseFormulation –**the base formulation is get from the database.

Method description

Same as InferenceEnigneCreator

###### HybridReasoningEnigne



**Figure 47: Rule Base Engine (Server Side)**

Hybrid Engine, the hybrid engine is the one of inference engine that use for reformulating the production. The Hybrid engine implement the method from inference engine creator.

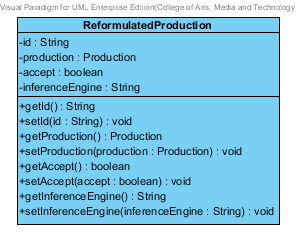
Attribute description

* + - **N/A**

Method description

Same as InferenceEnigneCreator

###### ReformulatedProduction



**Figure 48: Reformulated Production (Server Side)**

The Reformulated production is create after reformulation. The attribute of this class will show below this paragraph.

Attribute description

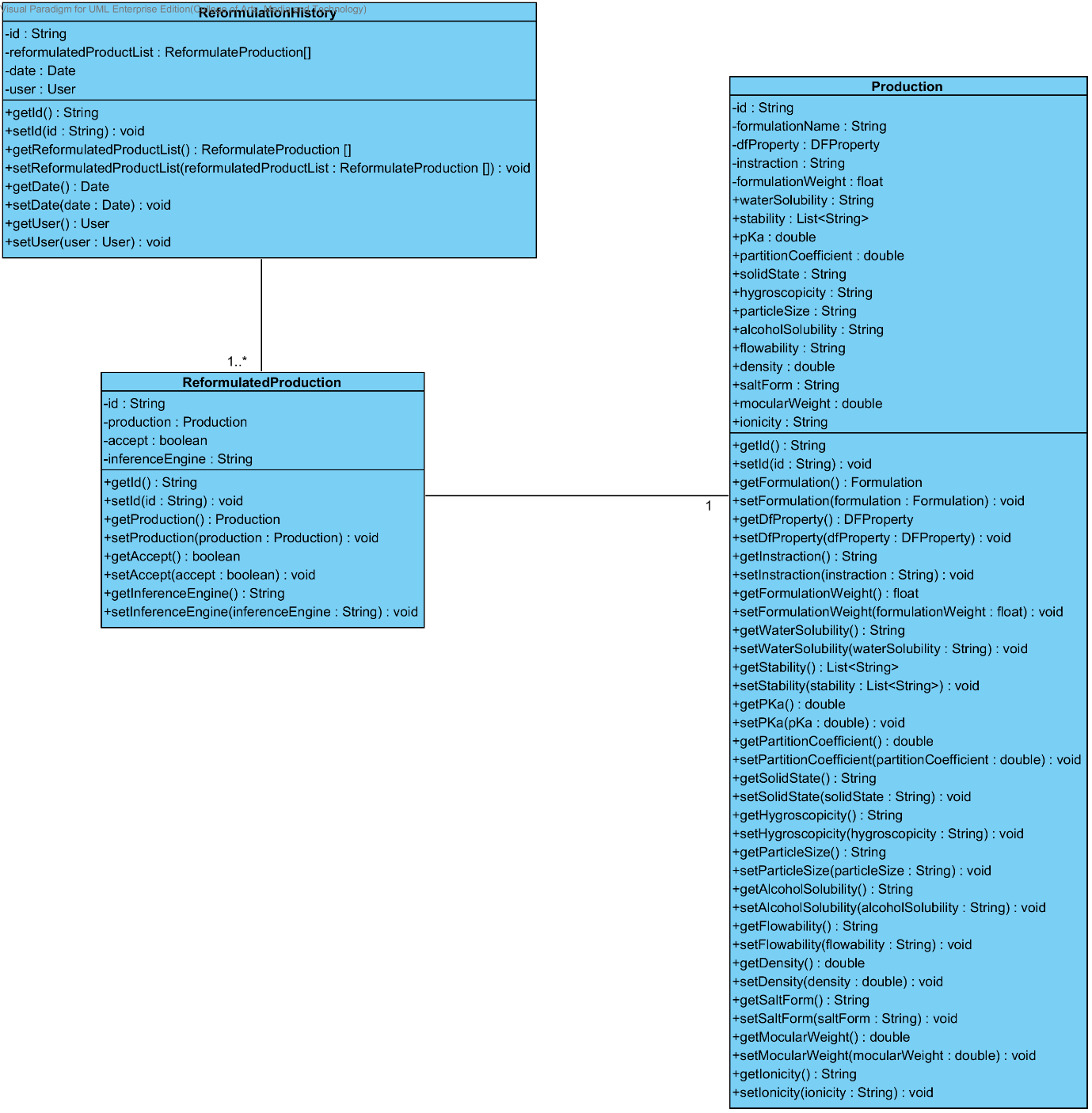
* + - **Id –** The identification of reformulated production
    - **Production –** the production has been reformulated by inference engine.
    - **Accept –** the default is “false”, if the accept equal “true”. Which mean this product is good for base data.
    - **Inference Engine –** the inference engine of this reformulated Production.

Method description

Getter and Setter Method.

## Sub-Feature 3: View the drug reformulation history.

The Sub-Feature 3 is view the drug reformulation history. The user can view the reformulation history that user save. The relationship between reformulation history and reformulated production is illustrated in the Figure 24 below on this passage.



**Figure 49: The Relationship between Reformulation History and Reformulated Production**

### CD-CI- 05: Reformulation History Class Diagram (Client Side)

##### D:\All for desktop\Senior project\Software design\DrugExpertClassDiagram(jpg file)\Class Diagram\CD-CI-05 _ Reformulation History Class Diagram (Client Side).pngClass diagram

**Figure 50: CD-CI-05 – Reformulation History Class Diagram (Client Side)**

##### Class description

From the Figure number 51 .It can divide into 3 Classes .The detail of each class is described on the next paragraph

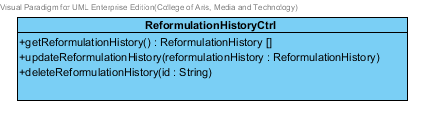
###### ReformulationHistory Class



**Figure 51: Reformulation History Class (Client Side)**

The Reformulation history Class in Client Side is the model class that used for receiving excipient data from the user and showing an excipient data to the user. This model is controlled by ReformulationHistoryCtrl.

###### ReformulationHistoryCtrl



**Figure 52 : Reformulation History Ctrl (Client Side)**

The ReformulationHistoryCtrl is the reformulation history controller that used for managing the excipient data that receive from the user or server. The ReformulationHistoryCtrl consists of 3 method. The detail of each method are shown on next paragraph.

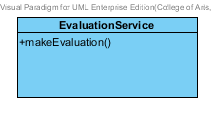
Attribute Description

N/A

Method Description

* + - **getReformulationHistory() : ReformulationHistory[]** - This method is used for getting the list of reformulation history from the server. The method will return the list of reformulation history object. On the other hand, if there are not any reformulation history object in the database. This method will return null.
    - **UpdateReformulationHistory( reformulationHistory : ReformulationHsitory) –** This method is used for updating the existing reformulation history that user selected. This method will use the service for saving an edited reformulation history.
    - **DeleteReformulationHistory(id : String) –** This method is used for deleting the existing excipient from the database.

###### EvaluationService



**Figure 53 : Substance Service (Client Side)**

The Evaluation Service is used for evaluate the reformulation that user making.

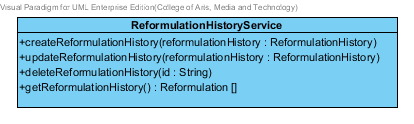
Attribute Description

* + - N/A

Method Description

* **makeEvaluation ()** – This method is used for evaluating the reformulation that user making.

###### ReformulationHistoryService



**Figure 54 : Reformulation History Service (Client Side)**

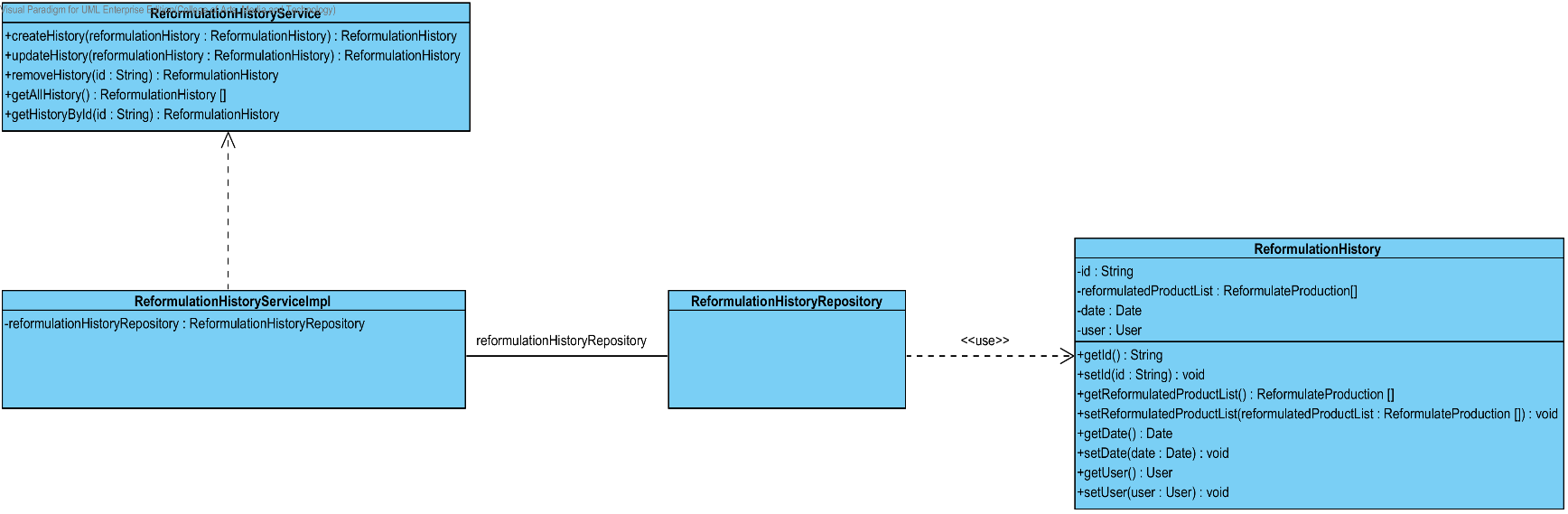
The Reformulation History Service is the service that use for adding, updating, deleting and getting data from the server. The Reformulation history Service can do its business by using http request such as POST, PUT, GET and DELELTE.

Method Description

* **createReformulationHistory (reformulationHistory: ReformulationHistory)** – This method is used for adding a new reformulation history to database. The createReformulationHistory method send the data that user input to the server by POST request.
* **updateReformulationHistory (reformulationHistory : ReformulationHistory ) –** This method is used for updating the existing reformulation history object. The updateReformulationHistory method send the edited data that user input to the server by PUT request.
* **deleteReformulationHistory (id: String) –** This method is used for deleting the existing reformulation history object. The deleteReformulationHistory method send the id of the reformulation history object to the server by DELETE request.
* **getReformulationHistory() : ReformulationHistory [] –** This method is used for getting the reformulation history object from the database. This method send the GET request to server for getting the data.

### CD-SV-05: Reformulation History Class Diagram (Server Side)

##### Class diagram

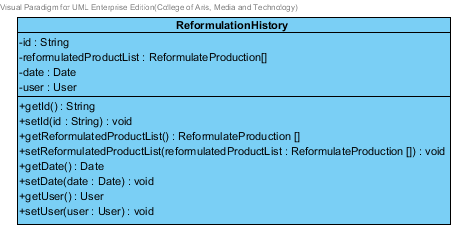


**Figure 55 : CD-SV-05: Reformulation History Class Diagram (Server Side)**

##### Class description

From the figure 57, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

###### ReformulationHistory class



**Figure 56 : Excipient Class (Server Side)**

Reformulation History is the history of the reformulated production. Reformulation History class is an entity class that will be saved to the system. The excipient class consists of 4 attributes follow the list below this passage.

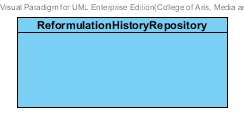
Attribute description

* + - **Id** – the identity of the reformulation history class. Id attribute is a String.
    - **ReformulationProductionlist** – the list of reformulated production.
    - **Date –**the created date.
    - **User** – the user object who create the reformulation history.

Method description

* + - **Getter and Setter method** – It used when the system set value and get value.

###### ReformulationRepository Interface



**Figure 57: Reformulation History Repository (Server Side)**

ReformulationHistoryRepository Interface is an interface that use for CRUD with entity classes in the system. All of ReformulationHistoryRepository interface’s method is generated from Spring data MongoDB framework. ReformulationHsitoryRepository interface consists of 4 methods is shown below this passage.

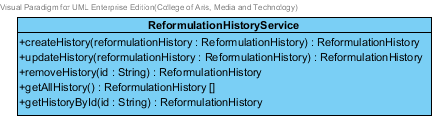
Attribute description

N/A

Method description

* + - **Save (reformulationHistory: ReformulationHistory)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new reformulation history or update existing reformulation history to the system. The input variable is the reformulation history object.
    - **Delete (reformulationHistory: ReformulationHistory)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the reformulation history from the system. The input variable is reformulation history object.
    - **findAll ():ReformulationHistory []** – The findAll method is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of reformulation history data from the system. The result of this method is a list of reformulation history object.
    - **findOne (id: String): ReformulationHistory** – The findOne method is used when the user wants to retrieve the reformulation history data from the system. The system gets the reformulation history object by the id of reformulation history.

###### ReformulationHistoryService



**Figure 58: Reformulation History Service (Server Side)**

ReformulationHistoryService is business processing logic for reformulation history entity. ReformulationHistoryService manages the reformulation history data through the ReformulationHistoryRepository interface. ReformulationHistoryService consists of 5 methods follow the list below this passage.

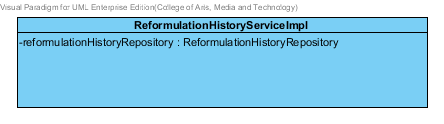
3.2.1.2.3.1: Attribute description

N/A

3.2.1.2.3.2: Method description

* + - **createHistory (reformulationHistory: ReformulationHistory)** – The create history method is used, when the user wants to add a new history to the database. This method adds a new history by input variable of history object. If the history object that input by the user is not contained in the database, this method will add a new reformulation history to the database and return the reformulation history object from the database to the user after the adding reformulation history is successful. On the other hand, when the reformulation history object that input by the user is contained in the database. This method will return a null value to the user.
    - **updateHistory (reformulationHistory: ReformulationHistory)** - The updating reformation history method is used, when the user wants to update an existing reformulation history on in the database. This method update the existing reformulation history by input variable of reformulation history object. If the reformulation history object that input by the user is contained in the database, this method will update an existing reformulation history in the database and return the reformulation history object from the database to the user after the updating reformulation history is successful. On the other hand, when the reformulation history object that input by the user is not contained in the database. This method will return a null value to the user.
    - **Removehistory (id : String)** – The deleting reformulation history method is used when the user wants to deletes the existing reformulation history from the database. This method delete the reformulation history by input variable of reformulation history object. If the reformulation history object that input by the user is contained in the database, this method will delete an existing reformulation history from the database and return the reformulation history object to the user after the deleting reformulation history is successful. On the other hand, when the reformulation history object that input by the user is not contained in the database. This method will return a null value to the user.
    - **getAllHistory() : ReformulationHistory []** – The getAllHistory method is used, when the user wants to get all reformulation history data in the database. This method is return as a list of reformulation history object from the database.
    - **getHistoryById(id : String) :** Excipient – The getHistoryById method is used, when the user wants to get the history data in the system. This method gets reformulation history object from the database by id that input by the user. On the other hand, if the id that input by user is not contained in the database. This method will return null value to the user.

###### ReformulationHistoryServiceImpl



**Figure 59: Reformulation History Service Implementation (Server Side)**

ReformulationHistoryServiceImpl is the reformulation history service class that implements the method from ReformulationHistoryService. So, the method of ReformulationHistoryServiceImpl is same as ReformulationHistoryService.

Attribute description

* + - **ReformulationHistoryRepository** – the repository of reformulation history. This attribute is used for reformulation history data management.

Method description

Same as ReformulationHistoryService

# Sequence Diagram

In the 2nd progress the URS is related with the list of sequence diagram that shown below this passage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Sub-Feature Name** | **URS No.** | **URS Name** | **Sequence Diagram** |
| 6 | Manage the drug substance | URS-09 | The user adds a new substance into the system. | SQD-CI-01, SQD-SV-01 |
| URS-10 | The user updates an existing substance into the system. | SQD-CI-02, SQD-SV-02 |
| URS-11 | The user deletes an existing substance from the system. | SQD-CI-03, SQD-SV-03 |
| URS-12 | The user views the substance in the system. | SQD-CI-04, SQD-SV-04 |
| 7 | Manage the drug excipient | URS-13 | The user adds a new excipient to the system. | SQD-CI-05, SQD-SV-05 |
| URS-14 | The user updates an existing drug excipient in the system. | SQD-CI-06, SQD-SV-06 |
| URS-15 | The user delete an existing drug excipient in the system. | SQD-CI-07, SQD-SV-07 |
| URS-16 | The user views all the drug excipient in the system. | SQD-CI-08, SQD-SV-08 |
| 8 | Manage the drug formulation | URS-17 | The user adds a new drug formulation case into the system. | SQD-CI-09, SQD-SV-09 |
| URS-18 | The user updates an existing drug formulation case in the system. | SQD-CI-10, SQD-SV-10 |
| URS-19 | The user deletes an existing drug formulation case in the system. | SQD-CI-11, SQD-SV-11 |
| URS-20 | The user views all of the formulation in the system. | SQD-CI-12, SQD-SV-12 |
| 2 | Calculate the drug reformulation by using the inference engine. | URS-06 | The user calculates a drug reformulation by using an inference engine. | SQD-CI-13 , SQD-CI-14, SQD-SV-13, SQD-SV-14 |
| 3 | View the drug reformulation history | URS-07 | The user views their drug reformulation history. | SQD-CI-15, SQD-SV-15 |

