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

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

[1]

[1] [1] NS – Narongrit Saisuwan , PW – Panupak Wichaidit , CD - Chartchai Doungsa-ard

Contents

Chapter 1 | Introduction

1.1. 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. In the 1st progress, the software design consists of the list of Sub-Feature in a software requirement specification that is illustrated in Figure 1.

No	Sub-Feature Name	URS No.	URS Name	Actor
5	Manage the drug substance property	URS-09	The user adds a new substance property into the system.	Expert Pharmacists, Administrator
		URS-10	The user updates an existing substance property into the system.	
		URS-11	The user deletes an existing substance property from the system.	
6	Manage the drug substance	URS-12	The user adds a new substance into the system.	Expert Pharmacists, Administrator
		URS-13	The user updates an existing substance into the system.	
		URS-14	The user deletes an existing substance from the system.	
		URS-15	The user views the substance in the system.	
7	Manage the drug excipient	URS-16	The user adds a new excipient to the system.	Expert Pharmacists, Administrator
		URS-17	The user updates an existing drug excipient in the system.	
		URS-18	The user delete an existing drug excipient in the system.	
		URS-19	The user views all the drug excipient in the system.	
8	Manage the drug formulation	URS-20	The user adds a new drug formulation case into the system.	Expert Pharmacists, Administrator
		URS-21	The user updates an existing drug formulation case in the system.	
		URS-22	The user deletes an existing drug formulation case in the system.	
		URS-23	The user views all of the formulation in the system.	

Table 1: The list of Sub-Feature in software requirement specification in the 1st progress

1.2. 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
SQD	Sequence Diagram
UI	User interface
SDD	Software Design Document

Chapter 2 | Sub-Feature 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.

Sub-Feature 5: PDPO System

Sub-Feature 5: Manage the drug substance property

- CD-01: solubility class diagram
- CD-02: Degradation mechanism Class Diagram
- CD-03: Kinetic Reaction Class Diagram
- CD-04 :PKA Class Diagram
- CD-05: Partition Coefficient Class Diagram
- CD-06 : Solid state class diagram
- CD-07 :Hygroscopicity class diagram
- CD-08 :Particle Size class diagram
- CD-09 :Flow ability class diagram
- CD-10 :Density class diagram
- CD-11 : Compound Function class diagram

Sub-Feature 6: Manage the drug substance

- CD-12 :Substance class diagram

Sub-Feature 7: Manage the drug excipient

- CD-13 : Excipient class diagram

Sub-Feature 8: Manage the drug formulation

- CD-14 :Tablet Formulation class diagram

Chapter 3 | Class Diagram

3.1. Sub-Feature 5: Manage the drug substance property

The Sub-Feature 5 is the substance property management. The users can add, update and delete the substance property. In the Ontology base expert system for generic drug production of pharmaceutical dosage form, there are 9 substance properties such as solubility, Degradation mechanism, Kinetic reaction, Pka, Partition Coefficient, Solid-state, Hygroscopicity, Flow ability, Particle Size and Density. Each substance property is a necessary attribute to make a substance. The relationship between substance entity and substance property entity is illustrated in the Figure 01 below on this passage.

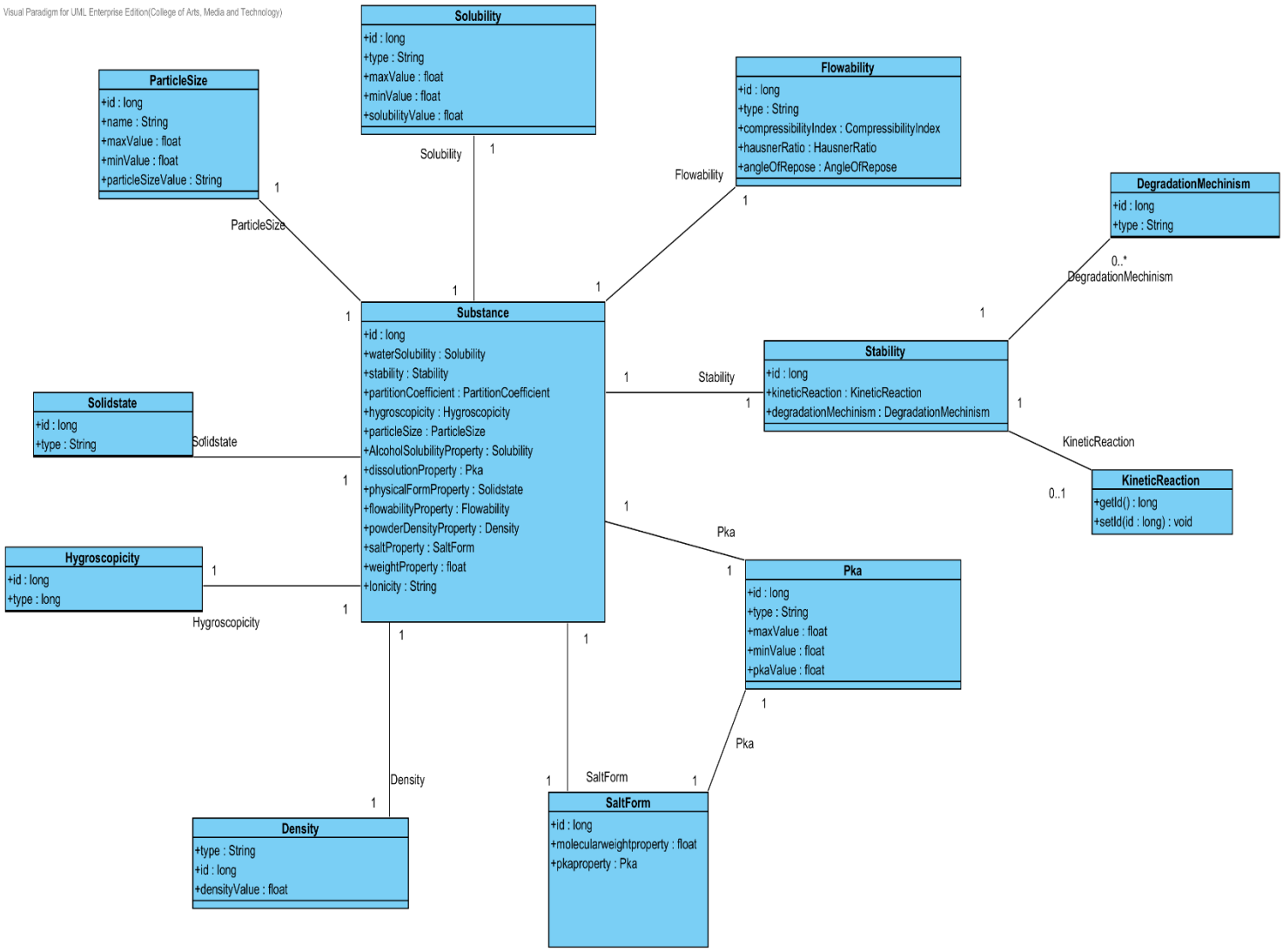


Figure- 01: The entity relationship between a substance property and a substance.

3.1.1- CD-01: Solubility Class diagram

3.1.1.1: Class diagram

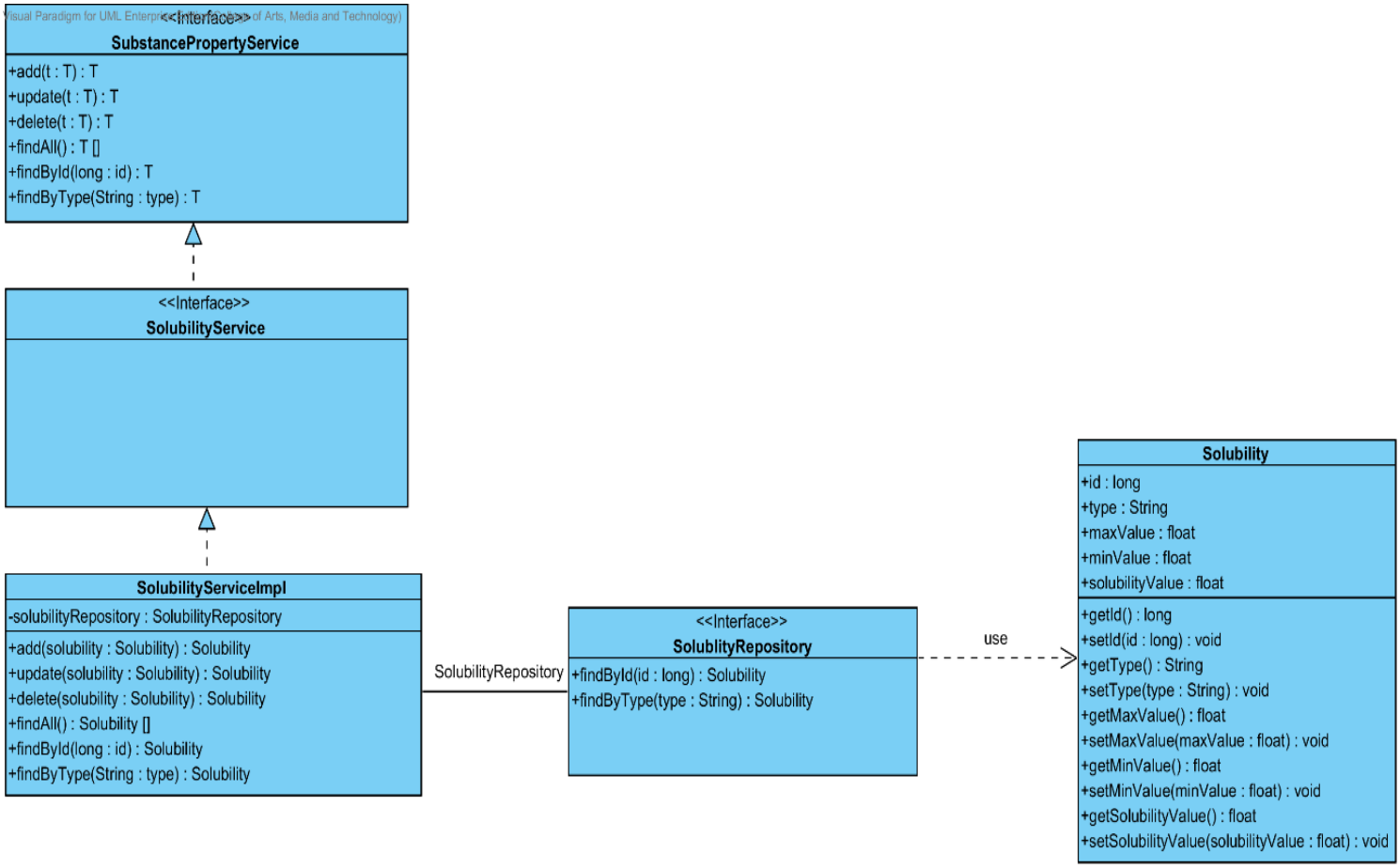


Figure 02 - CD-01: Solubility Class diagram

3.1.1.2: Class description

From the figure 2, it can divide into 5 important classes. The detail of each class is described on the next paragraph.

3.1.1.2.1 Solubility class

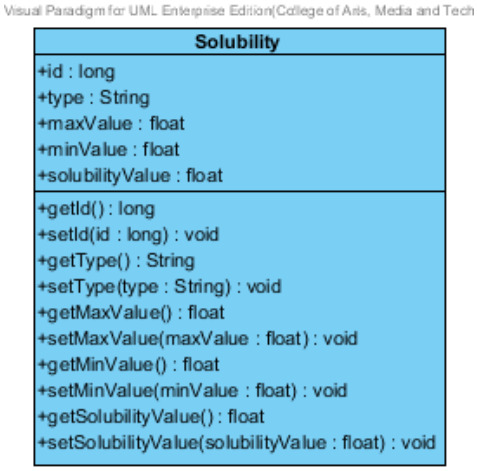


Figure 03 – The solubility class

Solubility class is a part of drug's substance. Solubility class is an entity class that will be saved to the system. The solubility class consists of 5 attributes follow the list below this passage.

3.1.1.2.1.1: Attribute description

- **Id** – the identity of the solubility class. Id attribute is a long number.
- **Type** – the type of the solubility class. In the pharmacy, the solubility type shows a good or bad soluble in each substance. The solubility type value is string.
- **Max Value** - The max value of each type of solubility. The pharmacists can use the max value and min value for solubility type identification.
- **Min Value** – The min value of each type of solubility. The pharmacists can use the max value and min value for solubility type identification.
- **Solubility value** - The solubility value is a value of each type of the solubility. The solubility value is stable value. The value is float number.

3.1.1.2.1.2: Method description

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

3.1.1.2.2 SolubilityRepository Interface

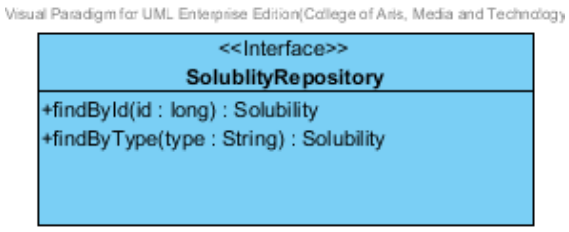


Figure 04 – The solubility repository interface

SolubilityRepository Interface is an interface that use for CRUD with entity classes in the system. The most of SolubilityRepository interface's method is generated from Spring data MongoDB framework. SolubilityRepository interface consists of 5 methods is shown below this passage.

3.1.1.2.2.1: Attribute description

N/A

3.1.1.2.2.2: Method description

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

3.1.1.2.3 SolubilityService

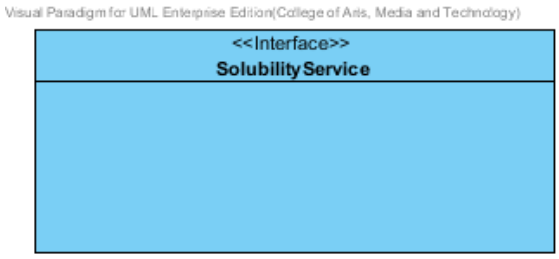


Figure 05 – The SolubilityService Interface.

SolubilityService is business processing logic for solubility entity. SolubilityService manages the solubility data through the SolubilityRepository interface. SolubilityService consists of 6 methods follow the list below this passage.

3.1.1.2.3.1: Attribute description

N/A

3.1.1.2.3.2: Method description

- **add (solubility: Solubility)** – The adding solubility method is used, when the user wants to add a new solubility to the database. This method adds a new solubility by input variable of solubility object. If the solubility object that input by the user is not contained in the database, this method will add a new solubility to the database and return the solubility object from the database to the user after the adding solubility is successful. On the other hand, when the solubility object that input by the user is contained in the database. This method will return a null value to the user.
- **update (solubility: Solubility)** - The updating solubility method is used, when the user wants to update an existing solubility in the database. This method update the existing solubility by input variable of solubility object. If the solubility object that input by the user is contained in the database, this method will update an existing solubility in the database and return the solubility object from the database to the user after the updating solubility is successful. On the other hand, when the solubility object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (solubility: Solubility)** – The deleting solubility method is used when the user wants to deletes the existing solubility from the database. This method delete the solubility by input variable of solubility object. If the solubility object that input by the user is contained in the database, this method will delete an existing solubility from the database and return the solubility object to the user after the deleting solubility is successful. On the other hand, when the solubility object that input by the user is not

contained in the database. This method will return a null value to the user.

- **findAll() : Solubility[]** – The findAll method is used, when the user wants to get all solubility data in the database. This method is return as a list of solubility object from the database.
- **findById(id : long) : Solubility** – The findById method is used, when the user wants to get the solubility data in the system. This method gets solubility 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.
- **findByType(type : String)** –The findByType method is used when the user wants to get solubility data in the system. This method gets solubility object from the database by type that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.1.2.4 SolubilityServiceImpl

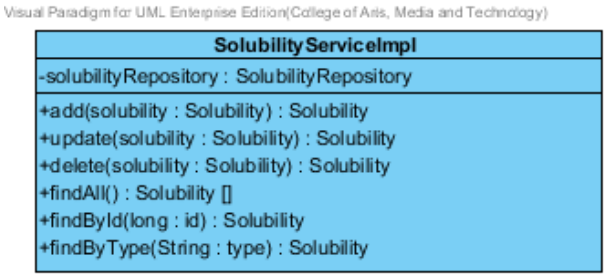


Figure 06 – The SolubilityServiceImpl class

SolubilityServiceImpl is the solubility service class that implements the method from SolubilityService. So, the method of SolubilityServiceImpl is same as SolubilityService.

3.1.1.2.4.1: Attribute description

- **SolubilityRepository** – the repository of solubility. This attribute is used for solubility data management.

3.1.1.2.4.2: Method description

Same as SolubilityService

3.1.1.2.5 SubstancePropertyService

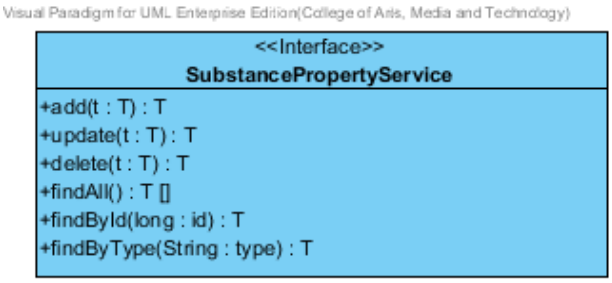


Figure 07 – The substance property service interface

SubstancePropertyService is the generic interface that is parameterized over a type. The type of parameter is followed the entity object that user inputs.

3.1.2- CD-02: Degradation mechanism Class Diagram

3.1.2.1: Class diagram

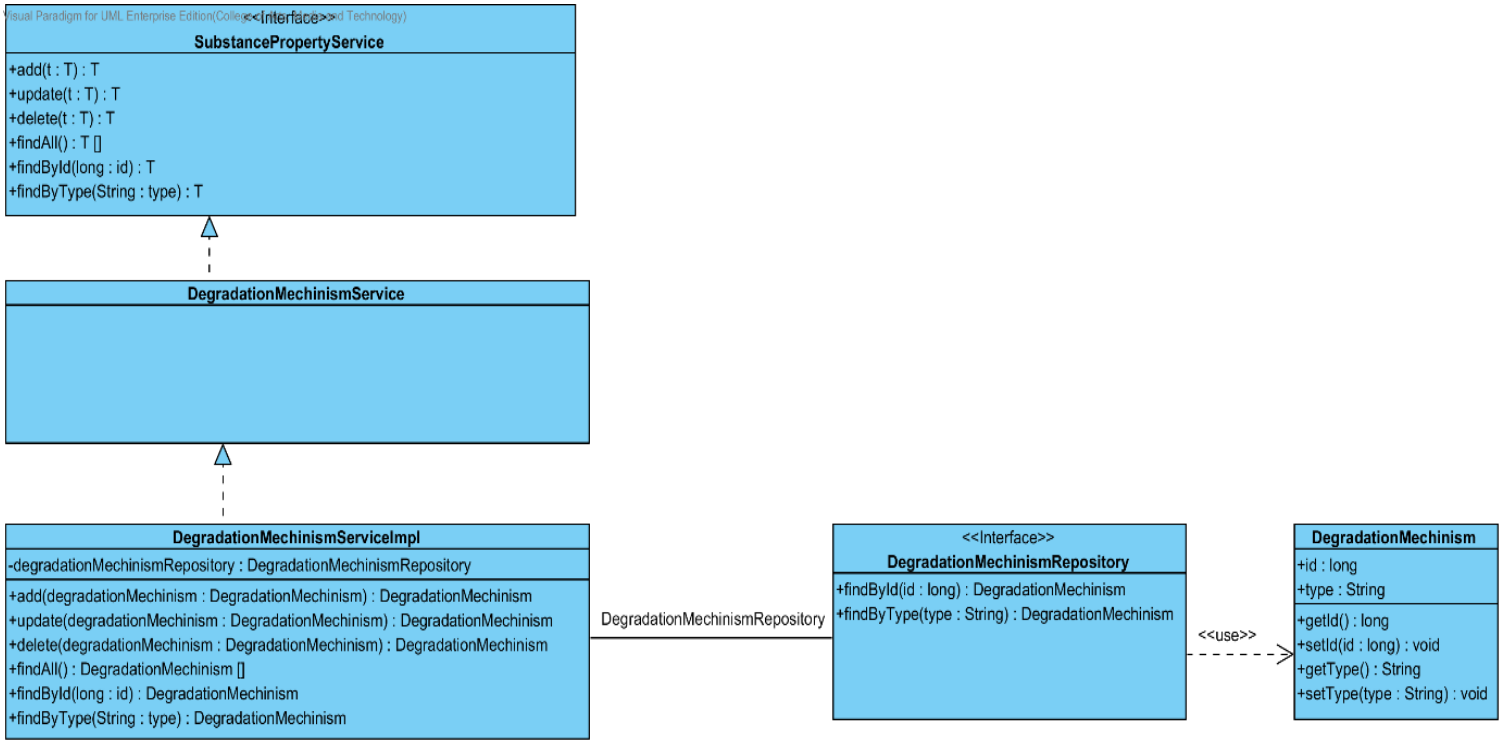


Figure 08 - CD-02: Degradation mechanism Class Diagram

3.1.2.2: Class description

From the figure 8, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.2.2.1 DegradationMechinism class

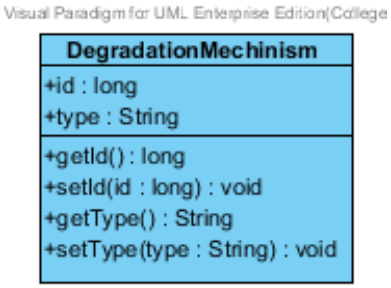


Figure 09 – The DegradationMechanism class

DegradationMechinism class is a part of drug's substance. DegradationMechinism class is an entity class that will be saved to the system. The DegradationMechinism class consists of 2 attributes follow the list below this passage.

3.1.2.2.1.1: Attribute description

- **Id** – The identity of the degradationMechinism class. Id attribute is a long number.
- **Type** – The type of the degradationMechinism class. In the pharmacy, the degradationMechinism type show process of degrading in each substance. The degradationMechinism type attribute is string.

3.1.2.2.1.2: Method description

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

3.1.2.2.2 DegradationMechinism Repository Interface

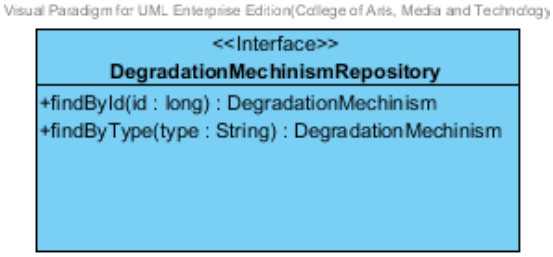


Figure -10: The Degradation Mechanism Repository

DegradationMechinismRepository Interface is an interface that use for CRUD with entity classes in the system. The most of DegradationMechinismRepository interface's method is generated from Spring data MongoDB framework. DegradationMechinismRepository interface consists of 5 methods is shown below this passage.

3.1.2.2.2.1: Attribute description

N/A

3.1.2.2.2.2: Method description

- **Save (degradationMechinism: DegradationMechinism)** - The save method is generated from Spring data MongoDB framework. This method is used when the user want to add a new DegradationMechinism or update existing DegradationMechinism to the system. The input variable is a DegradationMechinism object.
- **Delete (degradationMechinism: DegradationMechinism)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the degradationMechinism from the system. The input variable is degradationMechinism object.
- **findAll (): DegradationMechinism []** – The findAll is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of degradationMechinism data from the system. The result of this method is a list of degradationMechinism.
- **findById (id: long): DegradationMechinism** – The findById method is generated from Spring data MongoDB framework. The developer can makes configuration at this method for query the data by themselves. The findById is used when the user wants to retrieve the degradationMechinism data from the system. The input variable is id. The findById gets the DegradationMechinism object by the id of DegradationMechinism.
- **findByType (type: String)** – The findByType method is generated from Spring data MongoDB framework. The developer can make configuration at this method for query the data by themselves. The findBytype method is used when the user wants to retrieve the degradationMechinism data from the system. The findByType gets the DegradationMechinism object by the type of DegradationMechinism.

3.1.2.2.3 DegradationMechinismService

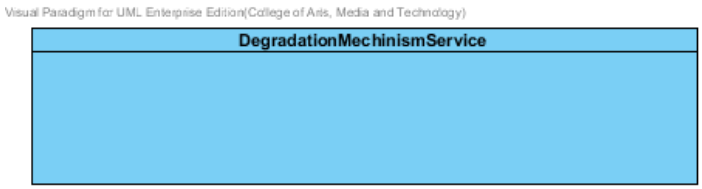


Figure -11: The DegradationMechanismService interface

DegradationMechinismService is business processing logic for DegradationMechinism entity. DegradationMechinismService manages the degradationMechinism data through the DegradationMechinismRepository interface. DegradationMechinismService consists of 6 methods follow the list below this passage.

3.1.2.2.3.1: Attribute description

N/A

3.1.2.2.3.2: Method description

- **add (degradationMechinism: DegradationMechinism)** – The adding degradationMechinism method is used, when the user wants to add a new degradationMechinism to the database. This method adds a new degradationMechinism by input variable of degradationMechinism object. If the degradationMechinism object that input by the user is not contained in the database, this method will add a new degradationMechinism to the database and return the degradationMechinism object from the database to the user after the adding degradationMechinism is successful. On the other hand, when the degradationMechinism object that input by the user is contained in the database. This method will return a null value to the user.
- **update (degradationMechinism: DegradationMechinism)** - The updating degradationMechinism method is used, when the user wants to update an existing degradationMechinism in the database. This method update the existing degradationMechinism by input variable of degradationMechinism object. If the degradationMechinism object that input by the user is contained in the database, this method will update an existing degradationMechinism in the database and return the degradationMechinism object from the database to the user after the updating degradationMechinism is successful. On the other hand, when the degradationMechinism object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (degradationMechinism: DegradationMechinism)** – The deleting degradationMechinism method is used when the user wants to deletes the existing degradationMechinism from the database. This method delete the degradationMechinism by input variable of degradationMechinism object. If the degradationMechinism object that input by the user is contained in the database, this method will delete an

existing degradationMechinism from the database and return the degradationMechinism object to the user after the deleting degradationMechinism is successful. On the other hand, when the degradationMechinism object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll():DegradationMechinism []** – The findAll method is used when the user wants to get all data of degradationMechinism in the database. This method is return as a list of degradationMechinism from the database.
- **findById (id: long): DegradationMechinism** – The findById method is used when the user wants to get the degradationMechinism data in the system. This method gets degradationMechinism 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.
- **findByType(type: String) : DegradationMechinism** –The findByType method is used when the user wants to get degradationMechinism data in the system. This method gets degradationMechinism object from the database by type that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.2.2.4 DegradationMechinismServiceImpl

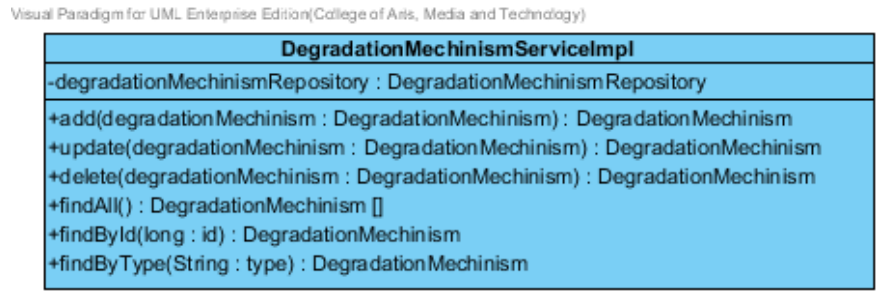


Figure -12: The DegradationMechanismServiceImpl class

DegradationMechinismServiceImpl is the service class that implements the method from DegradationMechinismService. So, the method of DegradationMechinismServiceImpl is same as DegradationMechinismService.

3.1.2.2.4.1: Attribute description

- **DegradationMechinismRepository** – the repository of DegradationMechinism. This attribute is used for DegradationMechinism data management.

3.1.2.2.4.2: Method description

Same as DegradationMechinismService

3.1.2.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.3- CD-03: Kinetic Reaction Class Diagram

3.1.3.1: Class diagram

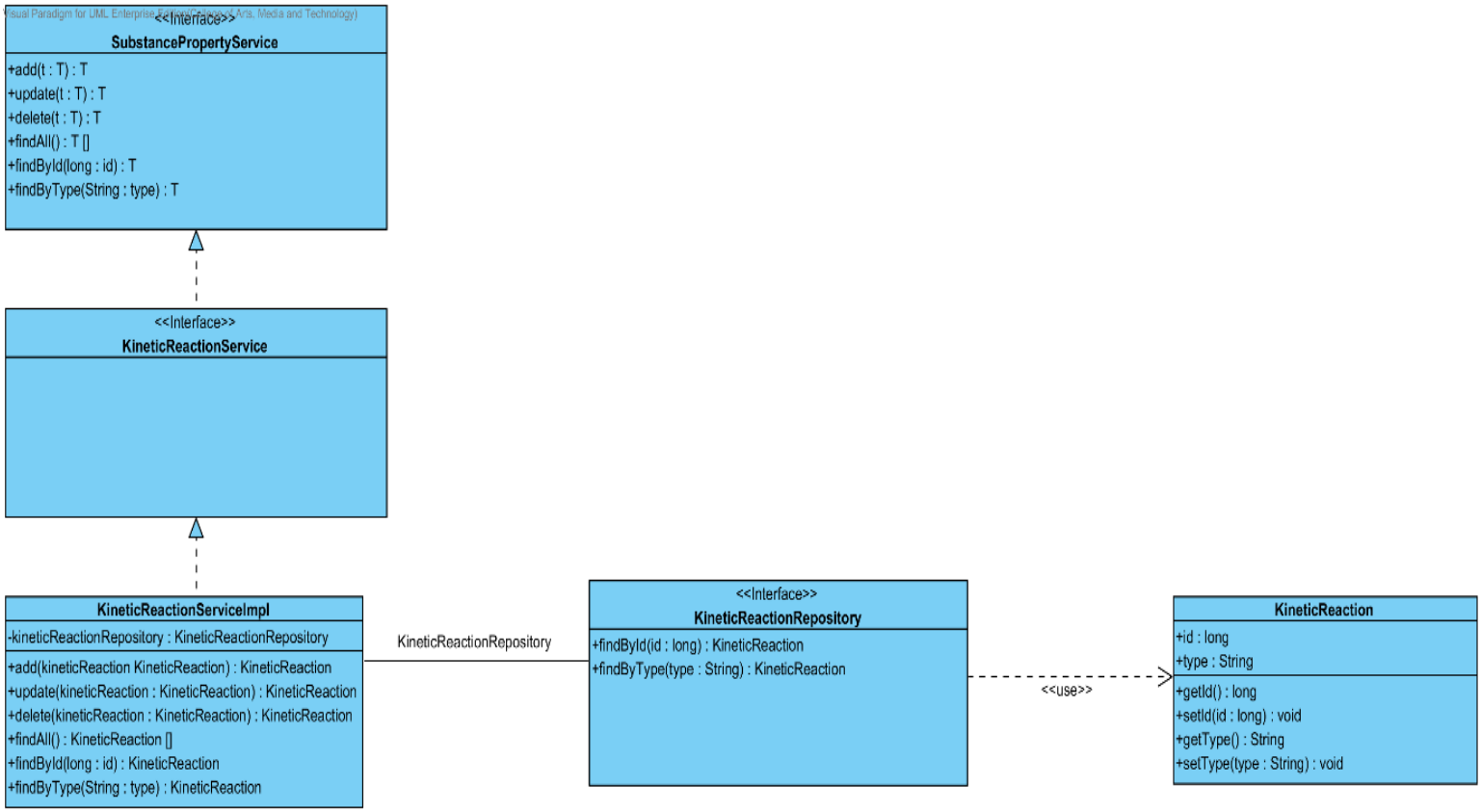


Figure 12 - CD-03: Kinetic Reaction Class Diagram

3.1.3.2: Class description

From the figure 12, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.3.2.1 Kinetic Reaction class

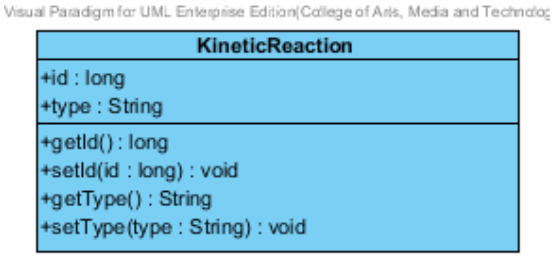


Figure 13 – The KineticReaction class

KineticReaction class is a part of drug's substance. KineticReaction class is an entity class that will be saved to the system. The KineticReaction class consists of 2 attributes follow the list below this passage.

3.1.3.2.1.1: Attribute description

- **Id** – The identity of the kineticReaction class. Id attribute is a long number.
- **Type** – The type of the kineticReaction class. In the pharmacy, the kineticReaction type shows the reaction of a substance motion. The kineticReaction type attribute is a string.

3.1.3.2.1.2: Method description

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

3.1.3.2.2 KineticReaction Repository Interface

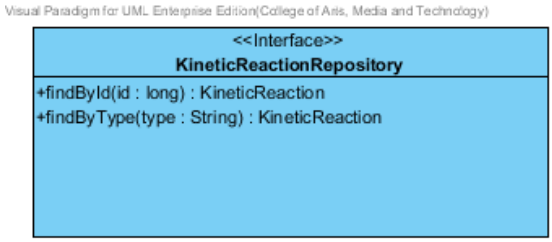


Figure -14: The KineticReaction Repository

KineticReactionRepository Interface is an interface that use for CRUD with entity classes in the system. The most of KineticReaction interface's method is generated from Spring data MongoDB framework. KineticReaction interface consists of 5 methods is shown below this passage.

3.1.3.2.2.1: Attribute description

N/A

3.1.3.2.2.2: Method description

- **Save (kineticReaction: KineticReaction)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new KineticReaction or update existing KineticReaction to the system. The input variable is a KineticReaction object.
- **Delete (kineticReaction: KineticReaction)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the KineticReaction from the system. The input variable is KineticReaction object.
- **findAll ():KineticReaction []** – The findAll is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of KineticReaction data from the system. The result of this method is a list of kineticReaction.
- **findById (id: long): KineticReaction** – The findById method is generated from Spring data MongoDB framework. The developers can make configuration at this method for query the data by themselves. The findById is used when the user wants to retrieve the kineticReaction data from the system. The input variable is id. The findById gets the kineticReaction object by the kineticReaction's Id.
- **findByType (type: String)** – The findByType method is generated from Spring data MongoDB framework. The developers can make configuration at this method for query the data by themselves. The findBytype method is used when the user wants to retrieve the kineticReaction data from the system. The findByType gets the kineticReaction object by the kineticReaction's type.

3.1.3.2.3 KineticReactionService

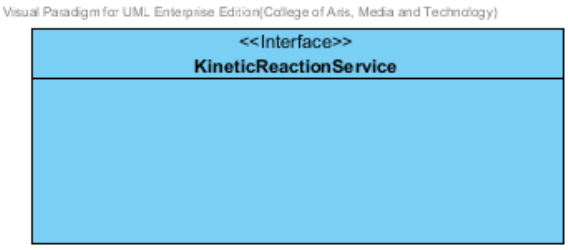


Figure -15: The KineticReaction Service

KineticReactionService is business processing logic for KineticReaction entity. KineticReactionService manages the kineticReaction data through the KineticReactionRepository interface. KineticReactionService consists of 6 methods follow the list below this passage.

3.1.3.2.3.1: Attribute description

N/A

3.1.3.2.3.2: Method description

- **add(kineticReaction:KineticReaction)** – The adding kineticReaction method is used, when the user wants to add a new kineticReaction to the database. This method adds a new kineticReaction by input variable of kineticReaction object. If the kineticReaction object that input by the user is not contained in the database, this method will add a new kineticReaction to the database and return the kineticReaction object from the database to the user after the adding kineticReaction is successful. On the other hand, when the kineticReaction object that input by the user is contained in the database. This method will return a null value to the user.
- **update (kineticReaction: KineticReaction)** - The updating kineticReaction method is used, when the user wants to update an existing kineticReaction in the database. This method update the existing kineticReaction by input variable of kineticReaction object. If the kineticReaction object that input by the user is contained in the database, this method will update an existing kineticReaction in the database and return the kineticReaction object from the database to the user after the updating kineticReaction is successful. On the other hand, when the kineticReaction object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (kineticReaction: KineticReaction)** – The deleting kineticReaction method is used when the user wants to deletes the existing kineticReaction from the database. This method delete the kineticReaction by input variable of kineticReaction object. If the kineticReaction object that input by the user is contained in the database, this method will delete an existing kineticReaction from the database and

return the kineticReaction object to the user after the deleting kineticReaction is successful. On the other hand, when the kineticReaction object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll():KineticReaction []** - The findAll method is used when the user wants to get all data of kineticReaction in the database. This method is return as a list of kineticReaction from the database.
- **findById (id: long): KineticReaction** – The findById method is used when the user wants to get the kineticReaction data in the database. This method gets kineticReaction 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.
- **findByType(type: String) : KineticReaction** –The findByType method is used when the user wants to get kineticReaction data in the system. This method gets kineticReaction object from the database by type that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.3.2.4 KineticReactionServiceImpl

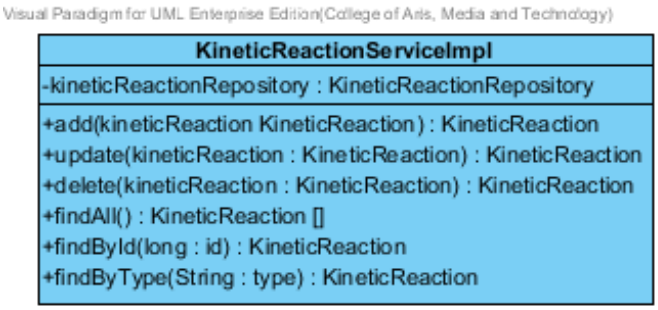


Figure -16: The KineticReactionServiceImpl

KineticReactionServiceImpl is the service class that implements the method from KineticReactionService. So, the method of KineticReactionServiceImpl is same as KineticReactionService.

3.1.3.2.4.1: Attribute description

- **KineticReactionRepository** – the repository of KineticReaction.

This attribute is used for KineticReaction data management.

3.1.3.2.4.2: Method description

Same as KineticReactionServiceImpl

3.1.3.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.4- CD-04: Pka Class diagram

3.1.4.1: Class diagram

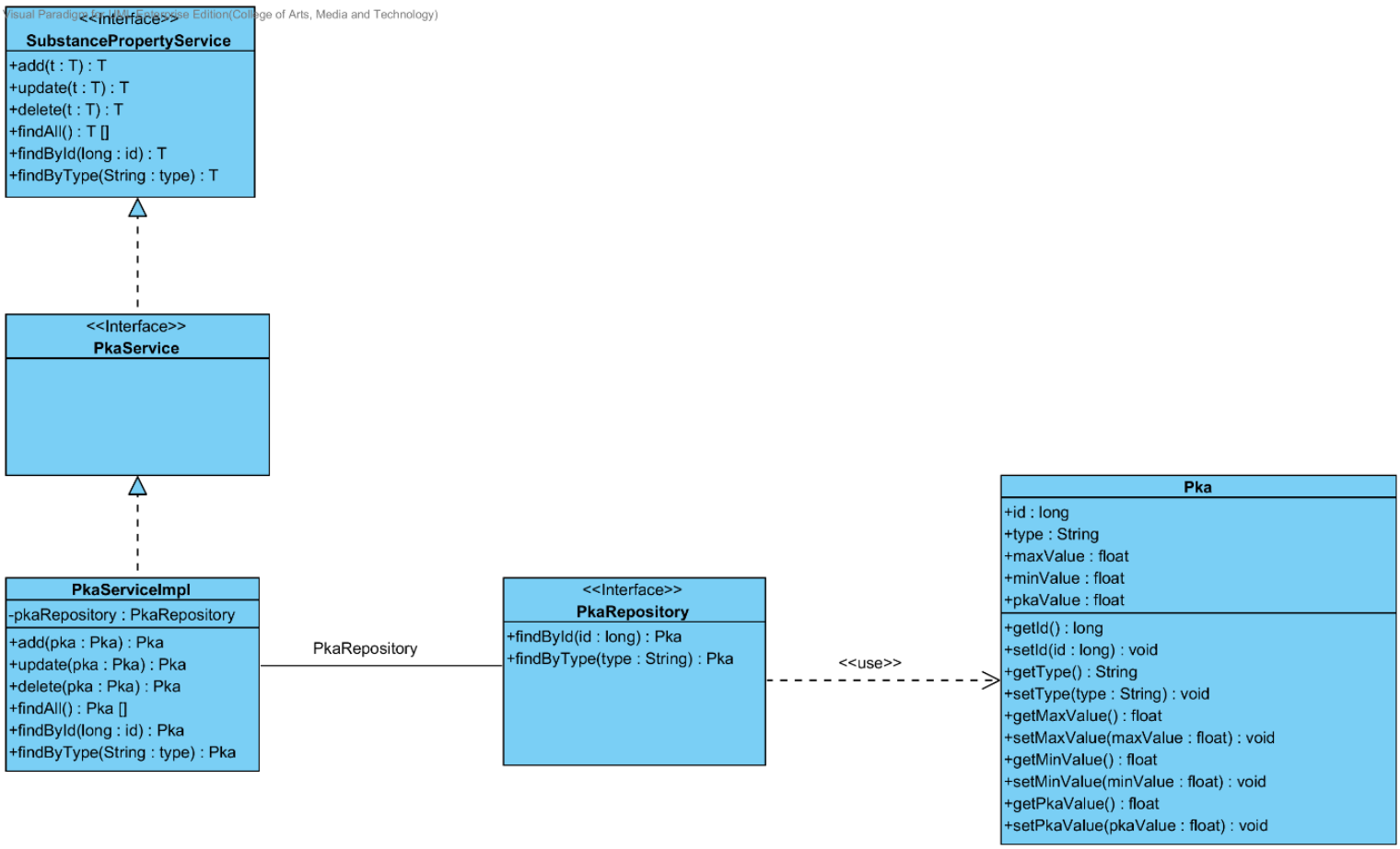


Figure 17 - CD-04: Pka Class diagram

3.1.4.2: Class description

From the figure 17, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.4.2.1 Pka class

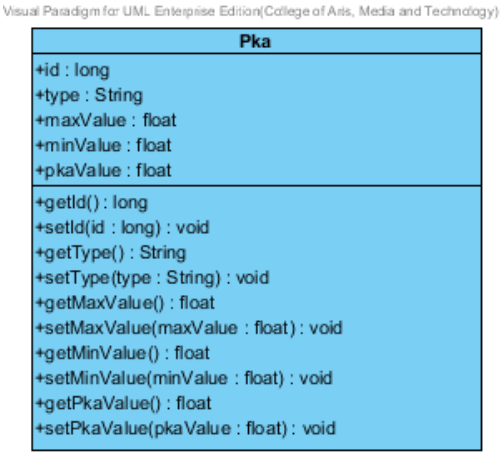


Figure 18 – The pka class

Pka class is a part of drug's substance. Pka class is an entity class that will be saved to the system. The pka class consists of 5 attributes follow the list below this passage.