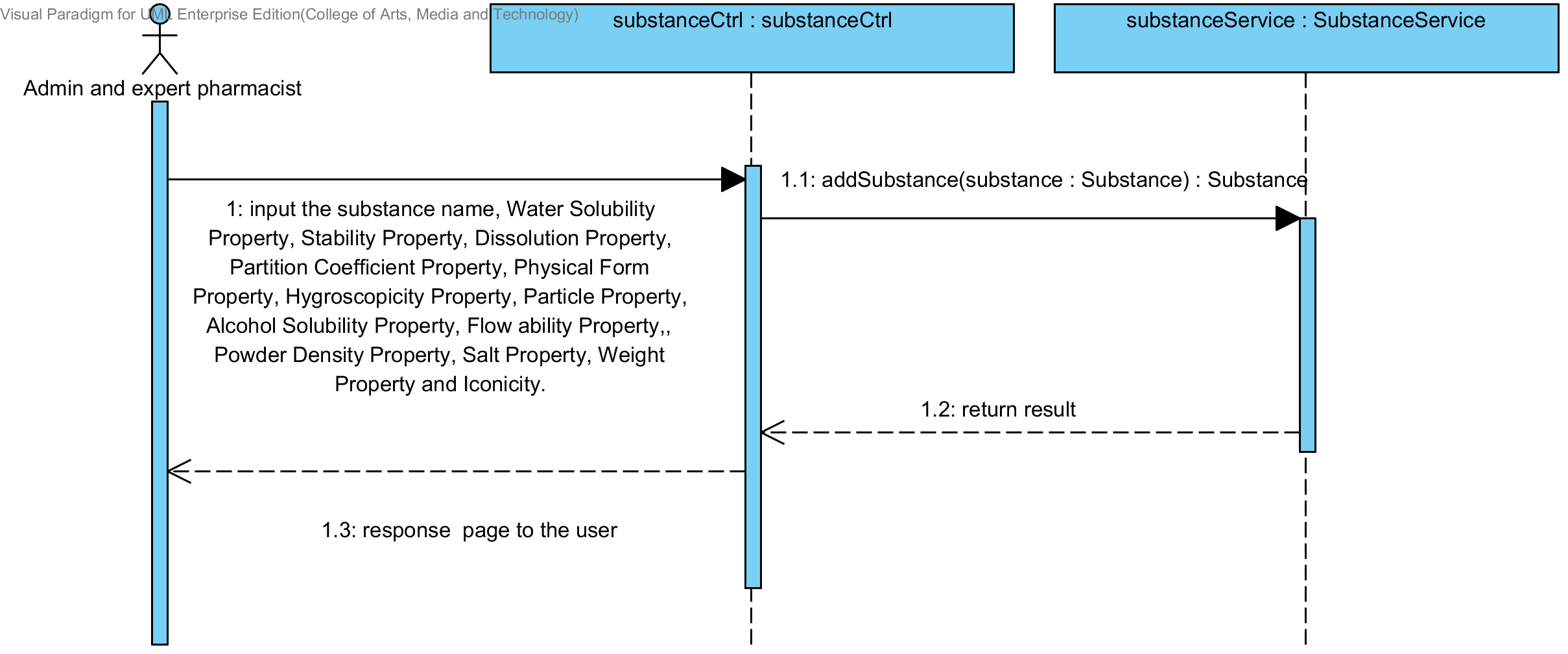
**Table 2: The Relationship between the Use Case and Sequence Diagram**

## Sub-Feature 6: Manage the drug substance

### URS-09: The user adds a new substance to the system.

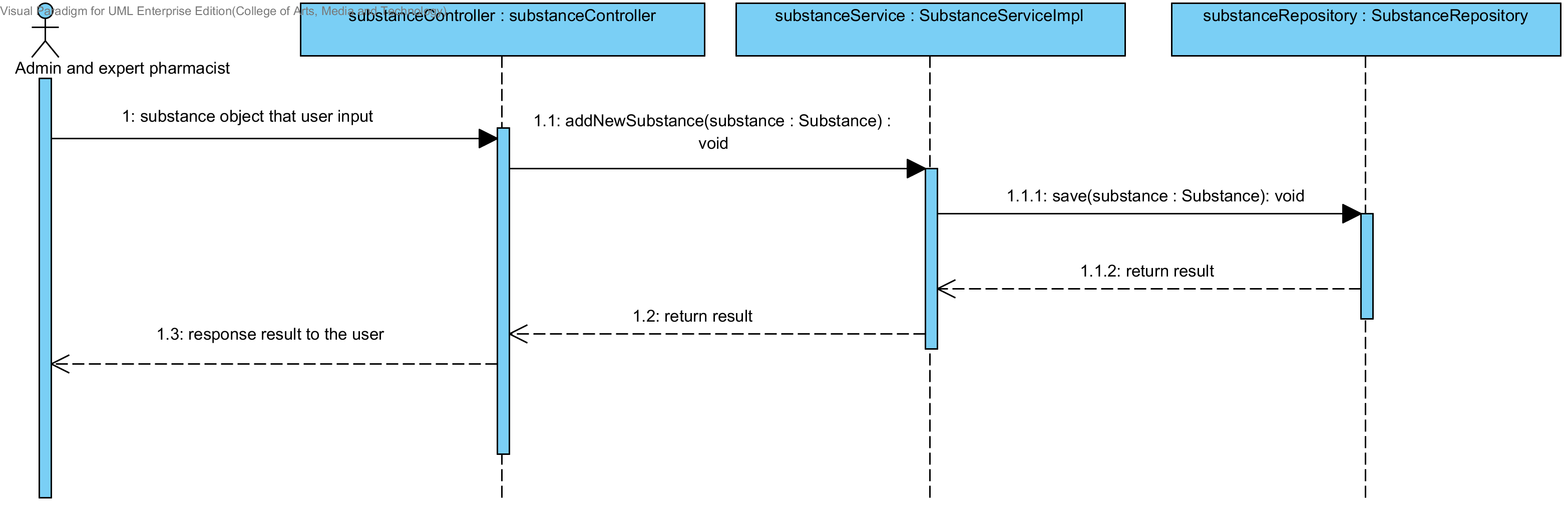
In a sequence diagram, the user can add a new substance to the system. Firstly, the user opens the substance adding page, then the user input substance data such as name, water solubility, stability and etc. The substance controller gets an input data from the user. After that, the substance controller send a new substance data to appropriate service for adding a new substance. Finally, the system show a new substance with the adding substance successful page.

##### SQD-CI-01: The user adds a new substance to the system (Client Side).



**Figure 60 : SQD-CI-01- The user adds a new substance to the system (Client Side).**

##### SQD-SV-01: The user adds a new substance to the system (Server Side)

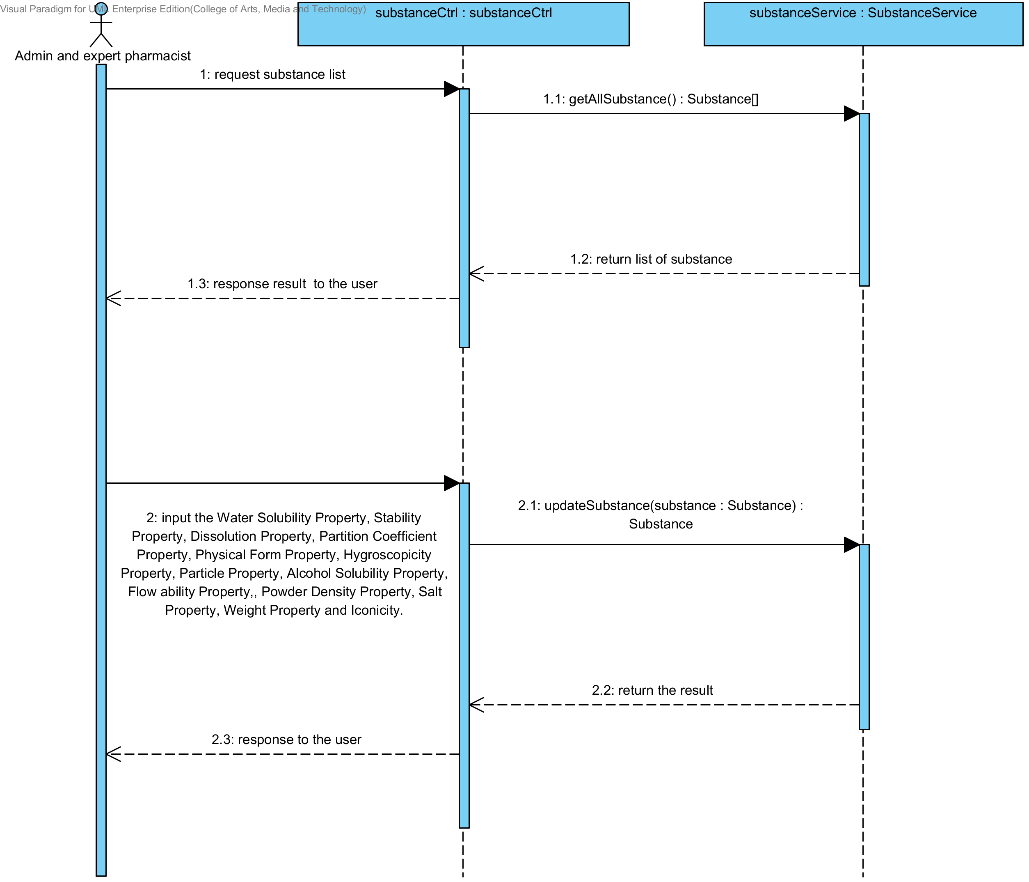


**Figure 61: SQD-SV-01: The user adds a new substance to the system (Server Side).**

### URS-10: The user updates an existing substance in the system.

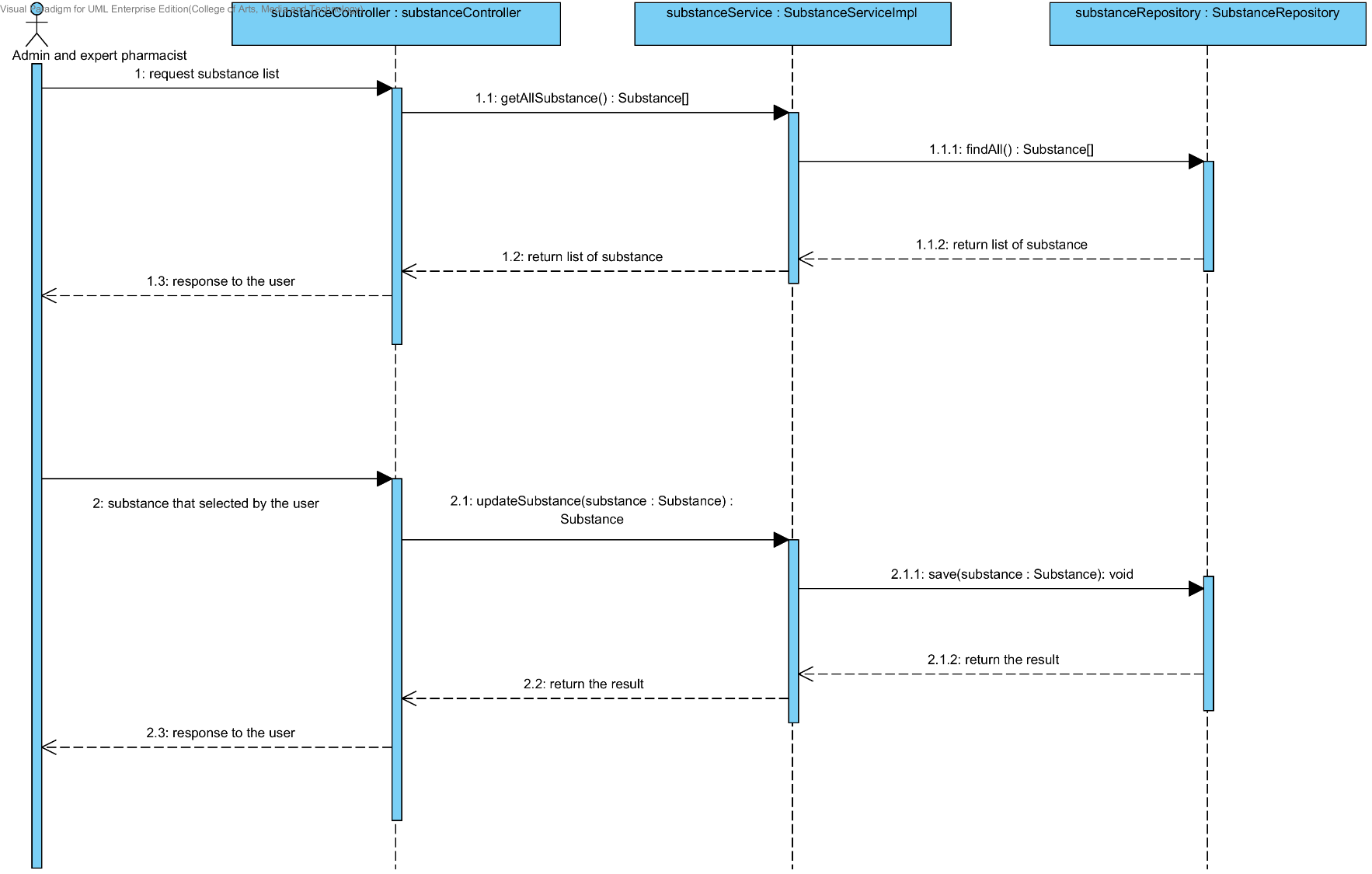
In a sequence diagram, the user can update an existing substance in the system. Firstly, the user opens the substance updating page, then the user input substance data such as water solubility, stability and etc. The substance controller gets a substance data from the user. After that, the substance controller send a new substance data to appropriate service for updating an existing substance in the system. Finally, the system show substance that already update with the substance updating successful page.

##### SQD-CI-02: The user updates an existing substance in the system (Client Side).



**Figure 62: SQD-CI-02 - The user updates an existing substance in the system (Client Side)**

##### SQD-SV-02: The user updates an existing substance in the system (Server Side).

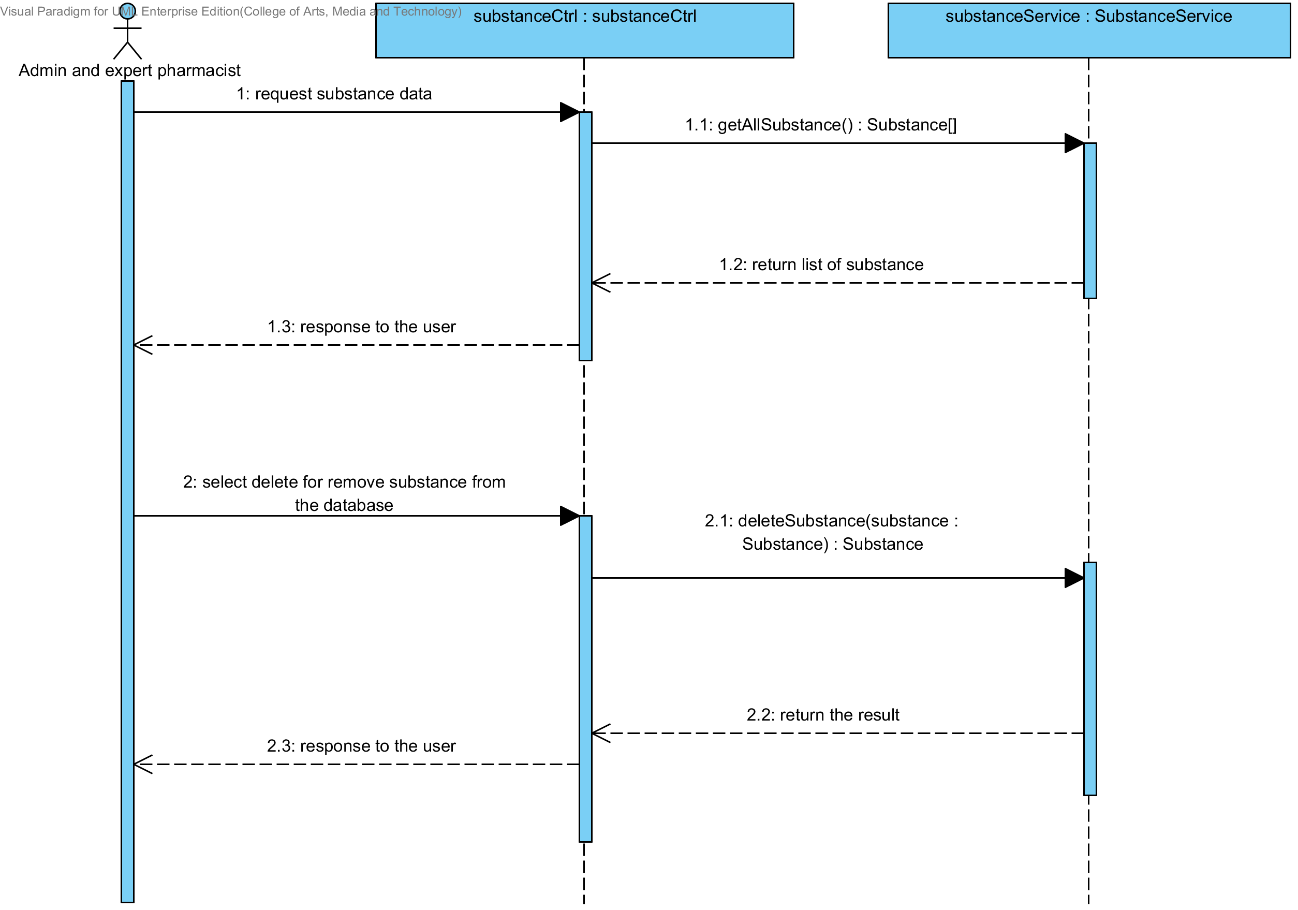


**Figure 63: SQD-SV-02- The user updates an existing substance in the system (Server Side)**

### URS-11: The user deletes an existing substance from the system.

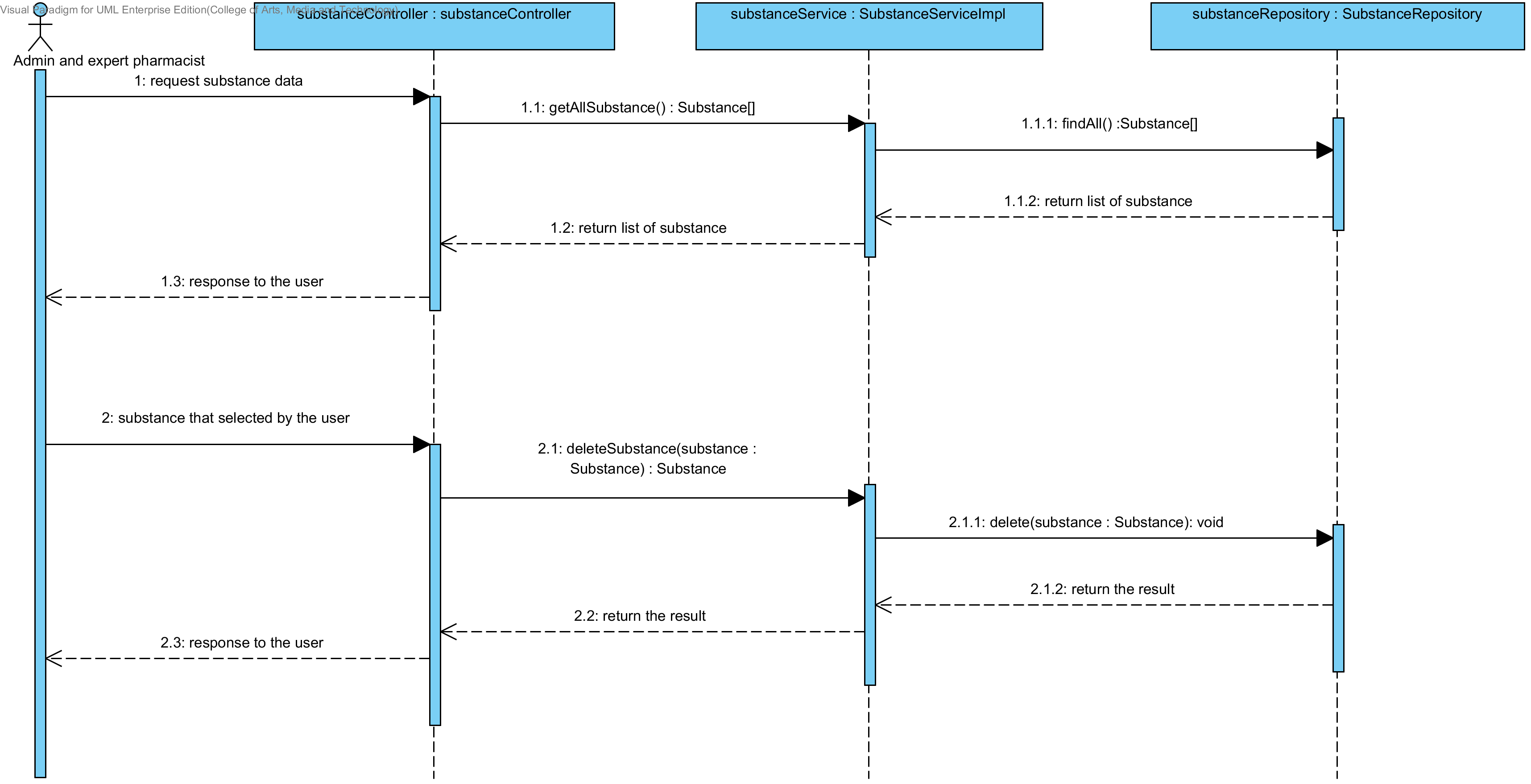
In the sequence diagram, the user can delete an existing substance from the system. Firstly, the user opens the substance deleting page. The system shows all substance data on the screen, then the user selects substance for deleting. After that, the substance controller finds an appropriate service for substance property deleting. Finally, the substance controller shows a substance that already deleted on the deleting substance successful page to the user.

##### SQD-CI-03: The user deletes an existing substance from the system (Client Side).



**Figure 64: SQD-CI-03- The user deletes an existing substance from the system (Client Side)**

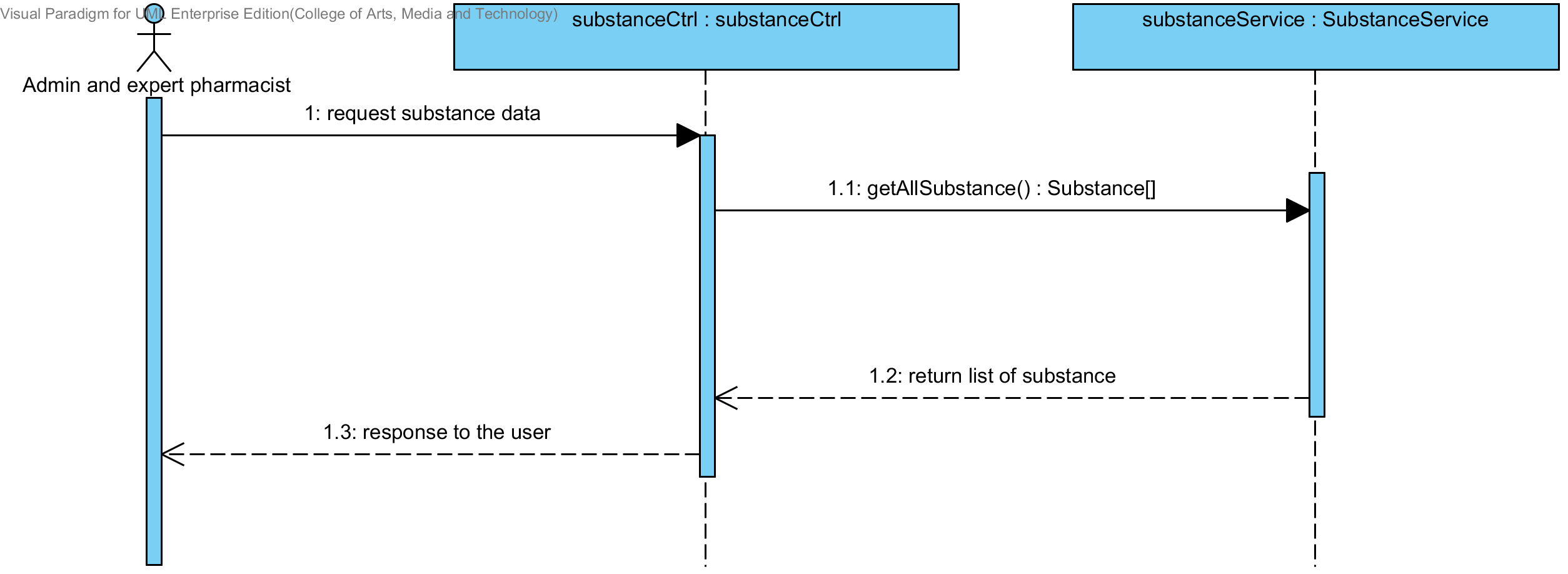
##### SQD-SV-03: The user deletes an existing substance from the system (Server Side).

**Figure 65: SQD-SV-03 – The user deletes an existing substance from the system (Server Side)**

### URS-12: The user views the substance in the system.

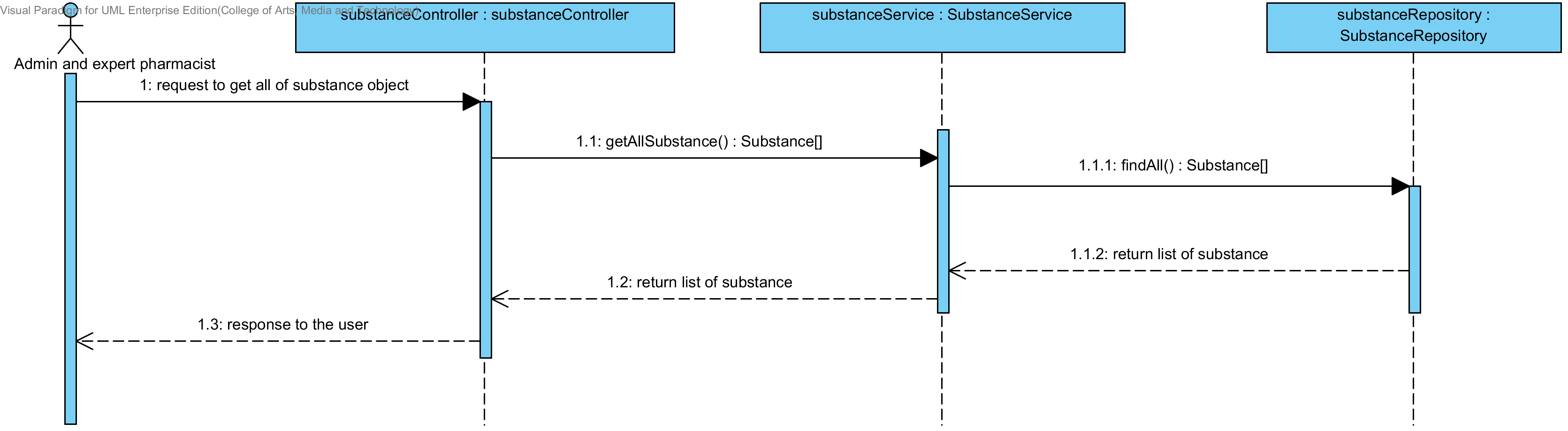
In the sequence diagram, the user can delete an existing substance from the system. Firstly, the user opens the substance deleting page, then the system shows all substance data on the screen.

##### SQD-CI-04: The user views the substance in the system (Client Side).



**Figure 66 : SQD-CI-04 – The user views the substance in the system (Client Side)**

##### SQD-SV-04: The user views the substance in the system (Server Side).



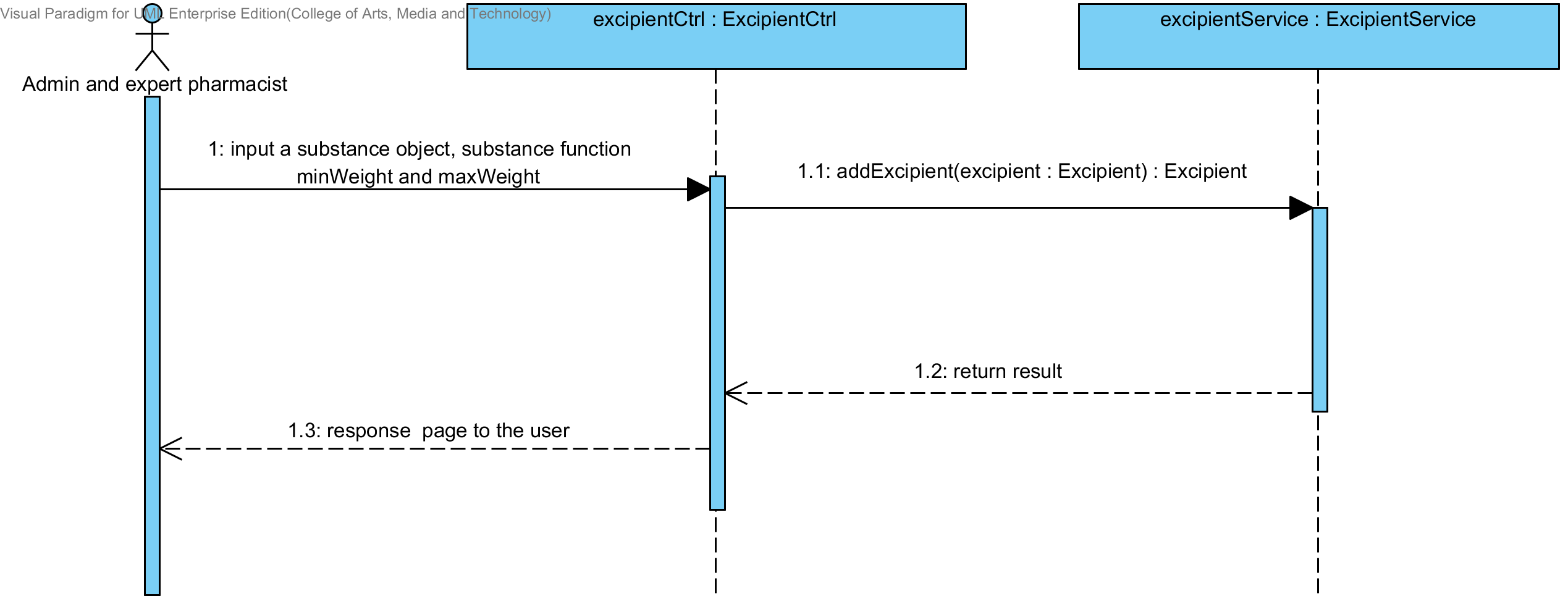
**Figure 67: SQD-SV-04 – The user views the substance in the system (Server Side)**

## Sub-Feature 7: Manage the drug excipient

### URS-13: The user adds a new excipient to the system.

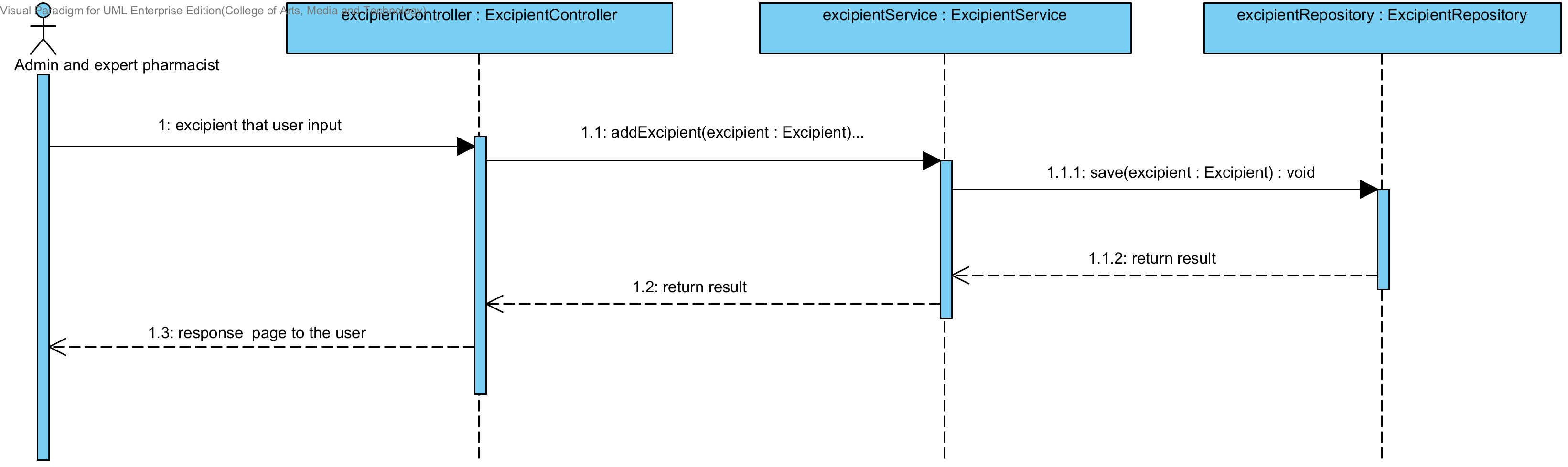
In a sequence diagram, the user can add a new excipient to the system. Firstly, the user opens the excipient adding page, then the user input excipient data such as substance object, substance function, min weight and max weight . The excipient controller gets an input data from the user. After that, the excipient controller send a new excipient data to appropriate service for adding a new excipient. Finally, the system show a new excipient with the adding excipient successful page.

##### SQD-CI-05: The user adds a new excipient to the system (Client Side).



**Figure 68: SQD-CI-05 – The user adds a new excipient to the system (Client Side)**

##### SQD-SV-05: The user adds a new excipient to the system (Server Side).

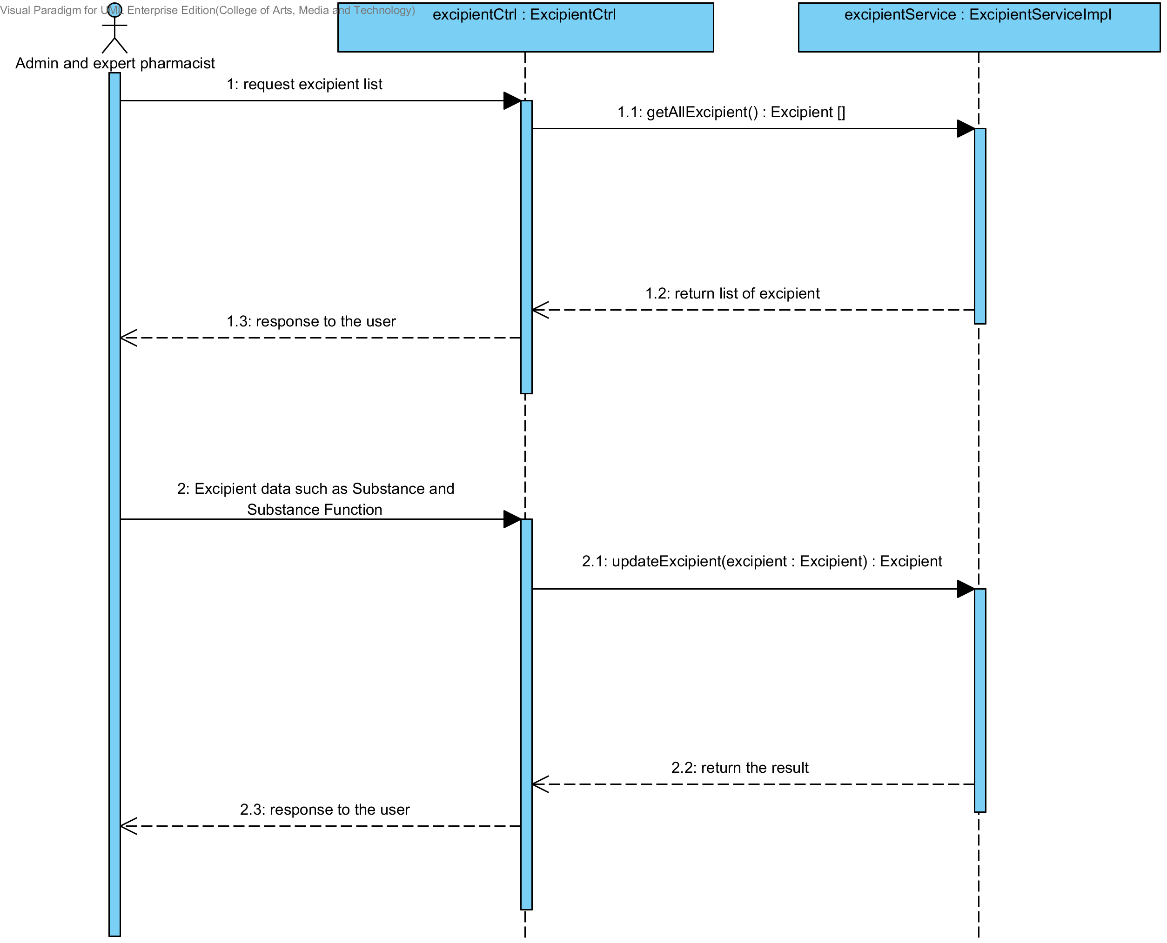


**Figure 69: SQD-SV-05 – The user adds a new excipient to the system (Server Side).**

### URS-14: The user updates an existing excipient in the system.

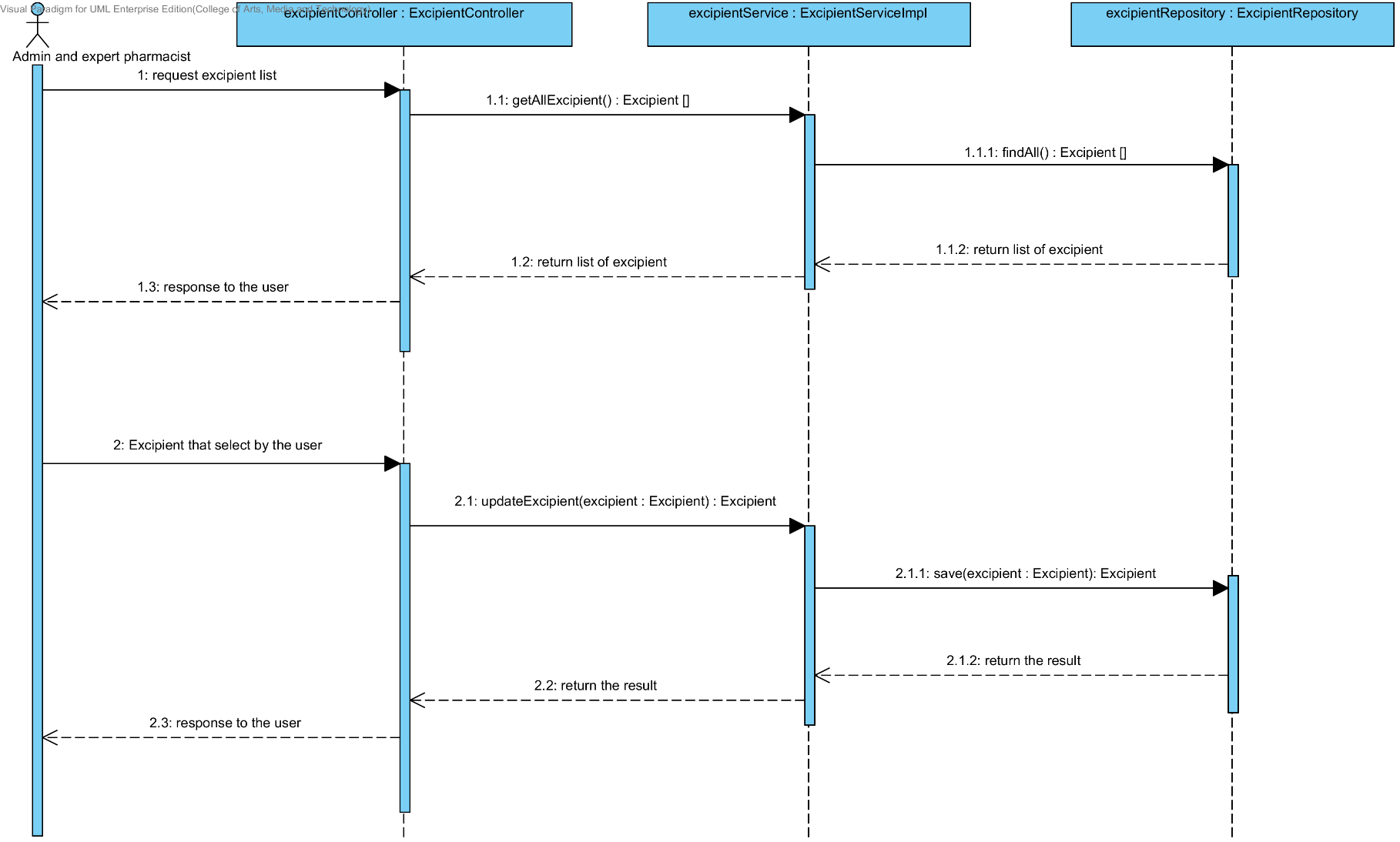
In a sequence diagram, the user can update an existing excipient in the system. Firstly, the user opens the excipient updating page, then the user input excipient data such as substance function. The excipient controller gets an excipient data from the user. After that, the excipient controller send a new excipient data to appropriate service for updating an existing excipient in the system. Finally, the system show excipient that already update with the excipient adding successful page.