3.1.4.2.1.1: Attribute description

- **Id** – the identity of the pka class. Id attribute is a long number.
- **Type** – the type of the pka class. In the pharmacy, the pka type shows a pka in each substance property. The pka type value is string.
- **Max Value** - The max value of each type of pka. The pharmacists can use the max value and min value for pka type identification.
- **Min Value** – The min value of each type of pka. The pharmacists can use the max value and min value for pka type identification.
- **Pka value** - The pka value is a value of each type of the pka. The pka value is stable value. The value is float number.

3.1.4.2.1.2: Method description

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

3.1.4.2.2 PkaRepository Interface

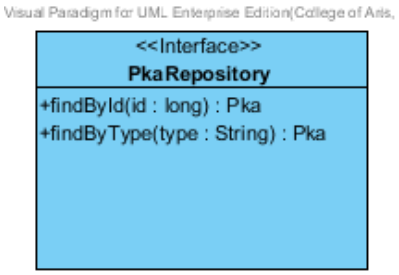


Figure 19 – The PkaRepository interface

PkaRepository Interface is an interface that use for CRUD with entity classes in the system. The most of PkaRepository interface's method is generated from Spring data MongoDB framework. PkaRepository interface consists of 5 methods is shown below this passage.

3.1.4.2.2.1: Attribute description

N/A

3.1.4.2.2.2: Method description

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

3.1.4.2.3 PkaService

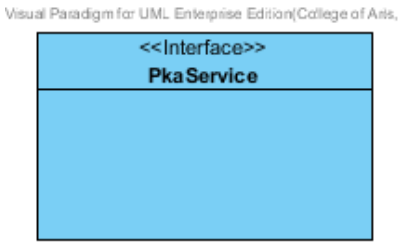


Figure 20 – The pka service Interface.

PkaService is business processing logic for the pka entity. PkaService manages the pka data through the PkaRepository interface. PkaService consists of 6 methods follow the list below this passage.

3.1.4.2.3.1: Attribute description

N/A

3.1.4.2.3.2: Method description

- **add (pka: Pka)** – The adding pka method is used, when the user wants to add a new pka to the database. This method adds a new pka by input variable of pka object. If the pka object that input by the user is not contained in the database, this method will add a new pka to the database and return the pka object from the database to the user after the adding pka is successful. On the other hand, when the pka object that input by the user is contained in the database. This method will return a null value to the user.
- **update (pka: Pka)** - The updating pka method is used, when the user wants to update an existing pka in the database. This method update the existing pka by input variable of pka object. If the pka object that input by the user is contained in the database, this method will update an existing pka in the database and return the pka object from the database to the user after the updating pka is successful. On the other hand, when the pka object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (pka: Pka)** – The deleting pka method is used when the user wants to deletes the existing pka from the database. This method delete the pka by input variable of pka object. If the pka object that input by the user is contained in the database, this method will delete an existing pka from the database and return the pka object to the user after the deleting pka is successful. On the other hand, when the pka object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll() : Pka[]** – The findAll method is used, when the user wants to get all pka data in the system. This method is return as a list of pka object from the database.
- **findById(id : long) : Pka** – The findById method is used, when the user wants to get the pka object from the system. . This method gets pka 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.
- **findByType(type : String)** –The findByType method is used when the user wants to get pka object from the system. . This method gets pka object from the database by type that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.4.2.4 PkaServiceImpl

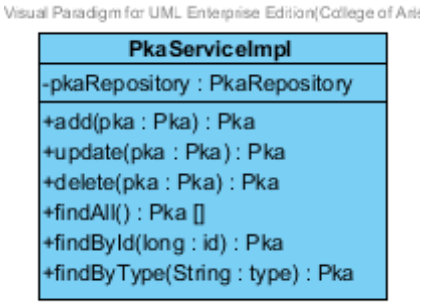


Figure 21 – The PkaServiceImpl class

PkaServiceImpl is the pka service class that implements the method from PkaService. So, the method of PkaServiceImpl is same as PkaService.

3.1.4.2.4.1: Attribute description

- **PkaRepository** – the repository of Pka. This attribute is used for Pka data management.

3.1.4.2.4.2: Method description

Same as PkaService

3.1.4.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.5- CD-05: PartitionCoefficient Class diagram

3.1.5.1: Class diagram

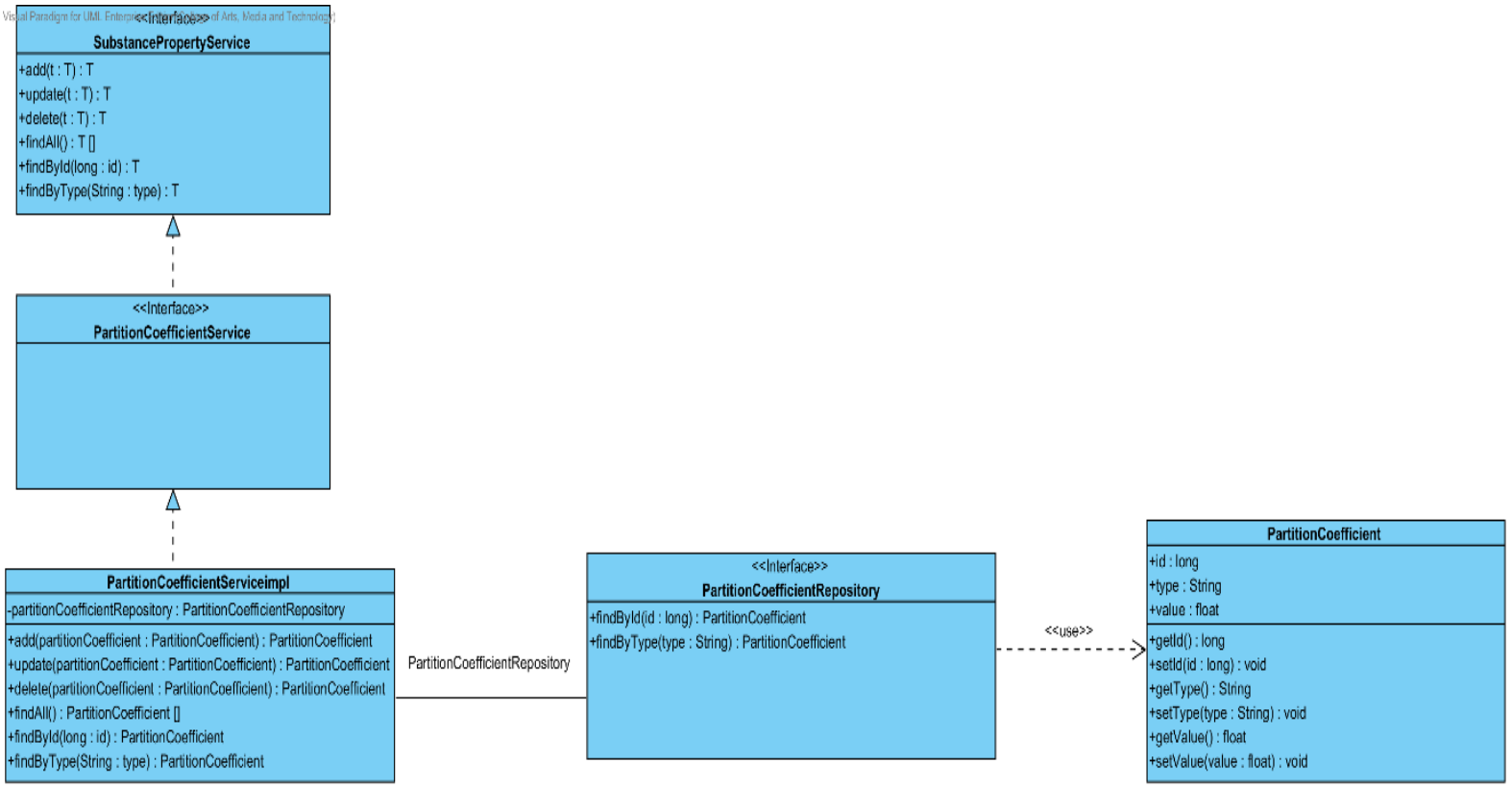


Figure 20 - CD-05: PartitionCoefficient Class diagram

3.1.5.2: Class description

From the figure 20, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.5.2.1 PartitionCoefficient class

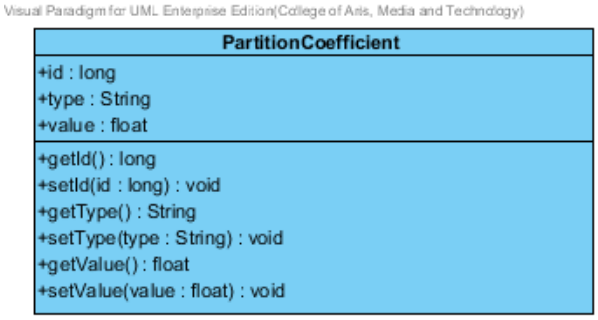


Figure 21 – The PartitionCoefficient class

PartitionCoefficient class is a part of drug's substance. PartitionCoefficient class is an entity class that will be saved to the system. The PartitionCoefficient class consists of 3 attributes follow the list below this passage.

3.1.5.2.1.1: Attribute description

- **Id** – the identity of the PartitionCoefficient class. Id attribute is a long number.
- **Type** – the type of the PartitionCoefficient class. In the pharmacy, the Partition Coefficient type shows a type of partition coefficient in each substance. The PartitionCoefficient type value is string.
- **PartitionCoefficient value** - The PartitionCoefficient value is a value of each type of the Partition Coefficient. The value is float number.

3.1.5.2.1.2: Method description

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

3.1.5.2.2 PartitionCoefficientRepository Interface

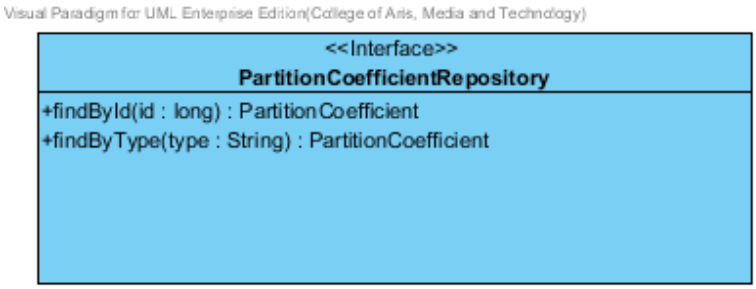


Figure 22 – The PartitionCoefficientRepository interface

PartitionCoefficient Repository Interface is an interface that use for CRUD with entity classes in the system. The most of PartitionCoefficientRepository interface's method is generated from Spring data MongoDB framework. PartitionCoefficient Repository interface consists of 5 methods is shown below this passage.

3.1.5.2.2.1: Attribute description

N/A

3.1.5.2.2.2: Method description

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

3.1.5.2.3 PartitionCoefficientService

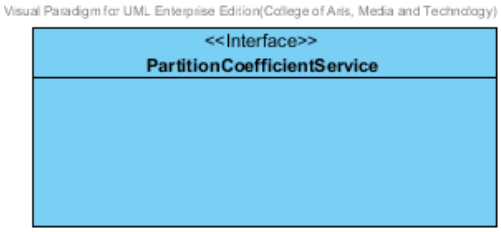


Figure 23 – The PartitionCoefficient service Interface.

PartitionCoefficientService is business processing logic for PartitionCoefficient entity. PartitionCoefficientService manages the partitionCoefficient data through the PartitionCoefficient Repository interface. PartitionCoefficientService consists of 6 methods follow the list below this passage.

3.1.5.2.3.1: Attribute description

N/A

3.1.5.2.3.2: Method description

- **add (partitionCoefficient: PartitionCoefficient)** – The adding partitionCoefficient method is used, when the user wants to add a new partitionCoefficient to the database. This method adds a new partitionCoefficient by input variable of partitionCoefficient object. If the partitionCoefficient object that input by the user is not contained in the database, this method will add a new partitionCoefficient to the database and return the partitionCoefficient object from the database to the user after the adding partitionCoefficient is successful. On the other hand, when the partitionCoefficient object that input by the user is contained in the database. This method will return a null value to the user.
- **update (partitionCoefficient: PartitionCoefficient)** - The updating partitionCoefficient method is used, when the user wants to update an existing partitionCoefficient in the database. This method update the existing partitionCoefficient by input variable of partitionCoefficient object. If the partitionCoefficient object that input by the user is contained in the database, this method will update an existing partitionCoefficient in the database and return the partitionCoefficient object from the database to the user after the updating partitionCoefficient is successful. On the other hand, when the partitionCoefficient object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (partitionCoefficient: PartitionCoefficient)** – The deleting partitionCoefficient method is used when the user wants to deletes the existing partitionCoefficient from the database. This method delete the partitionCoefficient by input variable of partitionCoefficient object. If the partitionCoefficient object that input by the user is contained in the database, this method will delete an existing partitionCoefficient from the

database and return the partitionCoefficient object to the user after the deleting partitionCoefficient is successful. On the other hand, when the partitionCoefficient object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll() : PartitionCoefficient []** – The findAll method is used, when the user wants to get all PartitionCoefficient data in the system. This method is return as a list of PartitionCoefficient object.
- **findById(id : long) : PartitionCoefficient** – The findById method is used, when the user wants to get the PartitionCoefficient object from the system. This method gets partitionCoefficient 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.
- **findByType(type : String) : PartitionCoefficient** –The findByType method is used when the user wants to get PartitionCoefficient object from the system. This method gets partitionCoefficient object from the database by id that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.5.2.4 PartitionCoefficientServiceImpl

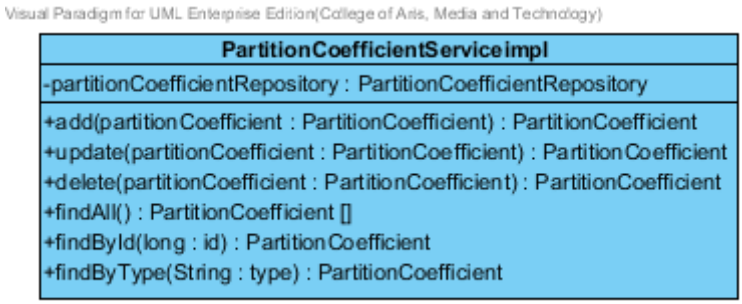


Figure 24 – The PartitionCoefficientImpl class

PartitionCoefficientImpl is the PartitionCoefficient service class that implements the method from PartitionCoefficientService. So, the method of PartitionCoefficientServiceImpl is same as PartitionCoefficientService.

3.1.4.2.4.1: Attribute description

- **PartitionCoefficientRepository** – the repository of PartitionCoefficient. This attribute is used for PartitionCoefficient data management.

3.1.4.2.4.2: Method description

Same as PartitionCoefficientService

3.1.4.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.6- CD-06: Solidstate Class Diagram

3.1.6.1: Class diagram

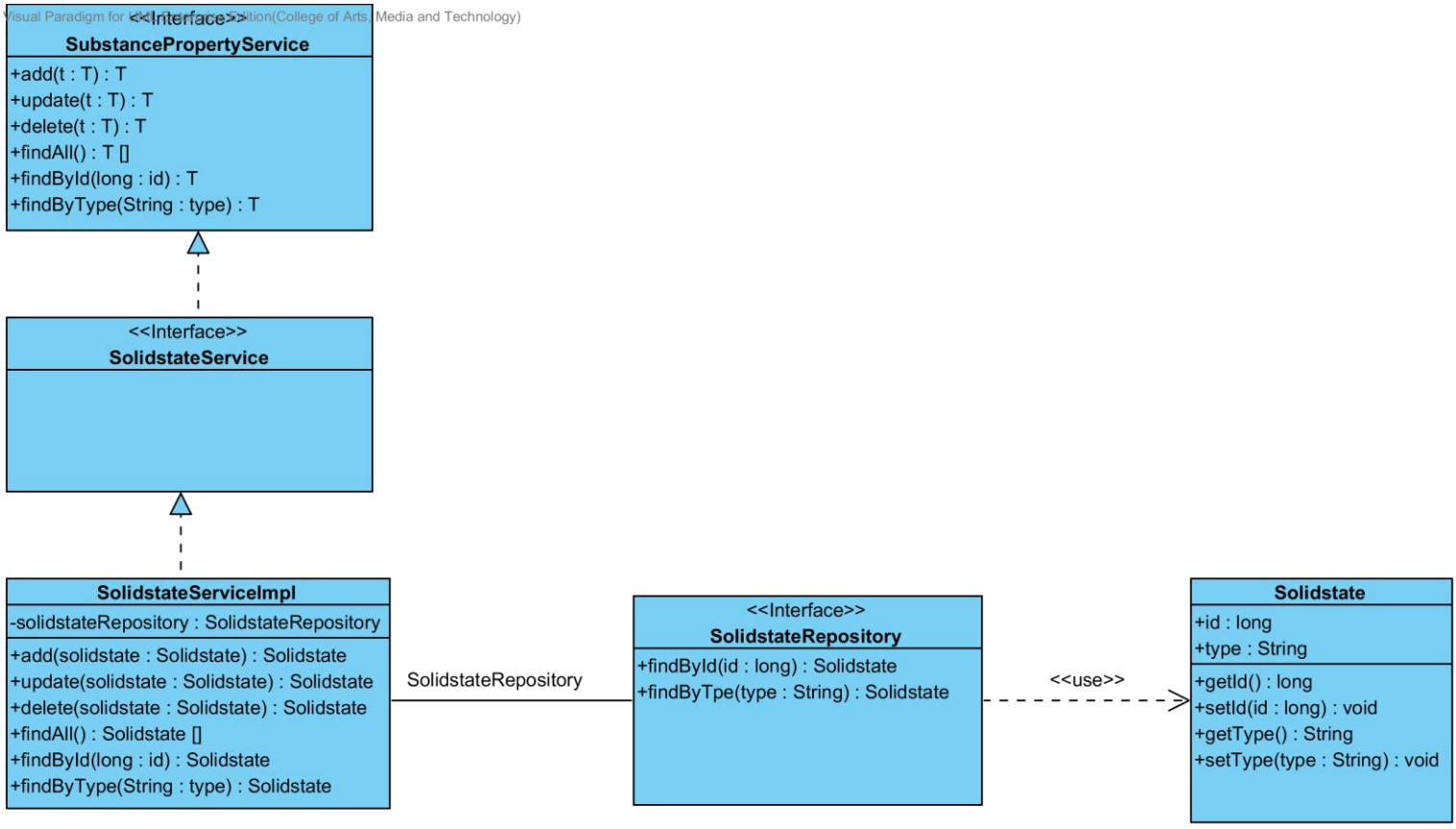


Figure 25 - CD-06: Solidstate Class Diagram

3.1.6.2: Class description

From the figure 25, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.6.2.1 Solidstate class

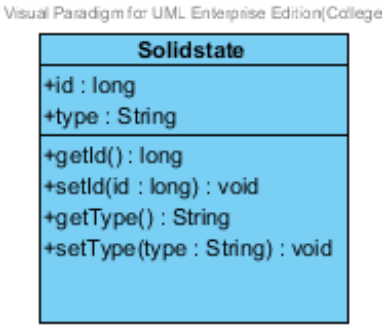


Figure 26 – The Solidstate class

Solidstate class is a part of drug's substance. Solidstate class is an entity class that will be saved to the system. The Solidstate class consists of 2 attributes follow the list below this passage.

3.1.6.2.1.1: Attribute description

- **Id** – The identity of the Solidstate class. Id attribute is a long number.
- **Type** – The type of the Solidstate class. In the pharmacy, the Solidstate type shows the level of solid in each Solidstate. The Solidstate type attribute is a string.

3.1.6.2.1.2: Method description

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

3.1.6.2.2 Solidstate Repository Interface

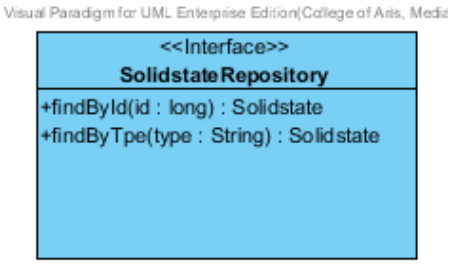


Figure -27: The Solidstate Repository

SolidstateRepository Interface is an interface that use for CRUD with entity classes in the system. The most of Solidstate interface's method is generated from Spring data MongoDB framework. Solidstate interface consists of 5 methods is shown below this passage.

3.1.6.2.2.1: Attribute description

N/A

3.1.6.2.2.2: Method description

- **Save (solidstate: Solidstate)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new Solidstate or update existing Solidstate to the system. The input variable is a Solidstate object.
- **Delete (solidstate: Solidstate)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the Solidstate from the system. The input variable is Solidstate object.
- **findAll (): Solidstate []** – The findAll is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of Solidstate data from the system. The result of this method is a list of Solidstate.
- **findById (id: long): Solidstate** – The findById method is generated from Spring data MongoDB framework. The developers can make configuration at this method for query the data by themselves. The findById is used when the user wants to retrieve the Solidstate data from the system. The findById gets the Solidstate object by the Solidstate's Id.
- **findByType (type: String)** – The findByType method is generated from Spring data MongoDB framework. The developers can make configuration at this method for query the data by themselves. The findBytype method is used when the user wants to retrieve the Solidstate data from the system. The findByType gets the Solidstate object by the Solidstate's type.

3.1.6.2.3 SolidstateService

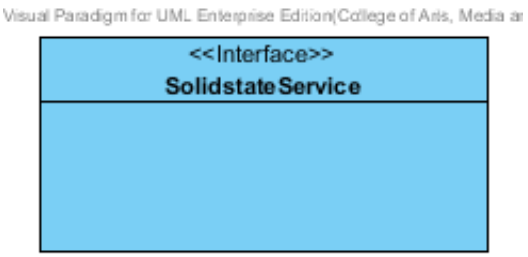


Figure -28: The Solidstate Service

SolidstateService is business processing logic for Solidstate entity. SolidstateService manages the Solidstate data through the SolidstateRepository interface. SolidstateService consists of 6 methods follow the list below this passage.

3.1.6.2.3.1: Attribute description

N/A

3.1.6.2.3.2: Method description

- **add (solidstate: Solidstate)** – The adding solidstate method is used, when the user wants to add a new solidstate to the database. This method adds a new solidstate by input variable of solidstate object. If the solidstate object that input by the user is not contained in the database, this method will add a new solidstate to the database and return the solidstate object from the database to the user after the adding solidstate is successful. On the other hand, when the solidstate object that input by the user is contained in the database. This method will return a null value to the user.
- **update (solidstate: Solidstate)** - The updating solidstate method is used, when the user wants to update an existing solidstate in the database. This method update the existing solidstate by input variable of solidstate object. If the solidstate object that input by the user is contained in the database, this method will update an existing solidstate in the database and return the solidstate object from the database to the user after the updating solidstate is successful. On the other hand, when the solidstate object that input by the user is not contained in the database. This method will return a null value to the user.

- **delete (solidstate: Solidstate)** – The deleting solidstate method is used when the user wants to deletes the existing solidstate from the database. This method delete the solidstate by input variable of solidstate object. If the solidstate object that input by the user is contained in the database, this method will delete an existing solidstate from the database and return the solidstate object to the user after the deleting solidstate is

successful. On the other hand, when the solidstate object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll():Solidstate []** – The findAll method is used when the user wants to get all data of Solidstate in the database. This method is return as a list of database.
- **findById (id: long): Solidstate** – The findById method is used when the user wants to get the Solidstate data in the system. This method gets solidstate 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.
- **findByType(type: String) : Solidstate** –The findByType method is used when the user wants to get Solidstate data in the system. This method gets solidstate object from the database by id that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.6.2.4 SolidstateServiceImpl

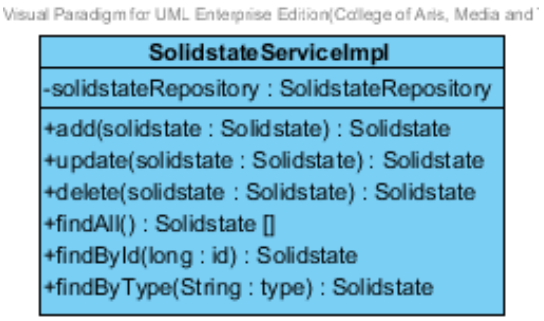


Figure -29: The SolidstateServiceImpl

SolidstateServiceImpl is the service class that implements the method from SolidstateService. So, the method of SolidstateServiceImpl is same as SolidstateService.

3.1.6.2.4.1: Attribute description

- **SolidstateRepository** – the repository of Solidstate. This attribute is used for Solidstate data management.

3.1.6.2.4.2: Method description

Same as KineticReactionServiceImpl

3.1.6.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.7- CD-07: Hygroscopicity Class Diagram

3.1.7.1: Class diagram

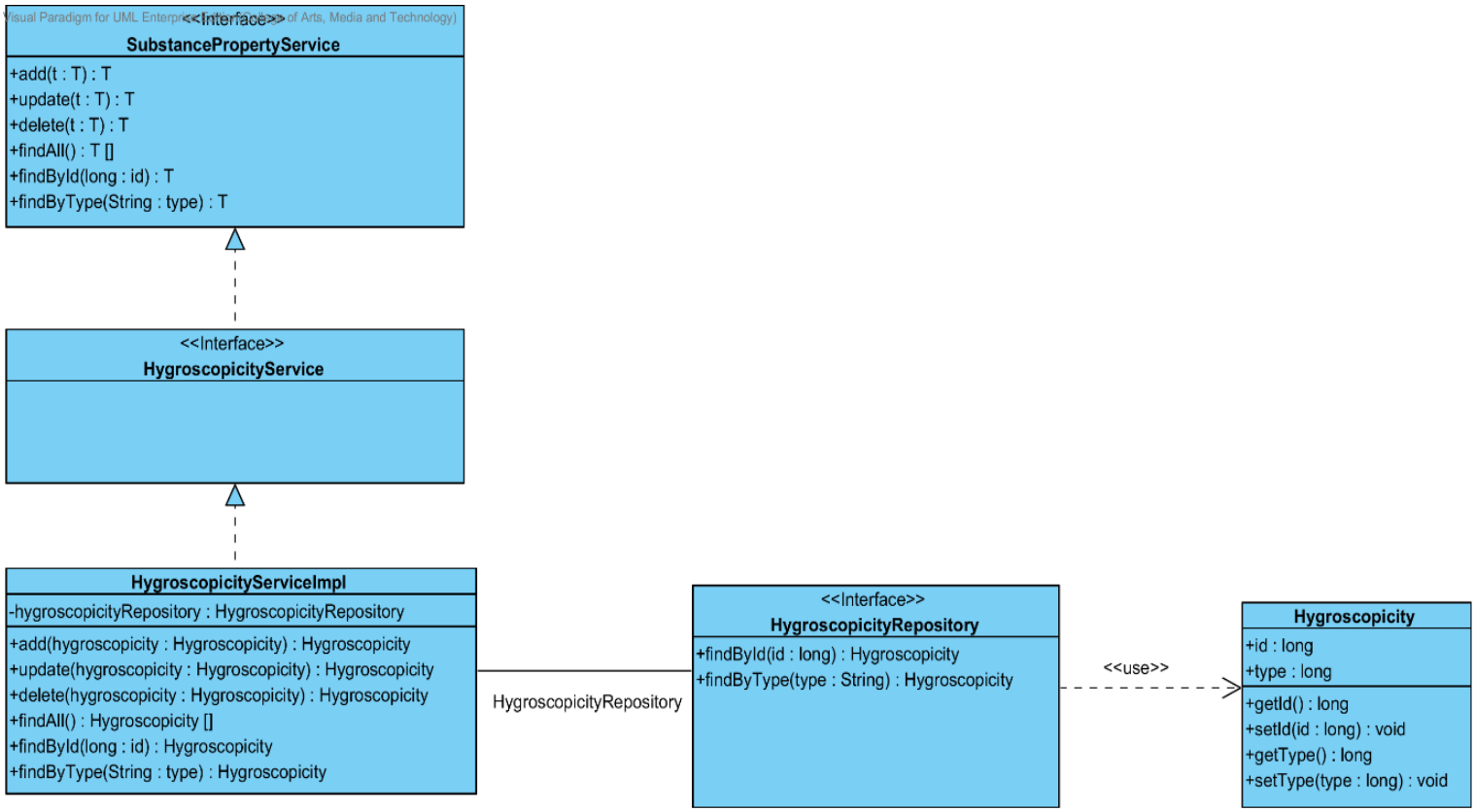


Figure 30 - CD-07: Hygroscopicity Class Diagram

3.1.7.2: Class description

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

3.1.7.2.1 Hygroscopicity class

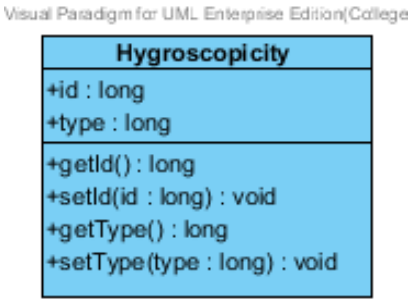


Figure 31 – The Hygroscopicity class

Hygroscopicity class is a part of drug's substance. Hygroscopicity class is an entity class that will be saved to the system. The Hygroscopicity class consists of 2 attributes follow the list below this passage.

3.1.7.2.1.1: Attribute description

- **Id** – The identity of the Hygroscopicity class. Id attribute is a long number.
- **Type** – The type of the Hygroscopicity class. The Hygroscopicity type attribute is a string.

3.1.7.2.1.2: Method description

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

3.1.7.2.2 HygroscopicityRepository Interface

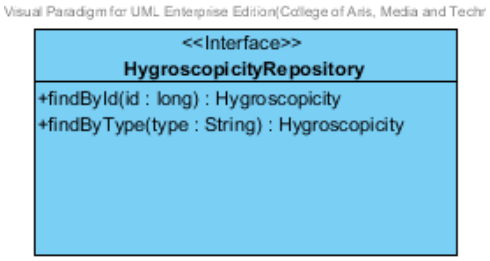


Figure -32: The Hygroscopicity Repository

HygroscopicityRepository Interface is an interface that use for CRUD with entity classes in the system. The most of Hygroscopicity interface's method is generated from Spring data MongoDB framework. Hygroscopicity interface consists of 5 methods is shown below this passage.

3.1.7.2.2.1: Attribute description

N/A

3.1.7.2.2.2: Method description

- **Save (hygroscopicity: Hygroscopicity)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new Hygroscopicity or update existing Hygroscopicity to the system. The input variable is a Hygroscopicity object.
- **Delete (hygroscopicity: Hygroscopicity)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the Hygroscopicity from the system. The input variable is Hygroscopicity object.
- **findAll ():Hygroscopicity []** – The findAll is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of Hygroscopicity data from the system. The result of this method is a list of Hygroscopicity.
- **findById (id: long): Hygroscopicity** – The findById method is generated from Spring data MongoDB framework. The developers can make configuration at this method for query the data by themselves. The findById is used when the user wants to retrieve the Hygroscopicity data from the system. The findById gets the Hygroscopicity object by the Hygroscopicity's Id.
- **findByType (type: String) : Hygroscopicity** – The findByType method is generated from Spring data MongoDB framework. The developers can make configuration at this method for query the data by themselves. The findBytype method is used when the user wants to retrieve the Hygroscopicity data from the system. The findByType gets the Hygroscopicity object by the Hygroscopicity's type.

3.1.7.2.3 HygroscopicityService

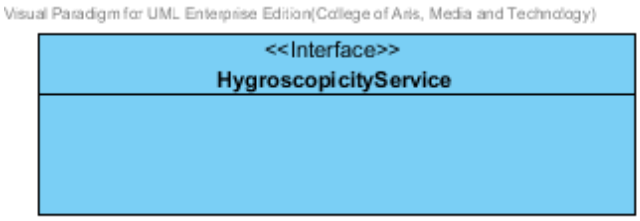


Figure -33: The Hygroscopicity Service

HygroscopicityService is business processing logic for Hygroscopicity entity. HygroscopicityService manages the Hygroscopicity data through the HygroscopicityRepository interface. HygroscopicityService consists of 6 methods follow the list below this passage.

3.1.7.2.3.1: Attribute description

N/A

3.1.7.2.3.2: Method description

- **add (hygroscopicity: Hygroscopicity)** – The adding hygroscopicity method is used, when the user wants to add a new hygroscopicity to the database. This method adds a new hygroscopicity by input variable of hygroscopicity object. If the hygroscopicity object that input by the user is not contained in the database, this method will add a new hygroscopicity to the database and return the hygroscopicity object from the database to the user after the adding hygroscopicity is successful. On the other hand, when the hygroscopicity object that input by the user is contained in the database. This method will return a null value to the user.
- **update (hygroscopicity: Hygroscopicity)** - The updating hygroscopicity method is used, when the user wants to update an existing hygroscopicity in the database. This method update the existing hygroscopicity by input variable of hygroscopicity object. If the hygroscopicity object that input by the user is contained in the database, this method will update an existing hygroscopicity in the database and return the hygroscopicity object from the database to the user after the updating hygroscopicity is successful. On the other hand, when the hygroscopicity object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (hygroscopicity: Hygroscopicity)** – The deleting hygroscopicity method is used when the user wants to deletes the existing hygroscopicity from the database. This method delete the hygroscopicity by input variable of hygroscopicity object. If the hygroscopicity object that input by the user is contained in the database, this method will delete an existing hygroscopicity from the database and return the hygroscopicity object to the user after the deleting hygroscopicity is successful. On the other hand, when the hygroscopicity object that input by the user is not

contained in the database. This method will return a null value to the user.

- **findAll():Hygroscopicity []** – The findAll method is used when the user wants to get all data of Hygroscopicity in the database. This method is return as a list of Hygroscopicity from the database.
- **findById (id: long): Hygroscopicity** – The findById method is used when the user wants to get the Hygroscopicity data in the system. This method gets hygroscopicity 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.
- **findByType(type: String) : Hygroscopicity** –The findByType method is used when the user wants to get Hygroscopicity data in the system This method gets hygroscopicity object from the database by id that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.7.2.4 HygroscopicityServiceImpl

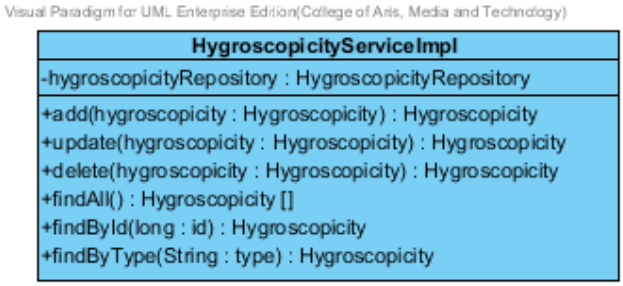


Figure -34: The HygroscopicityServiceImpl

HygroscopicityServiceImpl is the service class that implements the method from HygroscopicityService. So, the method of HygroscopicityServiceImpl is same as HygroscopicityService.

3.1.6.2.4.1: Attribute description

- **HygroscopicityRepository** – the repository of Hygroscopicity. This attribute is used for Hygroscopicity data management.

3.1.6.2.4.2: Method description

Same as HygroscopicityServiceImpl

3.1.6.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.8- CD-08: ParticleSize Class diagram

3.1.8.1: Class diagram

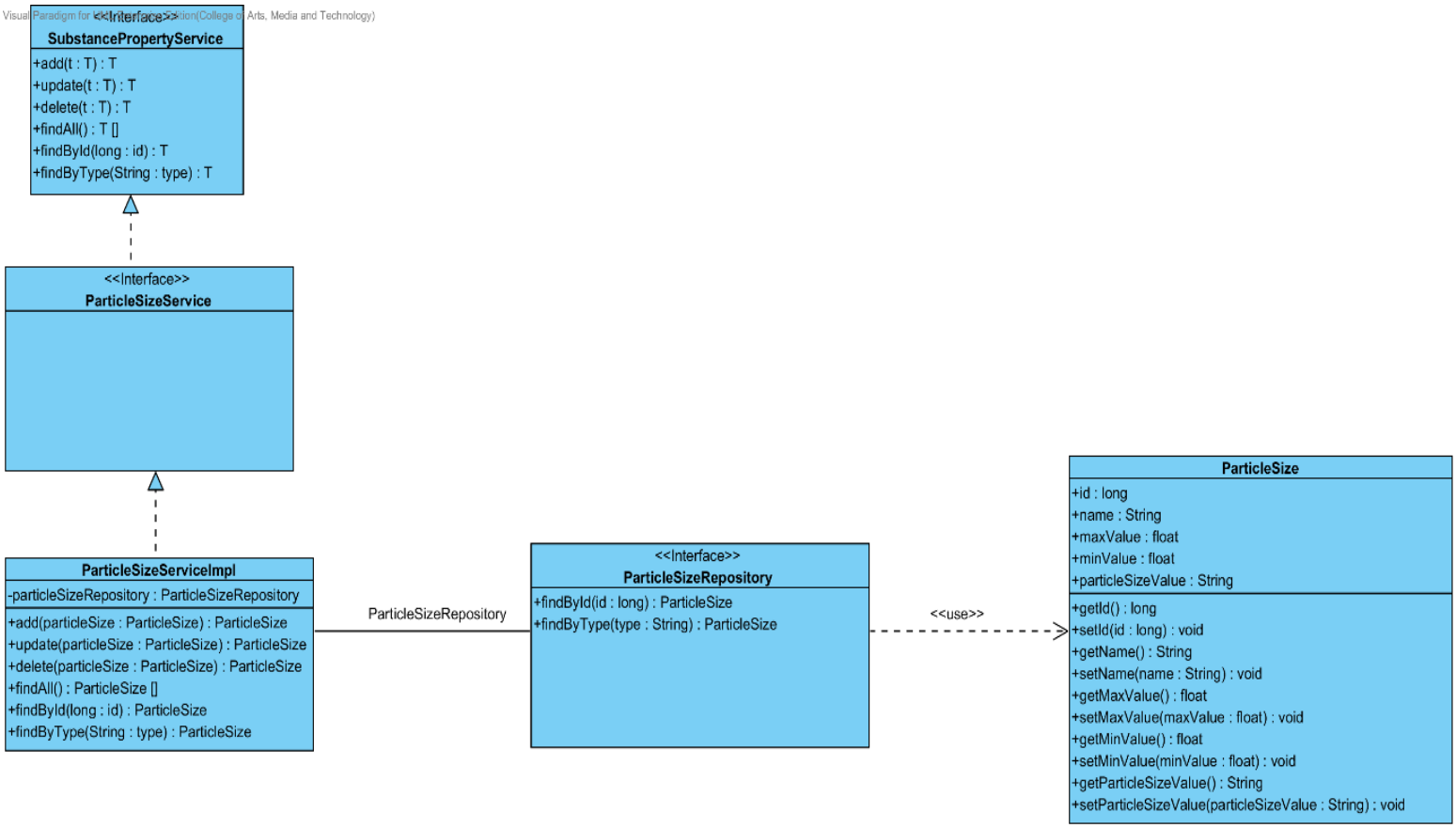


Figure 35 - CD-08: ParticleSize Class diagram

3.1.8.2: Class description

From the figure 35, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.8.2.1 ParticleSize class

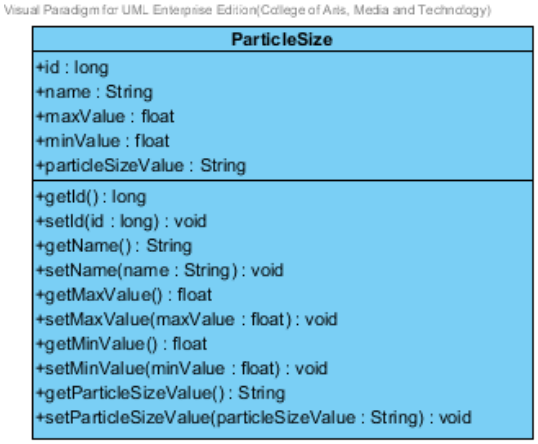


Figure 36 – The ParticleSize class

ParticleSize class is a part of drug's substance. ParticleSize class is an entity class that will be saved to the system. The ParticleSize class consists of 5 attributes follow the list below this passage.

3.1.8.2.1.1: Attribute description

- **Id** – the identity of the ParticleSize class. Id attribute is a long number.
- **Type** – the type of the ParticleSize class. The ParticleSize type value is string.
- **Max Value** - The max value of each type of ParticleSize. The pharmacists can use the max value and min value for ParticleSize type identification.
- **Min Value** – The min value of each type of ParticleSize. The pharmacists can use the max value and min value for ParticleSize type identification.
- **ParticleSize value** - The ParticleSize value is a value of each type of the ParticleSize. The ParticleSize value is stable value. The value is float number.