##### SQD-CI-06: The user updates an existing excipient in the system (Client Side)

****

**Figure 70 : SQD-CI-06 – The user updates an existing excipient in the system (Client Side).**

##### SQD-SV-06: The user updates an existing excipient in the system (Server Side)

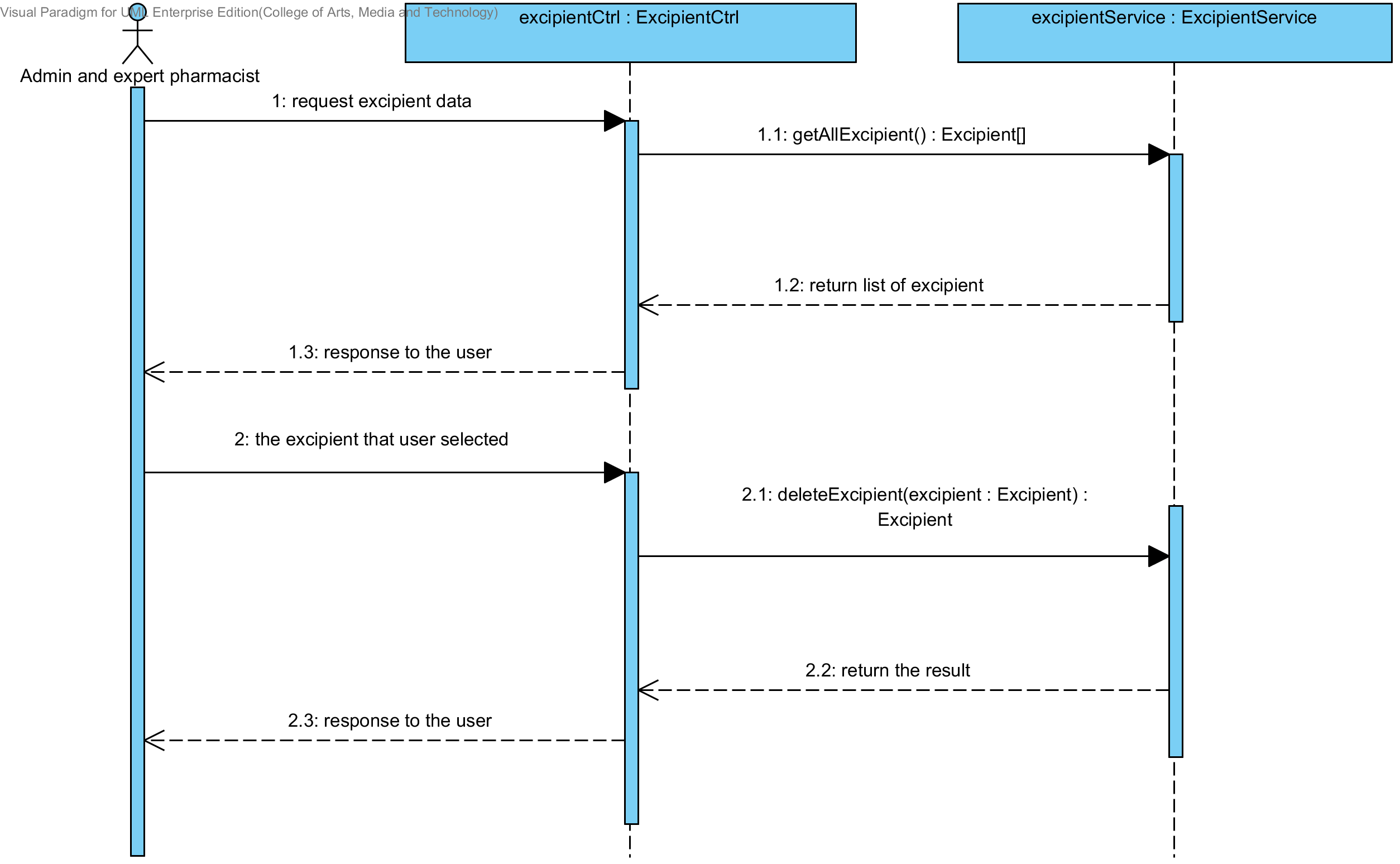


**Figure 71: SQD-SV-06 – The user updates an existing excipient in the system (Server Side)**

### URS-15: The user deletes an existing excipient from the system.

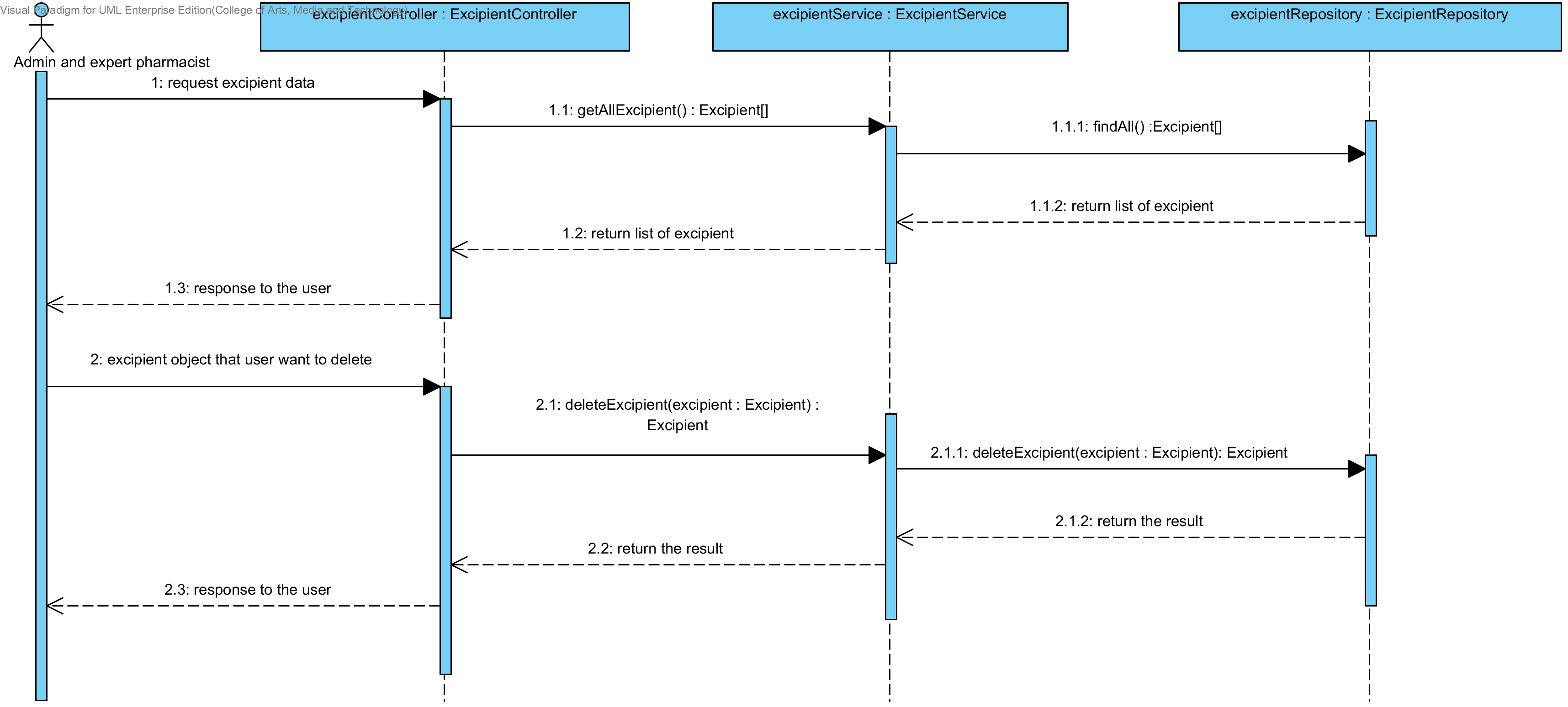
In the sequence diagram, the user can delete an existing excipient from the system. Firstly, the user opens the excipient deleting page. The system shows all excipient data on the screen, then the user selects excipient for deleting. After that, the excipient controller finds an appropriate service for excipient deleting. Finally, the excipient controller shows the excipient that already deleted on the deleting excipient successful page.

##### SQD-SV-07: The user deletes an existing excipient from the system (Client Side).



**Figure 72 : SQD-SV-07 – The user deletes an existing excipient from the system (Client Side)**

##### SQD-SV-07: The user deletes an existing excipient from the system (Server Side).

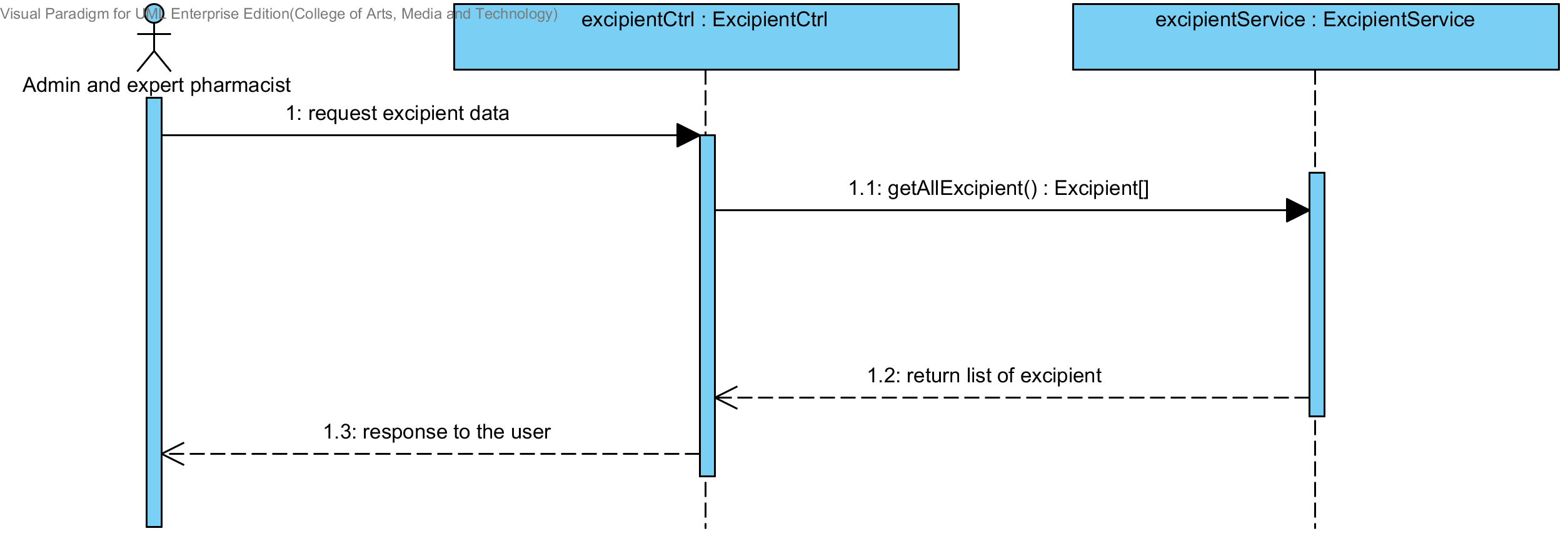


**Figure 73 : SQD-SV-07 – The user deletes an existing excipient from the system (Server Side)**

### URS-16: The user views the excipient in the system.

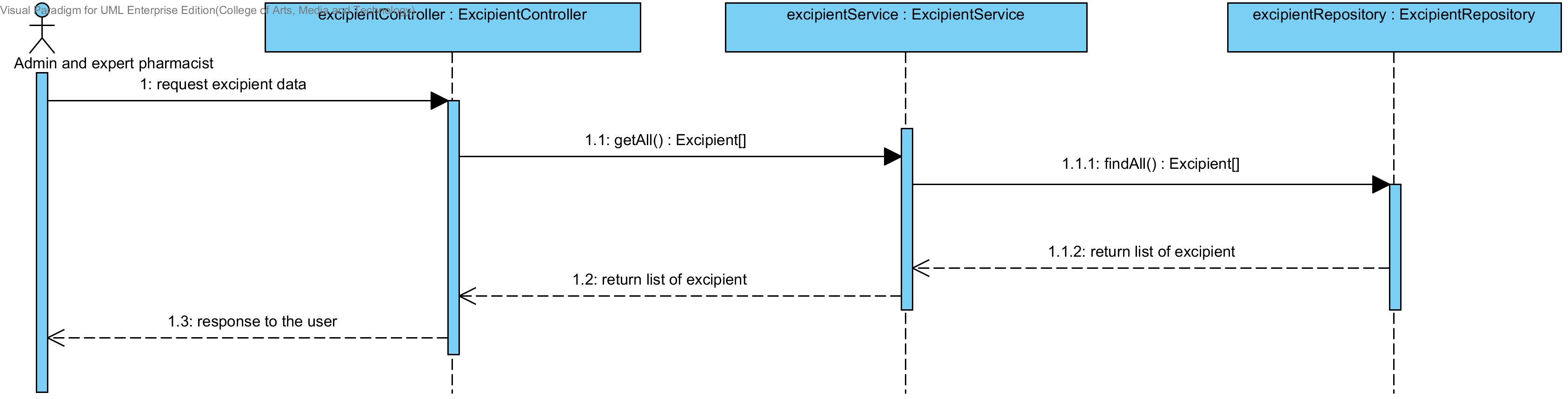
In the sequence diagram, the user can delete an existing excipient from the system. Firstly, the user opens the excipient deleting page, then the system shows all excipient data on the screen.

##### SQD-CI-08: The user views the excipient in the system (Client Side).



**Figure 74 : SQD-CI-08 – The user views the excipient in the system (Client Side)**

##### SQD-SV-08: The user views the excipient in the system (Server Side).



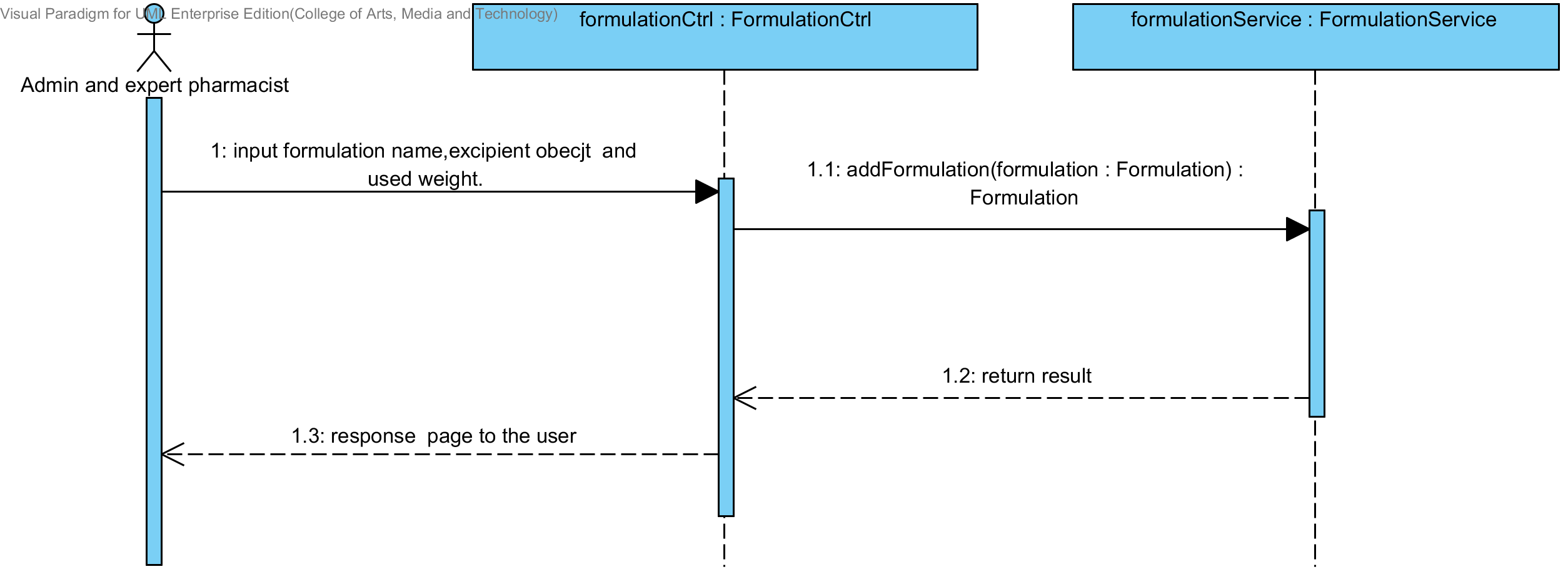
**Figure 75 : SQD-SV-08 – The user views the excipient in the system (Server Side)**

## Sub-Feature 8: Manage the drug formulation

### URS-17: The user adds a new drug’s formulation to the system.

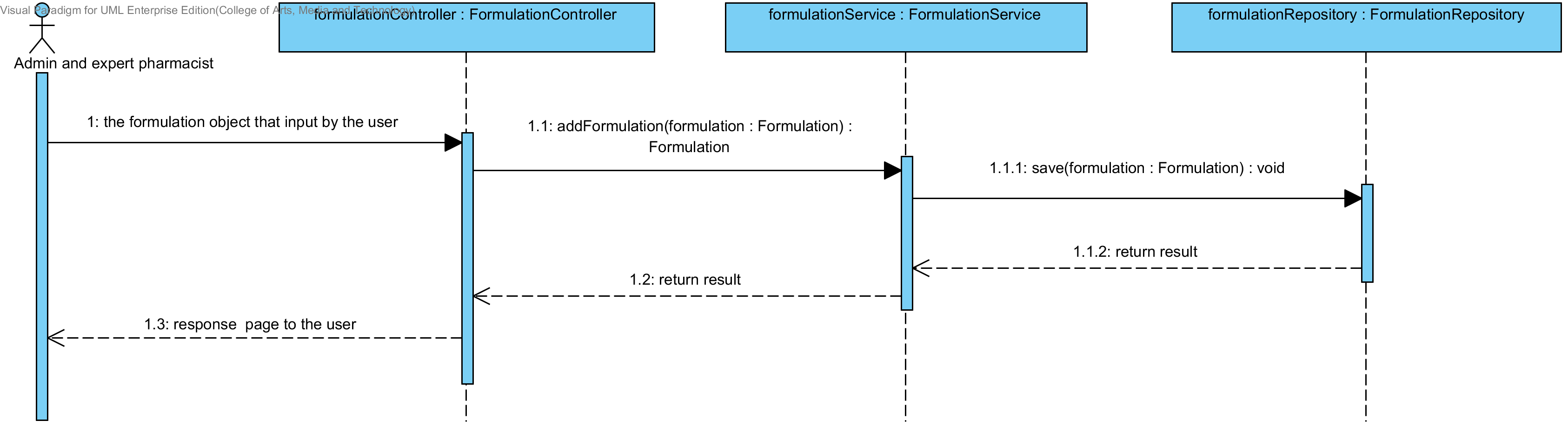
In a sequence diagram, the user can add a new drug’s formulation to the system. Firstly, the user opens the drug’s formulation adding page, then the user input drug’s formulation data such as name, excipient and used weight. The drug’s formulation controller gets an input data from the user. After that, the drug’s formulation controller send a new drug’s formulation data to appropriate service for adding a new drug’s formulation. Finally, the system show a new drug’s formulation with the adding drug’s formulation successful page.