3.1.8.2.1.2: Method description

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

3.1.8.2.2 ParticleSizeRepository Interface

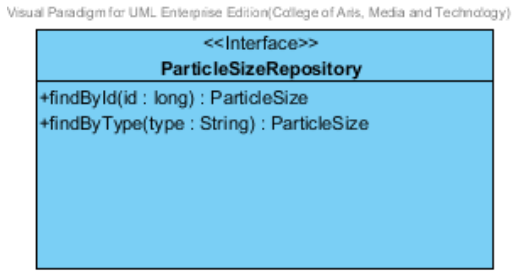


Figure 37 – The ParticleSizeRepository interface

ParticleSizeRepository Interface is an interface that use for CRUD with entity classes in the system. The most of ParticleSizeRepository interface's method is generated from Spring data MongoDB framework. ParticleSizeRepository interface consists of 5 methods is shown below this passage.

3.1.8.2.2.1: Attribute description

N/A

3.1.8.2.2.2: Method description

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

3.1.8.2.3 ParticleSize Service

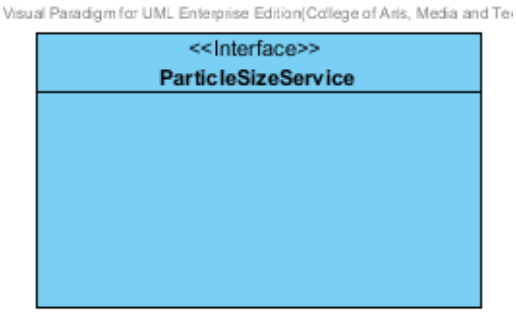


Figure 38 – The ParticleSizeService Interface.

ParticleSizeService is business processing logic for the particlesize entity. ParticleSizeService manages the ParticleSize data through the ParticleSizeRepository interface. ParticleSizeService consists of 6 methods follow the list below this passage.

3.1.8.2.3.1: Attribute description

N/A

3.1.8.2.3.2: Method description

- **add (particleSize: ParticleSize)** – The adding ParticleSize method is used, when the user wants to add a new ParticleSize to the database. This method adds a new ParticleSize by input variable of ParticleSize object. If the ParticleSize object that input by the user is not contained in the database, this method will add a new ParticleSize to the database and return the ParticleSize object from the database to the user after the adding ParticleSize is successful. On the other hand, when the ParticleSize object that input by the user is contained in the database. This method will return a null value to the user.
- **update (particleSize: ParticleSize)** - The updating ParticleSize method is used, when the user wants to update an existing ParticleSize in the database. This method update the existing ParticleSize by input variable of ParticleSize object. If the ParticleSize object that input by the user is contained in the database, this method will update an existing ParticleSize in the database and return the ParticleSize object from the database to the user after the updating ParticleSize is successful. On the other hand, when the ParticleSize object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (particleSize: ParticleSize)** – The deleting ParticleSize method is used when the user wants to deletes the existing ParticleSize from the database. This method delete the ParticleSize by input variable of ParticleSize object. If the ParticleSize object that input by the user is contained in the database, this method will delete an existing ParticleSize from the database and return the ParticleSize object to the user after the deleting ParticleSize is successful. On the other hand, when the

ParticleSize object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll() : ParticleSize []** – The findAll method is used, when the user wants to get all particleSize data in the database. This method is return as a list of particleSize object in the database.
- **findById(id : long) : ParticleSize** – The findById method is used, when the user wants to get the particleSize object from the system. This method gets ParticleSize 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.
- **findByType(type : String)** –The findByType method is used when the user wants to get the particleSize object from the system. This method gets ParticleSize object from the database by id that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.8.2.4 ParticleSizeServiceImpl

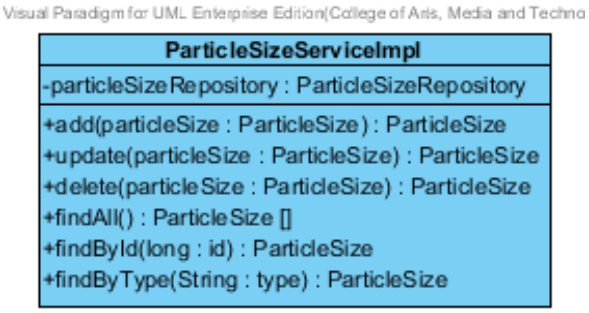


Figure 39 – The ParticleSizeServiceImpl class

ParticleSizeServiceImpl is the ParticleSize service class that implements the method from ParticleSizeService. So, the method of ParticleSizeServiceImpl is same as ParticleSizeService.

3.1.8.2.4.1: Attribute description

- **ParticleSizeRepository** – the repository of ParticleSize. This attribute is used for ParticleSize data management.

3.1.8.2.4.2: Method description

Same as ParticleSize Service

3.1.8.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.9- CD-09: Flowability Class diagram

3.1.9.1: Class diagram

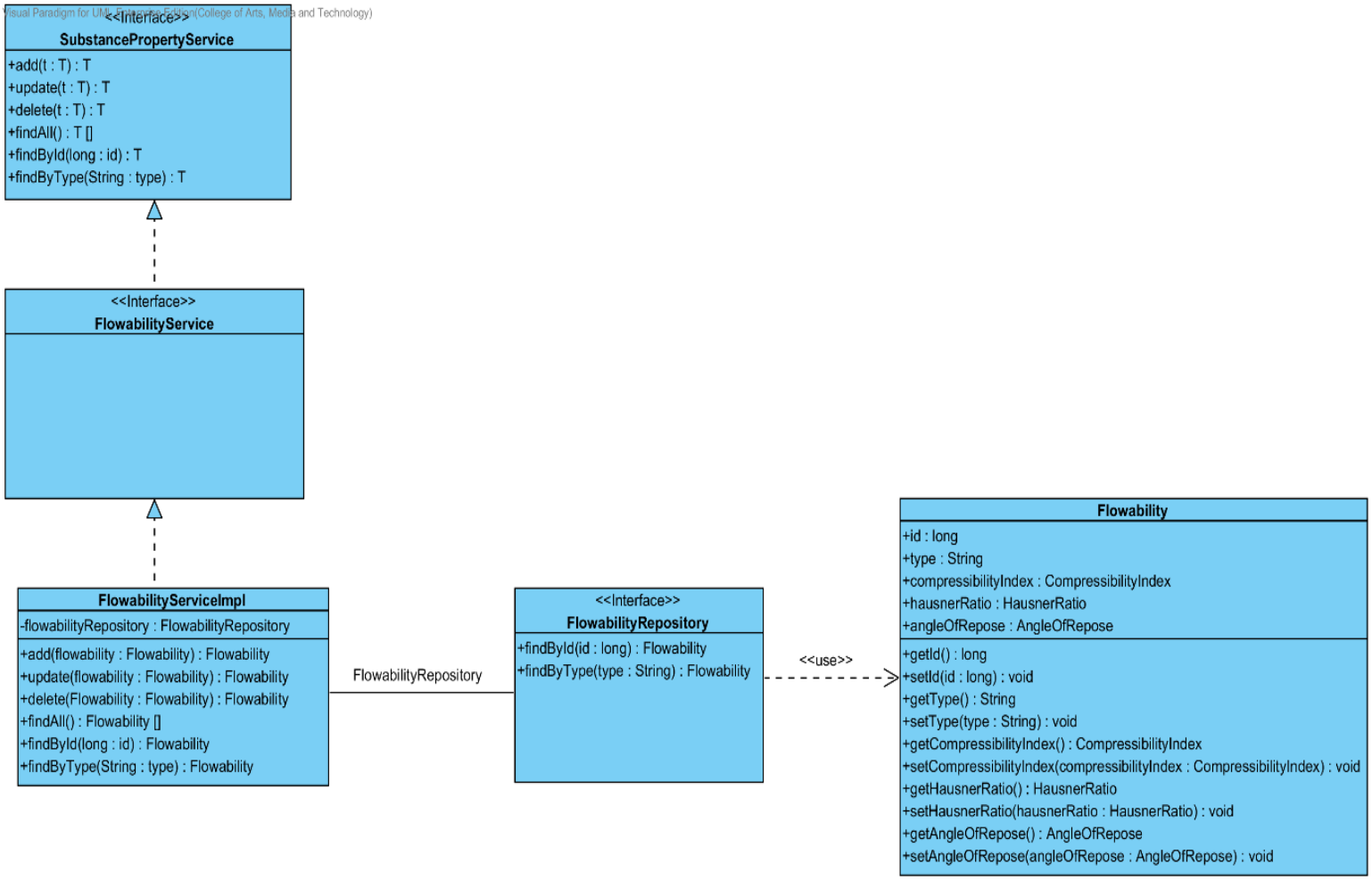


Figure 40 - CD-09: Flowability Class diagram

3.1.9.2: Class description

From the figure 40, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.9.2.1 Flowability class

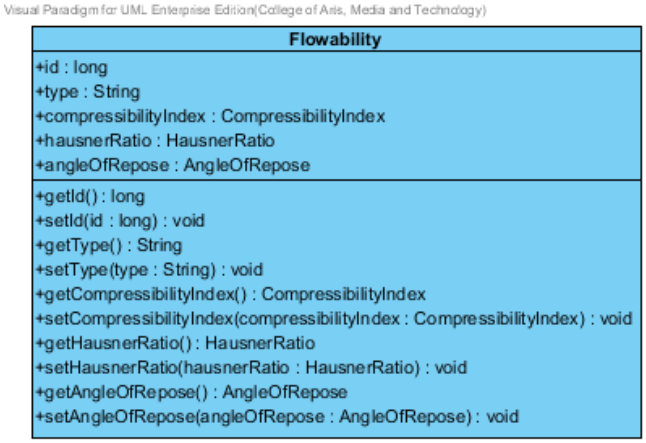


Figure 41 – The Flowability class

Flowability class is a part of drug's substance. Flowability class is an entity class that will be saved to the system. The Flowability class consists of 5 attributes follow the list below this passage.

3.1.9.2.1.1: Attribute description

- **Id** – the identity of the Flowability class. Id attribute is a long number.
- **Type** – the type of the Flowability class. The Flowability type value is string.
- **CompressibilityIndex** – The compressibilityIndex of the flowability. The compressibilityIndex attribute is a CompressibilityIndex object.
- **HausnerRatio** – The hausnerRatio of the flowability. The hausnerRatio attribute is a HausnerRatio object.
- **AngleOfRepose** – The AngleOfRepose of the flowability. The AngleOfRepose attribute is an AngleOfRepose object.

3.1.9.2.1.2: Method description

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

3.1.9.2.2 FlowabilityRepository Interface

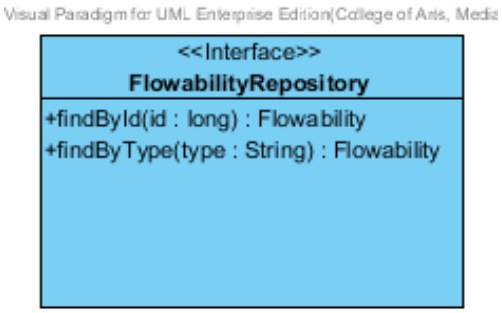


Figure 42 – The FlowabilityRepository interface

FlowabilityRepository Interface is an interface that use for CRUD with entity classes in the system. The most of FlowabilityRepository interface's method is generated from Spring data MongoDB framework. FlowabilityRepository interface consists of 5 methods is shown below this passage.

3.1.9.2.2.1: Attribute description

N/A

3.1.9.2.2.2: Method description

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

3.1.9.2.3 Flowability Service

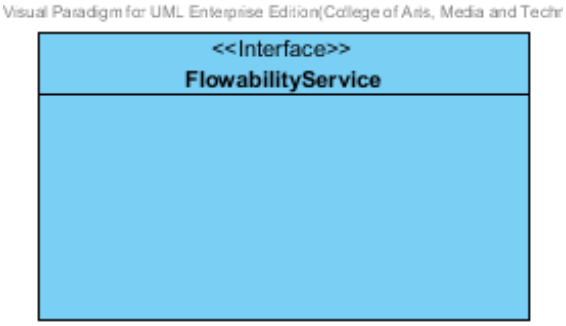


Figure 43 – The FlowabilityService Interface.

FlowabilityService is business processing logic for the flowability entity. FlowabilityService manages the Flowability data through the FlowabilityRepository interface. FlowabilityService consists of 6 methods follow the list below this passage.

3.1.9.2.3.1: Attribute description

N/A

3.1.9.2.3.2: Method description

- **add (flowability: Flowability)** – The adding flowability method is used, when the user wants to add a new flowability to the database. This method adds a new flowability by input variable of flowability object. If the flowability object that input by the user is not contained in the database, this method will add a new flowability to the database and return the flowability object from the database to the user after the adding flowability is successful. On the other hand, when the flowability object that input by the user is contained in the database. This method will return a null value to the user.
- **update (flowability: Flowability)** - The updating flowability method is used, when the user wants to update an existing flowability in the database. This method update the existing flowability by input variable of flowability object. If the flowability object that input by the user is contained in the database, this method will update an existing flowability in the database and return the flowability object from the database to the user after the updating flowability is successful. On the other hand, when the flowability object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (flowability: Flowability)** – The deleting flowability method is used when the user wants to deletes the existing flowability from the database. This method delete the flowability by input variable of flowability object. If the flowability object that input by the user is contained in the database, this method will delete an existing flowability from the database and return the flowability object to the user after the deleting flowability is successful. On the other hand, when the flowability object

that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll() : Flowability []** – The findAll method is used, when the user wants to get all Flowability data in the database. This method is return as a list of Flowability object from the database.
- **findById(id : long) : Flowability** – The findById method is used, when the user wants to get the flowability object from the system. This method gets flowability 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.
- **findByType(type : String)** –The findByType method is used when the user wants to get the flowability object from the system. This method gets flowability object from the database by type 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.

3.1.9.2.4 FlowabilityServiceImpl

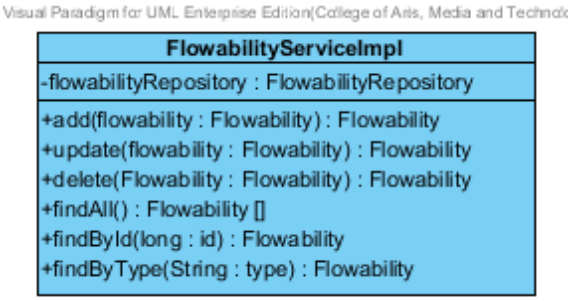


Figure 44 – The FlowabilityServiceImpl class

FlowabilityServiceImpl is the Flowability service class that implements the method from FlowabilityService. So, the method of FlowabilityServiceImpl is same as FlowabilityService.

3.1.9.2.4.1: Attribute description

- **FlowabilityRepository** – the repository of Flowability. This attribute is used for Flowability data management.

3.1.9.2.4.2: Method description

Same as Flowability Service

3.1.9.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.10- CD-10: Density Class diagram

3.1.10.1: Class diagram

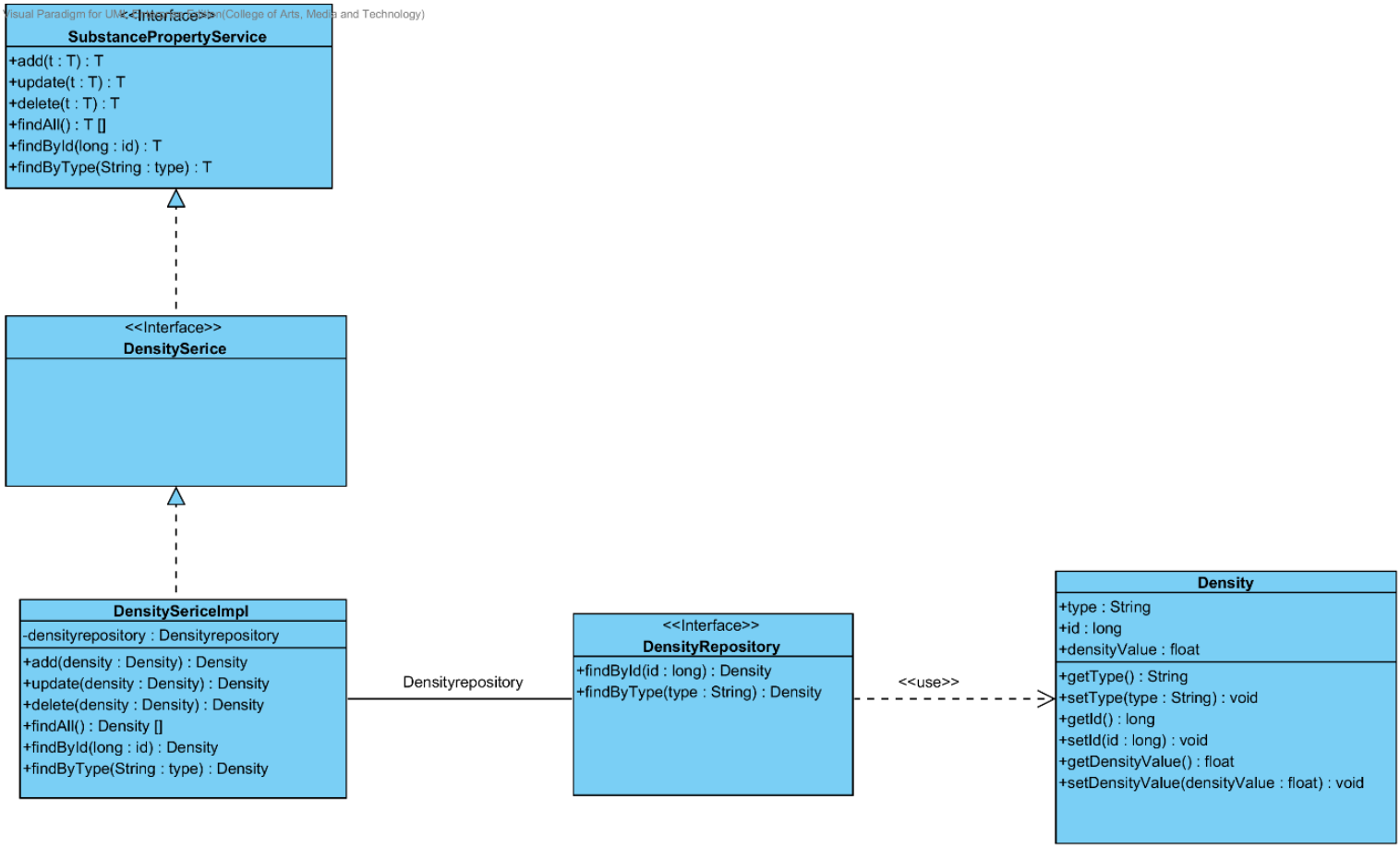


Figure 45 - CD-10: Density Class diagram

3.1.10.2: Class description

From the figure 45, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.10.2.1 Density class

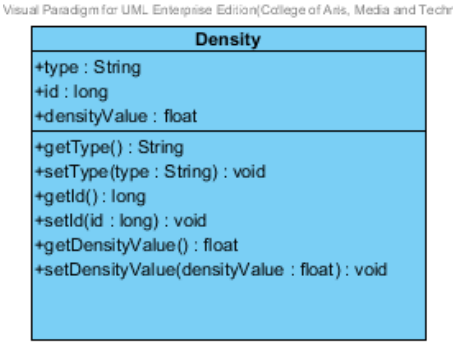


Figure 46 – The Density class

Density class is a part of drug's substance. Density class is an entity class that will be saved to the system. The Density class consists of 3 attributes follow the list below this passage.

3.1.8.2.1.1: Attribute description

- **Id** – the identity of the Density class. Id attribute is a long number.
- **Type** – the type of the Density class. The Density type value is string.
- **Density value** - The density value is a value of each type of the Density. The density value is stable value. The value is float number.

3.1.8.2.1.2: Method description

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

3.1.10.2.2 DensityRepository Interface

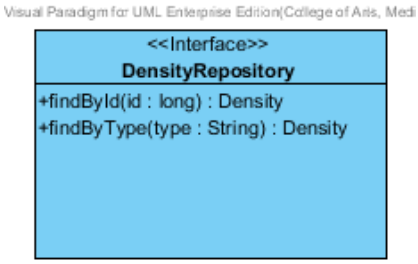


Figure 47 – The DensityRepository interface

DensityRepository Interface is an interface that use for CRUD with entity classes in the system. The most of DensityRepository interface's method is generated from Spring data MongoDB framework. DensityRepository interface consists of 5 methods is shown below this passage.

3.1.10.2.2.1: Attribute description

N/A

3.1.10.2.2.2: Method description

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

3.1.10.2.3 Density Service

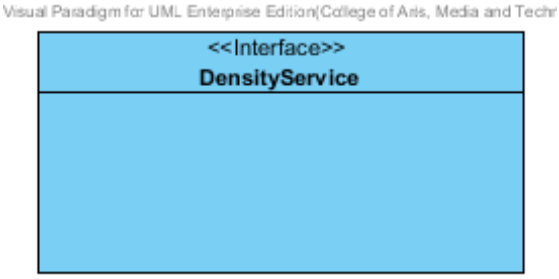


Figure 48 – The DensityService Interface.

DensityService is business processing logic for the density entity. DensityService manages the density data through the DensityRepository interface. DensityService consists of 6 methods follow the list below this passage.

3.1.10.2.3.1: Attribute description

N/A

3.1.10.2.3.2: Method description

- **add (density: Density)** – The adding density method is used, when the user wants to add a new density to the database. This method adds a new density by input variable of density object. If the density object that input by the user is not contained in the database, this method will add a new density to the database and return the f density object from the database to the user after the adding density is successful. On the other hand, when the density object that input by the user is contained in the database. This method will return a null value to the user.
- **update (density: Density)** - The updating density method is used, when the user wants to update an existing density in the database. This method update the existing density by input variable of density object. If the density object that input by the user is contained in the database, this method will update an existing density in the database and return the density object from the database to the user after the updating density is successful. On the other hand, when the density object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (density: Density)** – The deleting density method is used when the user wants to deletes the existing density from the database. This method delete the density by input variable of density object. If the density object that input by the user is contained in the database, this method will delete an existing density from the database and return the density object to the user after the deleting density is successful. On the other hand, when the density object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll() : Density []** – The findAll method is used, when the user wants to get all density data in the database. This method is return as a list of density object from the database.
- **findById(id : long) : Density** – The findById method is used, when the user wants to get the density object from the system. This method gets density 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.
- **findByType(type : String) : Density** –The findByType method is used when the user wants to get the density object from the system. This method gets density object from the database by id that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.10.2.4 DensityServiceImpl

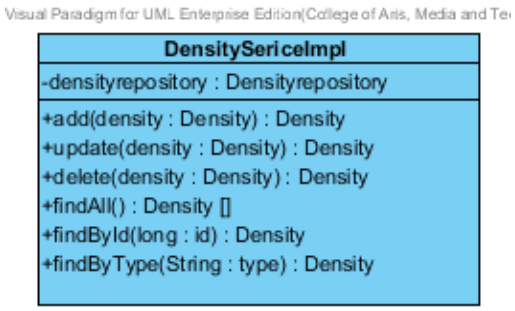


Figure 49 – The DensityServiceImpl class

DensityServiceImpl is the Density service class that implements the method from DensityService. So, the method of DensityServiceImpl is same as DensityService.

3.1.10.2.4.1: Attribute description

- **DensityRepository** – the repository of Density. This attribute is used for Density data management.

3.1.10.2.4.2: Method description

Same as Density Service

3.1.10.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.1.11- CD-11: CompoundFunction Class diagram

3.1.11.1: Class diagram

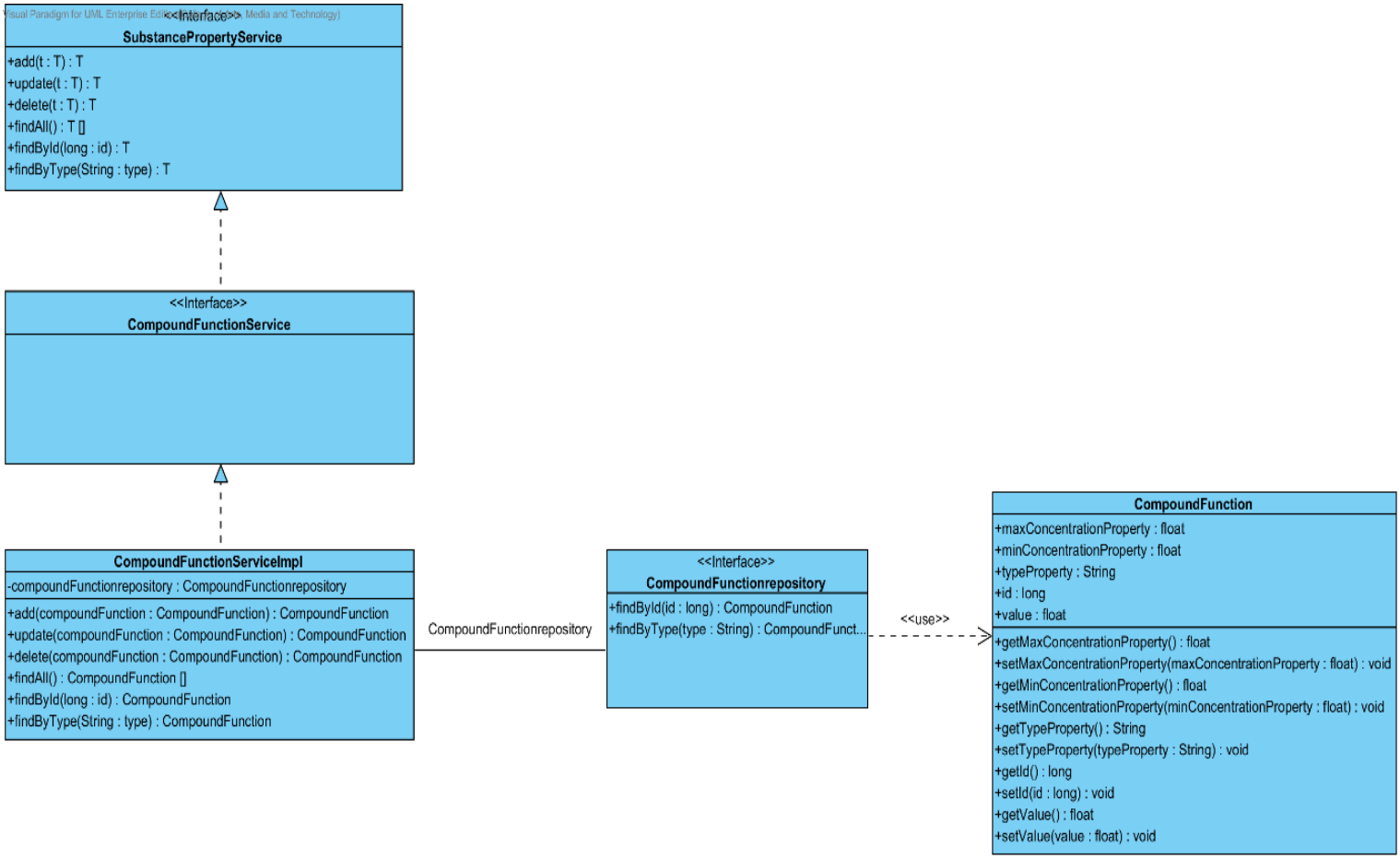


Figure 50 - CD-11: CompoundFunction Class diagram

3.1.11.2: Class description

From the figure 50, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.1.11.2.1 CompoundFunction class

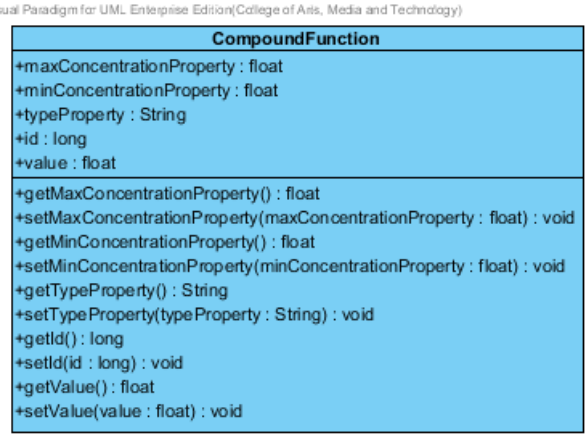


Figure 51 – The CompoundFunction class

CompoundFunction class is a part of drug's substance. CompoundFunction class is an entity class that will be saved to the system. The CompoundFunction class consists of 5 attributes follow the list below this passage.

3.1.8.2.1.1: Attribute description

- **Id** – the identity of the CompoundFunction class. Id attribute is a long number.
- **Type** – the type of the CompoundFunction class. The CompoundFunction type value is string.
- **Max Value** - The max value of each type of CompoundFunction. The pharmacists can use the max value and min value for CompoundFunction type identification.
- **Min Value** – The min value of each type of CompoundFunction. The pharmacists can use the max value and min value for CompoundFunction type identification.
- **CompoundFunction value** - The CompoundFunction value is a value of each type of the CompoundFunction. The CompoundFunction value is stable value. The value is float number.

3.1.8.2.1.2: Method description

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

3.1.11.2.2 CompoundFunctionRepository Interface

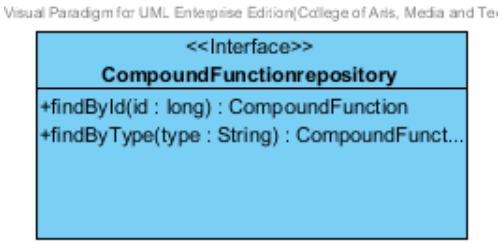


Figure 52 – The CompoundFunctionRepository interface

CompoundFunctionRepository Interface is an interface that use for CRUD with entity classes in the system. The most of CompoundFunctionRepository interface's method is generated from Spring data MongoDB framework. CompoundFunctionRepository interface consists of 5 methods is shown below this passage.

3.1.11.2.2.1: Attribute description

N/A

3.1.11.2.2.2: Method description

- **Save (compoundFunction: CompoundFunction)** - The save method is generated from Spring data MongoDB framework. This method is used when the user wants to add a new CompoundFunction or update existing CompoundFunction to the system. The input variable is a CompoundFunction object.
- **Delete (compoundFunction: CompoundFunction)** – The delete method is generated from Spring data MongoDB framework. This method is used when the user wants to delete the CompoundFunction from the system. The input variable is the CompoundFunction object.
- **findAll () : CompoundFunction []** – The findAll method is generated from Spring data MongoDB framework. This method is used when the user wants to retrieve all of CompoundFunction data from the system. The result of this method is a list of CompoundFunction object.
- **findById (id: long): CompoundFunction** – The findById method is used when the user wants to retrieve the CompoundFunction data from the system. The system gets a CompoundFunction object by the CompoundFunction's id.
- **findByType(type: String)** – The findBytype method is used when the user wants to retrieve the CompoundFunction data from the system. The system gets a CompoundFunction object by the type of CompoundFunction.

3.1.11.2.3 CompoundFunction Service

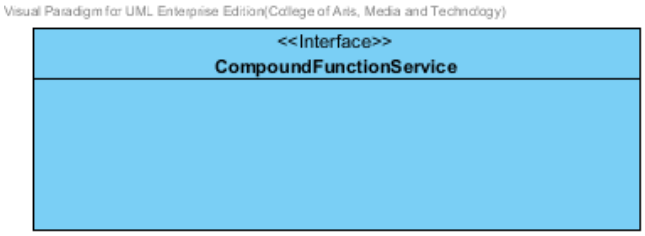


Figure 53 – The CompoundFunctionService Interface.

CompoundFunctionService is business processing logic for the CompoundFunction entity. CompoundFunctionService manages the CompoundFunction data through the CompoundFunction Repository interface. CompoundFunctionService consists of 6 methods follow the list below this passage.

3.1.11.2.3.1: Attribute description

N/A

3.1.11.2.3.2: Method description

- **add (compoundFunction: CompoundFunction)** – The adding compoundFunction method is used, when the user wants to add a new compoundFunction to the database. This method adds a new compoundFunction by input variable of compoundFunction object. If the compoundFunction object that input by the user is not contained in the database, this method will add a new compoundFunction to the database and return the compoundFunction object from the database to the user after the adding compoundFunction is successful. On the other hand, when the compoundFunction object that input by the user is contained in the database. This method will return a null value to the user.
- **update (compoundFunction: CompoundFunction)** - The updating compoundFunction method is used, when the user wants to update an existing compoundFunction in the database. This method update the existing compoundFunction by input variable of compoundFunction object. If the compoundFunction object that input by the user is contained in the database, this method will update an existing compoundFunction in the database and return the compoundFunction object from the database to the user after the updating compoundFunction is successful. On the other hand, when the compoundFunction object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (compoundFunction: CompoundFunction)** – The deleting compoundFunction method is used when the user wants to deletes the existing compoundFunction from the database. This method delete the compoundFunction by input variable of compoundFunction object. If the density object that input by the user is contained in the database, this method will delete an existing compoundFunction from the database and

return the compoundFunction object to the user after the deleting compoundFunction is successful. On the other hand, when the compoundFunction object that input by the user is not contained in the database. This method will return a null value to the user.

- **findAll() : compoundFunction []** – The findAll method is used, when the user wants to get all compoundFunction data in the database. This method is return as a list of compoundFunction object from the database.
- **findById(id : long) : CompoundFunction** – The findById method is used, when the user wants to get the CompoundFunction object from the system. This method gets compoundFunction 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.
- **findByType(type : String)** –The findByType method is used when the user wants to get the compoundFunction object from the system. This method gets compoundFunction object from the database by id that input by the user. On the other hand, if the type that input by user is not contained in the database. This method will return null value to the user.

3.1.11.2.4 CompoundFunctionServiceImpl

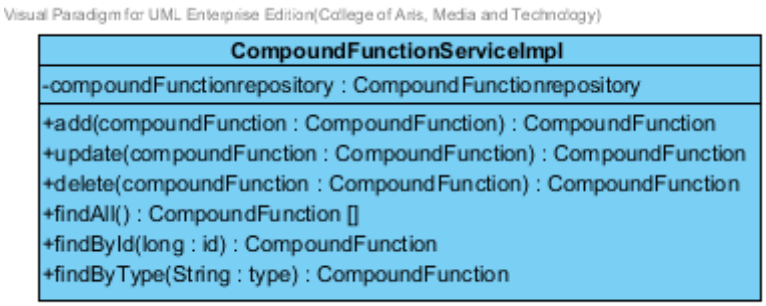


Figure 54 – The CompoundFunctionServiceImpl class

CompoundFunctionServiceImpl is the CompoundFunction service class that implements the method from CompoundFunctionService. So, the method of CompoundFunction ServiceImpl is same as the compoundFunctionService.

3.1.11.2.4.1: Attribute description

- **CompoundFunctionRepository** – the repository of CompoundFunction. This attribute is used for CompoundFunction data management.

3.1.11.2.4.2: Method description

Same as CompoundFunction Service

3.1.11.2.5 SubstancePropertyService

Same as SubstancePropertyService at Solubility class diagram.

3.2 Sub-Feature 6: Manage the drug substance

The Sub-Feature 6 is the substance management. The user can add, update, delete and view the substance. Each substance is created from 9 substance properties that shown on a previous Sub-Feature. An entity relationship between substance and substance's property show on the Figure 01.

3.2.1- CD-12: Substance Class diagram

3.2.1.1: Class diagram

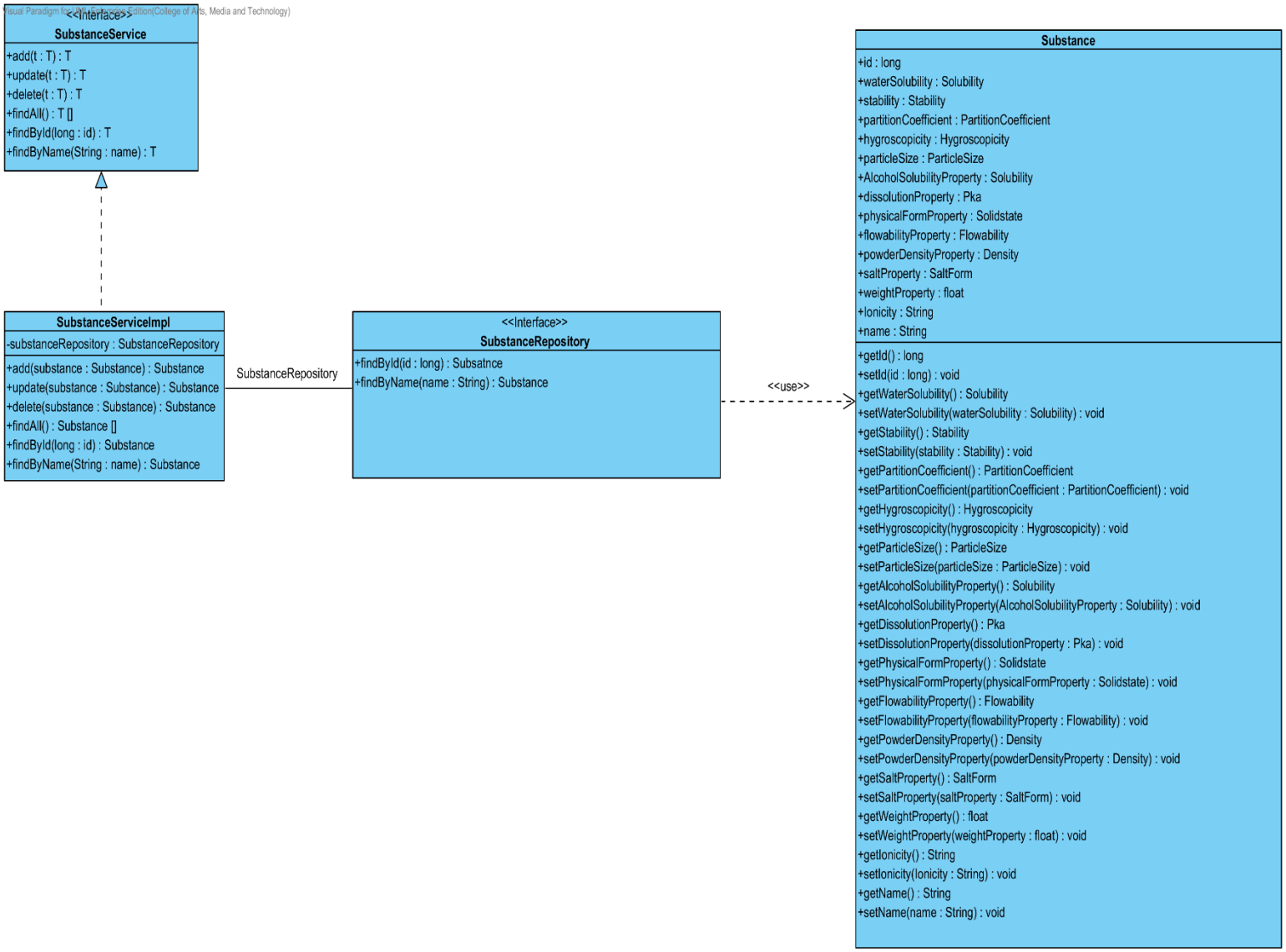


Figure 55 - CD-12: Substance Class diagram

3.2.1.2: Class description

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

3.2.1.2.1 Substance class

Figure 56 – The substance class

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.

3.2.1.2.1.1: Attribute description

- **Id** – the identity of the substance class. Id attribute is a long number.
- **Name** – the name of the substance class. The substance name is shown the name of each excipient.
- **waterSolubility** – the water solubility of substance. In the pharmacy, how good or bad of the water solubility. The water solubility attribute is Solubility object
- **Stability –** the stability of substance. The stability consist of Degradation mechanism and Kinetic reaction. The stability attribute is stability object.
- **PartitionCoefficient** – the partition coefficient of the substance. The partition Coefficient attribute is a PartitionCoefficient object.
- **hygroscopicity** - the hygroscopicity of the substance. The hygroscopicity attribute is a hygroscopicity object.
- **particleSize**- the particleSize of the substance. The particleSize attribute is a particleSize object.
- **alcoholSolubilityProperty**- the alcoholSolubilityProperty of the substance. The hygroscopicity attribute is a solubility object.
- **dissolutionProperty**- the dissolutionProperty of the substance. The dissolutionProperty attribute is a Pka object.
- **physicalFormProperty**- the physicalFormProperty of the substance. The physicalFormProperty attribute is a Solidstate object.
- **flowabilityProperty**- the flowabilityProperty of the substance. The flowabilityProperty attribute is a flowability object.
- **powderDensityProperty**- the powderDensityProperty of the substance. The powderDensityProperty attribute is a density object.
- **saltProperty**- the saltProperty of the substance. The saltProperty attribute is a saltForm object.
- **weightProperty**- the weightProperty of the substance. The weightProperty is show the weight value of each substance. The weightProperty attribute is a float number.
- **Ionicity**- the Ionicity of the substance. The Ionicity attribute is a String.

3.2.1.2.1.2: Method description

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

3.2.1.2.2 SubstanceRepository Interface

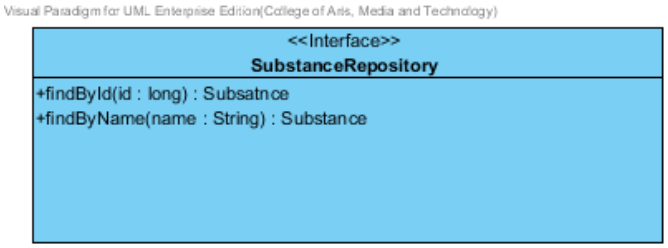


Figure 57 – The SubstanceRepository Interface

SubstanceRepository Interface is an interface that use for CRUD with entity classes in the system. The most of SubstanceRepository interface's method is generated from Spring data MongoDB framework. SubstanceRepository interface consists of 5 methods is shown below this passage.