##### SQD-CI-09: The user adds a new drug’s formulation to the system (Client Side).



**Figure 76: SQD-CI-09: The user adds a new drug’s formulation to the system (Client Side).**

##### SQD-SV-09: The user adds a new drug’s formulation to the system (Server Side).

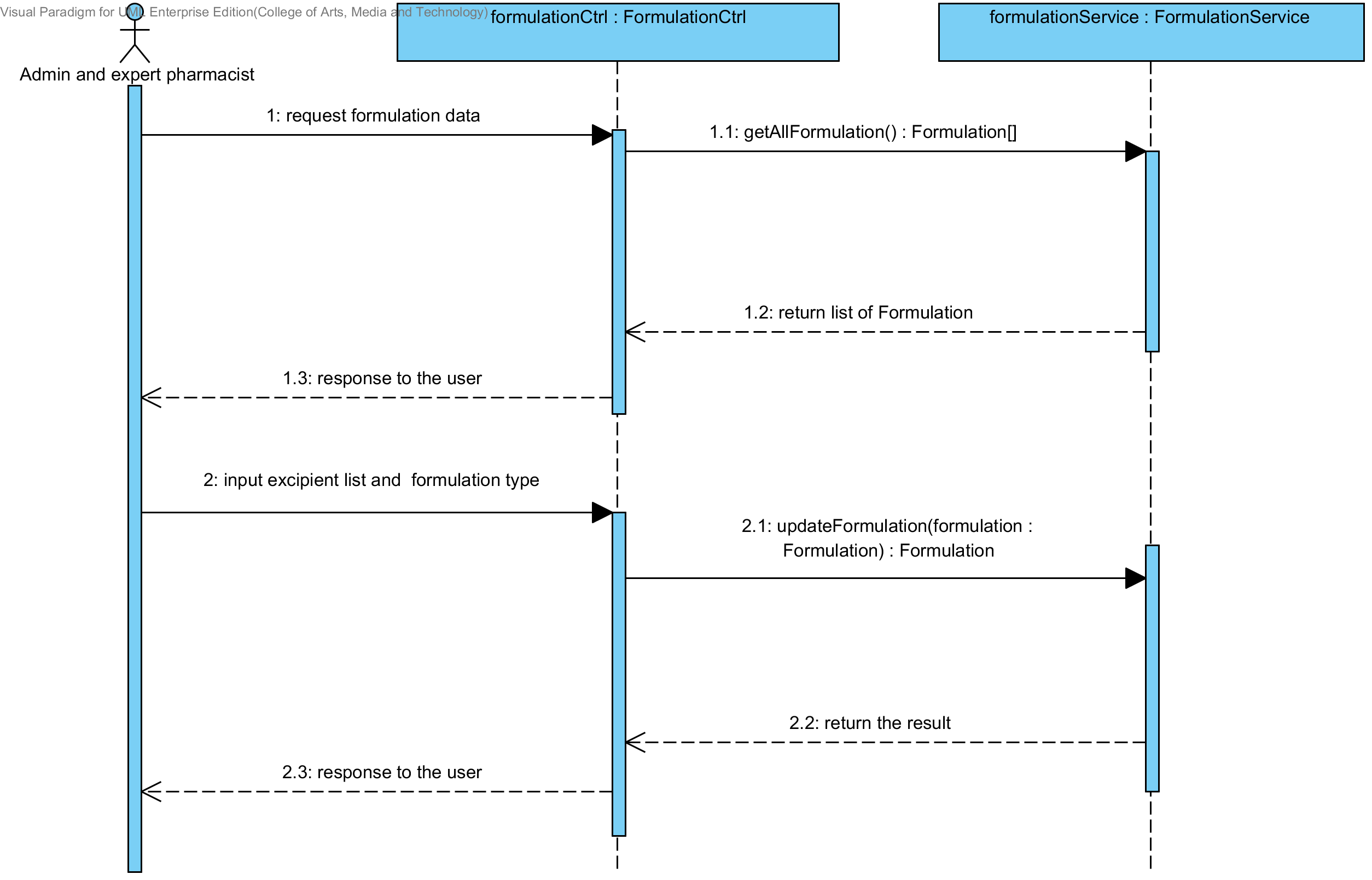


**Figure 77: SQD-SV-09 – The user adds a new drug formulation to the system (Server Side).**

### URS-18: The user updates an existing drug’s formulation in the system.

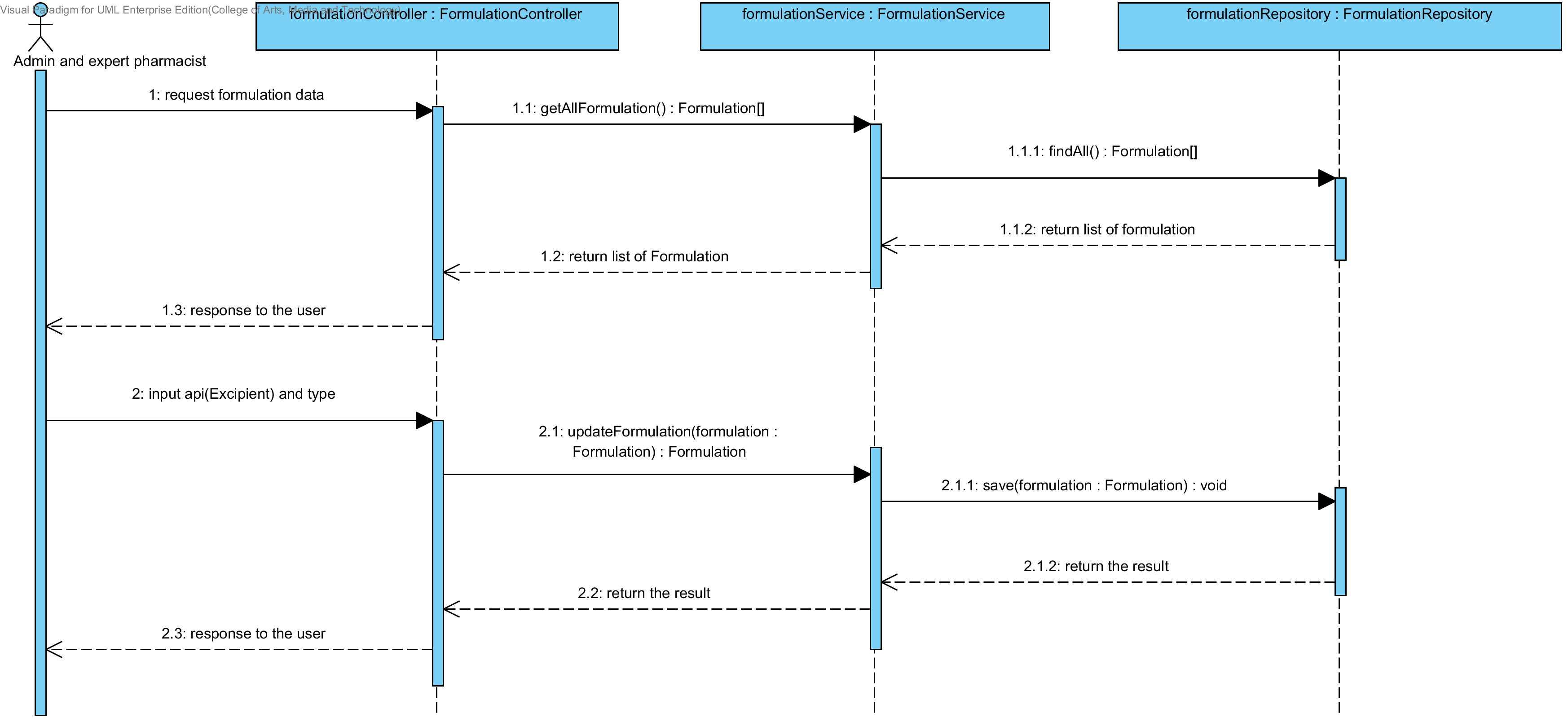
In a sequence diagram, the user can update an existing drug’s formulation in the system. Firstly, the user opens the drug’s formulation updating page, then the user input drug’s formulation data such as excipient and used weight. The drug’s formulation controller gets a drug’s formulation data from the user. After that, the drug’s formulation controller send a new drug’s formulation data to appropriate service for updating an existing substance in the system. Finally, the system show drug’s formulation that already update with the drug’s formulation updating successful page.

##### SQD-CI-10: The user updates an existing drug’s formulation in the system (Client Side).



**Figure 78: SQD-CI-10 – The user updates an existing drug’s formulation in the system (Client Side)**

##### SQD-SV-10: The user updates an existing drug’s formulation in the system (Server Side).

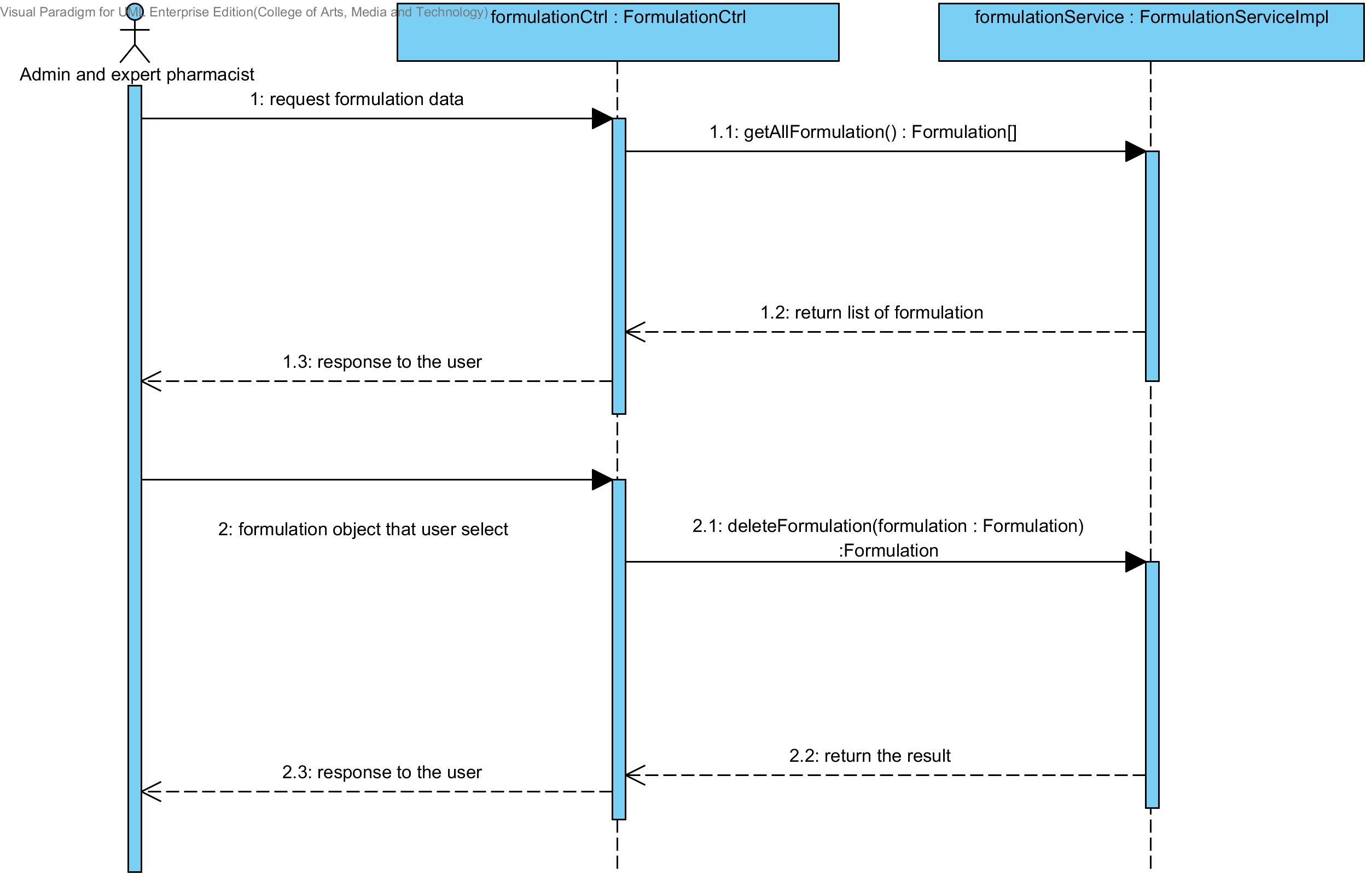


**Figure 79: SQD-SV-10: The user updates an existing drug‘s formulation in the system (Server Side).**

### URS-19: The user deletes an existing drug’s formulation from the system.

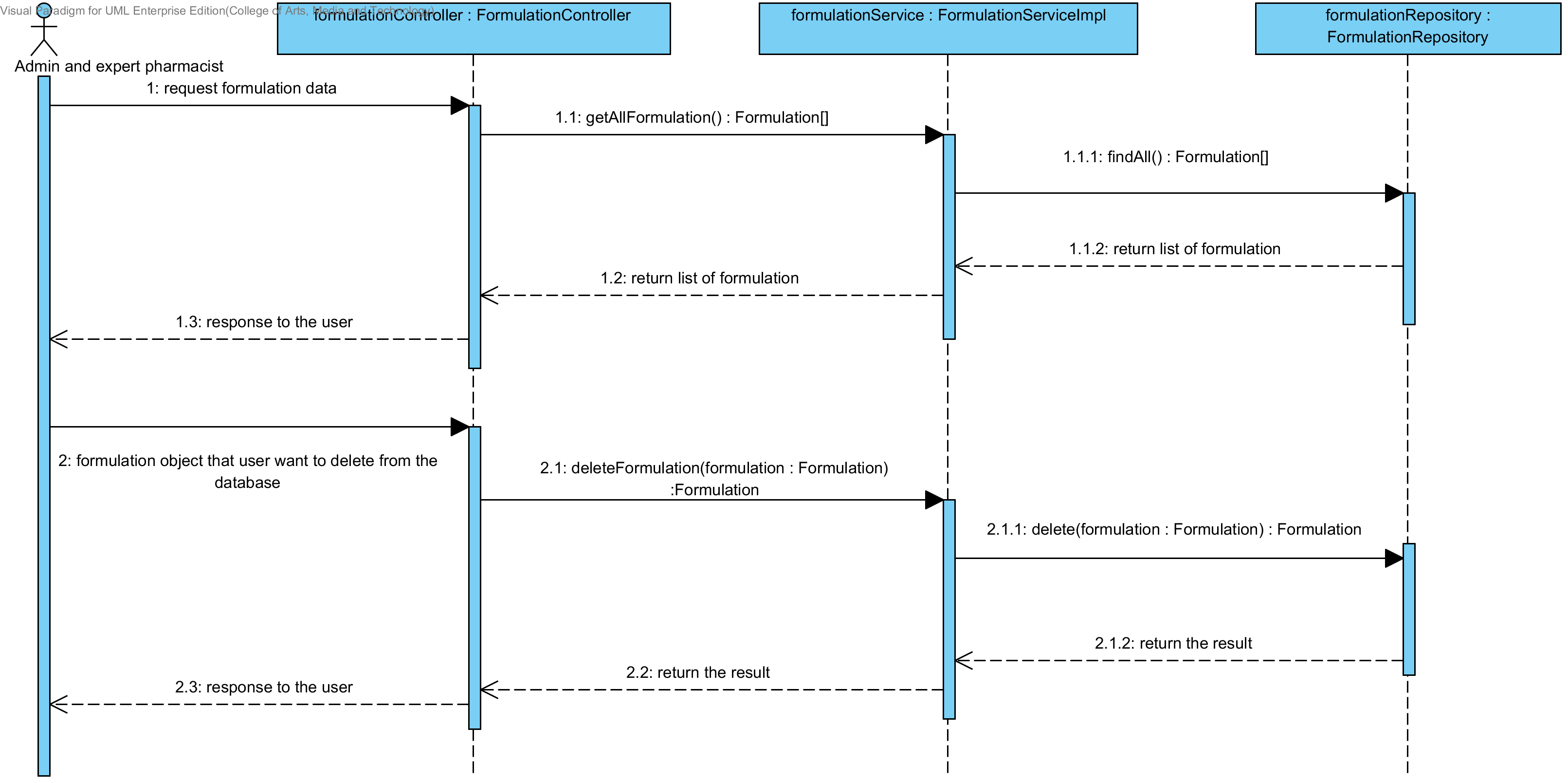
In the sequence diagram, the user can delete an existing drug’s formulation from the system. Firstly, the user opens the drug’s formulation deleting page. The system shows all drug’s formulation data on the screen, then the user selects drug’s formulation for deleting. After that, the drug’s formulation controller finds an appropriate service for drug’s formulation deleting. Finally, the drug formulation controller shows a drug’s formulation that already deleted on the deleting drug’s formulation successful page.

##### SQD-CI-11: The user deletes an existing drug’s formulation from the system (Client Side)

. 

**Figure 80: SQD-CI-11: The user deletes an existing drug formulation from the system (Client Side)**

##### SQD-SV-11: The user deletes an existing drug’s formulation from the system (Server Side).

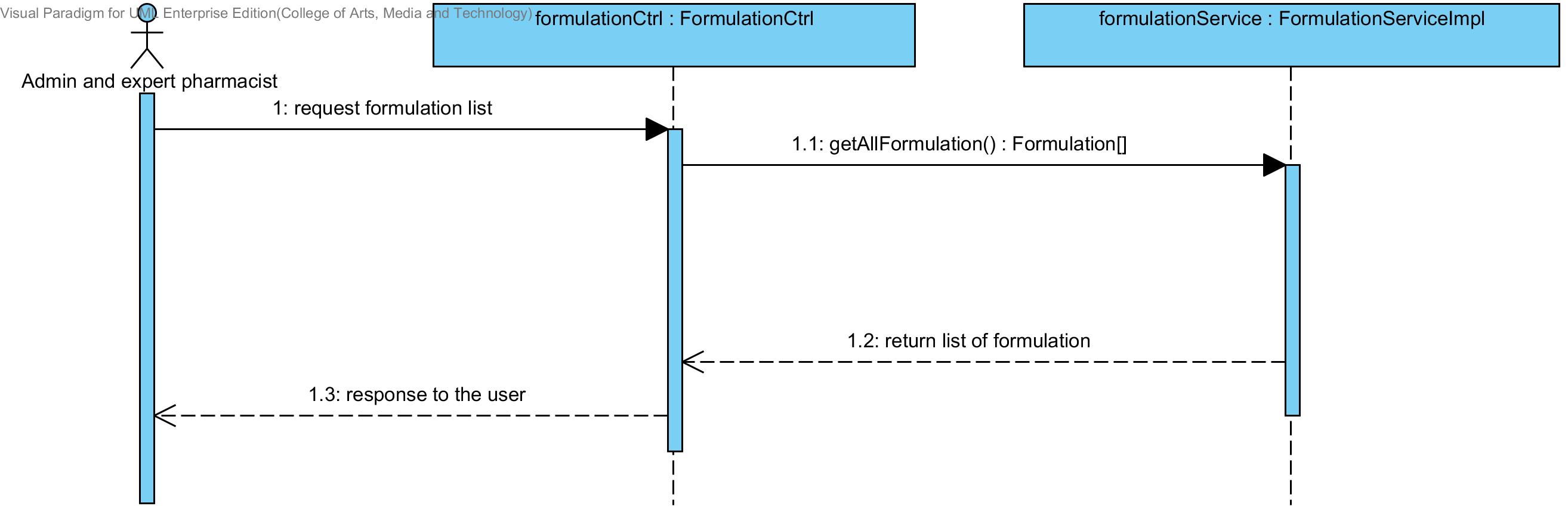


**Figure 81: SQD-SV-11: The user deletes an existing drug formulation from the system (Server Side)**

### URS-20: The user views the drug’s formulation in the system.

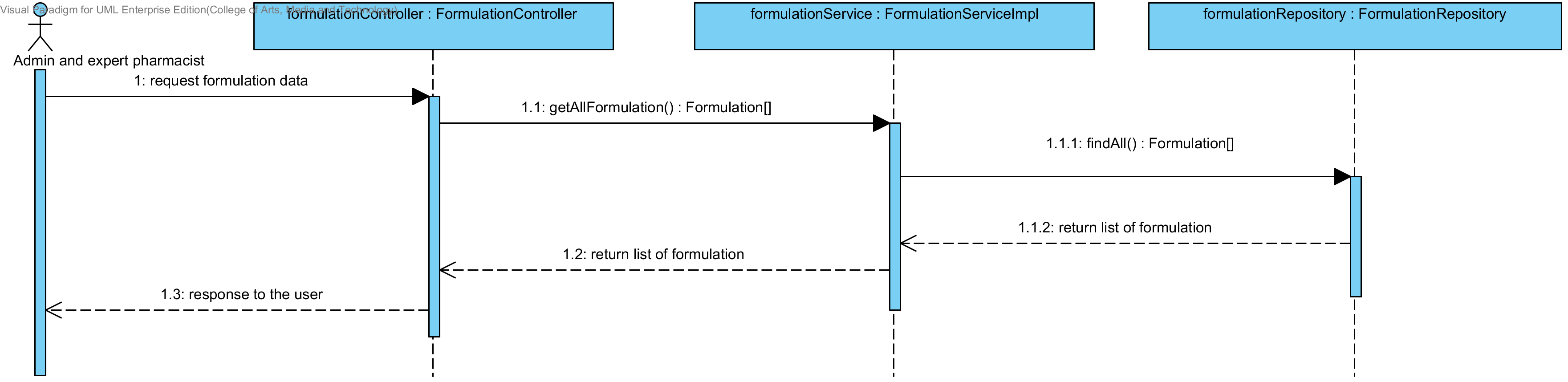
In the sequence diagram, the user can delete an existing drug’s formulation from the system. Firstly, the user opens the drug’s formulation deleting page, then the system shows all drug’s formulation data on the screen.

##### SQD-CI-12: The user views the drug’s formulation in the system (Client Side).



**Figure 82: SQD-CI-12: The user views the drug’s formulation in the system (Client Side).**

##### SQD-SV-12: The user views the drug’s formulation in the system (Server Side).



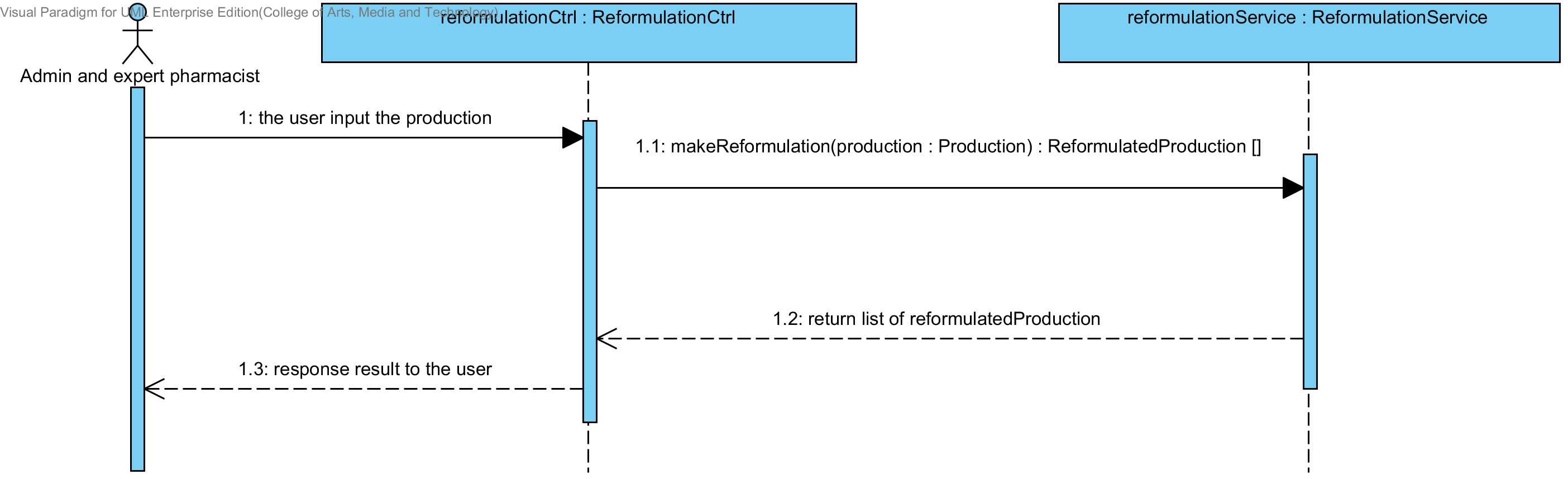
**Figure 83: SQD-SV-12 – The user views the drug’s formulation in the system (Server Side).**

## Sub-Feature 2: Calculate the drug reformulation by using the inference engine.

### URS-06: The user calculates a drug reformulation by using an inference engine.

##### SQD-CI-13: The user calculates a drug reformulation by using an inference engine (Client Side).

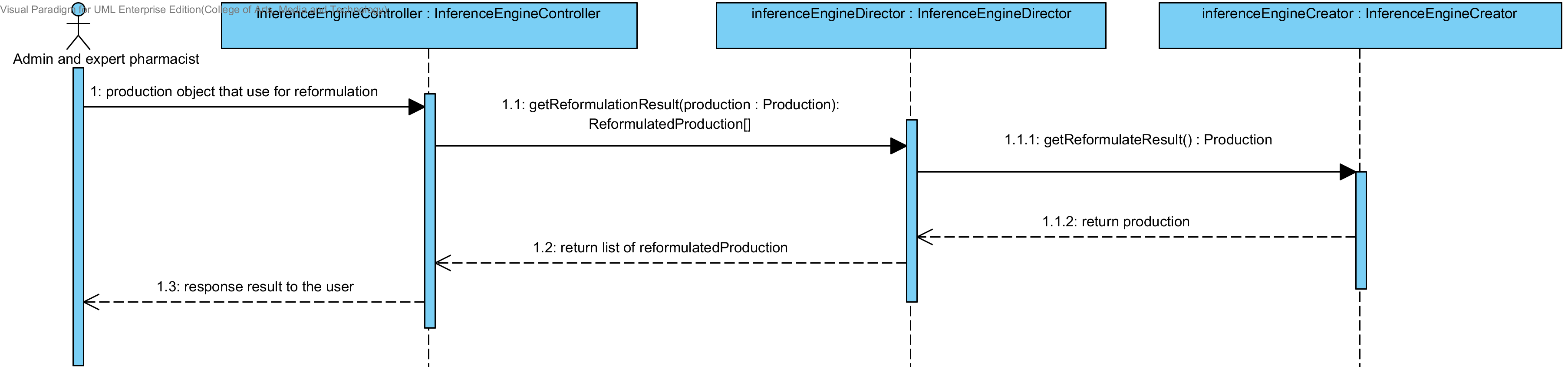
In a sequence diagram, the user can make drug reformulation by using the inference engine such as Rule Base System, Case Base Reasoning and Hybrid Reasoning. Firstly, the user input the production object that they want to reformulation. Then, the formulationCtrl will get the production data and call the formulation service for sending the production data to server. After that, the server make the reformulation by using the production object. Finally, the server will sends the reformulation result to client side for showing to the user.



**Figure 84: SQD-CI-13 – The user calculates a drug reformulation by using an inference engine (Client Side).**

##### SQD-SV-13: The user calculates a drug reformulation by using an inference engine (Server Side).

In a sequence diagram, the server can make drug reformulation by using the inference engine such as Rule Base System, Case Base Reasoning and Hybrid Reasoning. Firstly, the server receives production object from client Side by InferenceEngineController. Then, the InferenceEngineDirector will get the production data from the controller and call the inferenceEngineCreator for making drug reformulation. After that, the server will send the reformulation result to the client for showing to the user.



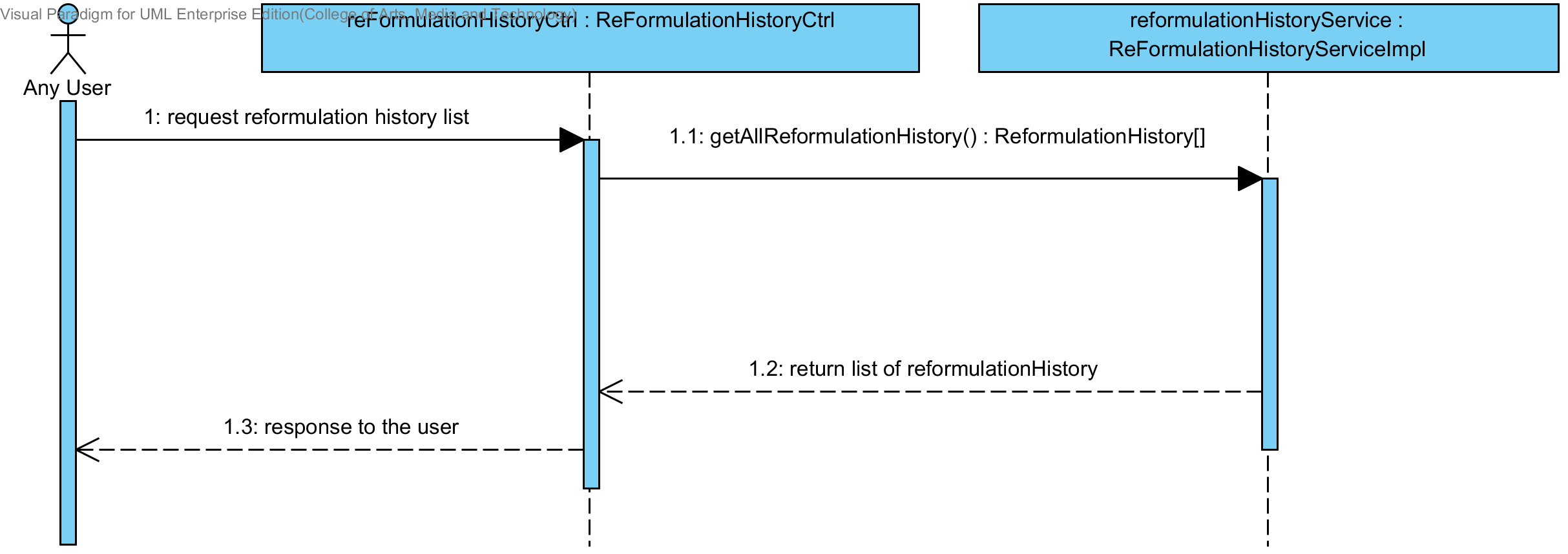
**Figure 85: SQD-SV-13 – The user calculates a drug reformulation by using an inference engine (Server Side).**

## Sub-Feature 3: View the drug reformulation history

### URS-07: The user views their drug reformulation history.

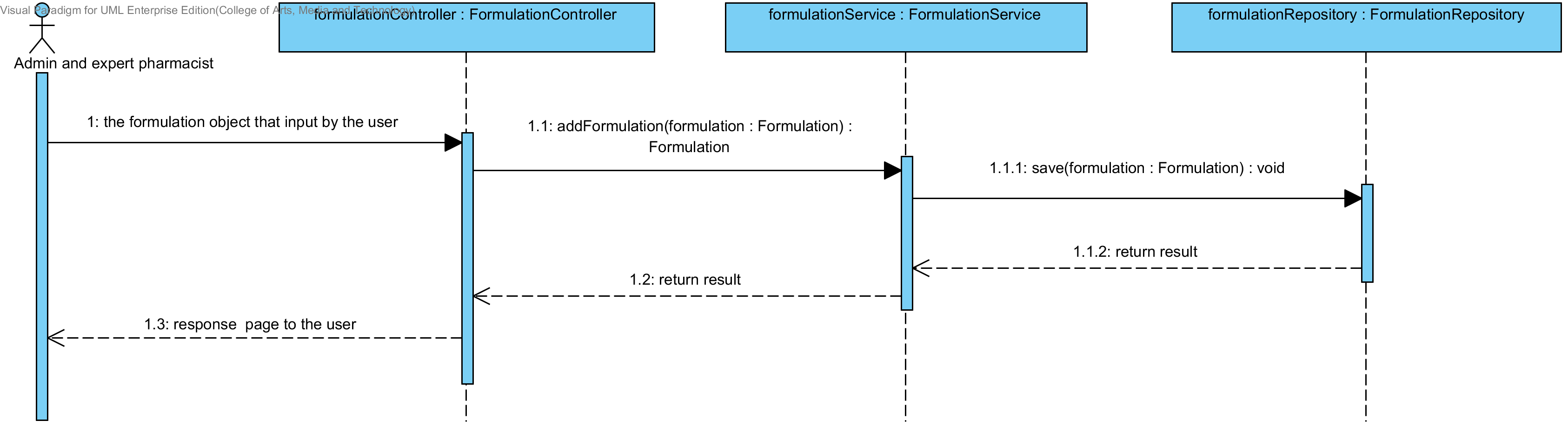
In a sequence diagram, the user can view the reformulation history Firstly, the open the reformulation history page. Then the system will show the reformulation history list to the user. The reformulation history gets from the database.

##### SQD-CI-14: The user adds a new drug’s formulation to the system (Client Side).



**Figure 86: SQD-CI-14 – The user adds a new drug formulation to the system (Client Side).**

##### SQD-SV-14: The user adds a new drug’s formulation to the system (Server Side).



**Figure 87: SQD-SV-14 – The user adds a new drug formulation to the system (server side).**

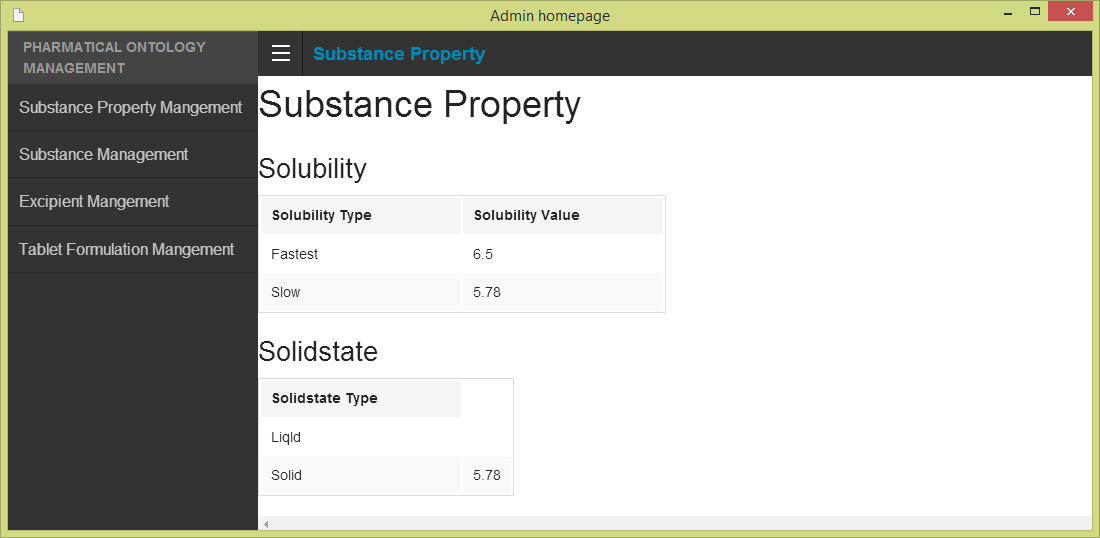
# Chapter 5 | User Interface

In the 1st progress the URS is related with the list of user interface that shown below this passage.