3.2.1.2.2.1: Attribute description

N/A

3.2.1.2.2.2: 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.
- **findById (id: long): Substance** – The findById 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.
- **findByName(name: String) : Substance** – The findBytype method is used when the user wants to retrieve the substance data from the system. The system gets a substance object by the name of substance.

3.2.1.2.3 SubstanceService

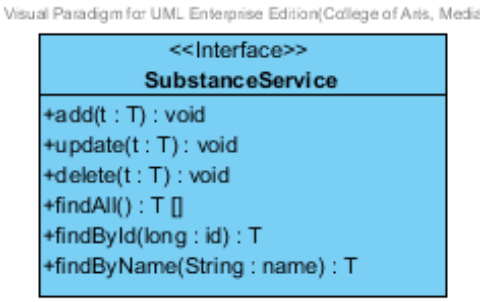


Figure 58 – The SubstanceService Interface.

SubstanceService is business processing logic for solubility entity. SubstanceService manages the solubility data through the SubstanceRepository interface. SubstanceService consists of 6 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

- **add (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.
- **update (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.
- **delete (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.

- **findAll() : Substance []** – The findAll 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.
- **findById(id : long) : Substance** – The findById 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.
- **findByName(name : String)** –The findByName method is used when the user wants to get 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 name that input by user is not contained in the database. This method will return null value to the user.

3.2.1.2.4 SubstanceServiceImpl

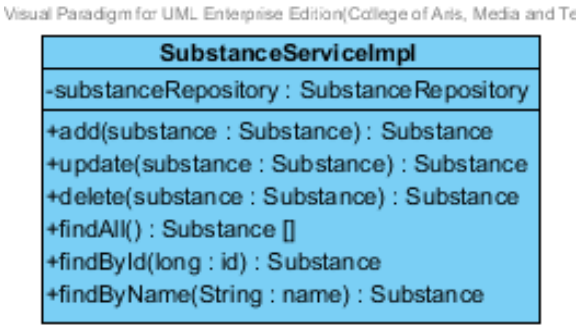


Figure 59 – The SubstanceServiceImpl class

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

3.2.1.2.4.1: Attribute description

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

3.2.1.2.4.2: Method description

Same as SubstanceService

3.3 Sub-Feature 7: Manage the drug Excipient

The Sub-Feature 7 is the excipient management. The user can add, update, delete and view the excipient. Each excipient is created from the substance and compoundfunction. The relationship between excipient entity, substance entity, and compoundfunction is illustrated in the Figure 60 below on this passage.

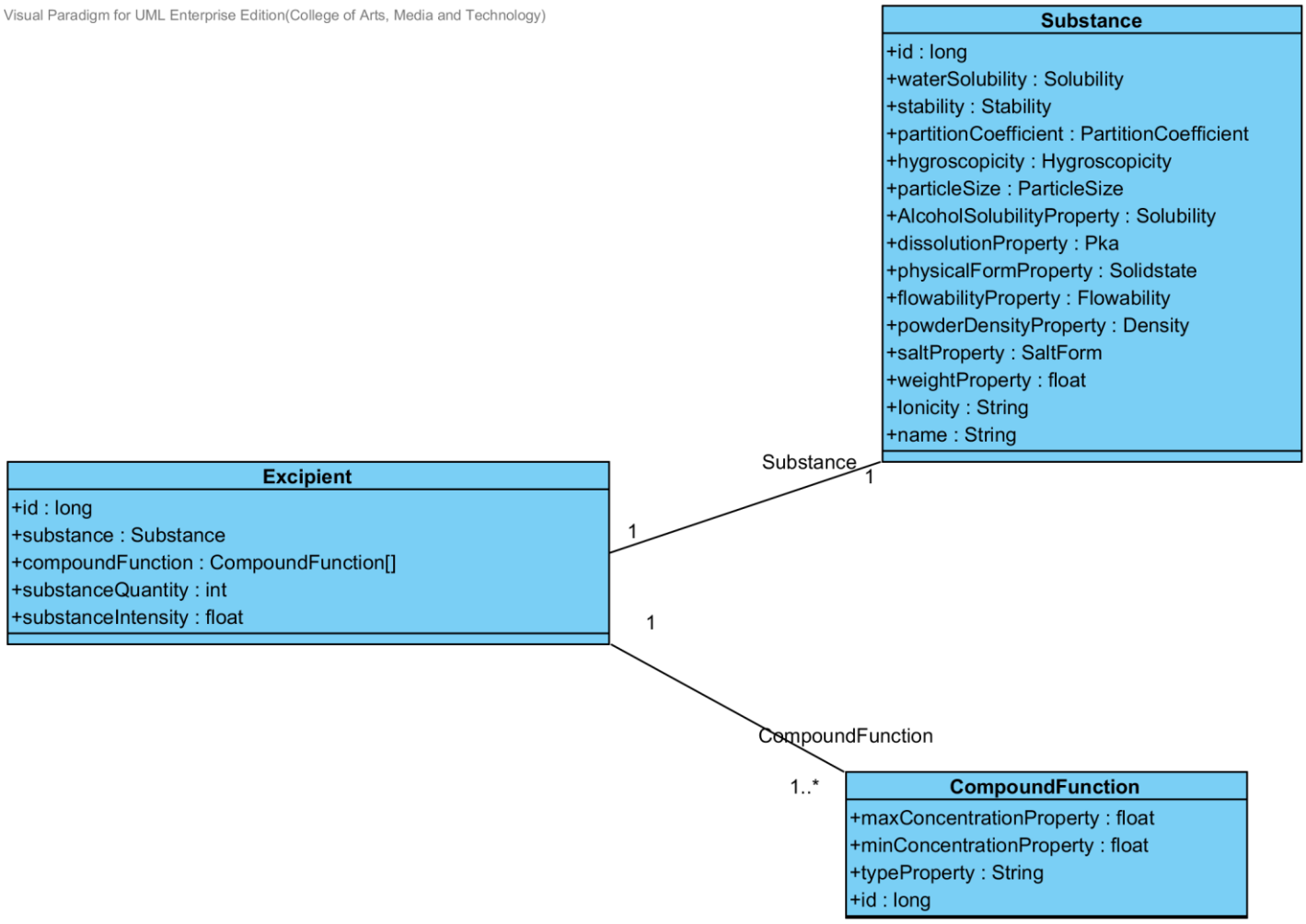


Figure- 60: The entity relationship between an excipient, substance and compoundFunction.

3.3.1- CD-13: Excipient Class diagram

3.3.1.1: Class diagram

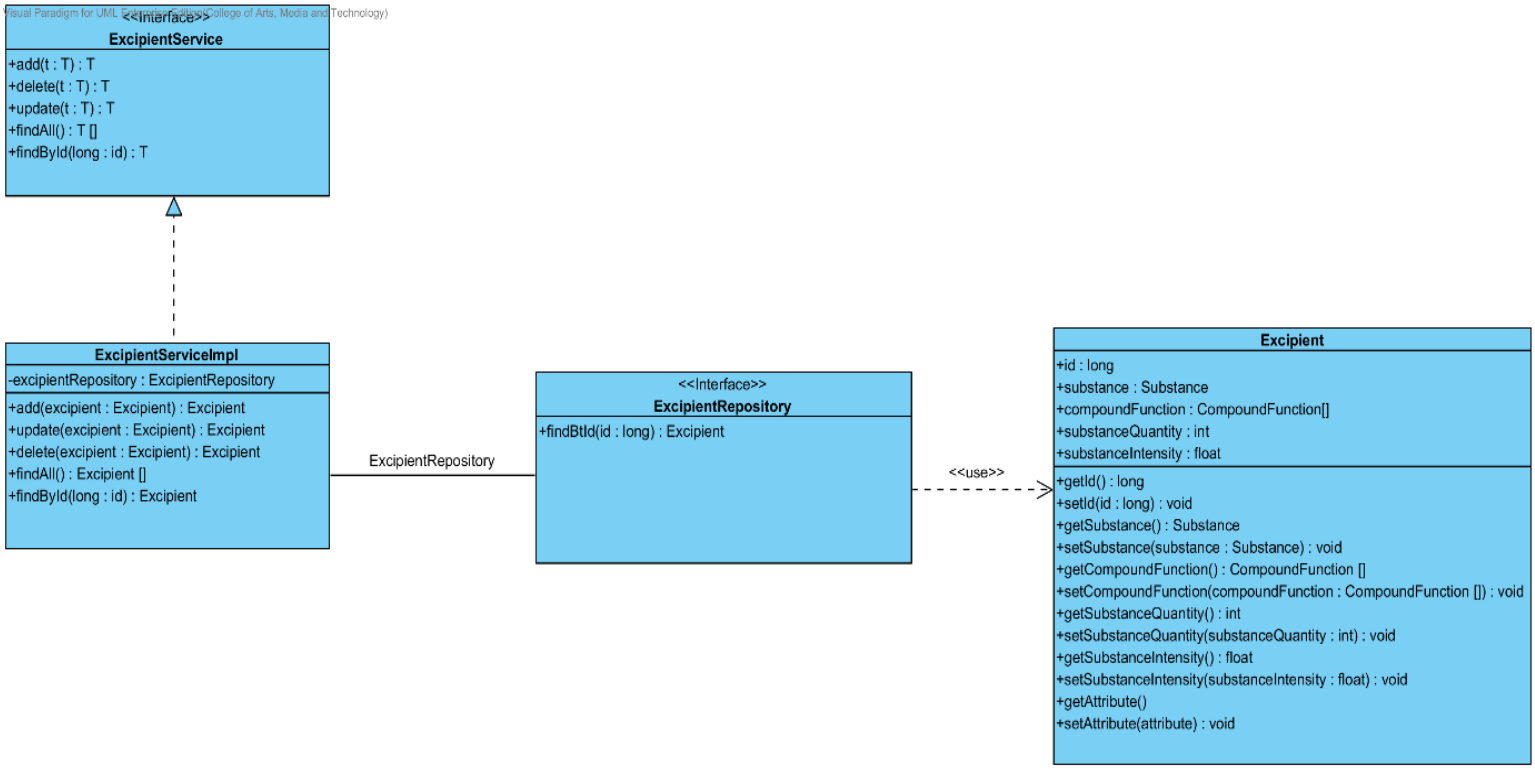


Figure 61- CD-13: Excipient Class diagram

3.3.1.2: Class description

From the figure 61, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.3.1.2.1 Excipient class

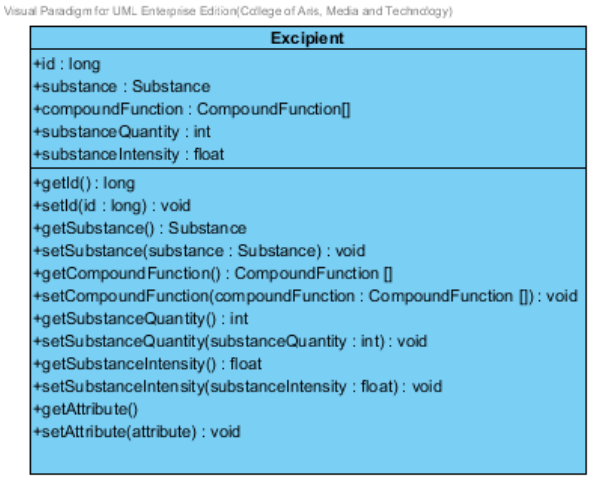


Figure 62 – The excipient class

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.

3.3.1.2.1.1: 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.
- **CompoundFunction –**the compoundFunction of the excipient class .The user can set the compoundFunction in each substance.The compoundFunction is a compoundFunction object.
- **substanceQuantity** – the substance quantity of the excipient. The substance quantity is integer number.
- **substanceIntensity** –the substance intensity of the excipient. The substance intensity is the float value.

3.3.1.2.1.2: Method description

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

3.3.1.2.2 ExcipientRepository Interface

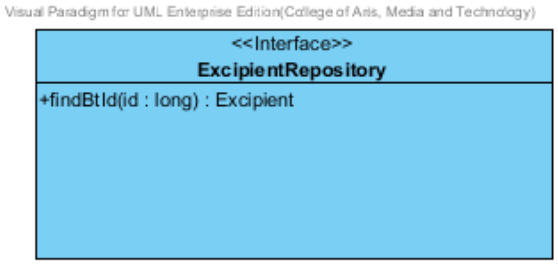


Figure 63 – The excipient repository interface

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.

3.2.1.2.2.1: Attribute description

N/A

3.2.1.2.2.2: 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 a 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.
- **findById (id: long): Excipient** – The findById 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.

3.3.1.2.3 ExcipientService

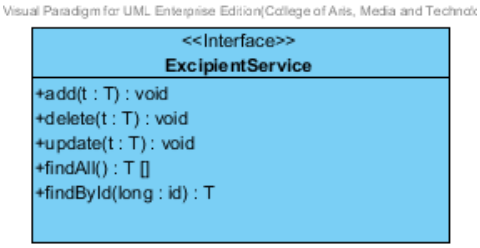


Figure 64 – The ExcipientService Interface.

ExcipientService is business processing logic for excipient entity. ExcipientService manages the excipient data through the ExcipientRepository interface. ExcipientService consists of 6 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

- **add (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.
- **update (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.
- **delete (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.

- **findAll() : Excipient []** – The findAll 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.
- **findById(id : long) :** Excipient – The findById 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.

3.3.1.2.4 ExcipientServiceImpl

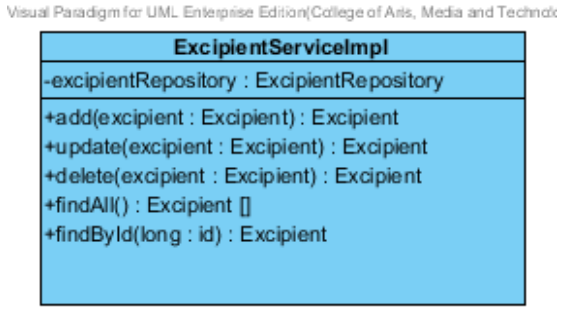


Figure 65 – The ExcipientServiceImpl class

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

3.3.1.2.4.1: Attribute description

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

3.3.1.2.4.2: Method description

Same as ExcipientService

3.4 Sub-Feature 8: Manage the drug formulation

The Sub-Feature 8 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 66 below on this passage.

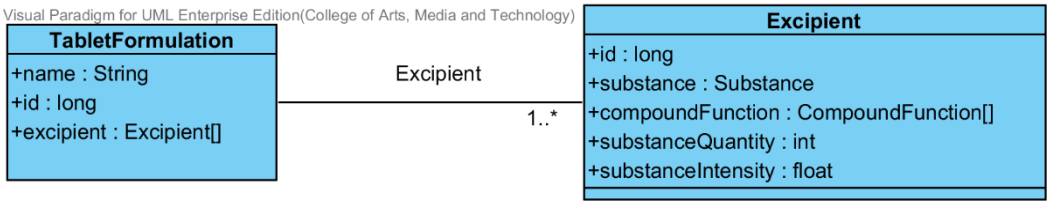


Figure- 66: The entity relationship between a drug's formulation and the excipient.

3.4.1- CD-14: TabletFormulation Class diagram

3.4.1.1: Class diagram

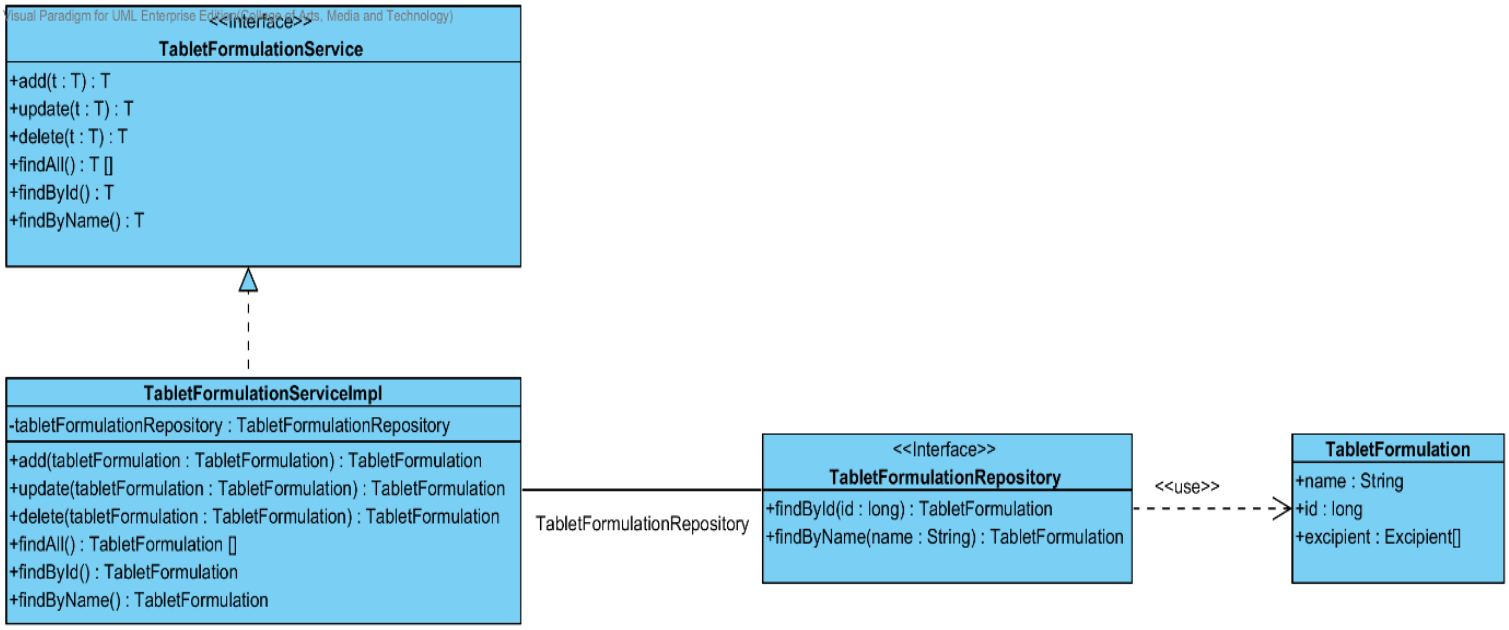


Figure 67- CD-14: TabletFormulation Class diagram

3.4.1.2: Class description

From the figure 67, it can divide into 4 important classes. The detail of each class is described on the next paragraph.

3.4.1.2.1 TabletFormulation class

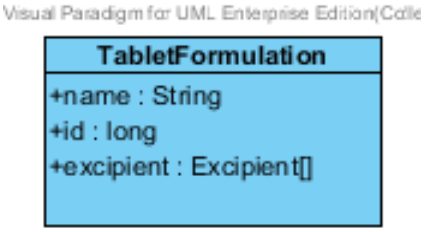


Figure 68 – The TabletFormulation class

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

3.4.1.2.1.1: Attribute description

- **Id** – the identity of the TabletFormulation class. Id attribute is a long number.
- **Name** – the name of the TabletFormulation class. The TabletFormulation attribute is a TabletFormulation object.
- **Excipient –**the excipient of the TabletFormulation. The user can set the excipient more than 1 excipient in each tabletformulation.

3.4.1.2.1.2: Method description

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

3.4.1.2.2 TabletFormulationRepository Interface

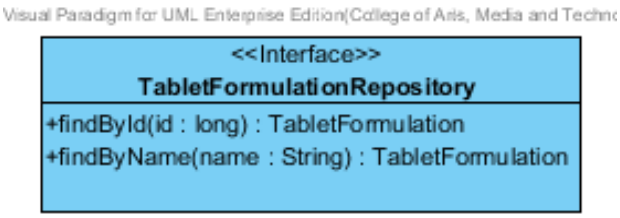


Figure 69 – The TabletFormulation interface

TabletFormulationRepository Interface is an interface that use for CRUD with entity classes in the system. The most of TabletFormulationRepository interface's method is generated from Spring data MongoDB framework. TabletFormulationRepository interface consists of 5 methods is shown below this passage.

3.4.1.2.2.1: Attribute description

N/A

3.4.1.2.2.2: Method description

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

3.4.1.2.3 TabletFormulationService

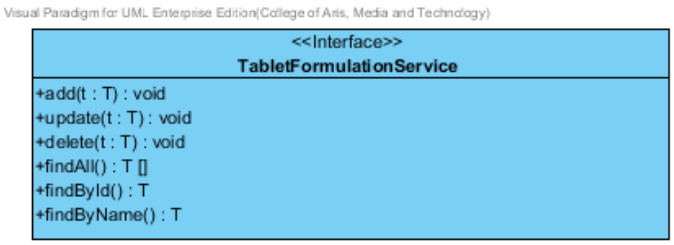


Figure 70 – The TabletFormulationService Interface.

TabletFormulationService is business processing logic for TabletFormulation entity. TabletFormulationService manages the TabletFormulation data through the TabletFormulationRepository interface. TabletFormulationService 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

- **add (tabletFormulation: TabletFormulation)** – The adding tabletFormulation method is used, when the user wants to add a new tabletFormulation to the database. This method adds a new tabletFormulation by input variable of tabletFormulation object. If the tabletFormulation object that input by the user is not contained in the database, this method will add a new tabletFormulation to the database and return the tabletFormulation object from the database to the user after the adding tabletFormulation is successful. On the other hand, when the tabletFormulation object that input by the user is contained in the database. This method will return a null value to the user.
- **update (tabletFormulation: TabletFormulation)** - The updating tabletFormulation method is used, when the user wants to update an existing tabletFormulation on in the database. This method update the existing tabletFormulation by input variable of tabletFormulation object. If the tabletFormulation object that input by the user is contained in the database, this method will update an existing tabletFormulation in the database and return the tabletFormulation object from the database to the user after the updating tabletFormulation is successful. On the other hand, when the tabletFormulation object that input by the user is not contained in the database. This method will return a null value to the user.
- **delete (tabletFormulation: TabletFormulation)** The deleting tabletFormulation method is used when the user wants to deletes the existing tabletFormulation from the database. This method delete the tabletFormulation by input variable of tabletFormulation object. If the tabletFormulation object that input by the user is contained in the

database, this method will delete an existing tabletFormulation from the database and return the tabletFormulation object to the user after the deleting tabletFormulation 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.

- **findAll() : TabletFormulation []** – The findAll method is used, when the user wants to get all tabletFormulation data in the database. This method is return as a list of tabletFormulation object from the database.
- **findById(long : id) : TabletFormulation** – The findById method is used, when the user wants to get the tabletFormulation data in the system. This method gets tabletFormulation 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.
- **findByName(name : String) : TabletFormulation** – The findById method is used, when the user wants to get the tabletFormulation data in the system. This method gets tabletFormulation object from the database by id that input by the user. On the other hand, if the name that input by user is not contained in the database. This method will return null value to the user.

3.4.1.2.4 TabletFormulationServiceImpl

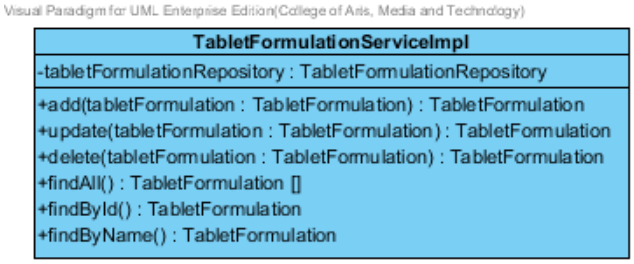


Figure 71 – The TabletFormulationServiceImpl class

TabletFormulationServiceImpl is the tabletformulation service class that implements the method from TabletFormulationService. So, the method of TabletFormulationServiceImpl is same as TabletFormulationService.

3.4.1.2.4.1: Attribute description

- **TabletFormulationRepository** – the repository of tabletFormulation. This attribute is used for tabletFormulation data management.

3.4.1.2.4.2: Method description

Same as TabletFormulationService

Chapter 4 | Sequence Diagram

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

No	Sub 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.	SQD-9A, SQD-9B, SQD-9C, SQD-9D, SQD-9E, SQD-9F, SQD-9G, SQD-9H, SQD-9I, SQD-9J, SQD-9K
		URS-10	The user updates an existing substance property into the system.	SQD-10A, SQD-10B, SQD-10C, SQD-10D, SQD-10E, SQD-10F, SQD-10G, SQD-10H, SQD-10I, SQD-10J, SQD-10K
		URS-11	The user deletes an existing substance property from the system.	SQD-11A, SQD-11B, SQD-11C, SQD-11D, SQD-11E, SQD-11F, SQD-11G, SQD-11H, SQD-11I, SQD-11J, SQD-11K
6	Manage the drug substance	URS-12	The user adds a new substance into the system.	SQD-12
		URS-13	The user updates an existing substance into the system.	SQD-13
		URS-14	The user deletes an existing substance from the system.	SQD-14
		URS-15	The user views the substance in the system.	SQD-15
7	Manage the drug excipient	URS-16	The user adds a new excipient to the system.	SQD-16
		URS-17	The user updates an existing drug excipient in the system.	SQD-17
		URS-18	The user delete an existing drug excipient in the system.	SQD-18
		URS-19	The user views all the drug excipient in the system.	SQD-19
8	Manage the drug formulation	URS-20	The user adds a new drug formulation case into the system.	SQD-20
		URS-21	The user updates an existing drug formulation case in the system.	SQD-21
		URS-22	The user deletes an existing drug formulation case in the system.	SQD-22

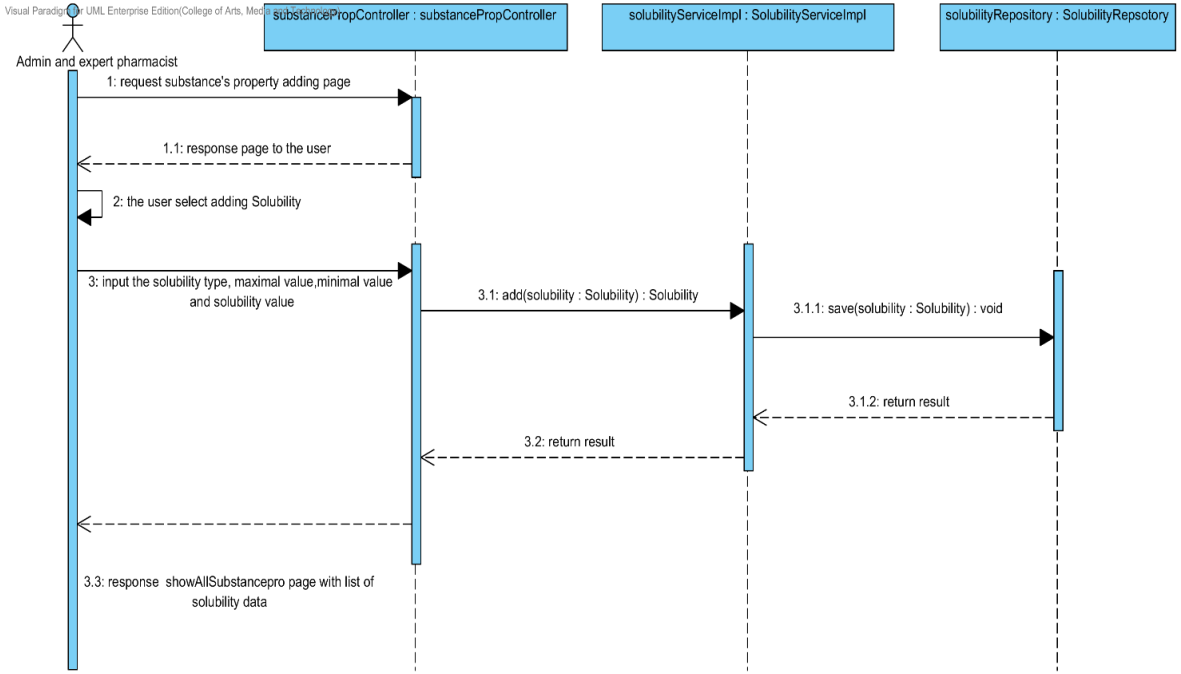
		URS-23	The user views all of the formulation in the system.	SQD-23

4.1- Sub-Feature 5: Manage the drug substance property

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

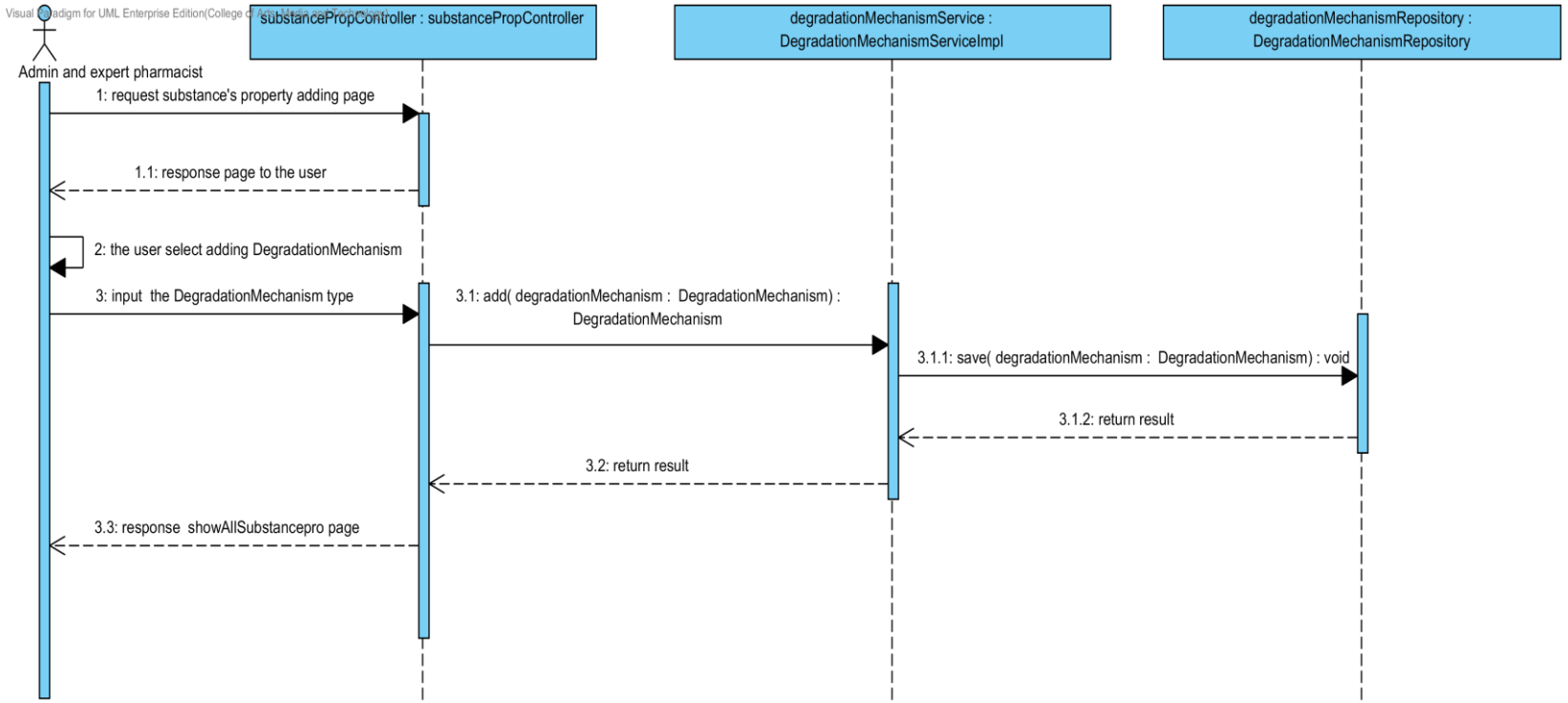
In the sequence diagram, the user can add a new substance property to the system. Firstly, the user opens the substance property adding page, then the user inputs substance property value such as name, max value min value and substance value. The substance property controller gets the data from the user, after that the controller chooses appropriate the SubstancePropertyService for saving a new substance property to the system. Finally, the substance property controller shows a new substance property on the successful adding substance property page to the user.

4.1.1.1 SQD-09A: The user add a new drug substance property into the system (Solubility).

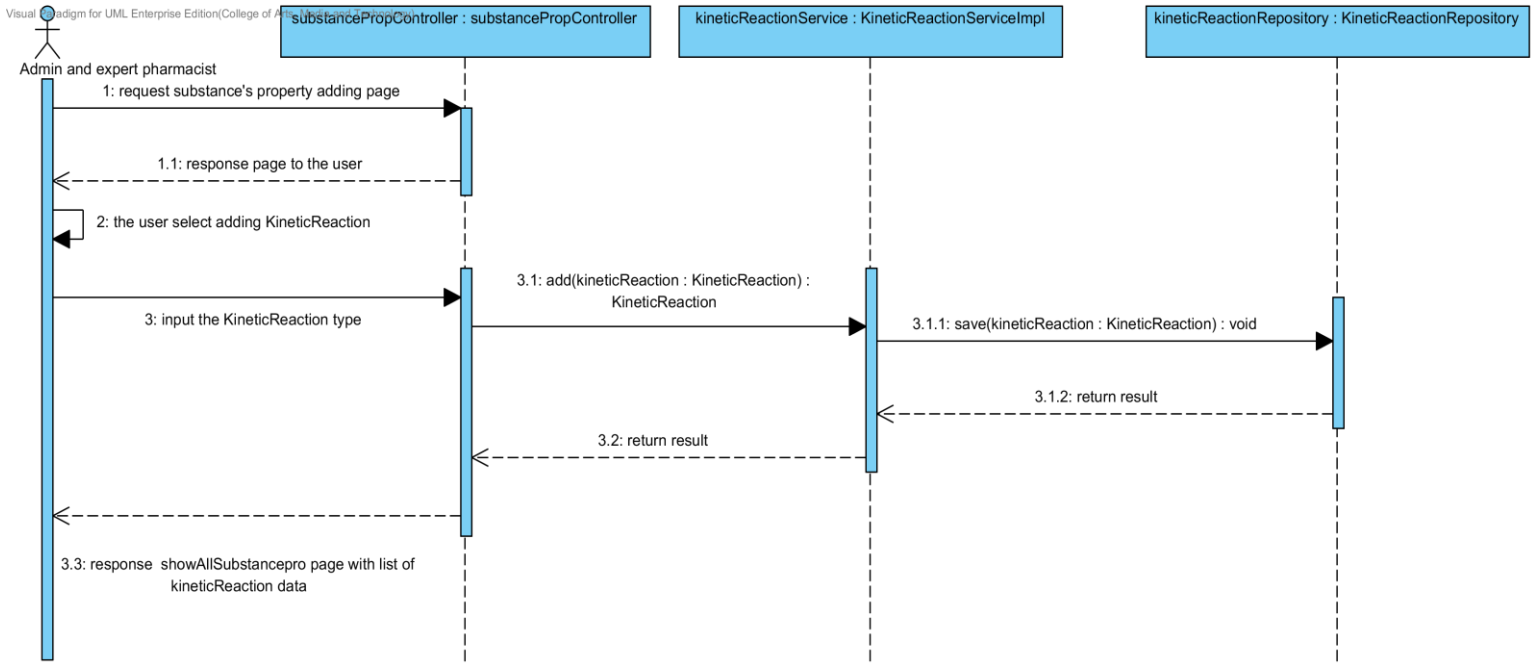


SD-09A : The user add a new drug substance property to the database

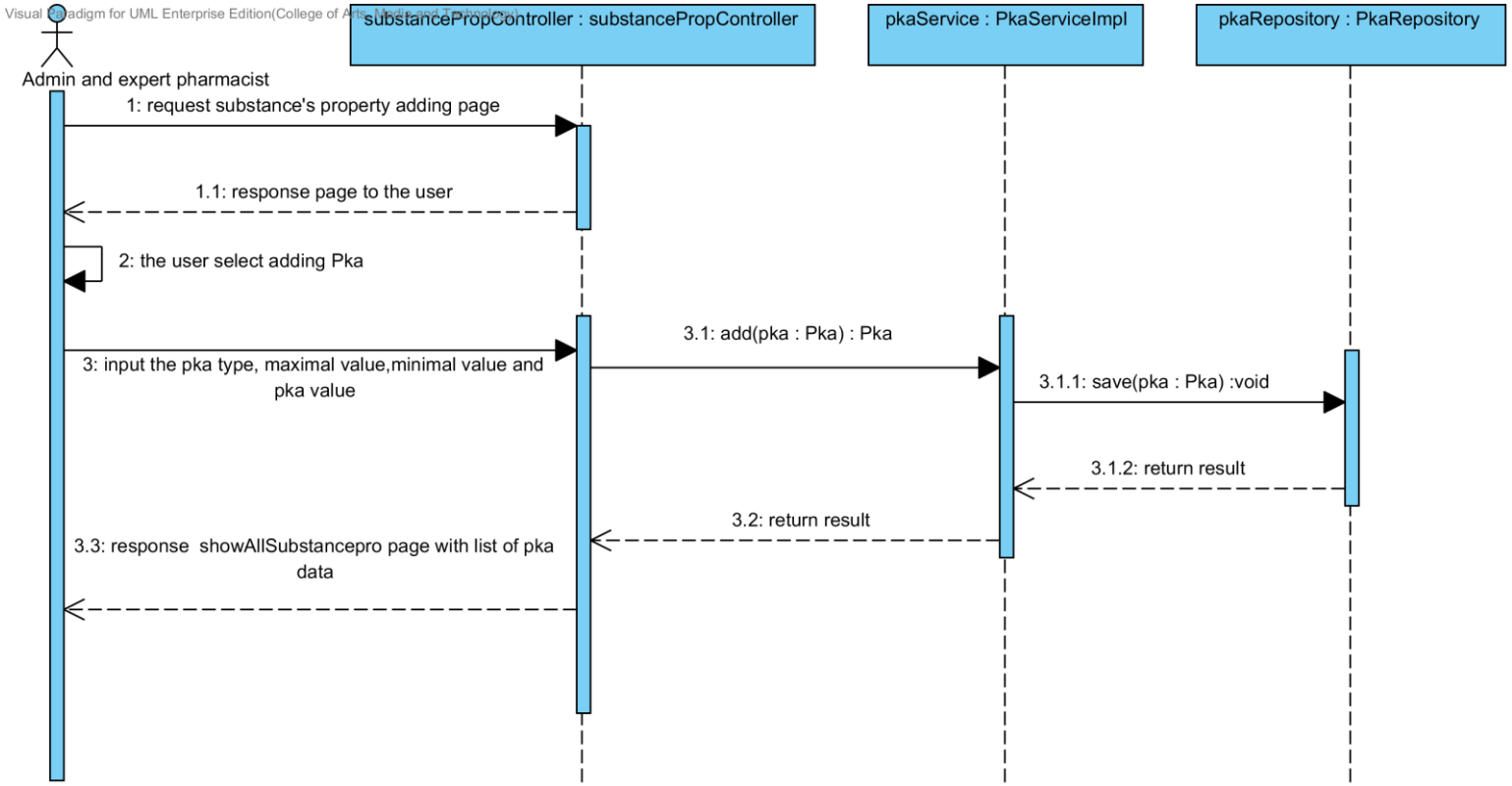
4.1.1.2 SQD-09B: The user add a new drug substance property into the system (DegradationMechanism).



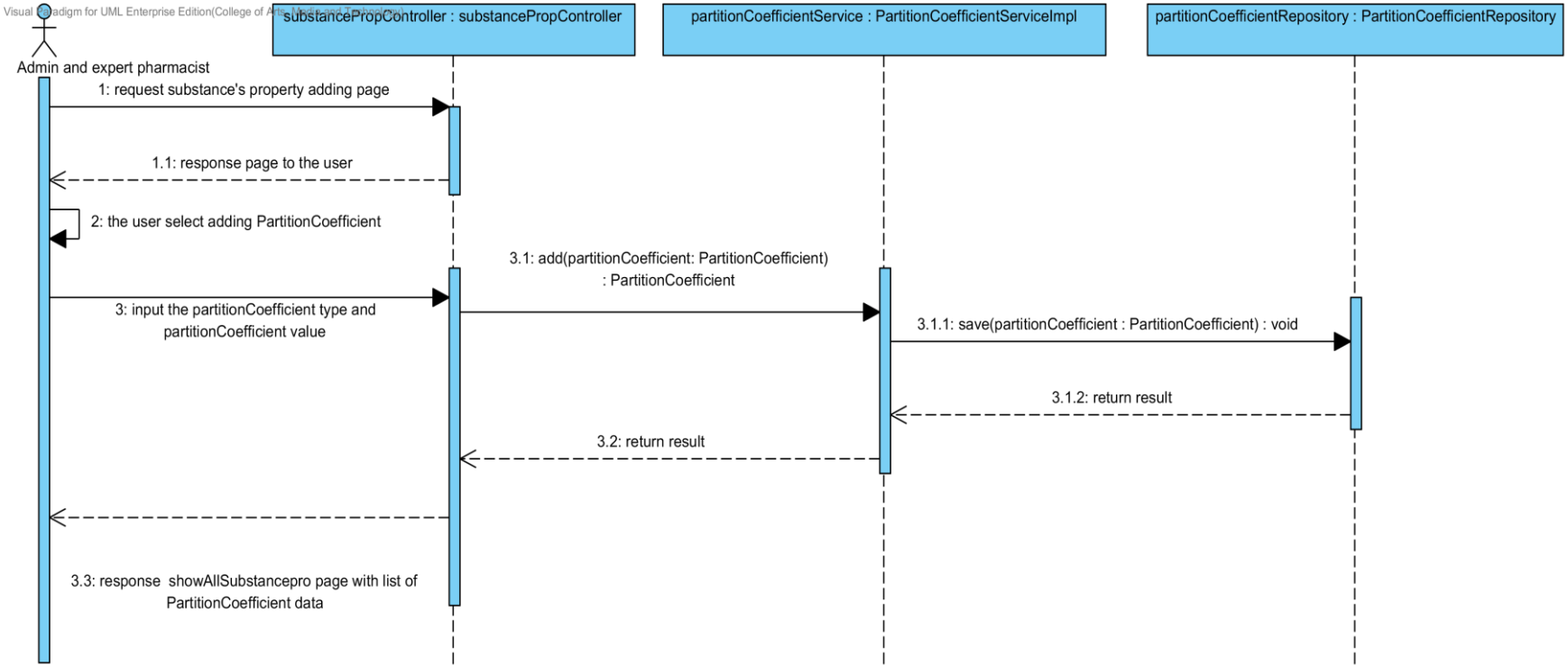
4.1.1.3 SQD-09C: The user add a new drug substance property into the system (Kinetic Reaction).