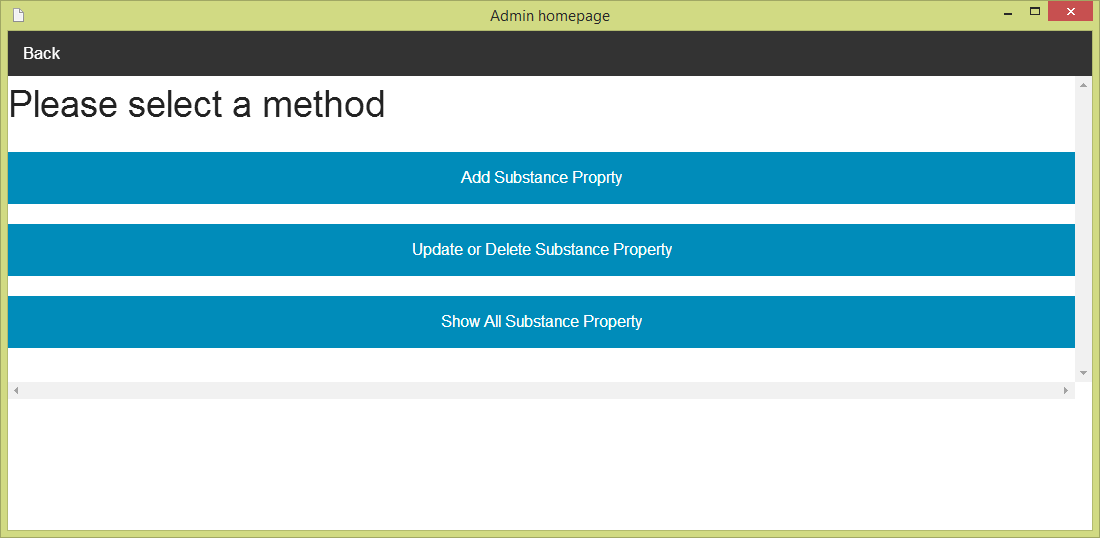
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Sub-Feature Name** | **URS No.** | **URS Name** | **Sequence Diagram** |
| 5 | Manage the drug substance property | URS-09 | The user adds a new substance property into the system. | UI-01,UI-02,UI-03,UI-04 |
| URS-10 | The user updates an existing substance property into the system. | UI-01,UI-02,UI-03,UI-05,UI-06 |
| URS-11 | The user deletes an existing substance property from the system. | UI-01,UI-02,UI-03,UI-05,UI-06 |
| 6 | Manage the drug substance | URS-12 | The user adds a new substance into the system. | UI-01,UI-07,UI-08 |
| URS-13 | The user updates an existing substance into the system. | UI-01,UI-07,UI-09,UI-10 |
| URS-14 | The user deletes an existing substance from the system. | UI-01,UI-07,UI-09,UI-10 |
| URS-15 | The user views the substance in the system. | UI-01,UI-07,UI-09,UI-10 |
| 7 | Manage the drug excipient | URS-16 | The user adds a new excipient to the system. | UI-01,UI-11,UI-12 |
| URS-17 | The user updates an existing drug excipient in the system. | UI-01,UI-09,UI-11,UI-13 |
| URS-18 | The user delete an existing drug excipient in the system. | UI-01,UI-09,UI-11,UI-13 |
| URS-19 | The user views all the drug excipient in the system. | UI-01,UI-09,UI-11,UI-13 |
| 8 | Manage the drug formulation | URS-20 | The user adds a new drug formulation case into the system. | UI-01,UI-14,UI-15 |
| URS-21 | The user updates an existing drug formulation case in the system. | UI-01,UI-14,UI-16,UI-17 |
| URS-22 | The user deletes an existing drug formulation case in the system. | UI-01,UI-14,UI-16,UI-17 |
| URS-23 | The user views all of the formulation in the system. | UI-01,UI-14,UI-16,UI-17 |

## 5.1- Sub-Feature 5: Manage the drug substance property

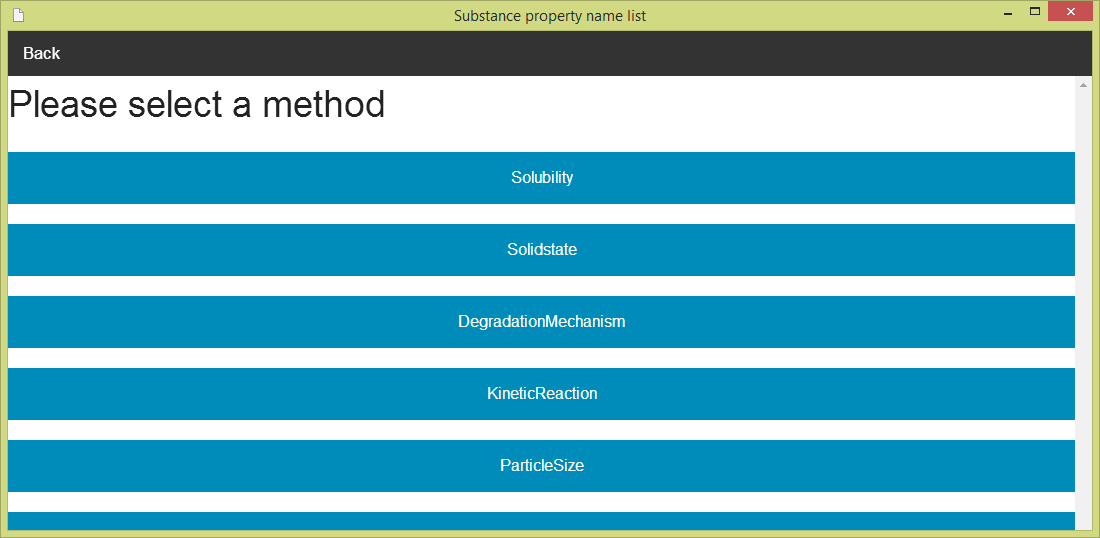
### 5.1.1 URS-09: The user adds a new substance property into the system.

 In the user interface design, the user can add a new substance property by opening the main page of a program, then the user select “Substance Property Management”. The main menu shows on the figure 68.

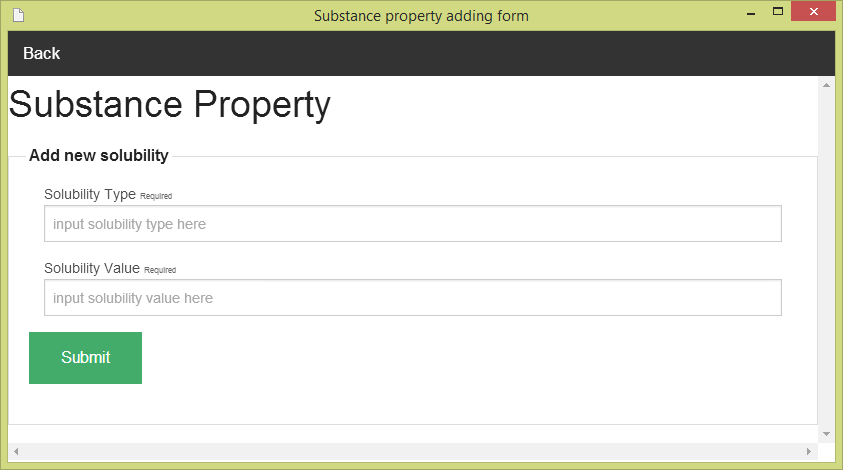
**Figure 68- UI-01: Admin main menu**

After the user selects “Substance Property Management”. The Substance management menu will show on the screen. The user must selects “Adding substance property” for making substance property adding. The substance management menu is shown on the figure 69.

**Figure 69- UI-02: Substance property management menu**

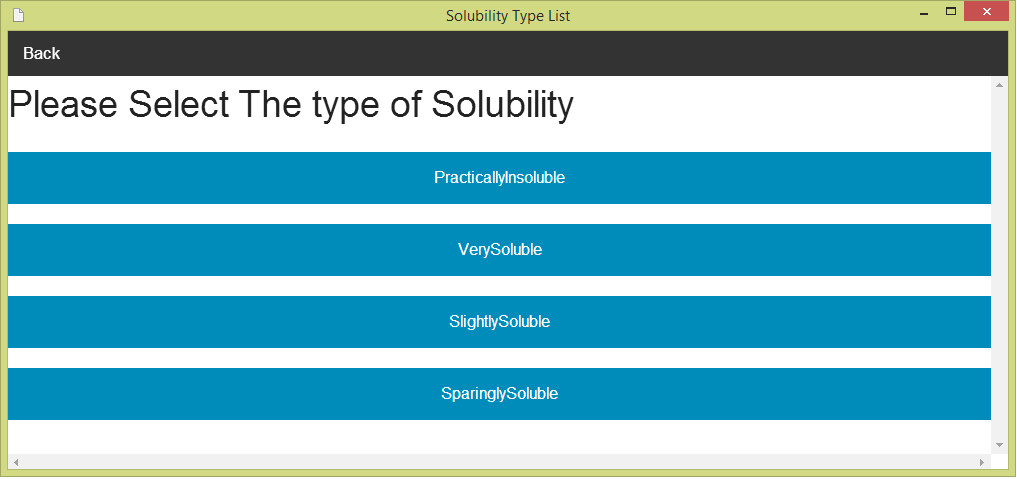
After the user selects “Add Substance Property”. The Substance property name list will show on the screen. The user must can select one substance property (e.g. Solubility) for making substance property adding. The substance property name list is shown on the figure 70.

**Figure 70 - UI-03: Substance property management menu**

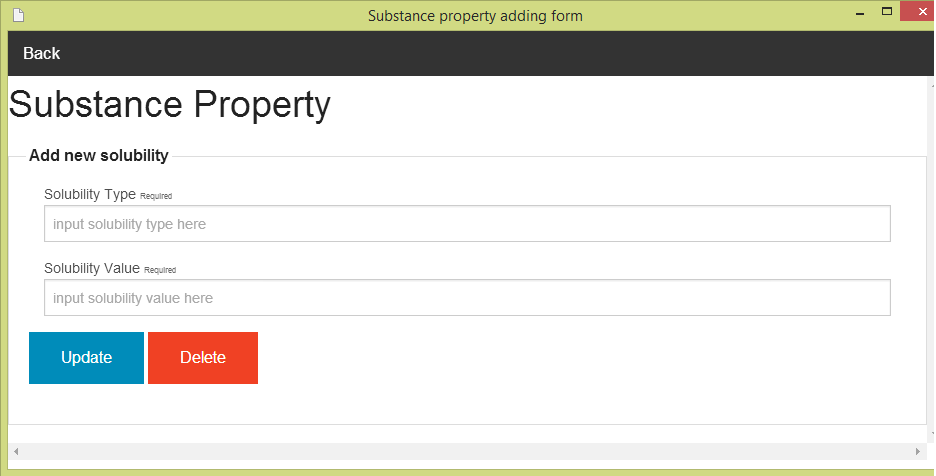
****After the user selects one of substance property in the list (e.g. solubility). The Substance property adding form will show on the screen. The user must input data of substance property to the form. The Substance adding form can check the input format. The substance property adding form is show on the figure 71

**Figure 71 - UI-04: Substance property adding form**

### 5.1.2 URS-10: The user updates an existing substance property into the system.

 In the user interface design, the user can update an existing substance property by opening the main page of the program (figure 68).Then, the user select “Substance Property Management." After that, the system will show substance management menu on the screen (figure 69). The user can select “update and delete substance property” for making update and delete substance property. Next, the system will show the substance property name list on the screen (figure 70).The user must select one of substance property (e.g. solubility). Then, the system will show a list of substance property type (e.g. if user selects solubility, the list of a type will be solubility type).The user must select one type from the list of substance property type. The list of substance property type is shown on the figure 72.

**Figure 72 - UI-05: Substance property adding form**

After the user selects one of substance property type in the list (e.g. Very soluble). The Substance property updating and deleing form will show on the screen. The user must input data of substance property to the form. The substance property updating can check the input format. The substance property updating and deleting form is show on the figure 73

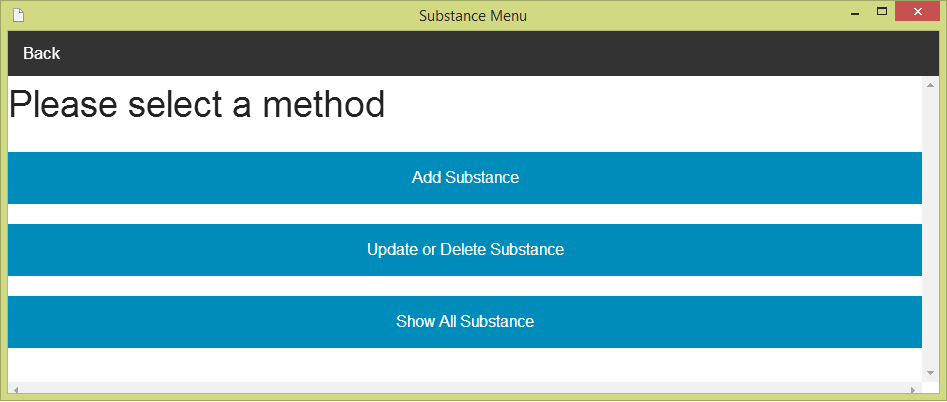
**Figure 73 - UI-06: Substance property updating and deleting form**

### 5.1.3 URS-11: The user deletes an existing substance property from the system.

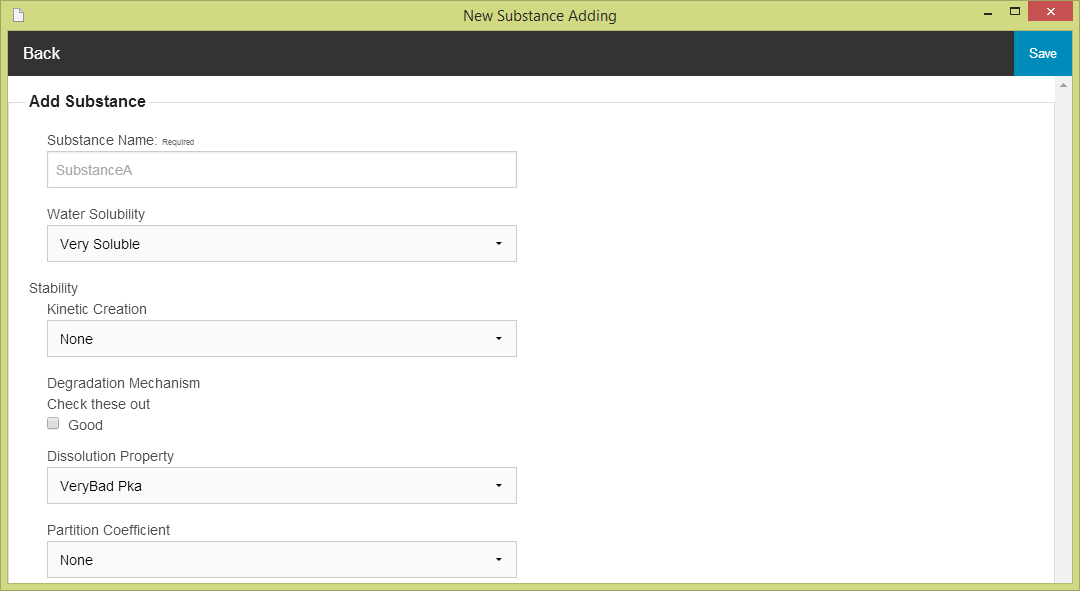
The user interface of this URS is same as URS-10.

## 5.2- Sub-Feature 6: Manage the drug substance

### 5.2.1 URS-12: The user adds a new substance to the system.

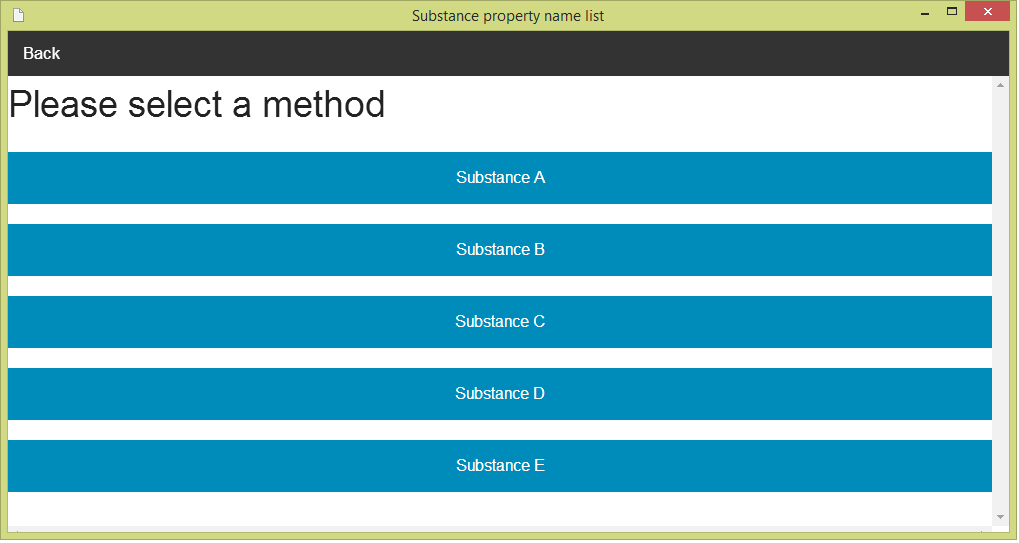
 In the user interface design, the user can add a new substance by opening the main page of a program (figure 68).Then the user select “Substance Management”. After that, the system will show substance management menu on the screen. The user can select “Add a new substance” for making substance adding. The substance management menu is show on the figure 74.

**Figure 74 - UI-07: Substance management menu**

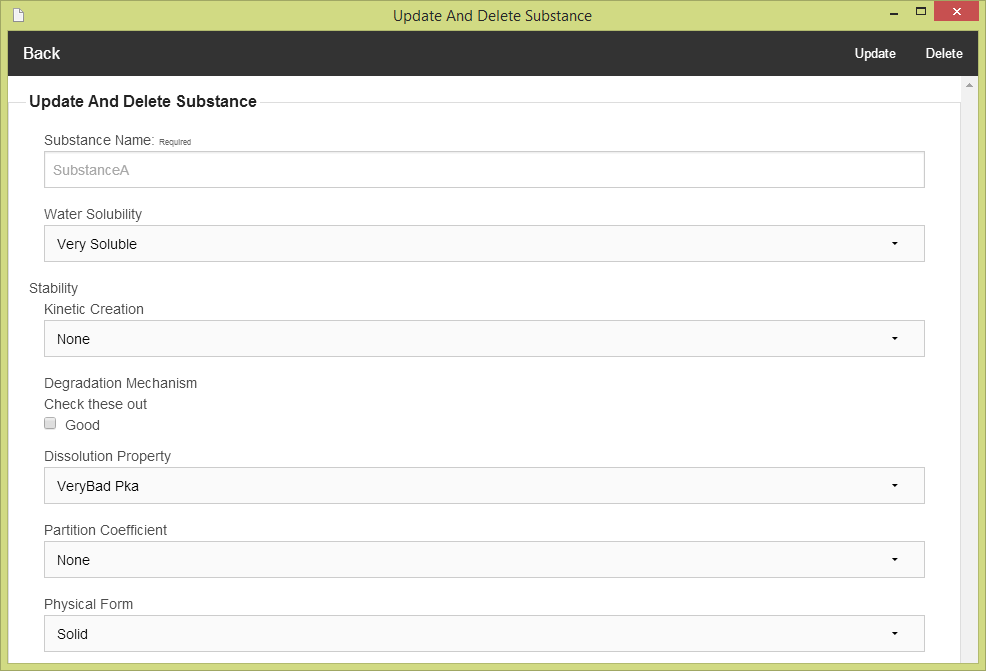
After the user selects “Adding substance”. The Substance adding form will show on the screen. The user must input data of substance to the form. The substance adding form can check the input format for the user. The substance adding form is show on the figure 75

**Figure 75 - UI-08: Substance adding form**

### 5.2.2 URS-13: The user updates an existing substance in the system.

 In the user interface design, the user can update an existing substance by opening the main page of the program (figure 68).Then, the user select “Substance Management." After that, the system will show substance management menu on the screen (figure 74). The user can select “update and delete substance”. Next, the system will show the substance name list on the screen .The user must select one of substance property (e.g. Substance A) for making update and delete the substance. The substance name list is show the figure 76.

**Figure 76 - UI-09: Substance name list**

****After the user selects one of substance property (e.g. Substance A). The Substance updating and deleting form will show on the screen. The user must input data of substance to the form. The substance updating and deleting form can check the input format from the user. The substance updating and deleting form is show on the figure 77.

**Figure 77 - UI-10: Substance updating and deleting form**

### 5.2.3 URS-14: The user deletes an existing substance from the system.

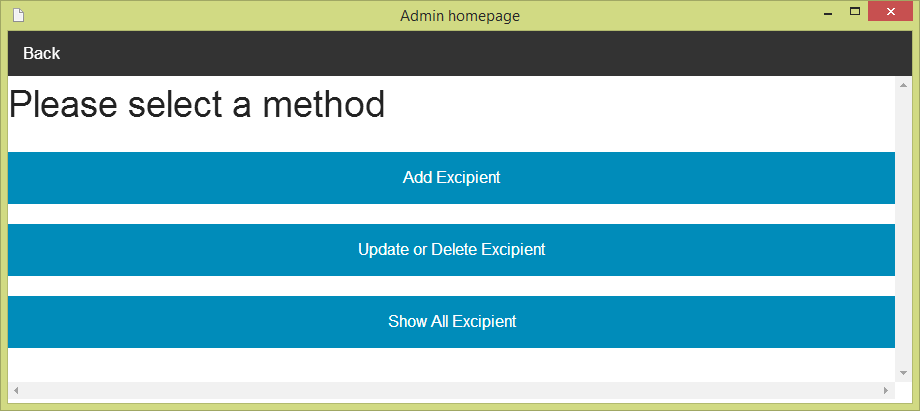
The user interface of this URS is same as URS-13.

### 5.2.4 URS-15: The user views the substance in the system

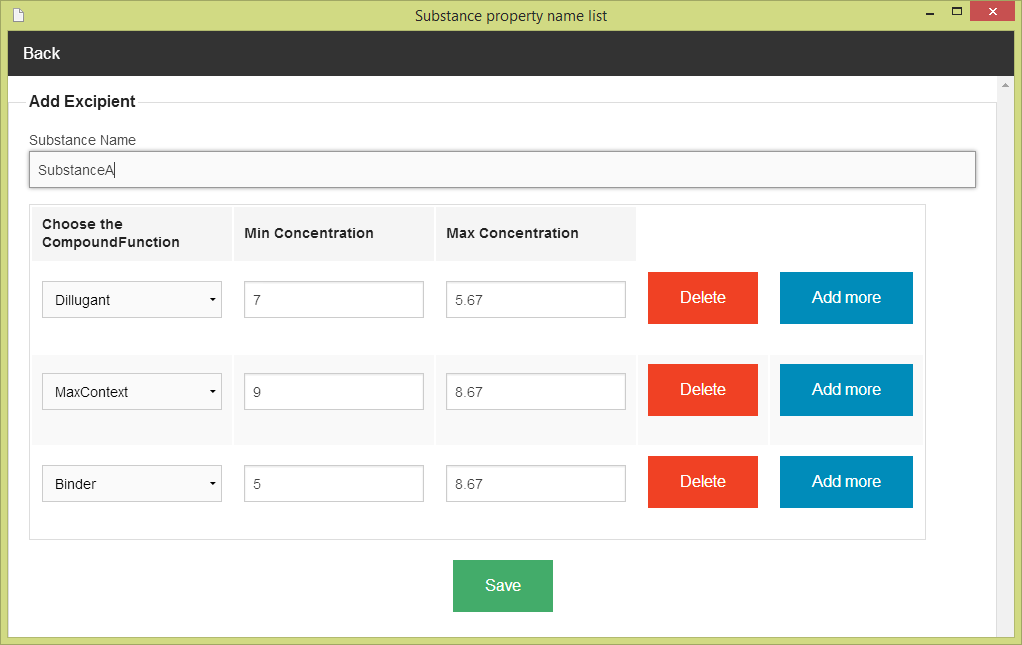
The user interface of this URS is same as URS-13.

## 5.3- Sub-Feature 7: Manage the drug excipient

### 5.3.1 URS-16: The user adds a new excipient to the system.

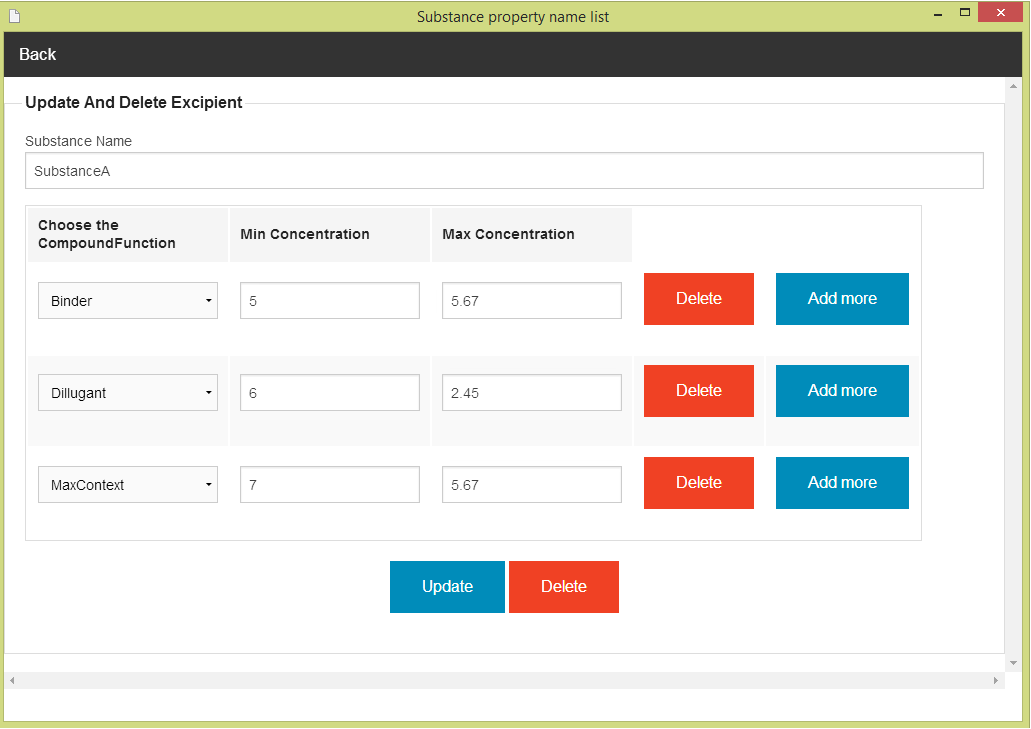
**** In the user interface design, the user can add a new excipient by opening the main page of a program (figure 68).Then the user select “Excipient Management”. After that, the system will show Excipient management menu on the screen. The user can select “Add Excipient” for making excipient adding. The excipient management menu is show on the figure 78.

**Figure 78 - UI-11: Excipient management menu**

****After the user selects “Adding Excipient”. The system show the list of substance name on the screen (Figure 76). The user must select one of substance from the substance name list. After that, the system will show excipient adding form. The user must input excipient data to the form. The excipient adding form can check the input format from the user. The excipient adding form is show on the figure 79.

**Figure 79 - UI-12: Excipient adding form**

### 5.3.2 URS-17: The user updates an existing excipient in the system.

**** In the user interface design, the user can update an existing excipient by opening the main page of the program (figure 68).Then, the user select “Excipient Management." After that, the system will show excipient management menu on the screen (figure 78). The user can select “update and delete excipient”. Next, the system will show the substance name list on the screen (Figure 76) .The user must select one of substance property (e.g. Substance A) for making update and delete the substance. After that, the system will show the excipient updating and deleting form. The user must input data of excipient to the form. The excipient adding form can check the input format from the user. The excipient updating and deleting form is show on the figure 80

**Figure 80 - UI-13: Excipient updating and deleting form**

### 5.2.3 URS-18: The user deletes an existing excipient from the system.

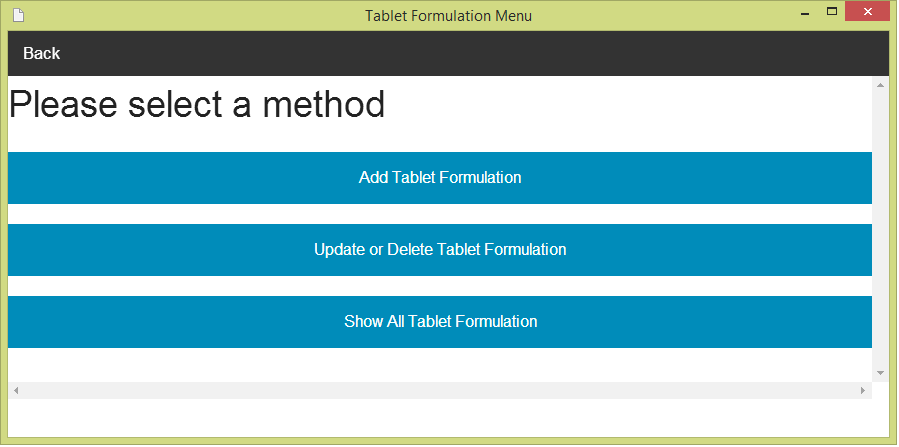
The user interface of this URS is same as URS-17.

### 5.2.4 URS-19: The user views the excipient in the system

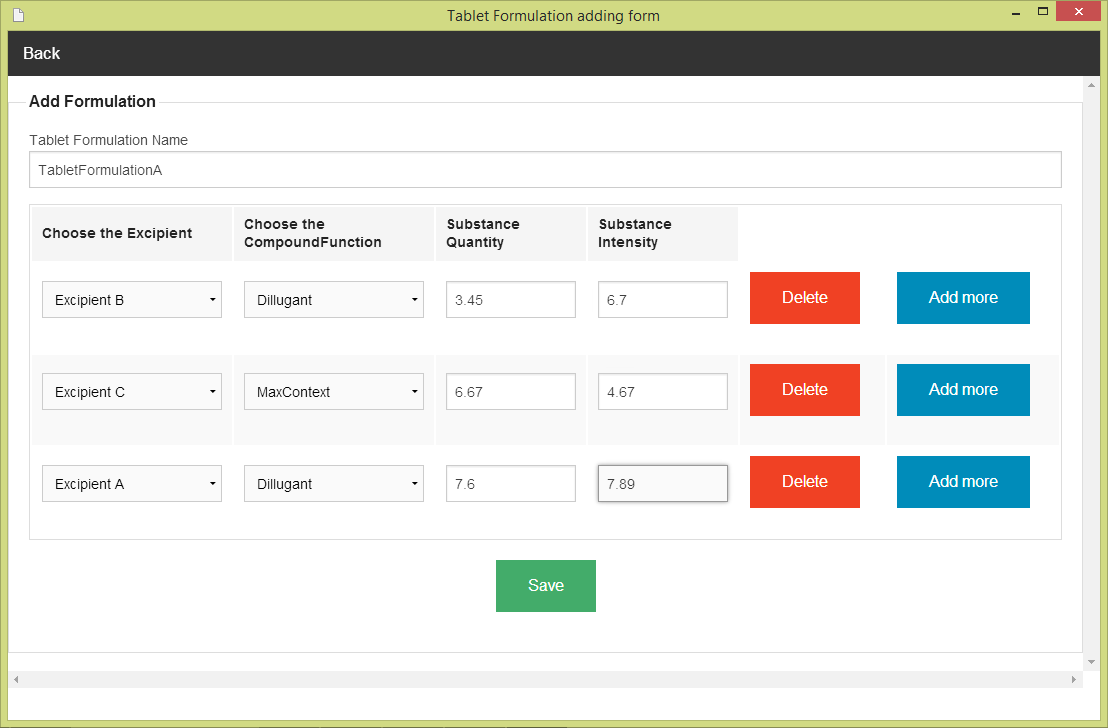
The user interface of this URS is same as URS-17.

## 5.4- Sub-Feature 8: Manage the drug formulation

### 5.4.1 URS-20: The user adds a new formulation to the system.

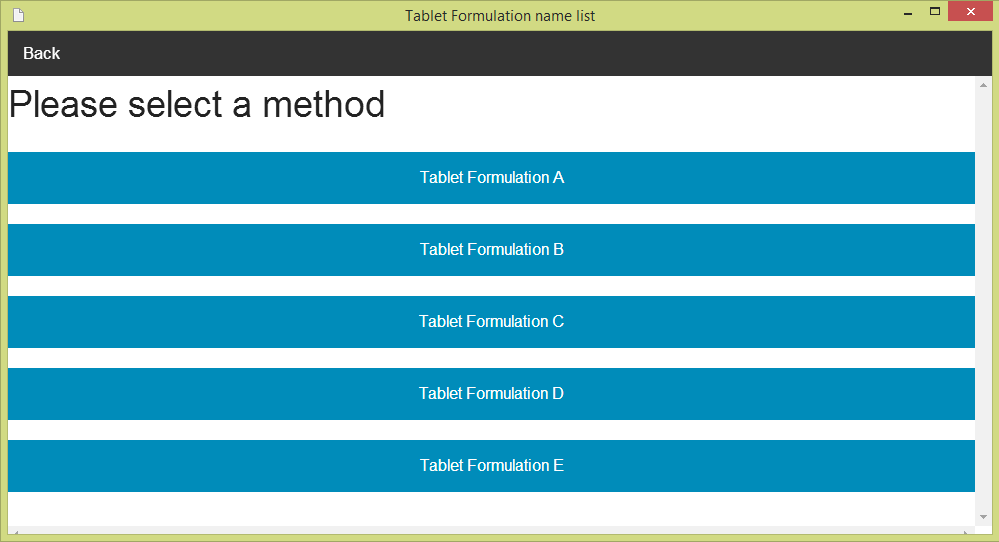
 In the user interface design, the user can add a new formulation by opening the main page of a program (figure 68).Then the user select “Tablet formulation Management”. After that, the system will show tablet formulation management menu on the screen. The user can select “Add a new formulation” for making tablet formulation adding. The tablet formulation management menu is show on the figure 81.

**Figure 81 - UI-14: Tablet Formulation management menu**

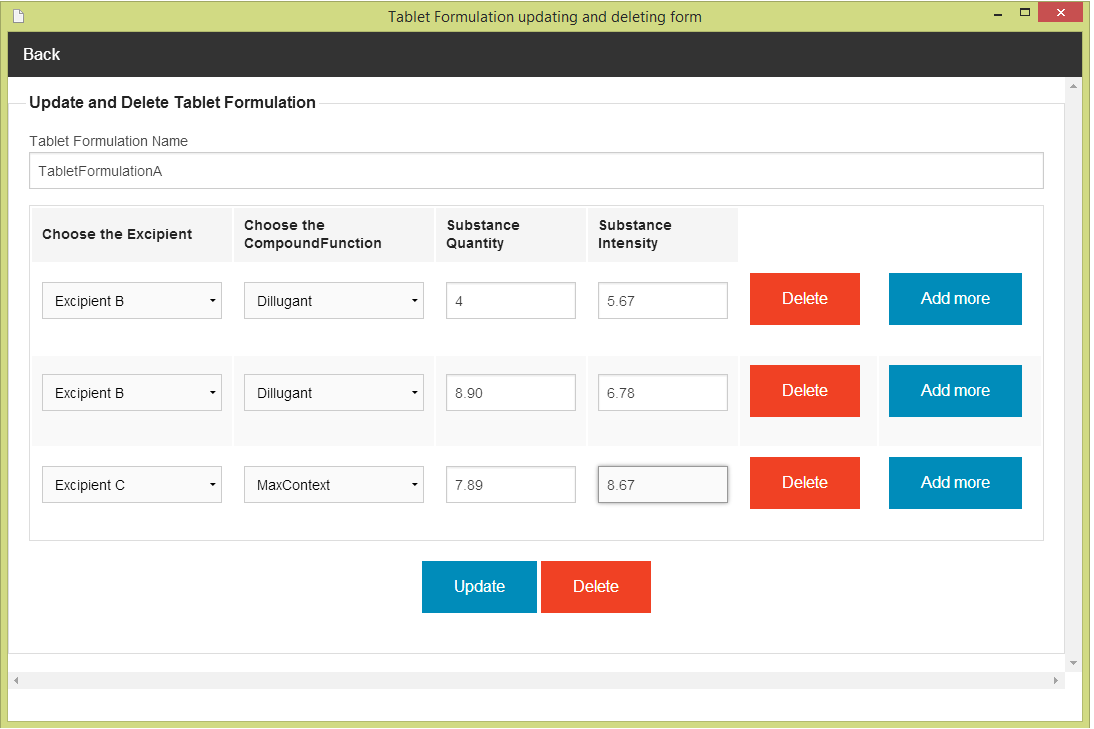
****After the user selects “Adding Tablet Formulation”. The Tablet Formulation adding form will show on the screen. The user must input the tablet formulation data to the form. The tablet formulation adding form can check the input format from the user. The tablet formulation adding form is show on the figure 82.

**Figure 82 - UI-15: Tablet Formulation adding form**

### 5.4.2 URS-21: The user updates a tablet formulation in the system.

 In the user interface design, the user can update an existing tablet formulation by opening the main page of the program (figure 68).Then, the user select “Tablet Formulation Management." After that, the system will show tablet formulation management menu on the screen (figure 81). The user can select “update and delete TabletFormulation”. Next, the system will show the list of tablet formulation name on the screen .The user must select one of tablet formulation (e.g. TabletFormulation A) for making update and delete the TabletFormulation. The list of tablet formulation is show the figure 83.

**Figure 83 - UI-16: The list of tablet formulation name**

After the user selects one of tablet formulation (e.g. TabletFormulation A). The TabletFormulation updating and deleting form will show on the screen. The user must input tablet formulation data to the form. The tabletformulation updating and deleting form can check the input format from the user. The tabletformulation updating and deleting form is show on the figure 84

**Figure 84 - UI-17: TabletFormulation updating and deleting form**

### 5.4.3 URS-22: The user deletes an existing tablet formulation from the system.

The user interface of this URS is same as URS-21.

### 5.4.4 URS-23: The user views the tablet formulation in the system

The user interface of this URS is same as URS-21.

1. NS – Narongrit Saisuwan , PW – Panupak Wichaidit , CD - Chartchai Doungsa-ard [↑](#footnote-ref-1)