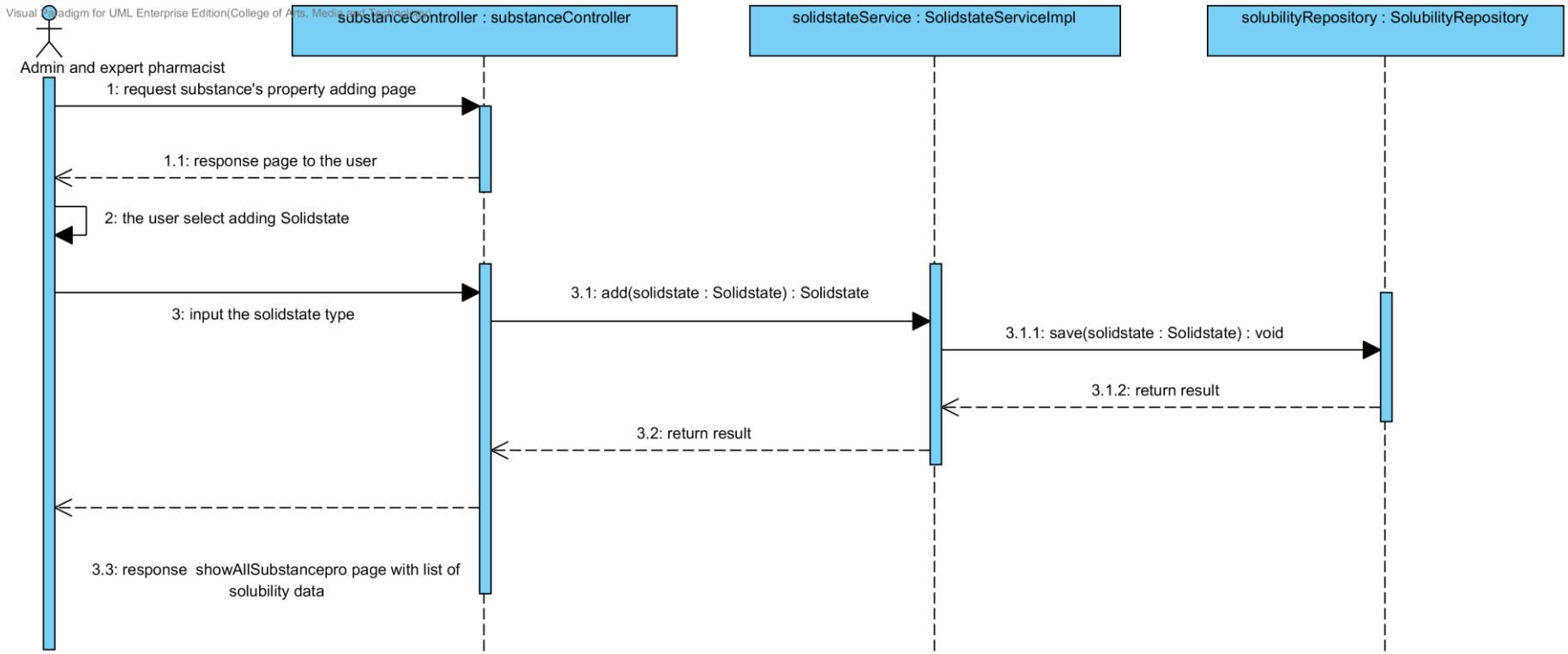
4.1.1.4 SQD-09D: The user add a new drug substance property into the system (Pka).



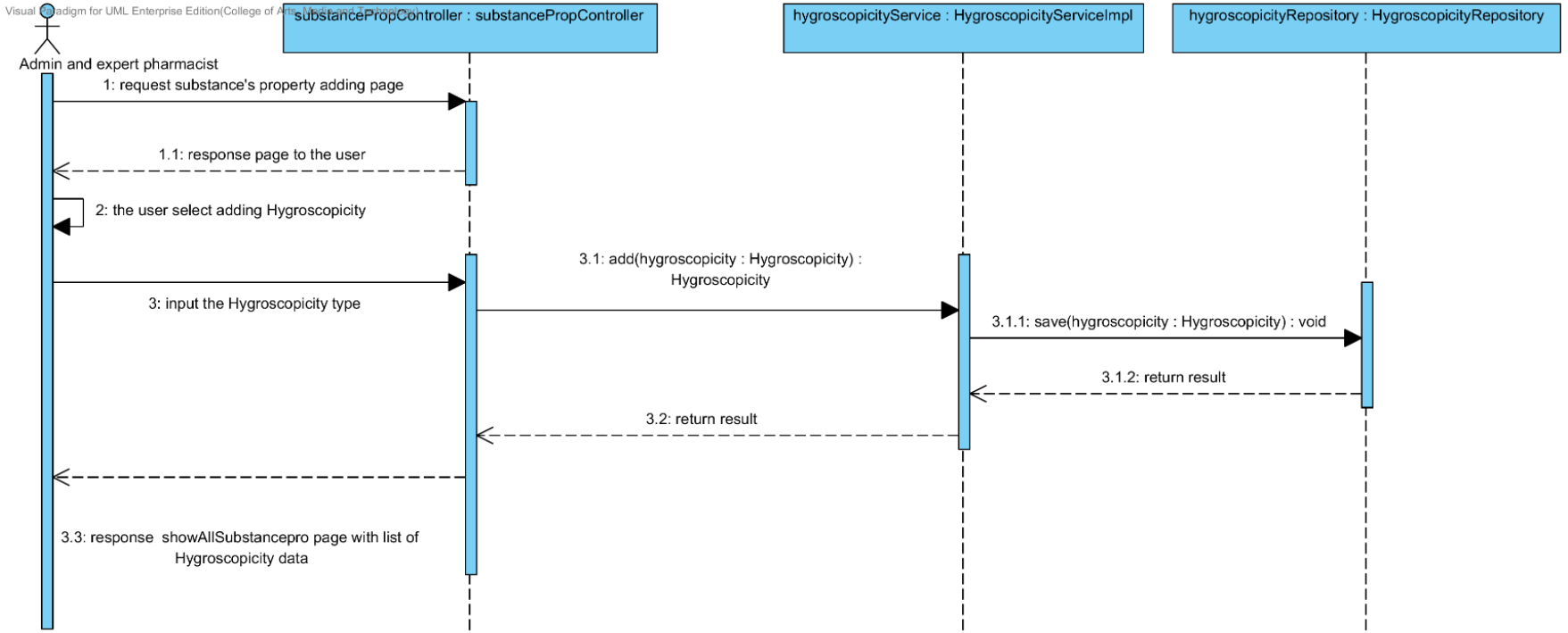
4.1.1.5 SQD-09E: The user add a new drug substance property into the system (PartitionCoefficient).



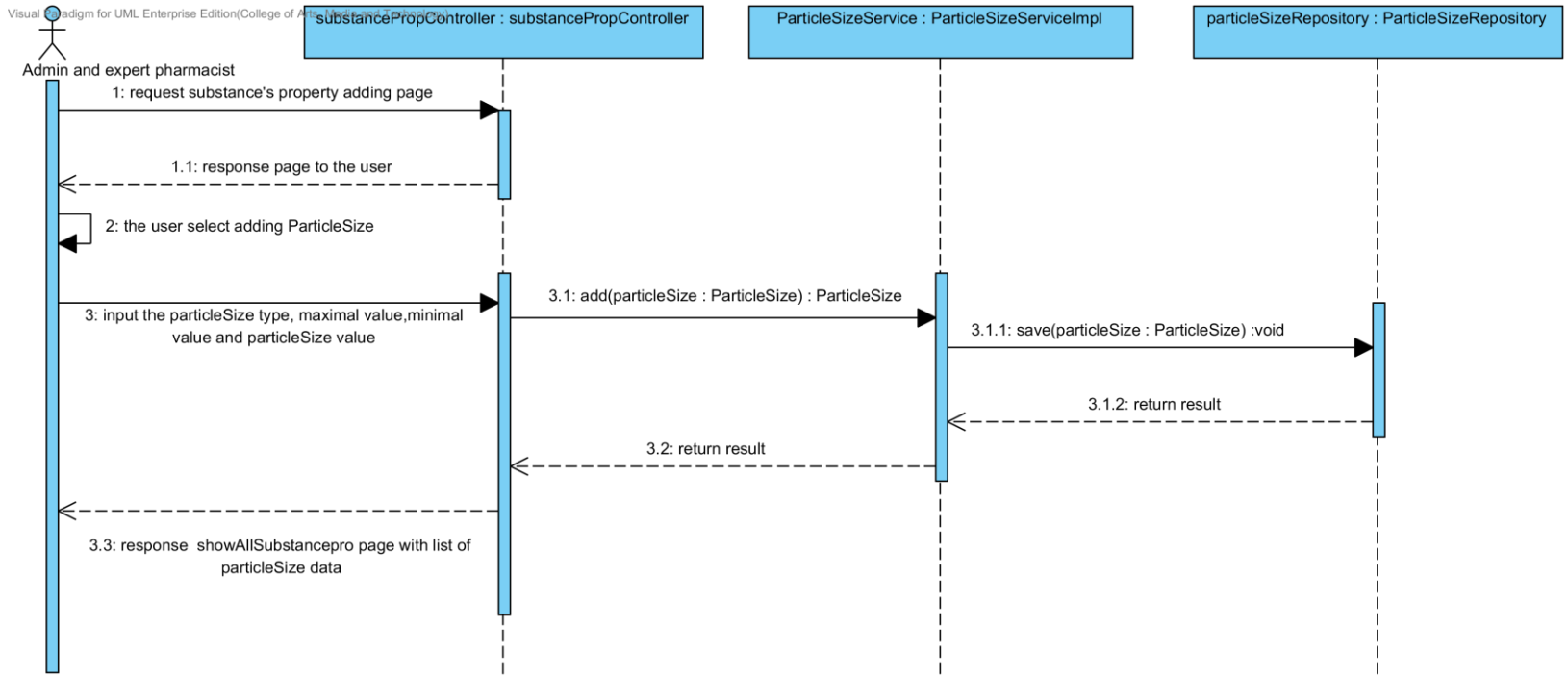
4.1.1.6 SQD-09F: The user add a new drug substance property into the system (Solidstate).



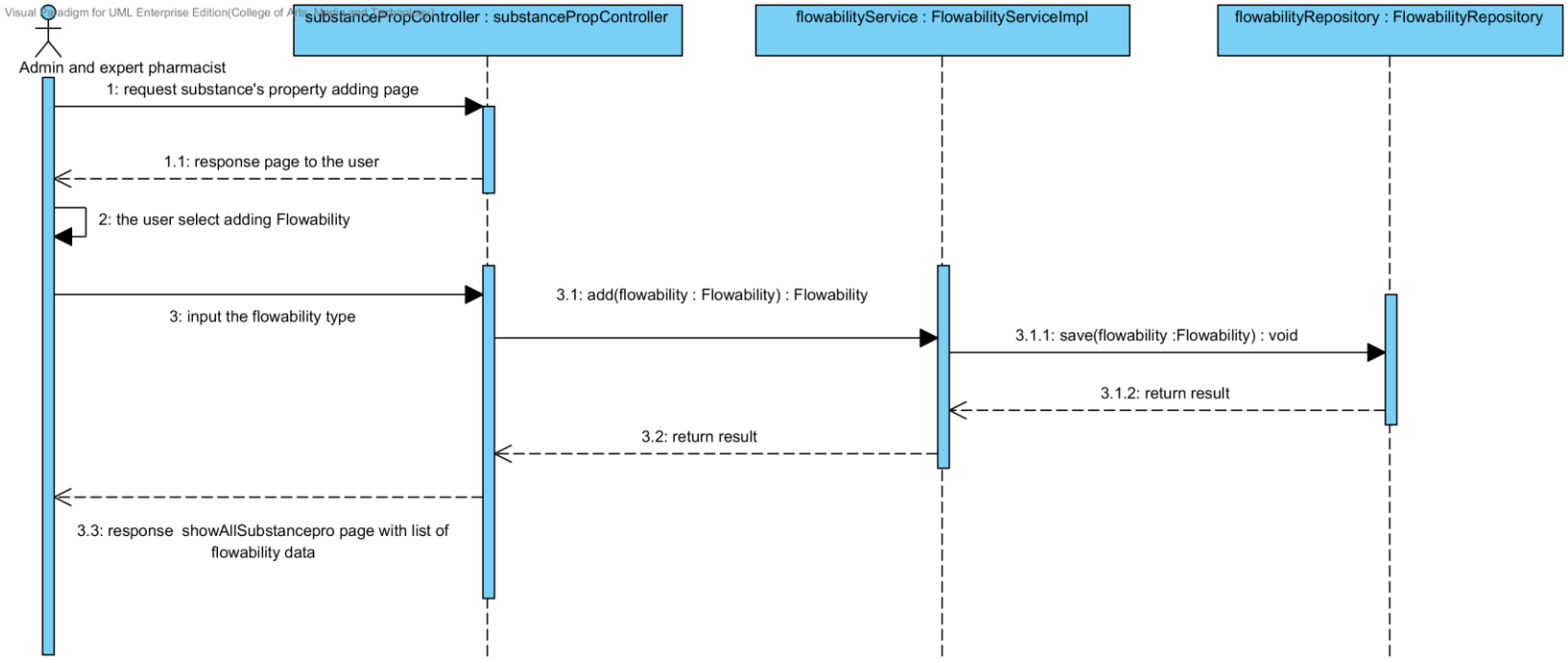
4.1.1.7 SQD-09G: The user add a new drug substance property into the system (Hygroscopicity).



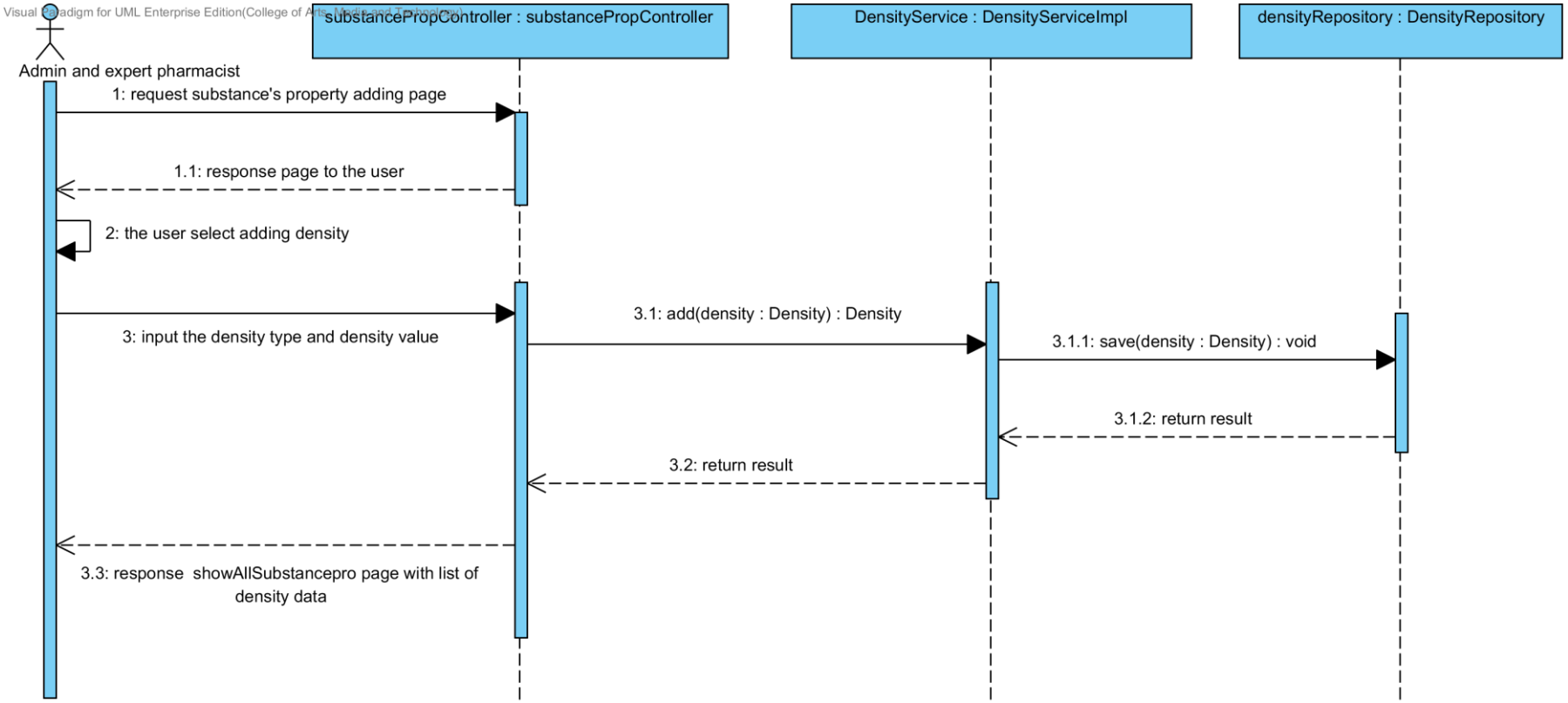
4.1.1.8 SQD-09H: The user add a new drug substance property into the system (ParticleSize).



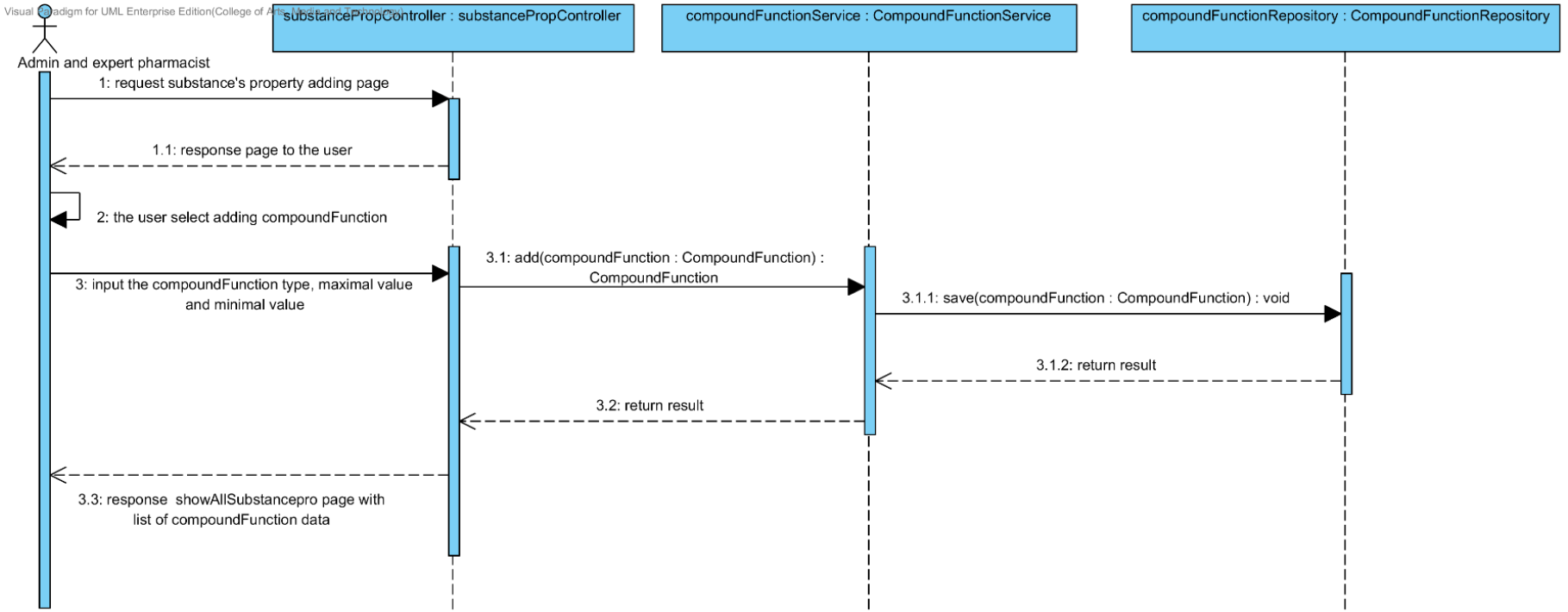
4.1.1.9 SQD-09I: The user add a new drug substance property into the system (Flowability).



4.1.1.10 SQD-09J: The user add a new drug substance property into the system (Density).



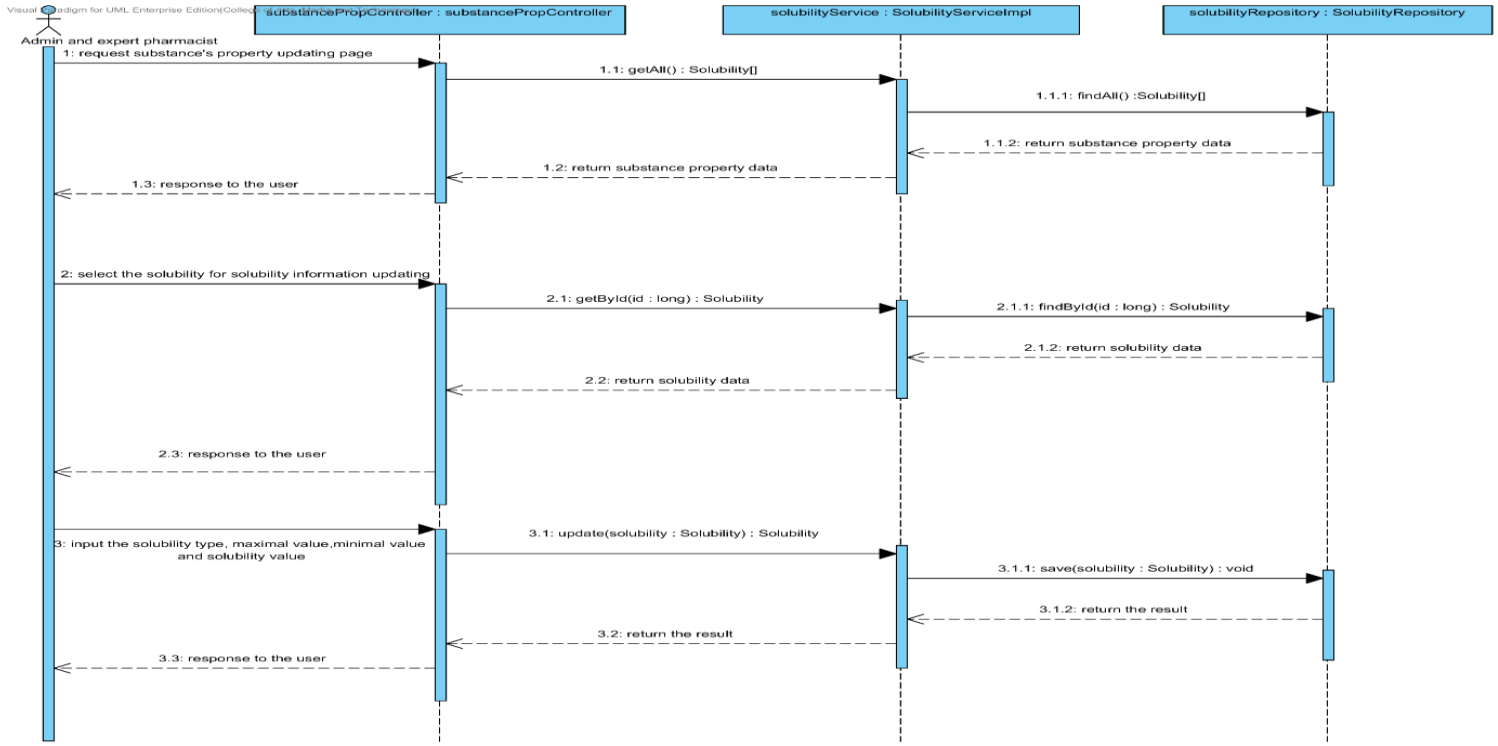
4.1.1.11 SQD-09K: The user add a new drug substance property into the system (Compound function).



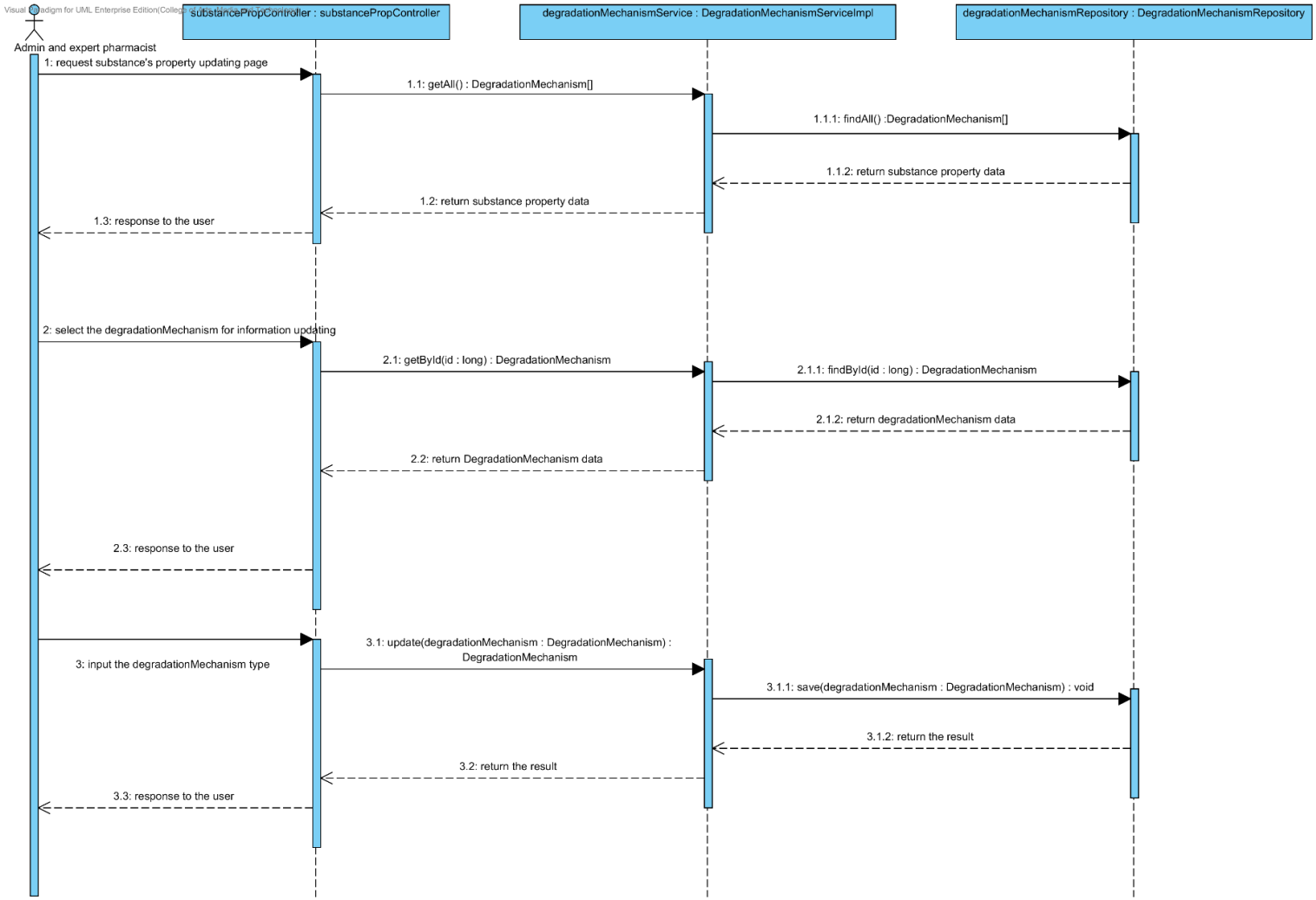
4.1.2 URS-10: The user updates an existing substance property into the system.

In the sequence diagram, the user can updates an existing substance property to the system. Firstly, the user opens the substance property updating page. The system shows all substance property data to the user, then the user selects the substance property for making substance property updating, after that, the user inputs substance property value such as name, max value min value and substance's property value. The substance property controller gets the data from the user, after that the controller chooses appropriate SubstanceProperty's service for updating the existing substance property to the system. Finally, the substance property controller shows a substance property that already update on the successful adding substance property page to the user.

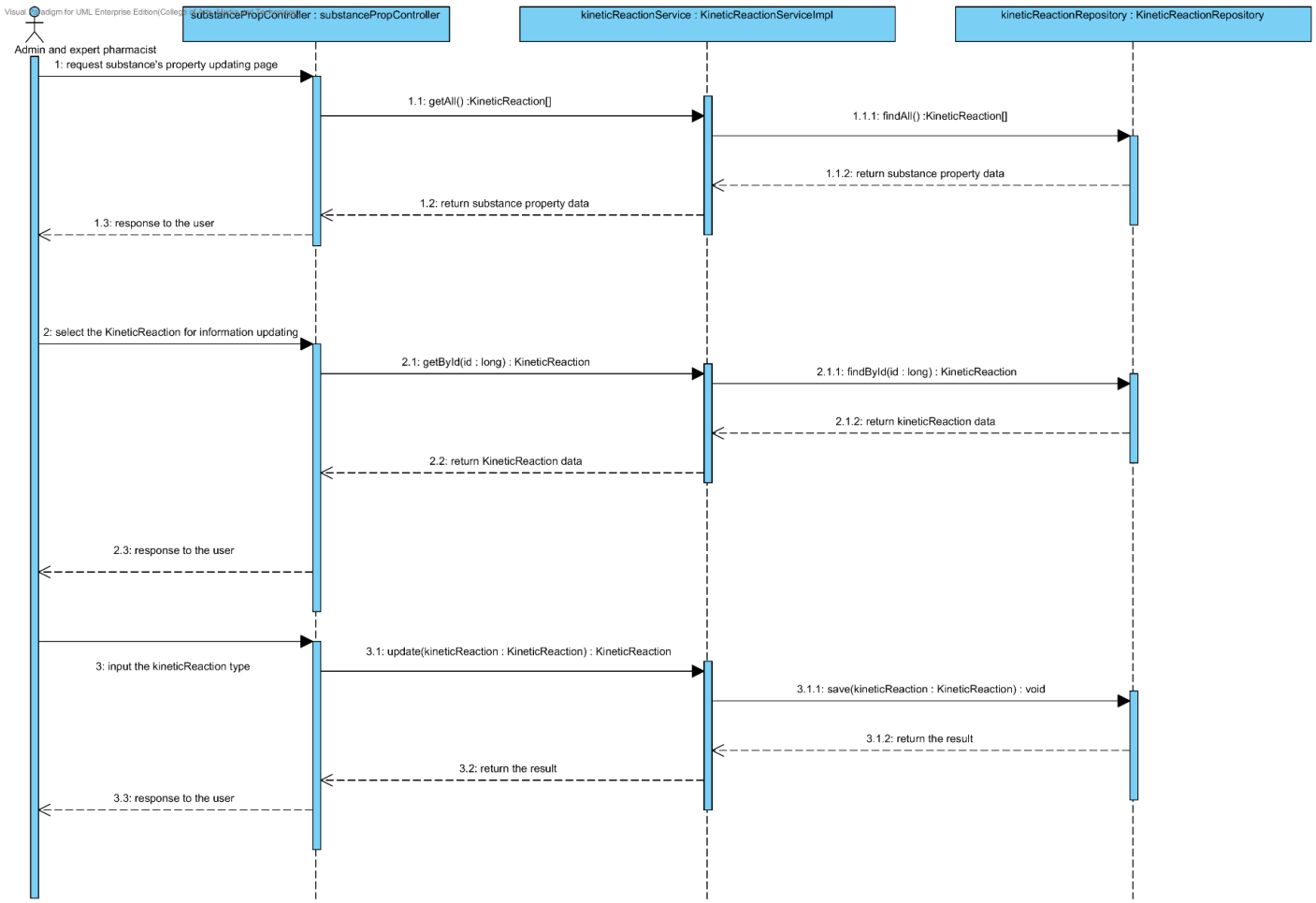
4.1.2.1 SQD-10A: The user updates an existing substance's property into the system (Solubility).



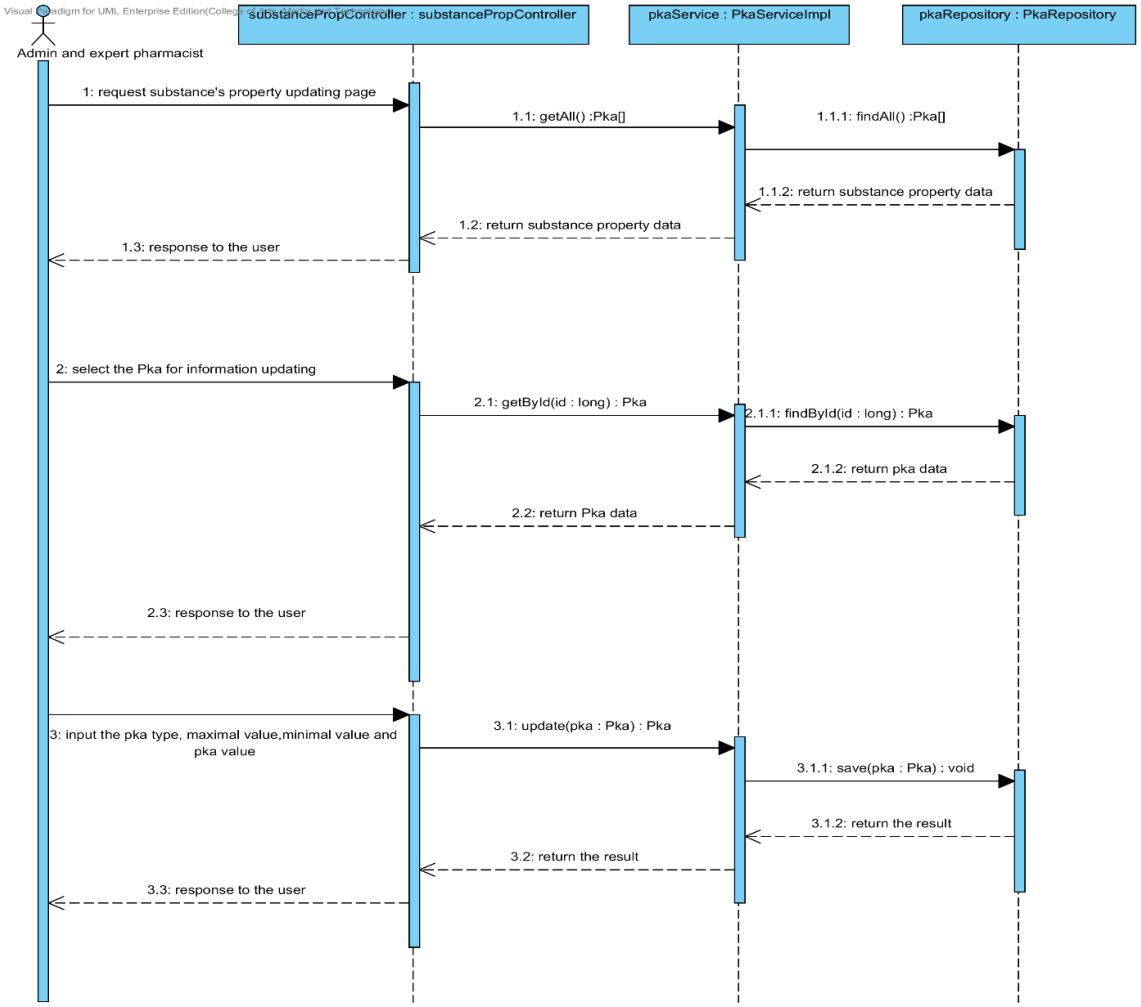
4.1.2.2 SQD-10B: The user updates an existing substance property into the system (DegradationMechanism).



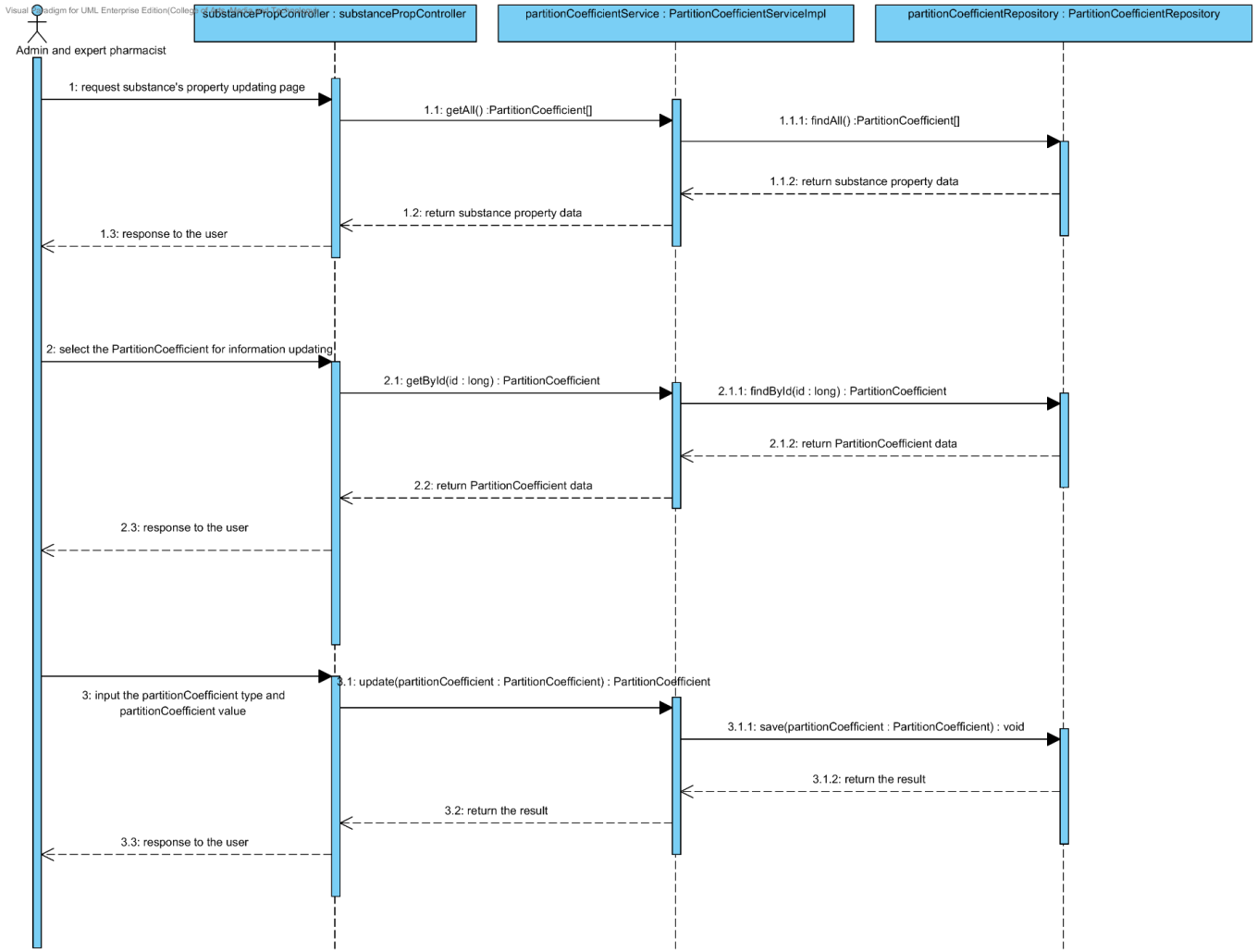
4.1.2.3 SQD-10C: The user updates an existing substance property into the system (Kinetic Reaction).



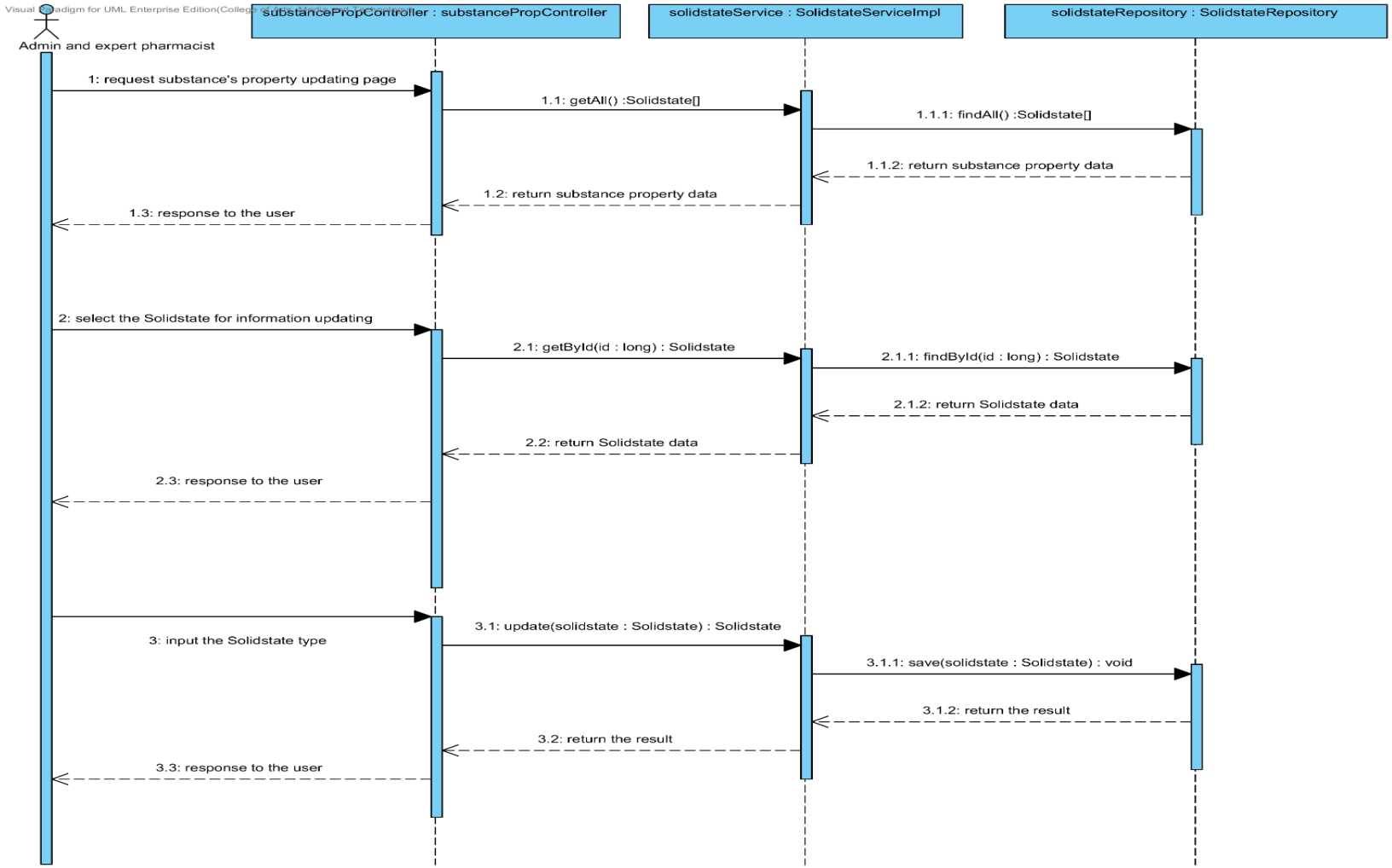
4.1.2.4 SQD-10D: The user updates an existing substance property into the system (Pka).



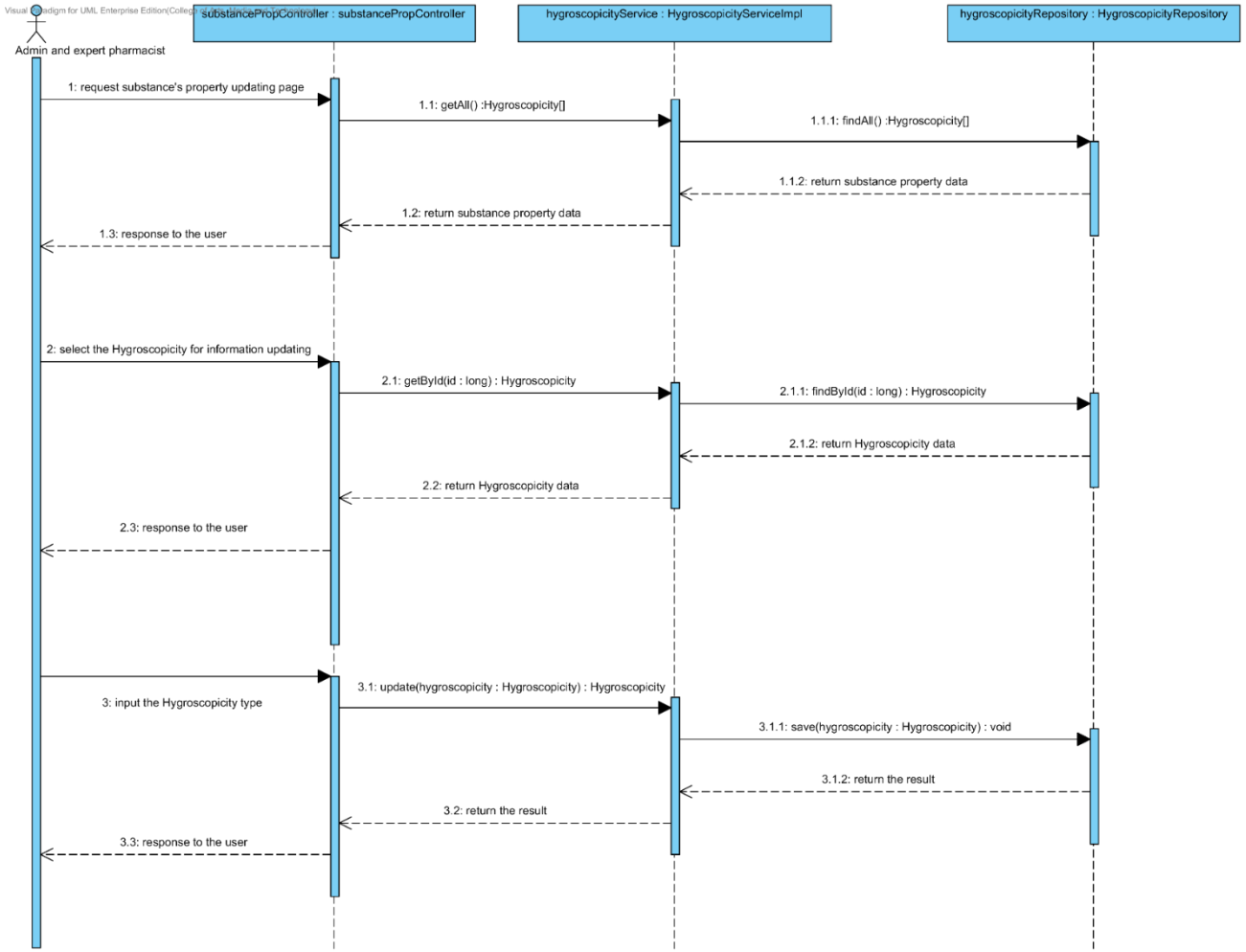
4.1.2.5 SQD-10E: The user updates an existing substance property into the system (PartitionCoefficient).



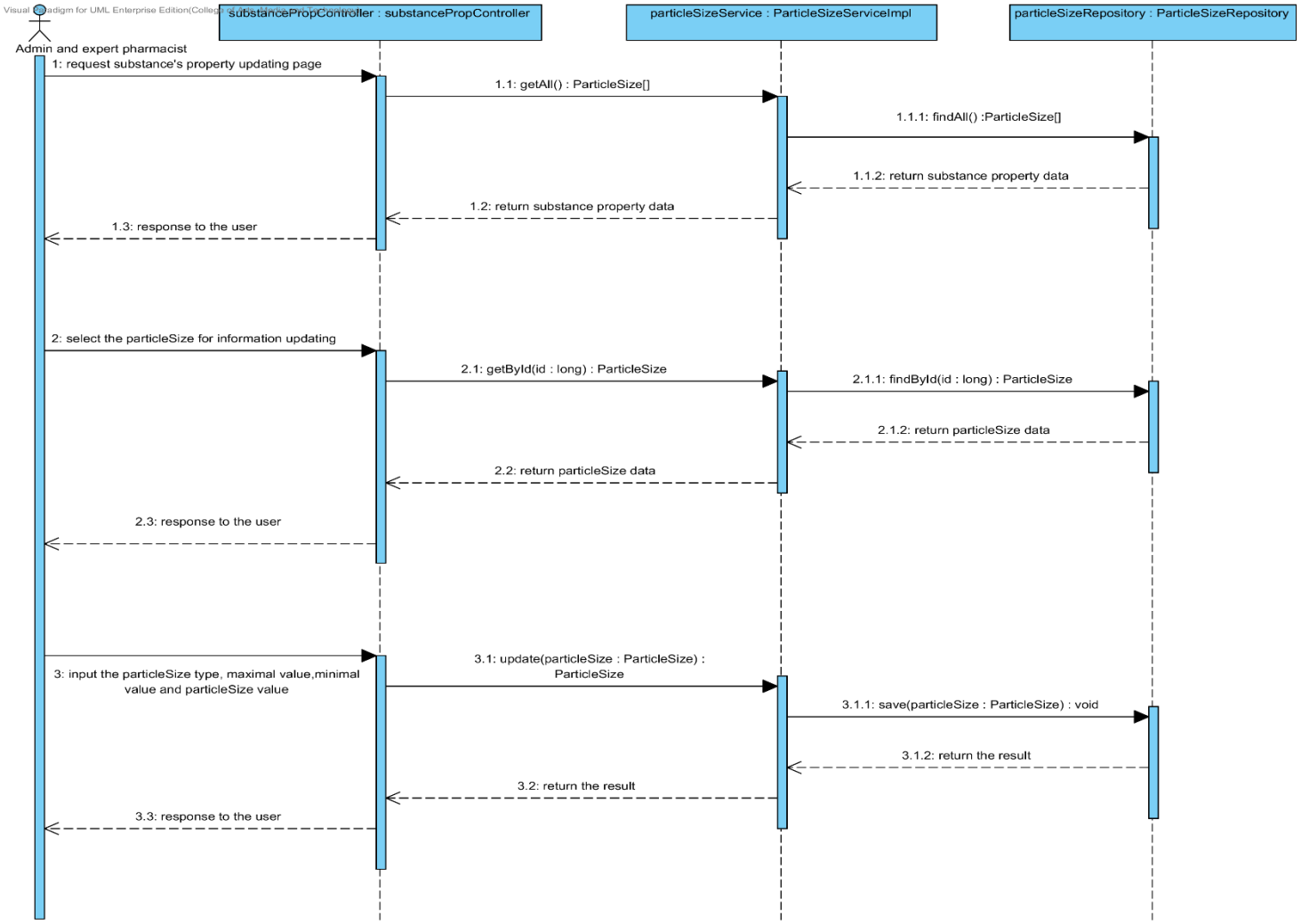
4.1.2.6 SQD-10F: The user updates an existing substance property into the system (Solidstate).



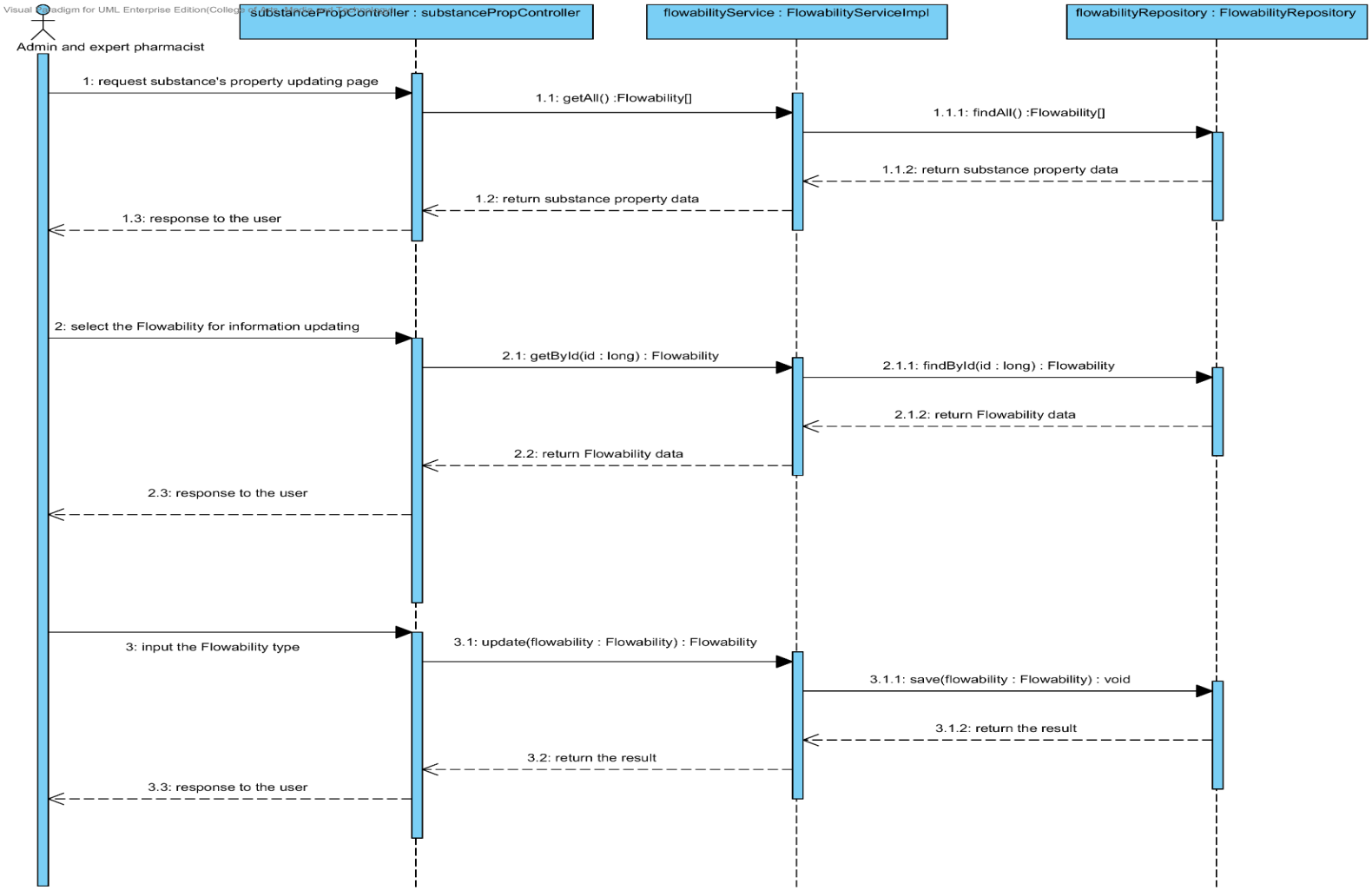
4.1.2.7 SQD-10G: The user updates an existing substance property into the system (Hygroscopicity).



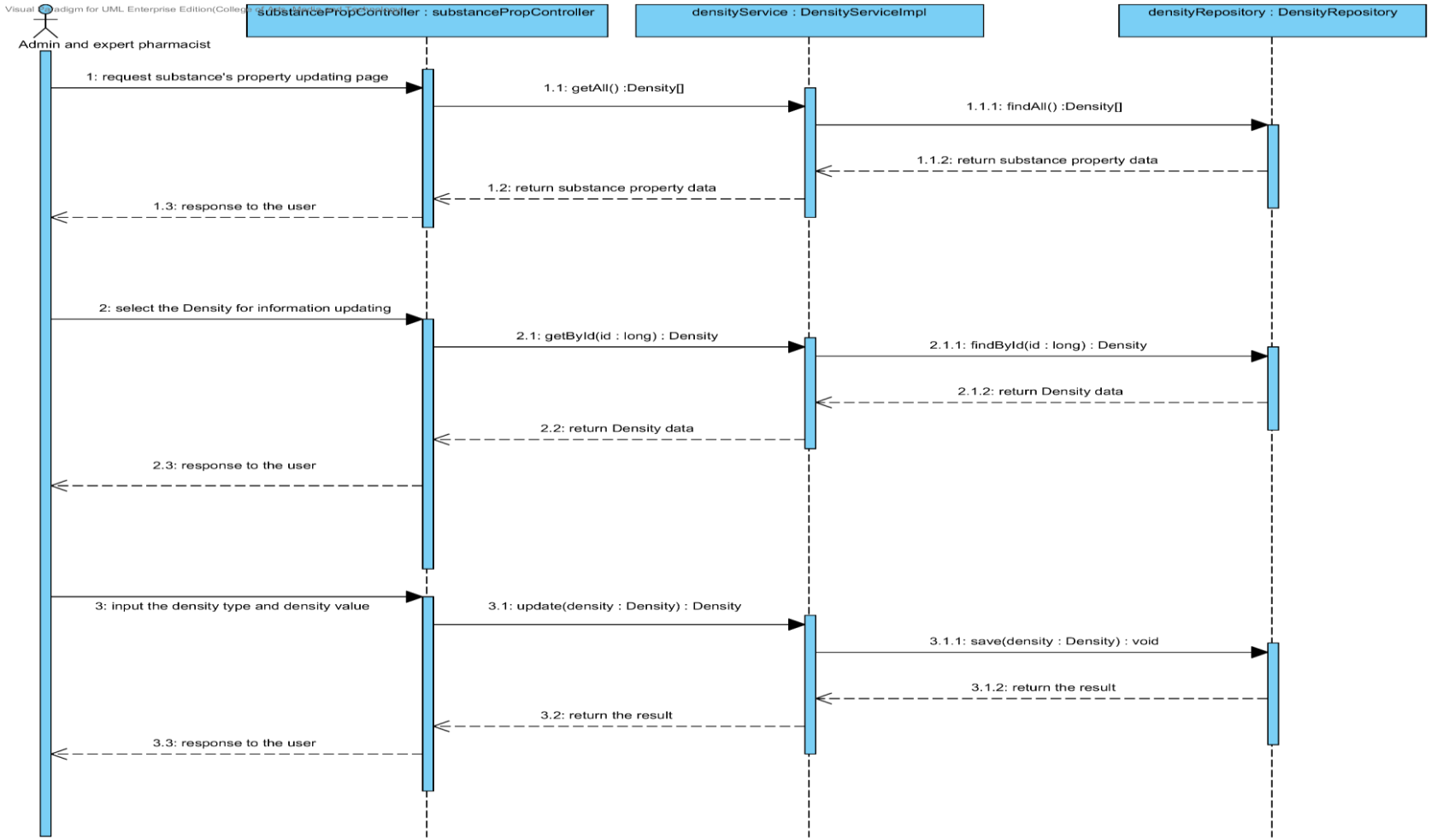
4.1.2.8 SQD-10H: The user updates an existing substance property into the system (ParticleSize).



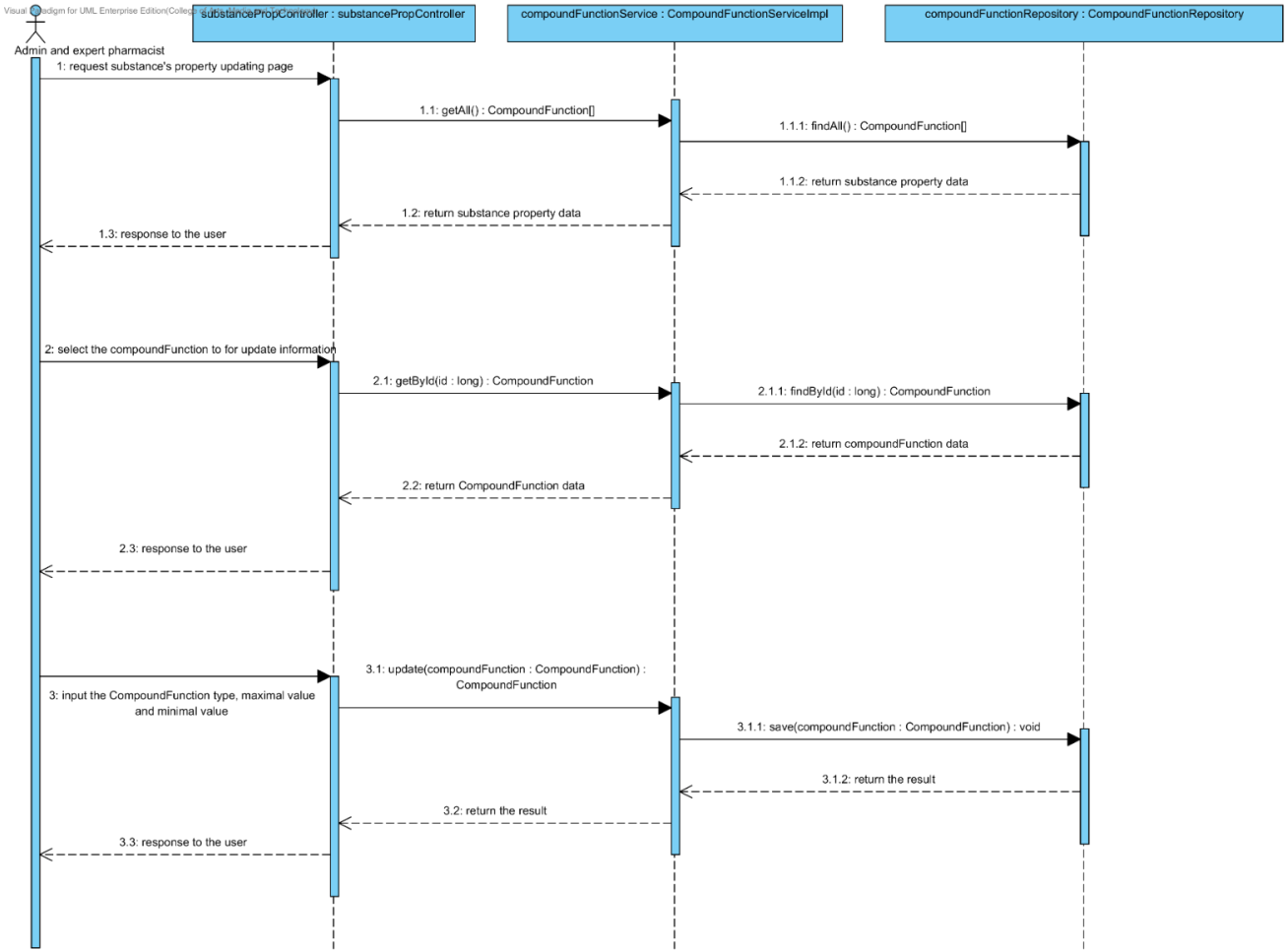
4.1.2.9 SQD-10I: The user updates an existing substance property into the system (Flowability).



4.1.2.10 SQD-10J: The user updates an existing substance property into the system (Density).



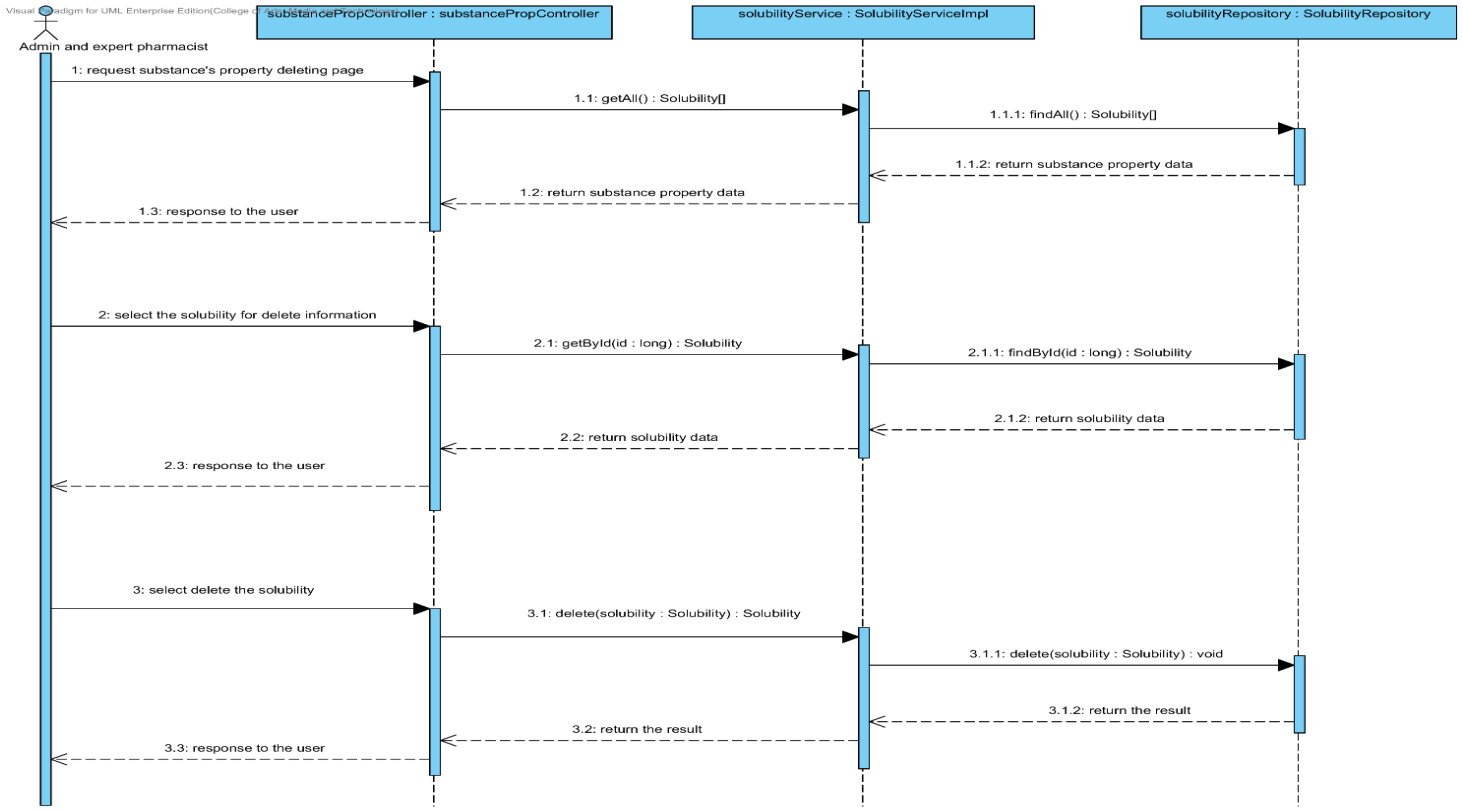
4.1.2.11 SQD-10K: The user updates an existing substance property into the system (Compound function).



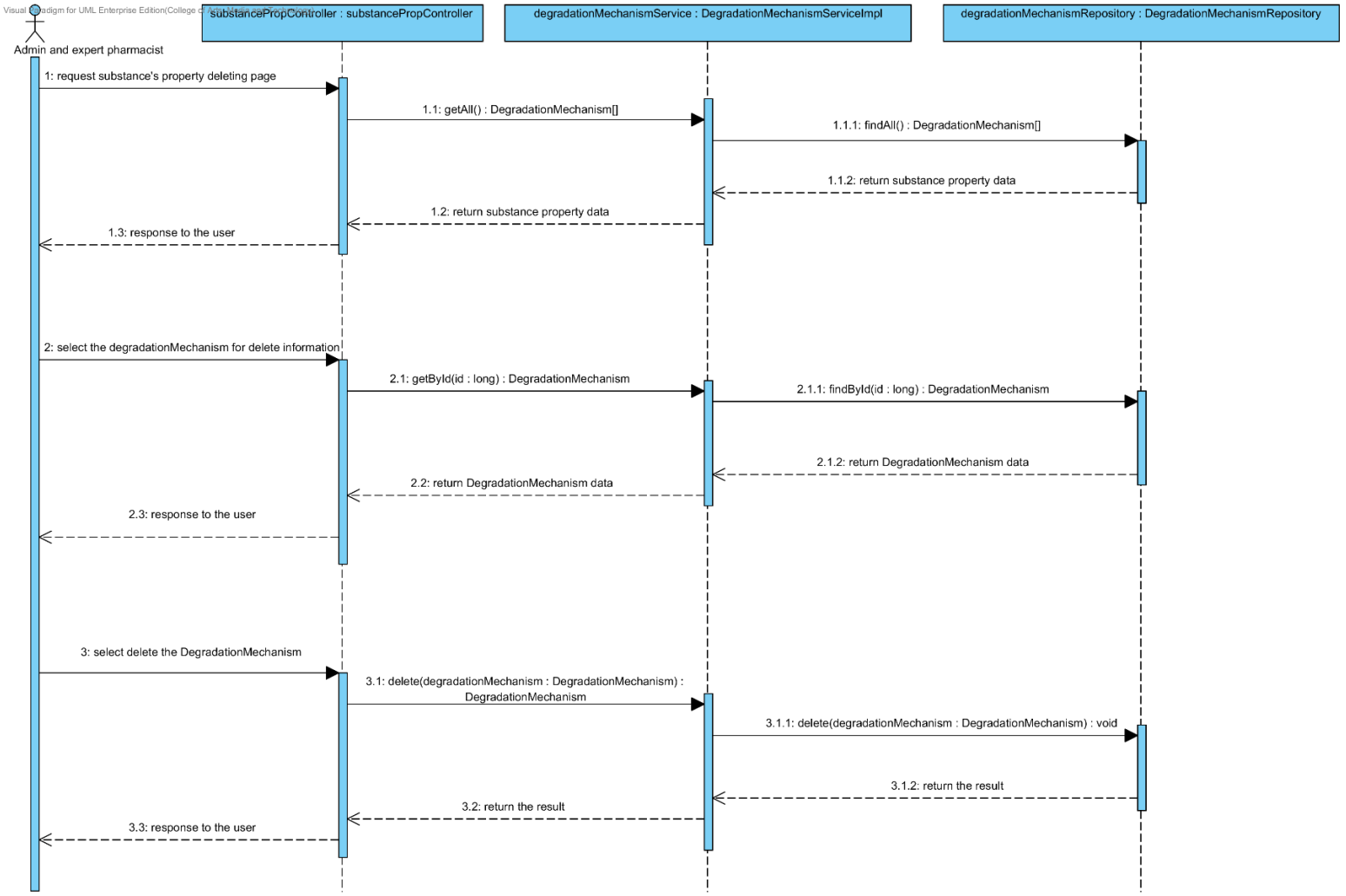
4.1.3 URS-11: The user deletes an existing substance property from the system.

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

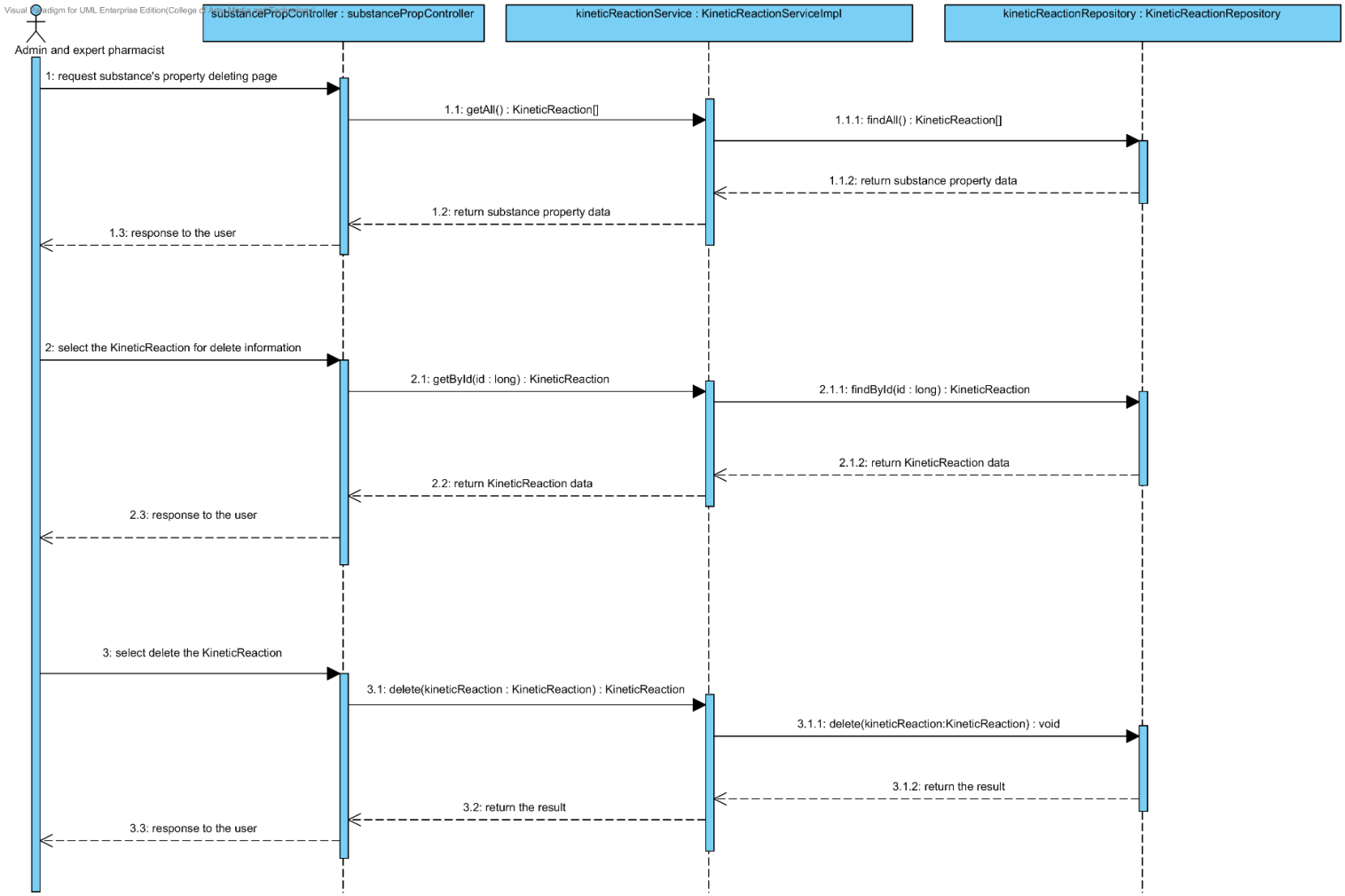
4.1.3.1 SQD-11A: The user deletes an existing substance property from the system (Solubility).



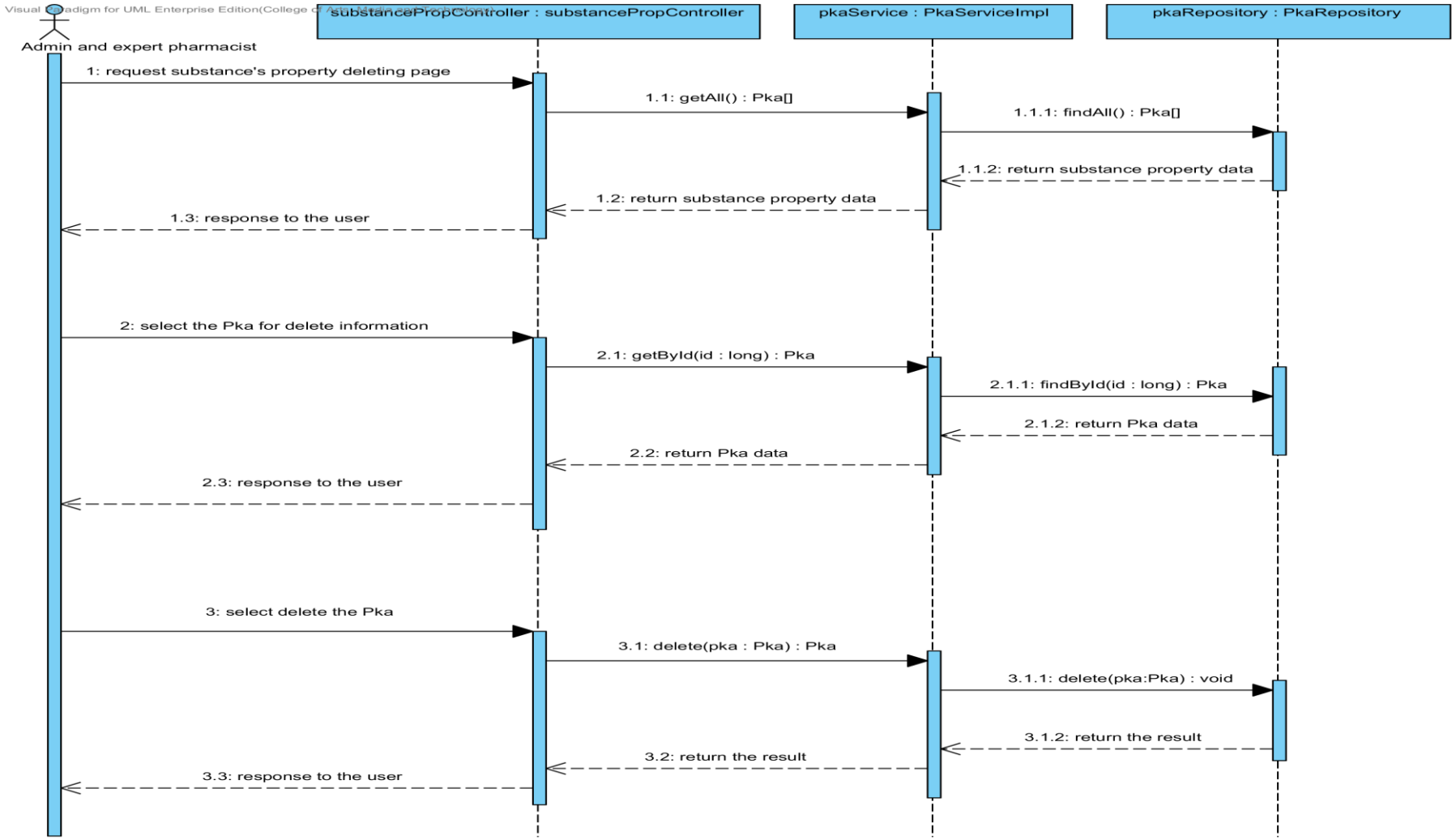
4.1.3.2 SQD-11B: The user deletes an existing substance property from the system (DegradationMechanism).



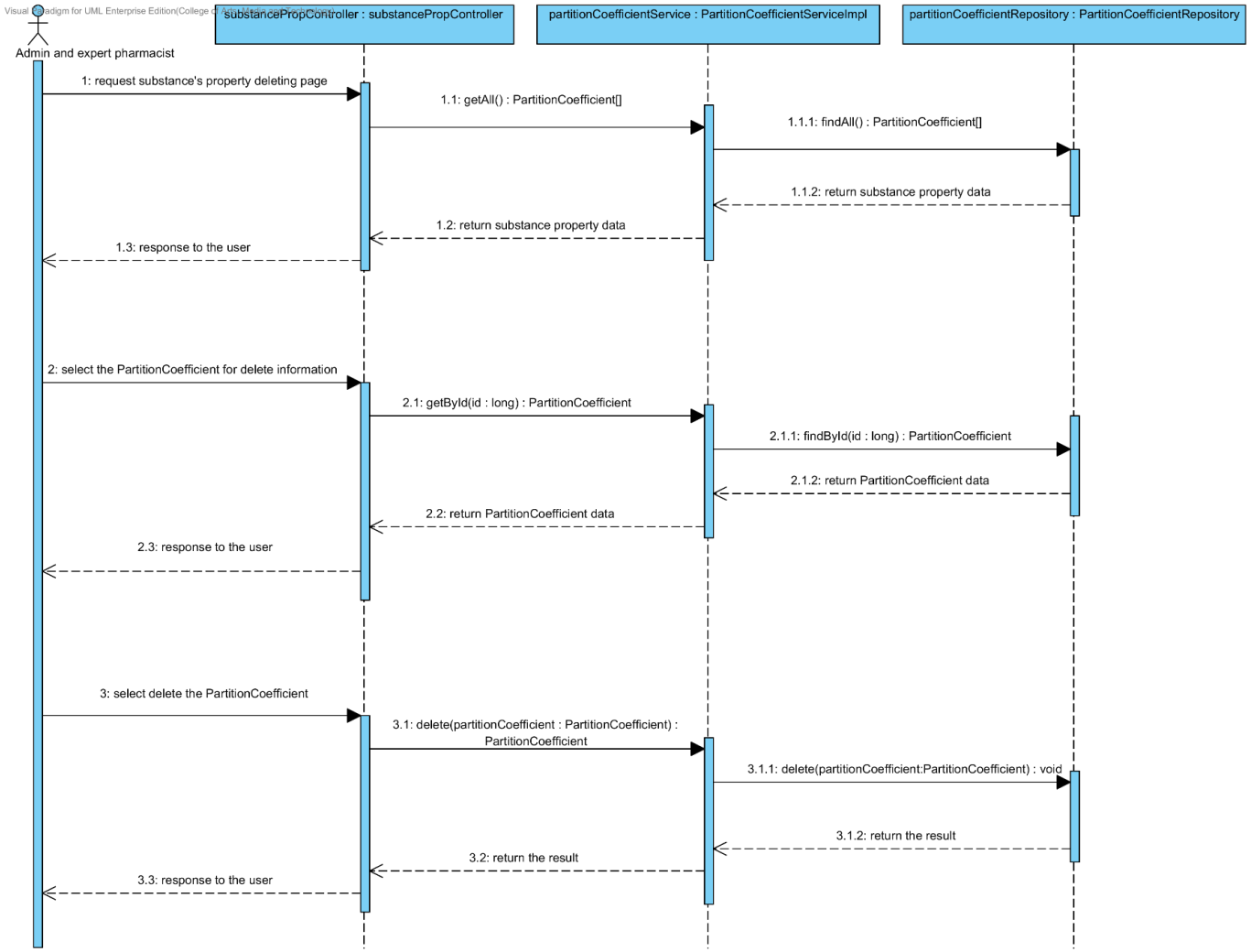
4.1.3.3 SQD-11C: The user deletes an existing substance property into the system (Kinetic Reaction).



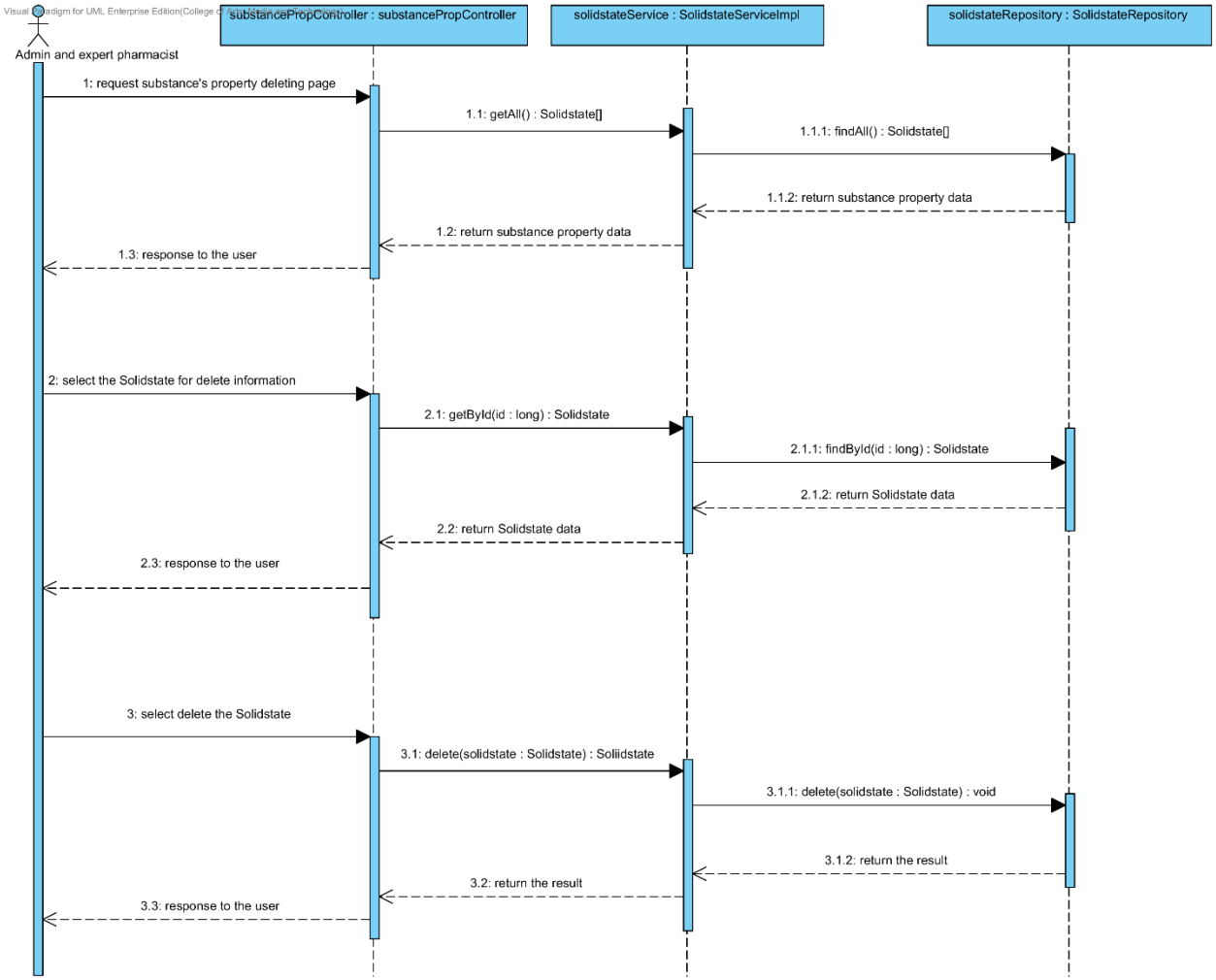
4.1.3.4 SQD-11D: The user deletes an existing substance property from the system (Pka).



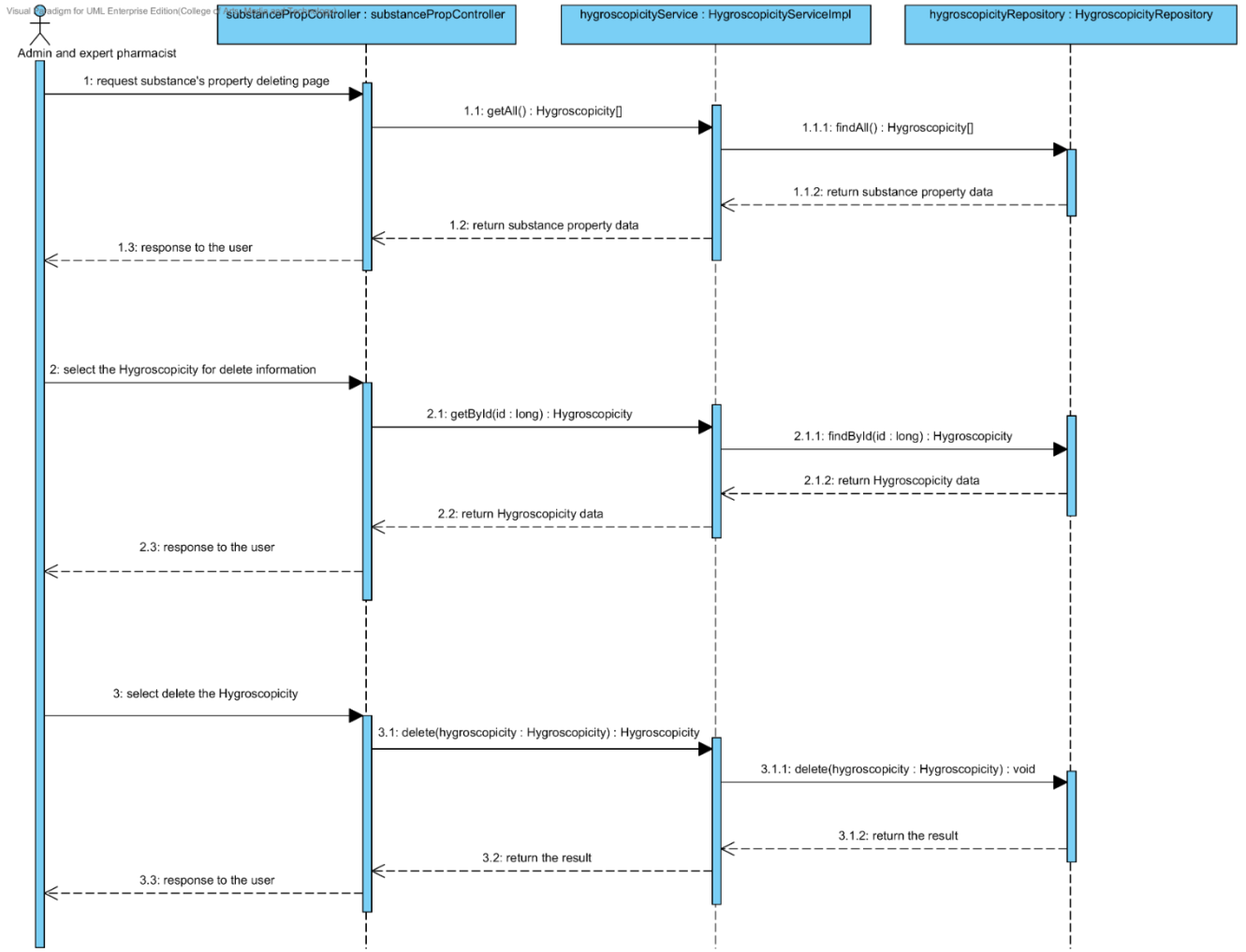
4.1.3.5 SQD-11E: The user deletes an existing substance property from the system (PartitionCoefficient).



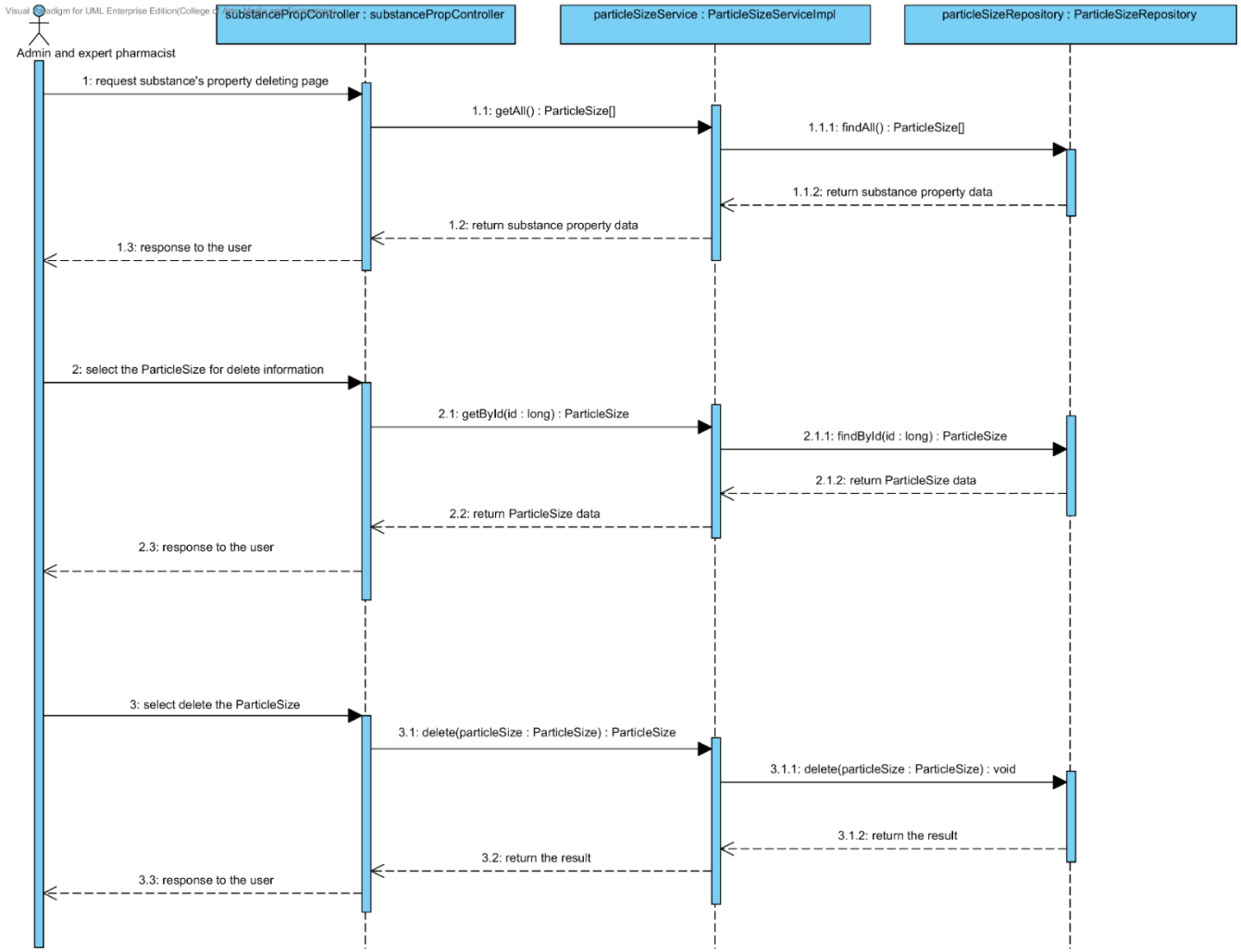
4.1.3.6 SQD-11F: The user deletes an existing drug substance property from the system (Solidstate).



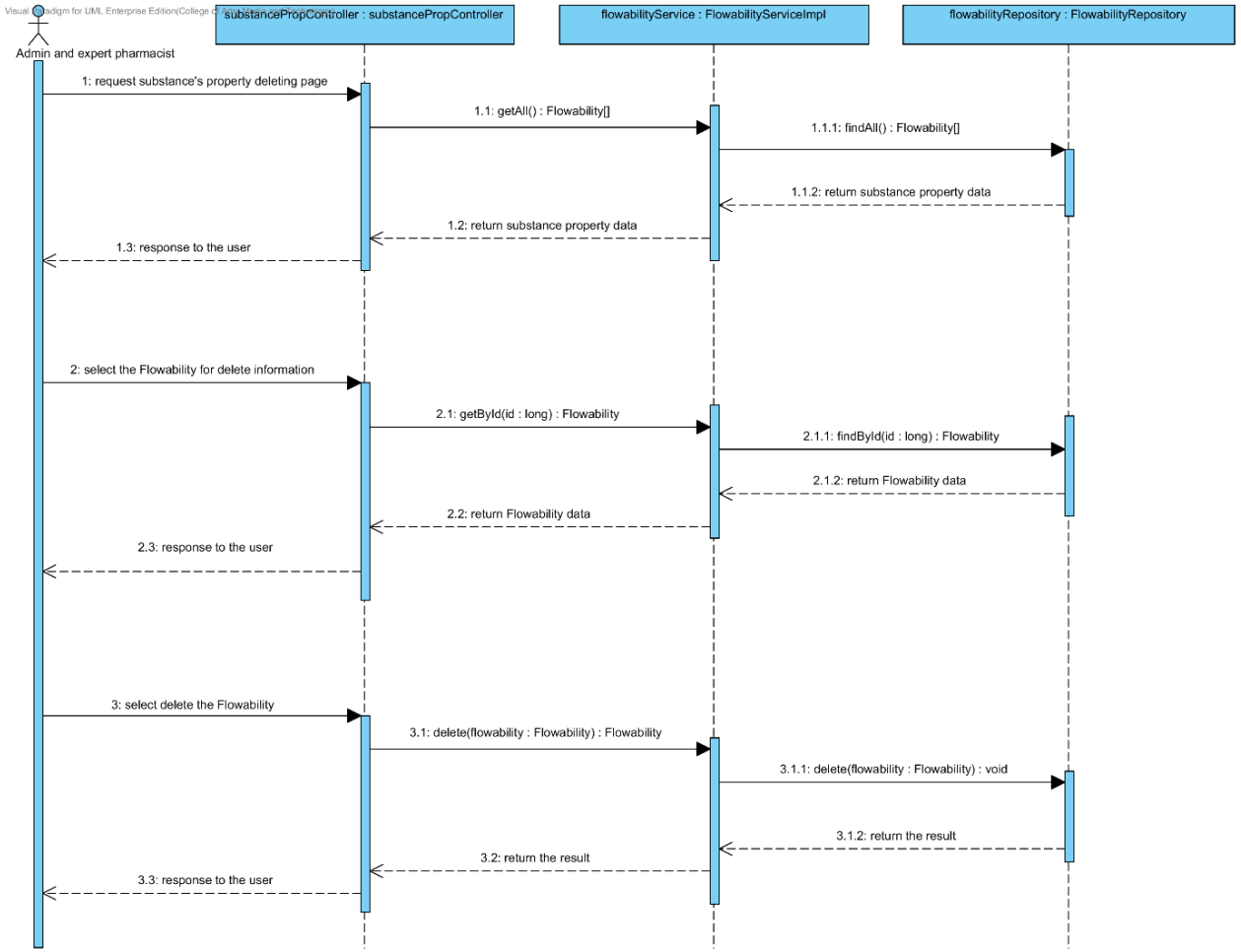
4.1.3.7 SQD-11G: The user deletes an existing substance property from the system (Hygroscopicity).



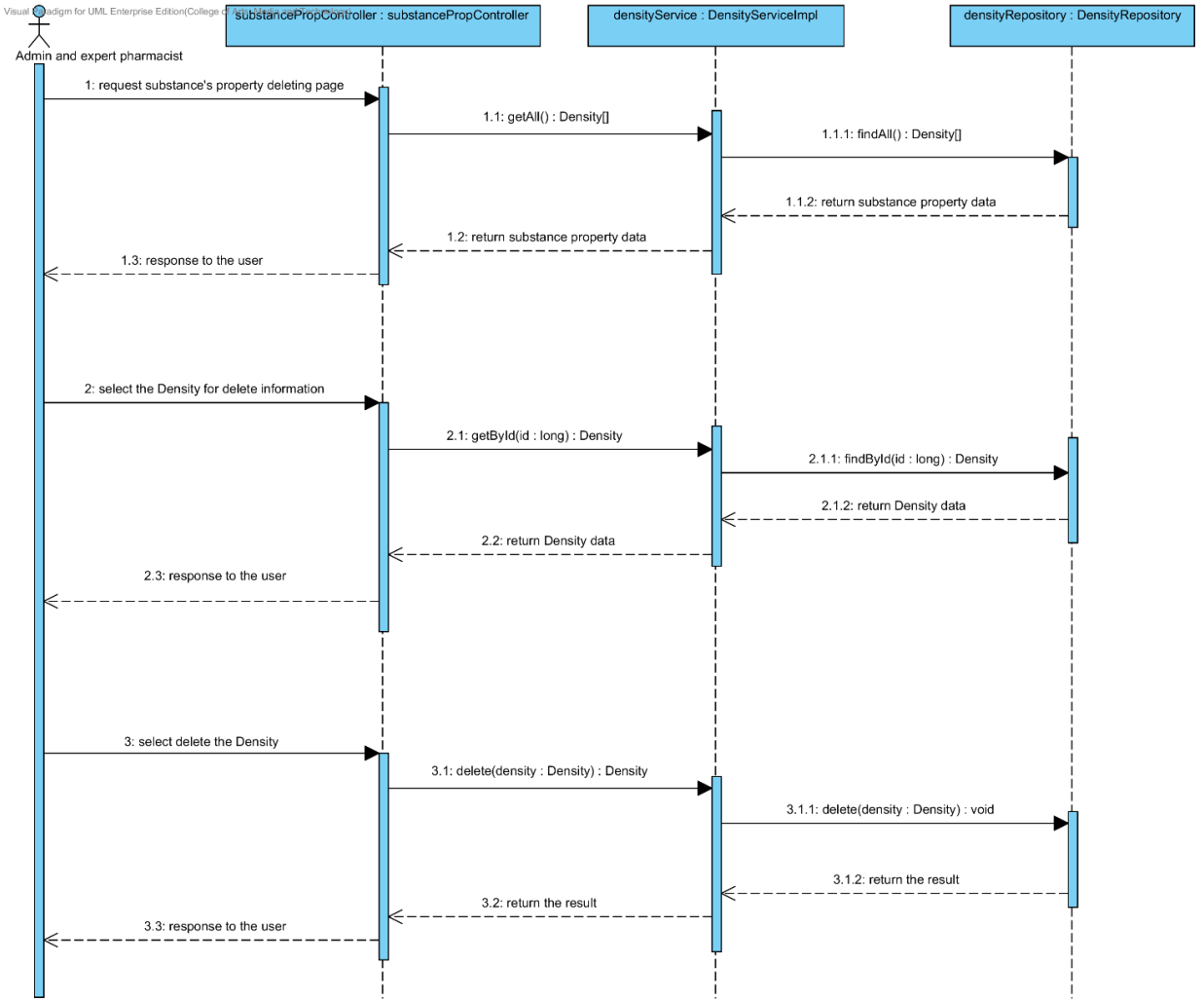
4.1.3.8 SQD-11H: The user deletes an existing substance property from the system (ParticleSize).



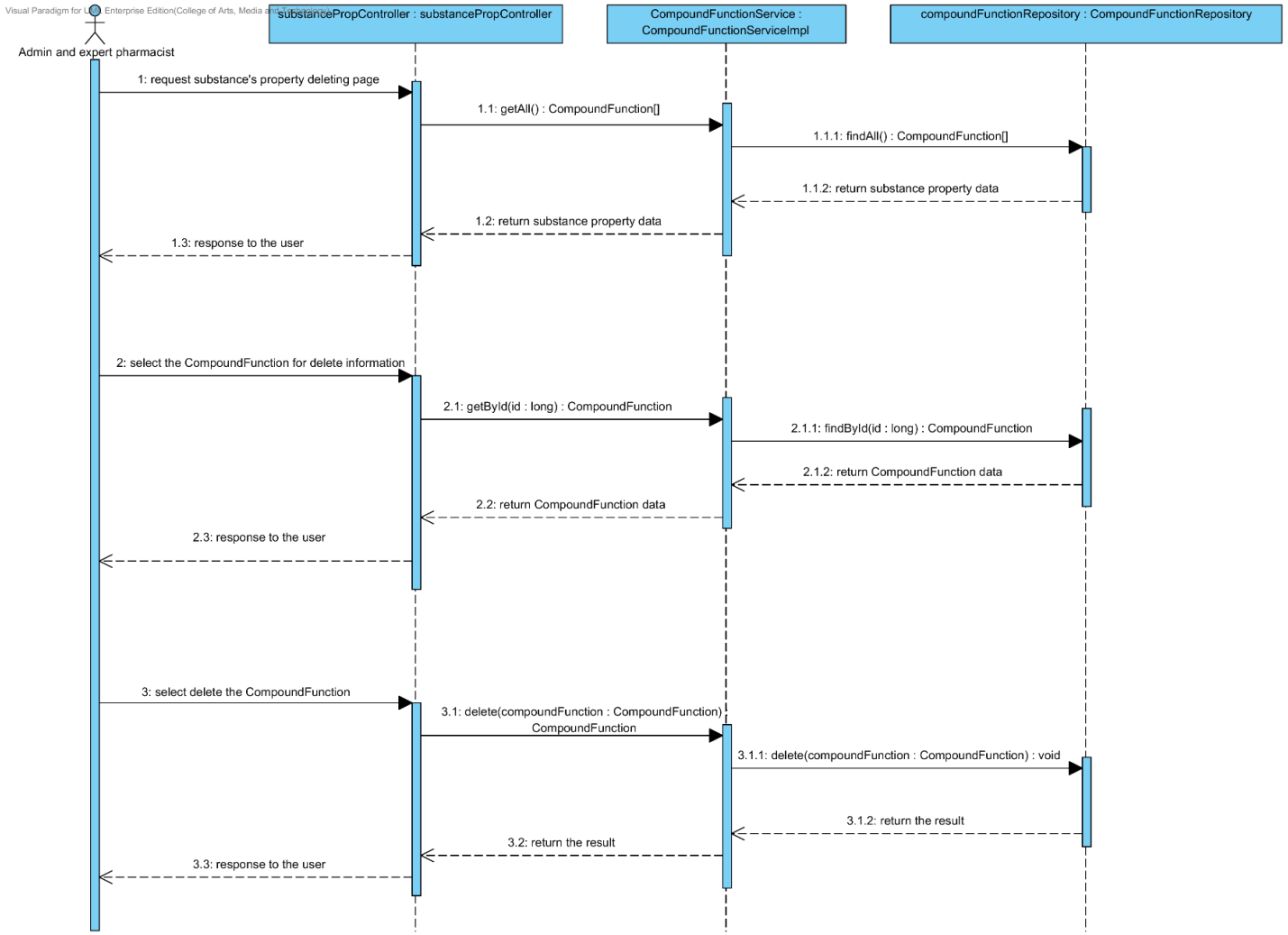
4.1.3.9 SQD-11I: The user deletes an existing substance property from the system (Flowability).



4.1.3.10 SQD-11J: The user deletes an existing substance property from the system (Density).



4.1.3.11 SQD-11K: The user deletes an existing substance property from the system (Compound function).

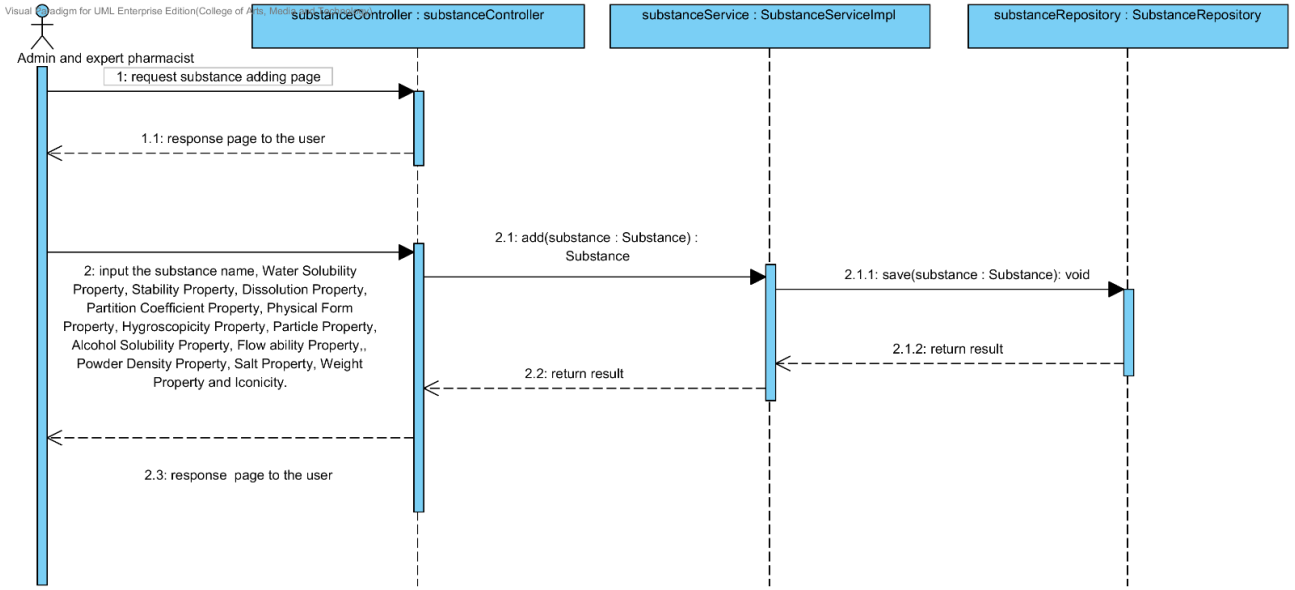


4.2- Sub-Feature 6: Manage the drug substance

4.2.1 URS-12: 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, substance property, weight property and lonicity. 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.

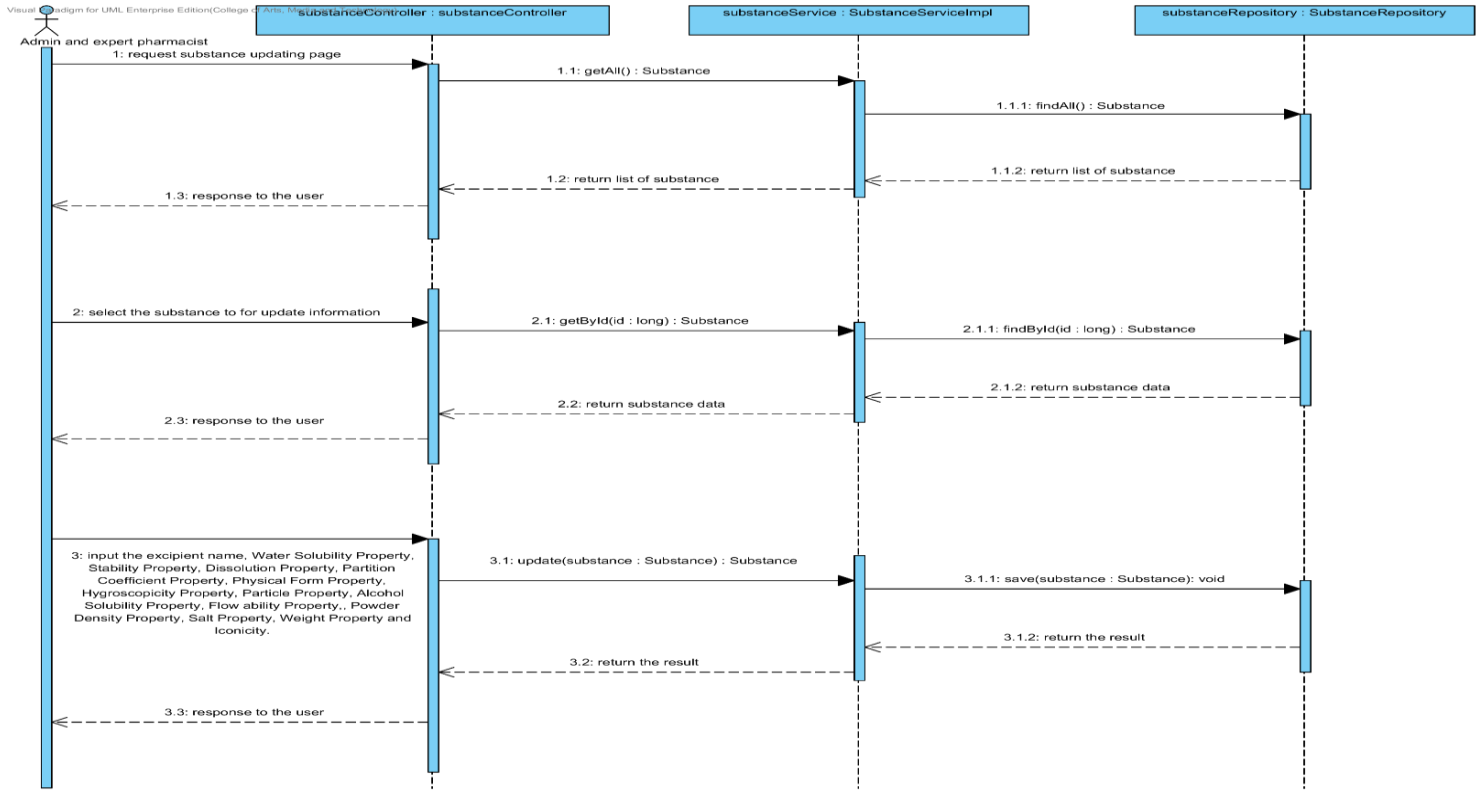
4.2.1.1 SQD-12: The user adds a new substance to the system.



4.2.2 URS-13: 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 substance property, weight property and lonicity. 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 adding successful page.

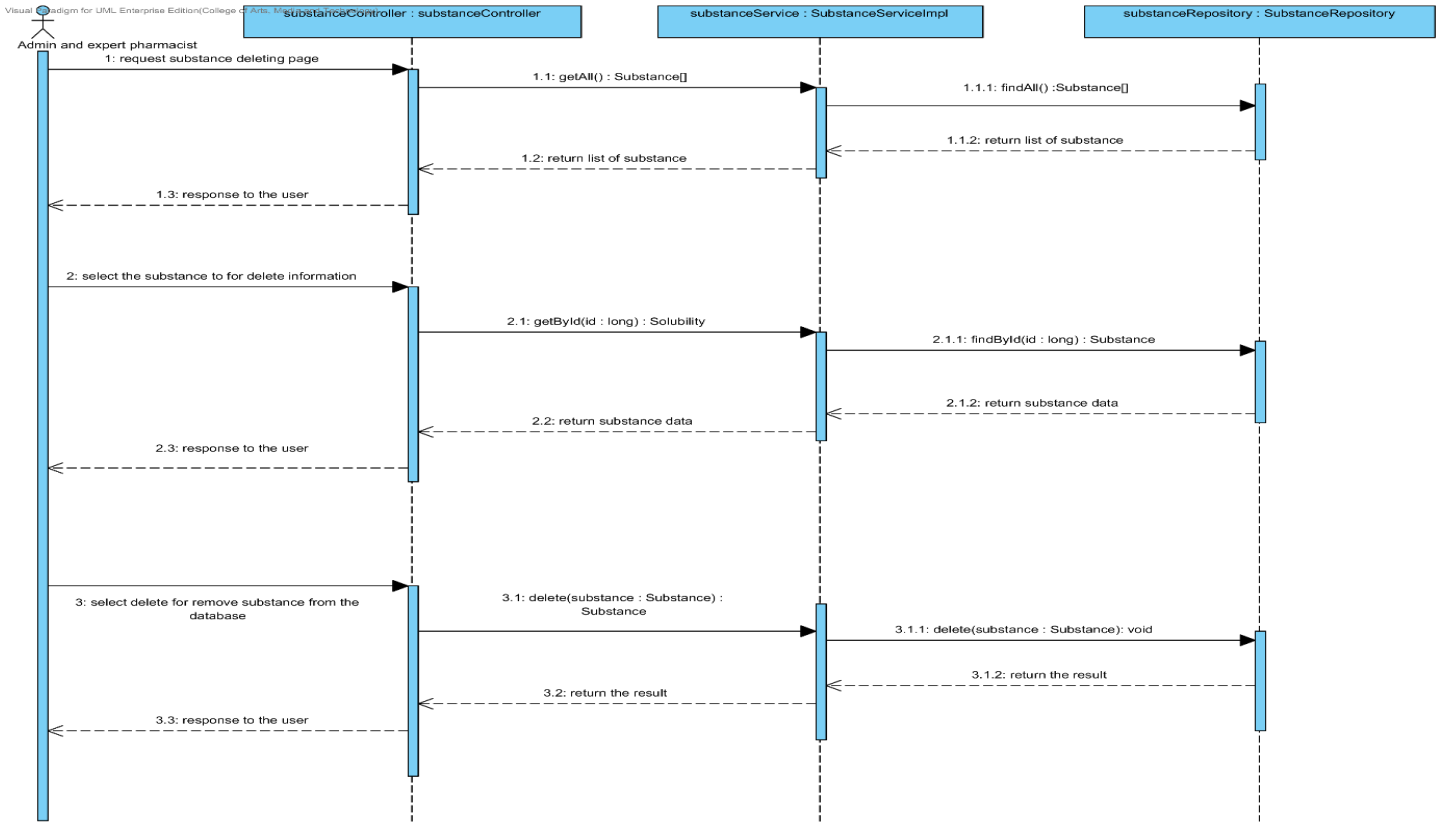
4.2.2.1 SQD-13: The user updates an existing substance in the system.



4.2.3 URS-14: 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.

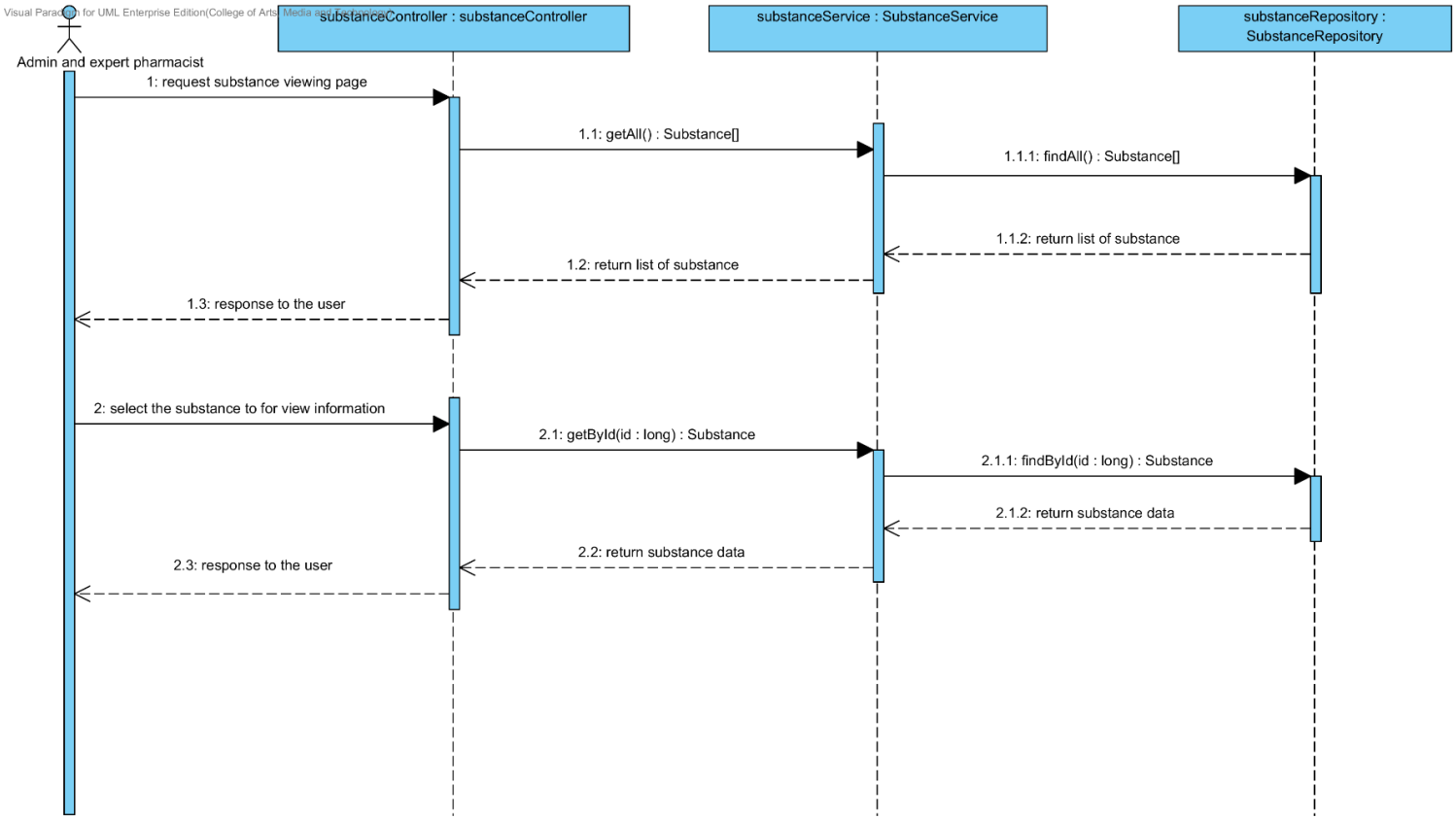
4.2.3.1 SQD-14: The user deletes an existing substance from the system.



4.2.4 URS-15: 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.

4.2.4.1 SQD-15: The user views the substance in the system.

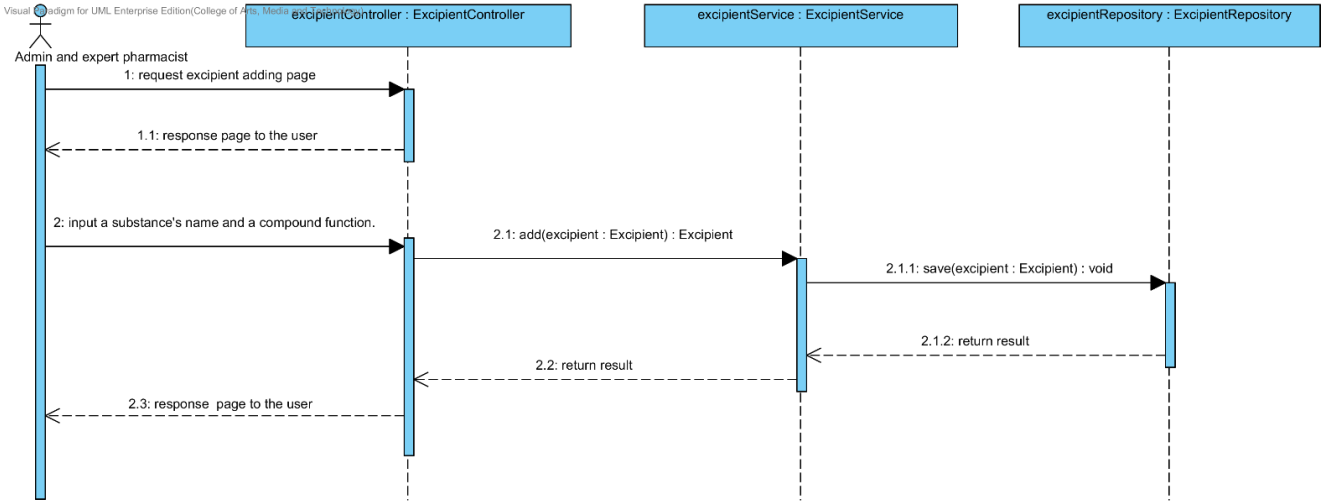


4.3- Sub-Feature 7: Manage the drug substance

4.3.1 URS-16: 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 property and compound function. 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.

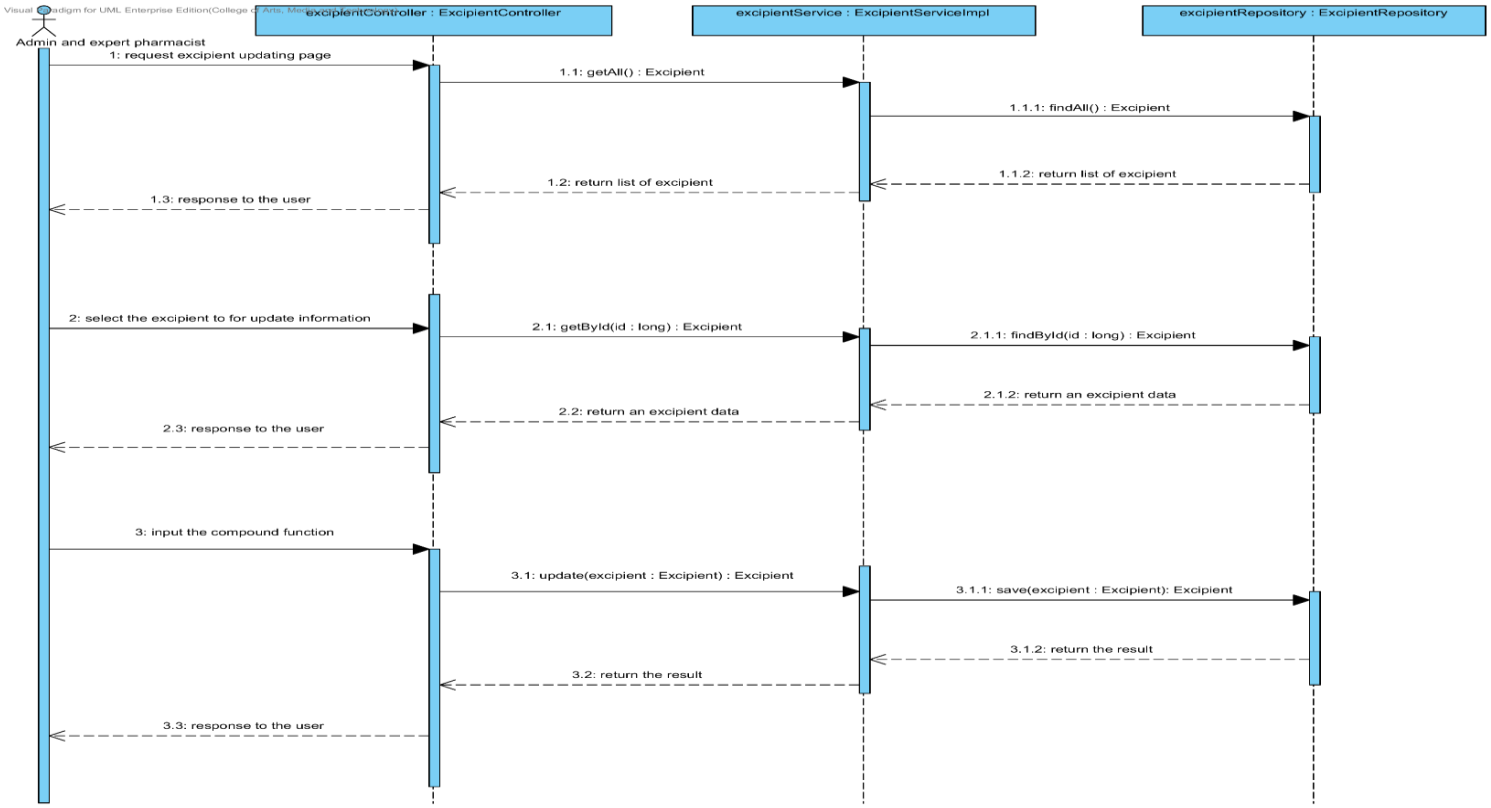
4.3.1.1 SQD-16: The user adds a new excipient to the system.



4.3.2 URS-17: 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 property and compound 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.

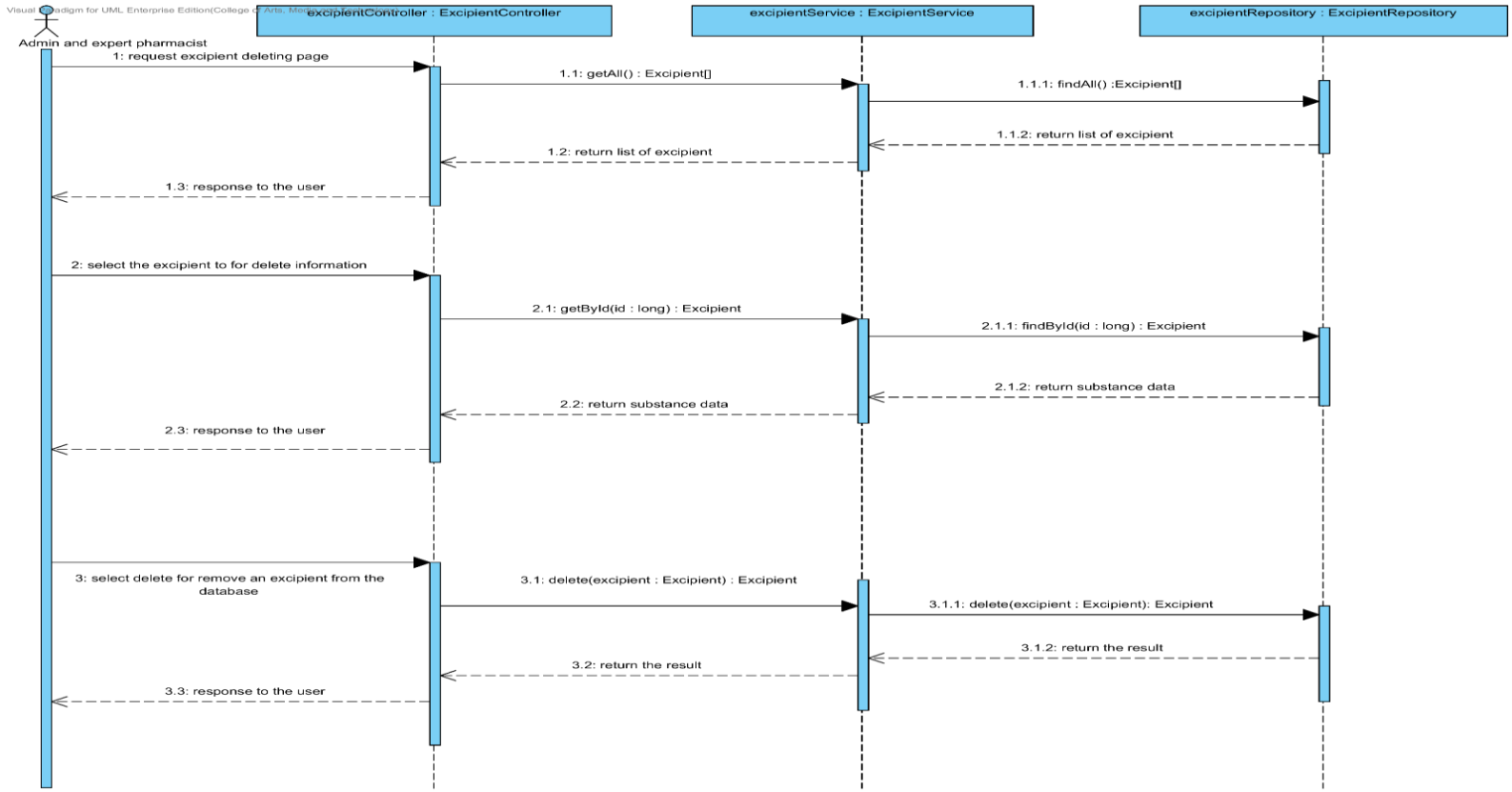
4.3.2.1 SQD-17: The user updates an existing excipient in the system



4.3.3 URS-18: 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.

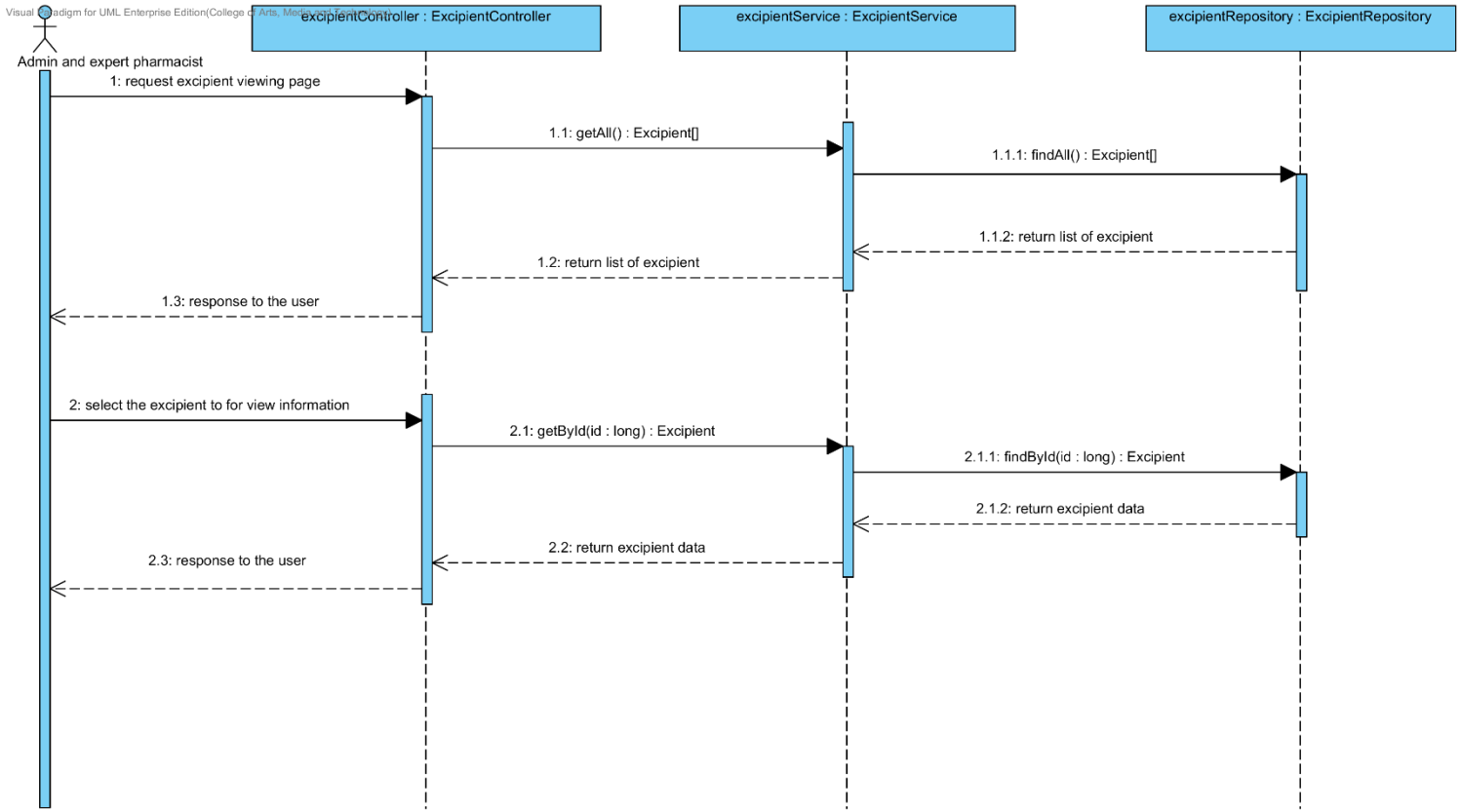
4.3.3.1 SQD-18: The user deletes an existing excipient from the system.



4.3.4 URS-19: 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.

4.3.4.1 SQD-19: The user views the excipient in the system.

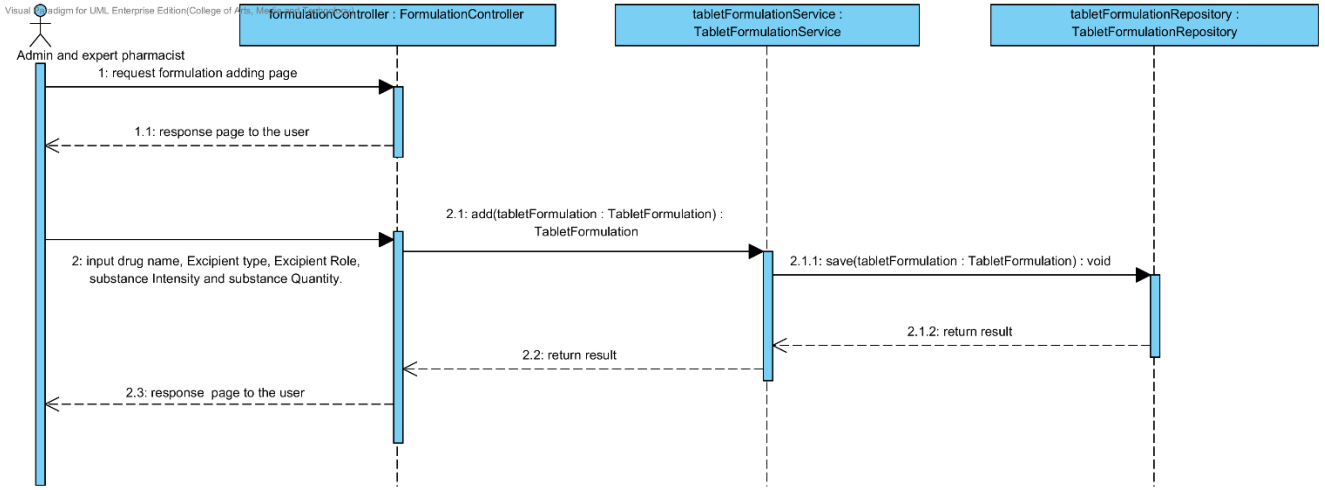


4.4- Sub-Feature 8: Manage the drug formulation

4.4.1 URS-20: 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 data such as name, excipient, substance quantity and substance intensity. 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.

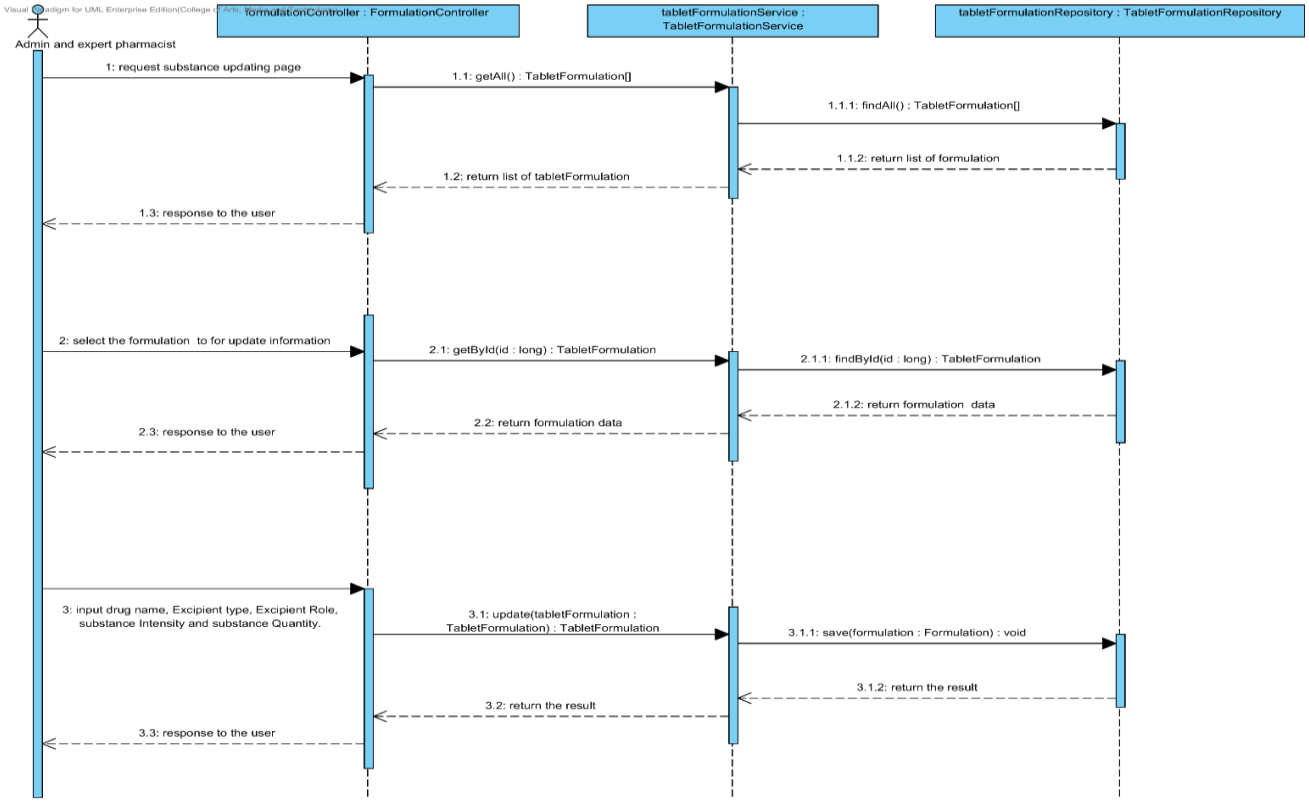
4.4.1.1 SQD-20: The user adds a new drug's formulation to the system.



4.4.2 URS-21: 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 name, excipient, substance quantity and substance intensity. 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.

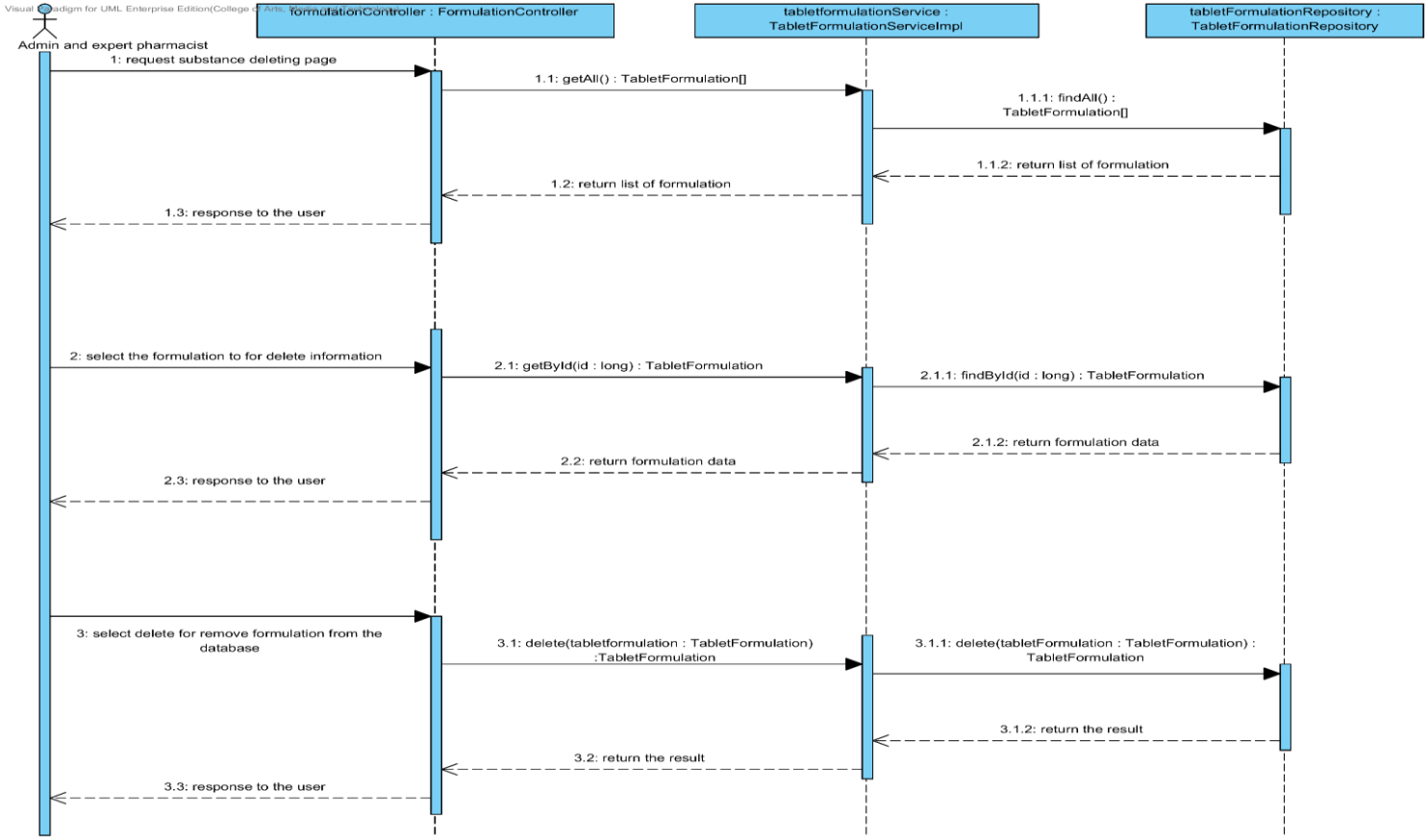
4.4.2.1 SQD-21: The user updates an existing drug's formulation in the system.



4.4.3 URS-22: 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.

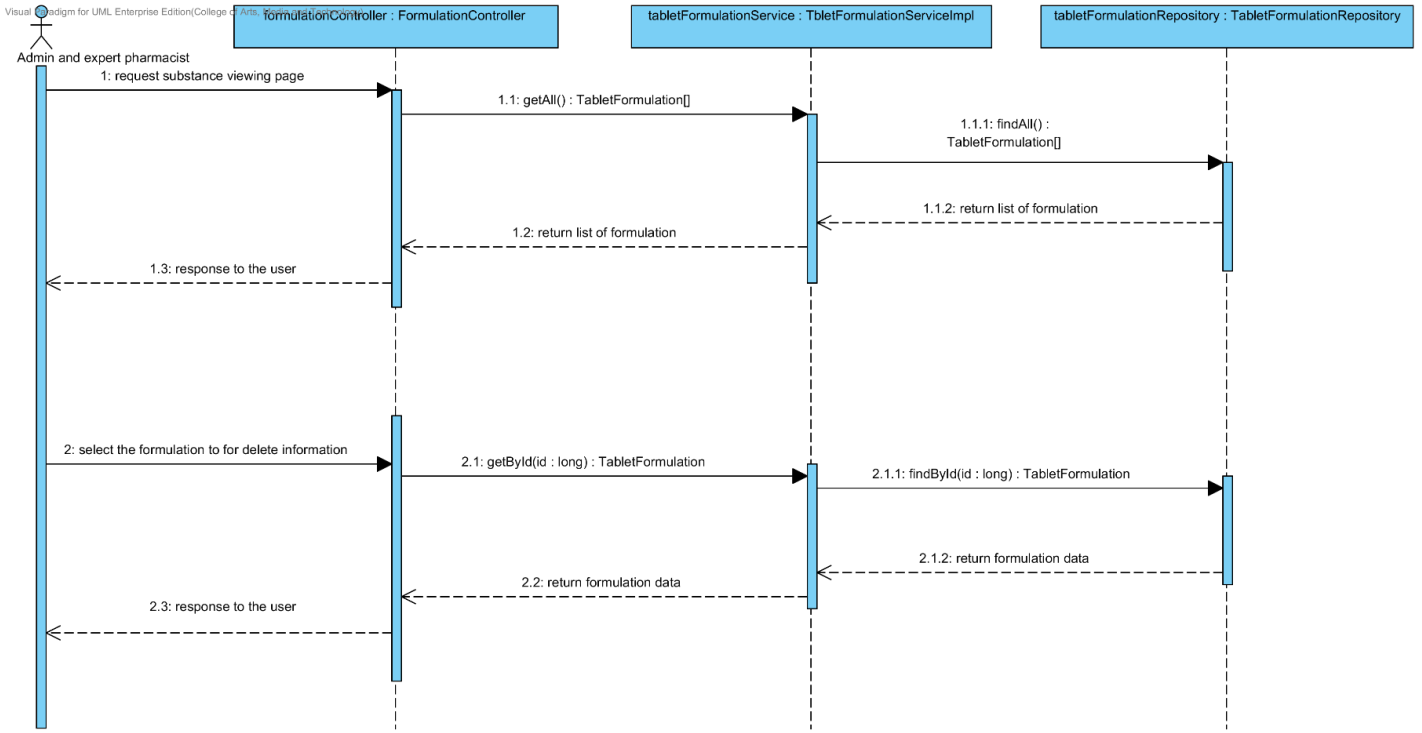
4.4.3.1 SQD-22: The user deletes an existing drug's formulation from the system.



4.4.4 URS-23: 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.

4.4.4.1 SQD-23: The user views the drug's formulation in the system.



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.

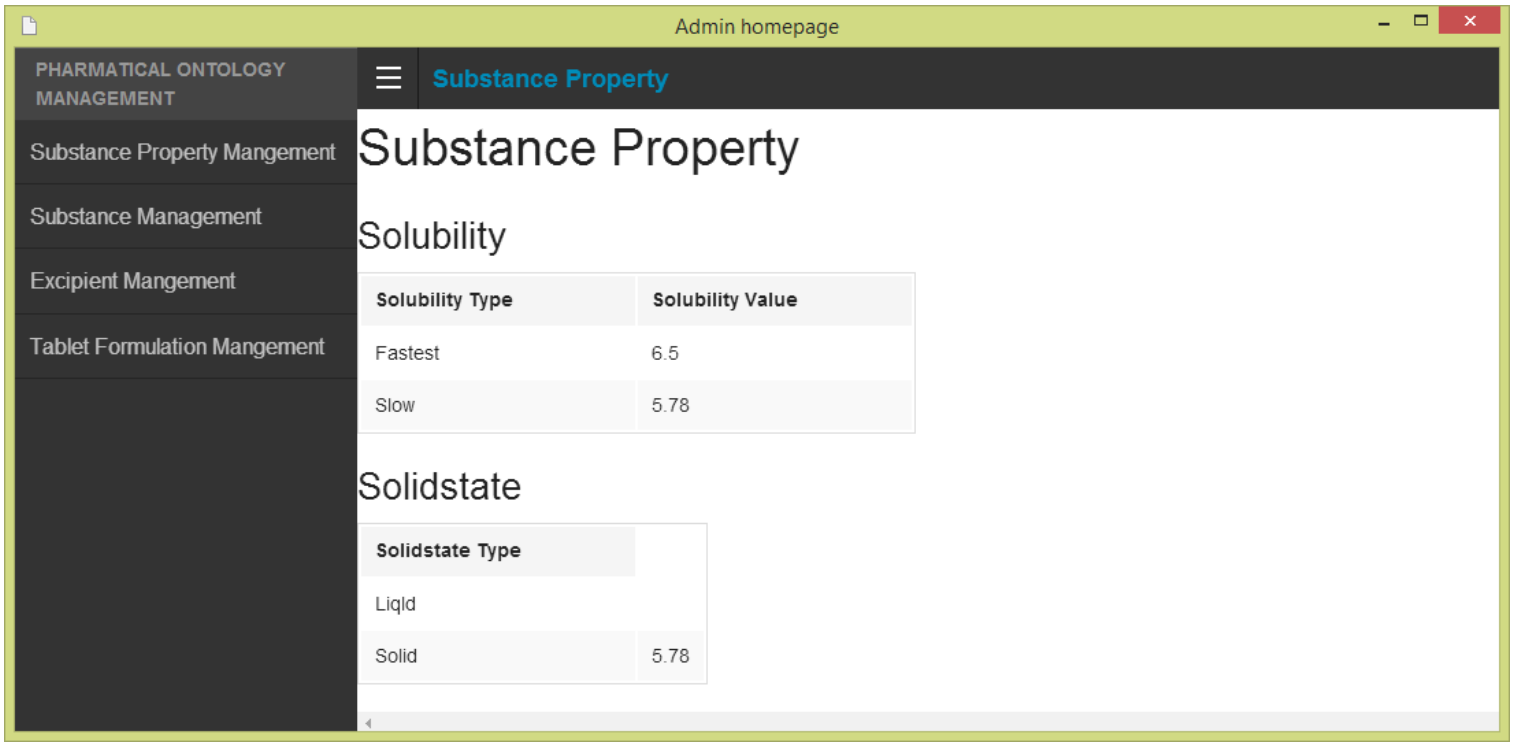


Figure 68- UI-01: Admin main menu

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.

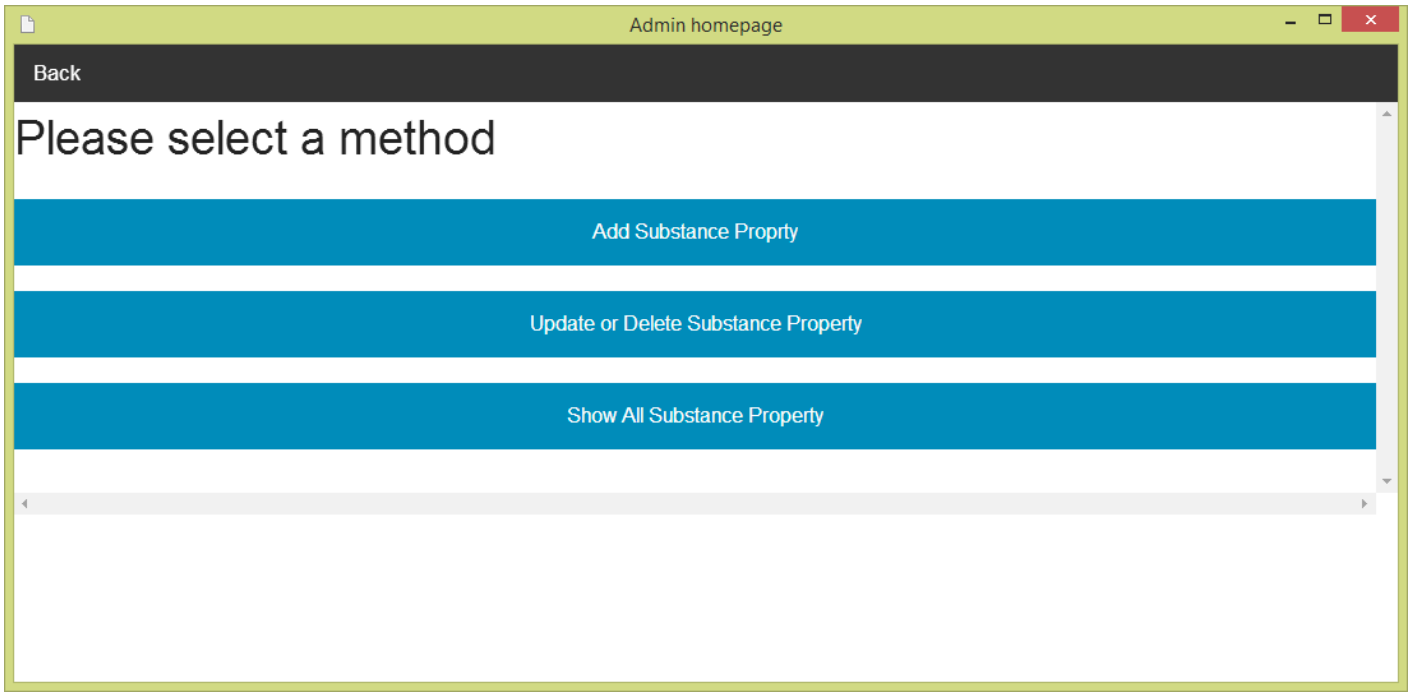


Figure 69- UI-02: Substance property management menu

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.

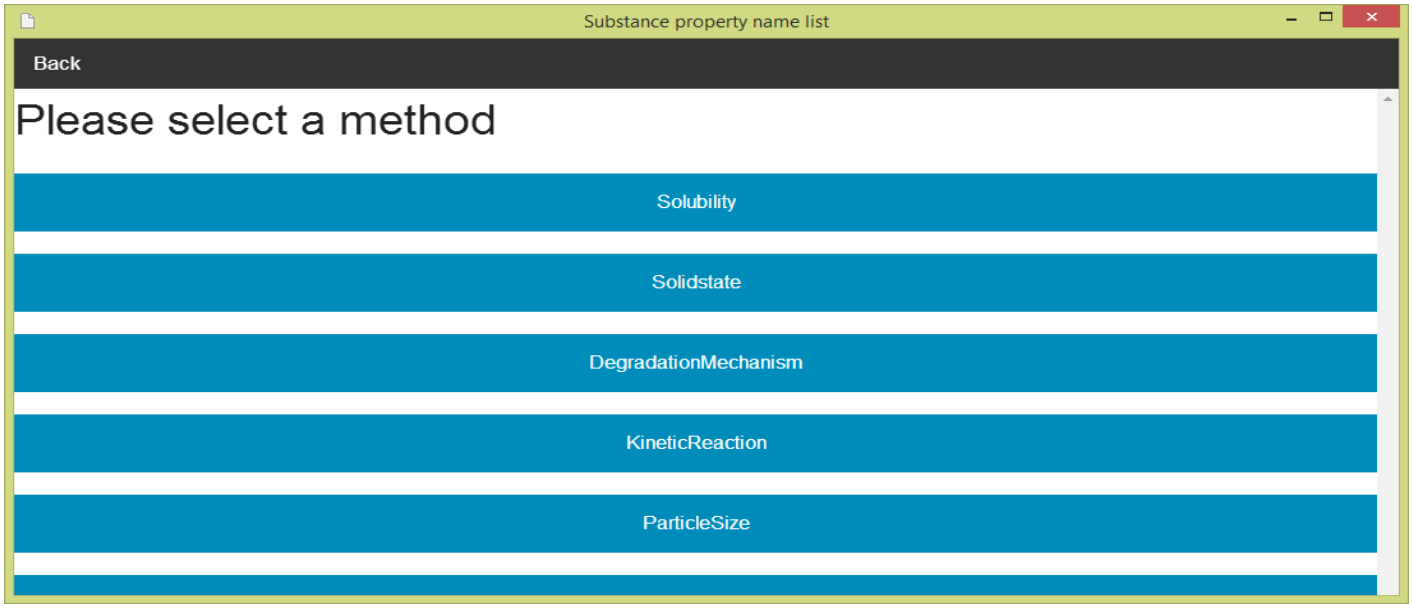


Figure 70 - UI-03: Substance property management menu

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

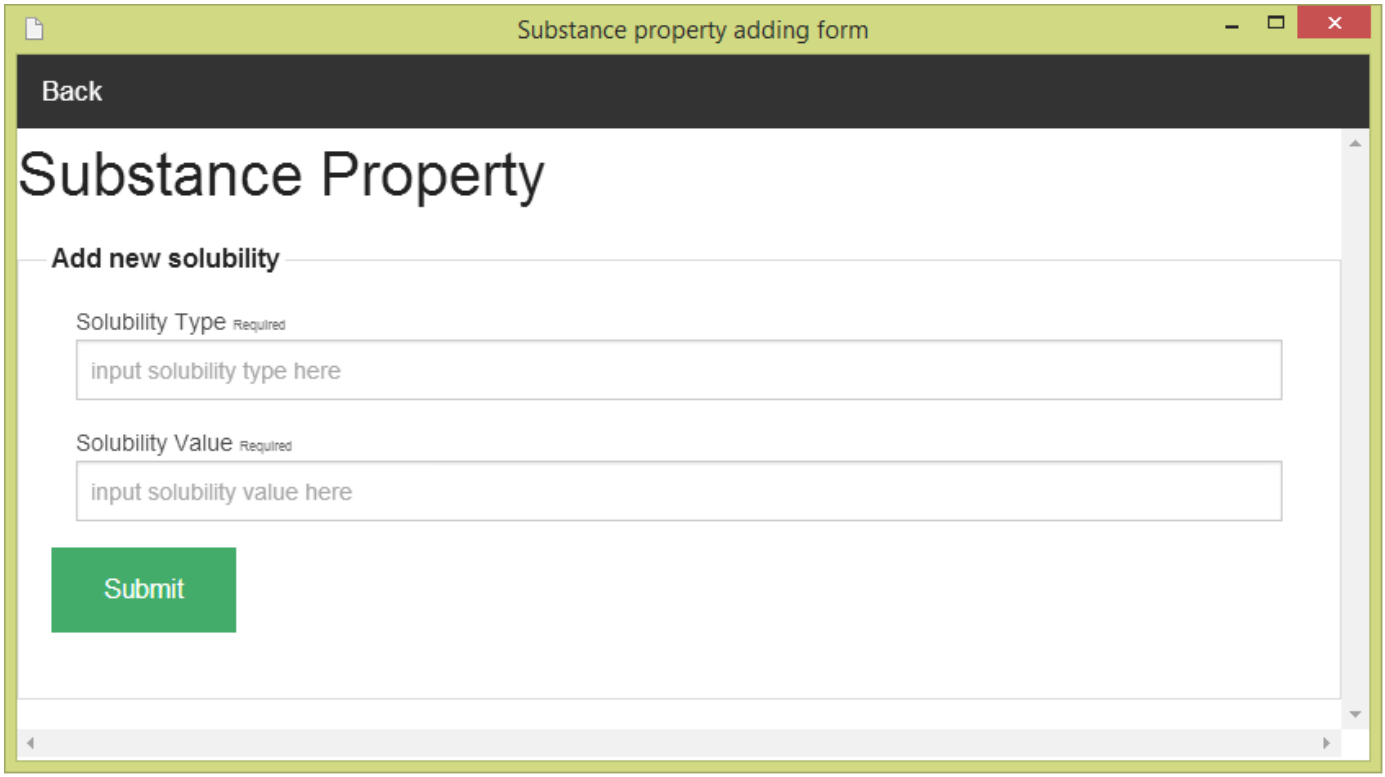


Figure 71 - UI-04: Substance property adding form

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.

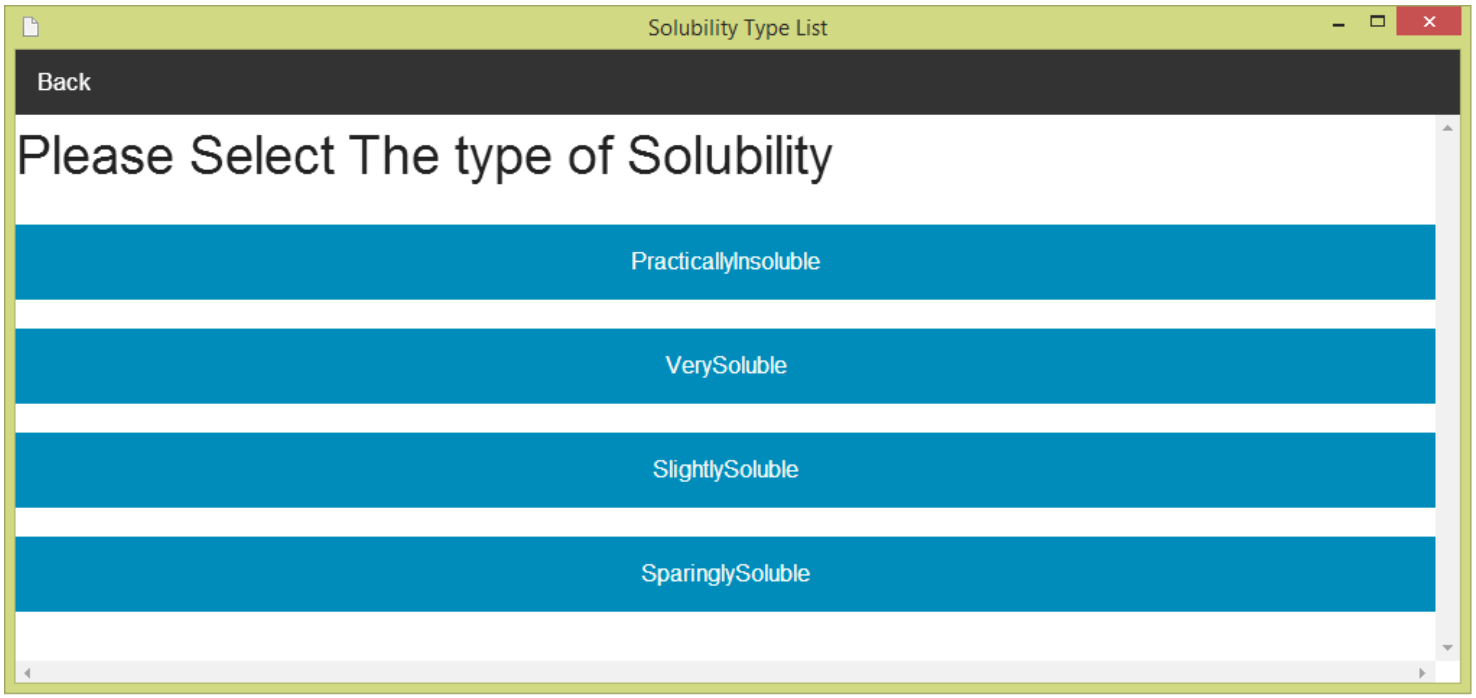


Figure 72 - UI-05: Substance property adding form

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

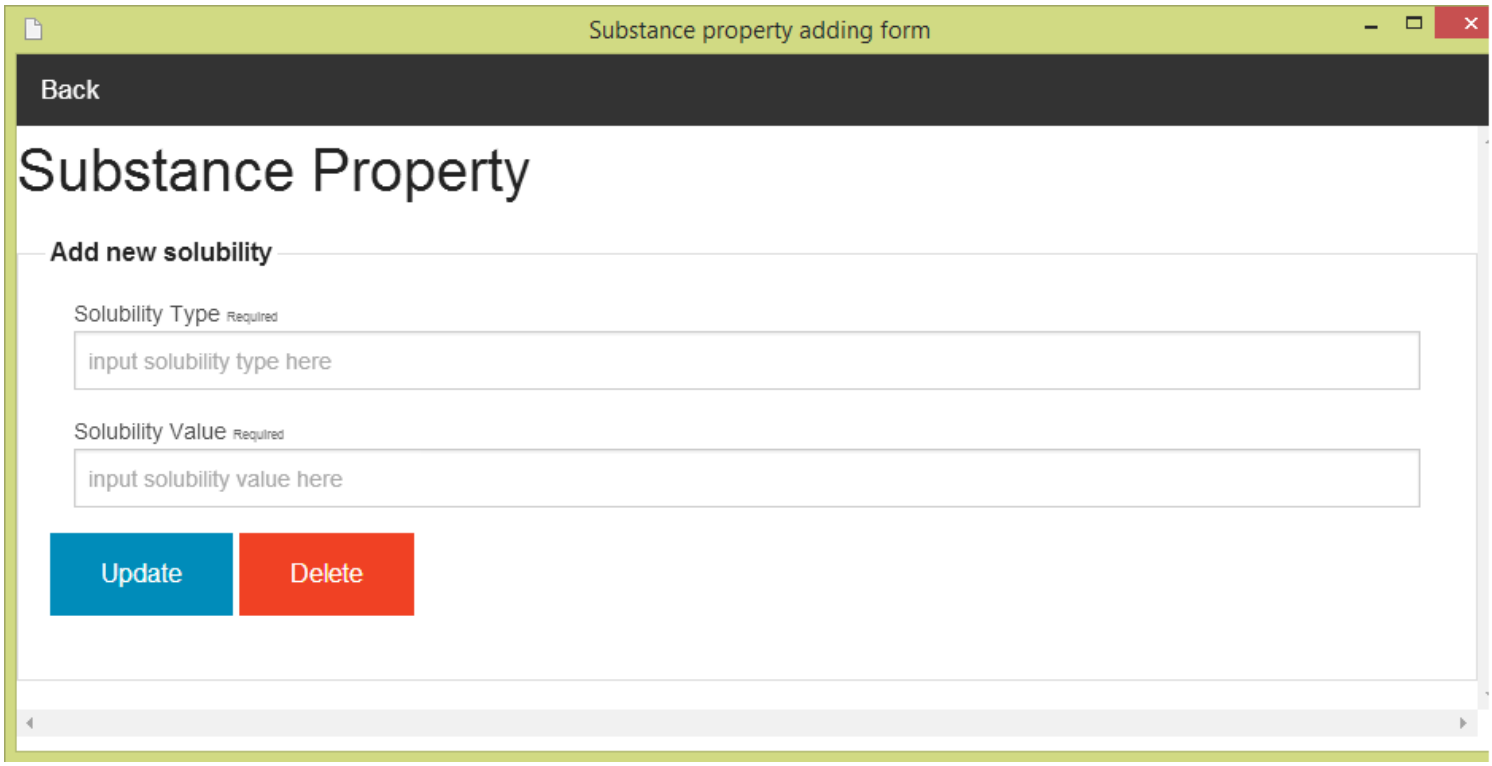


Figure 73 - UI-06: Substance property updating and deleting form

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.

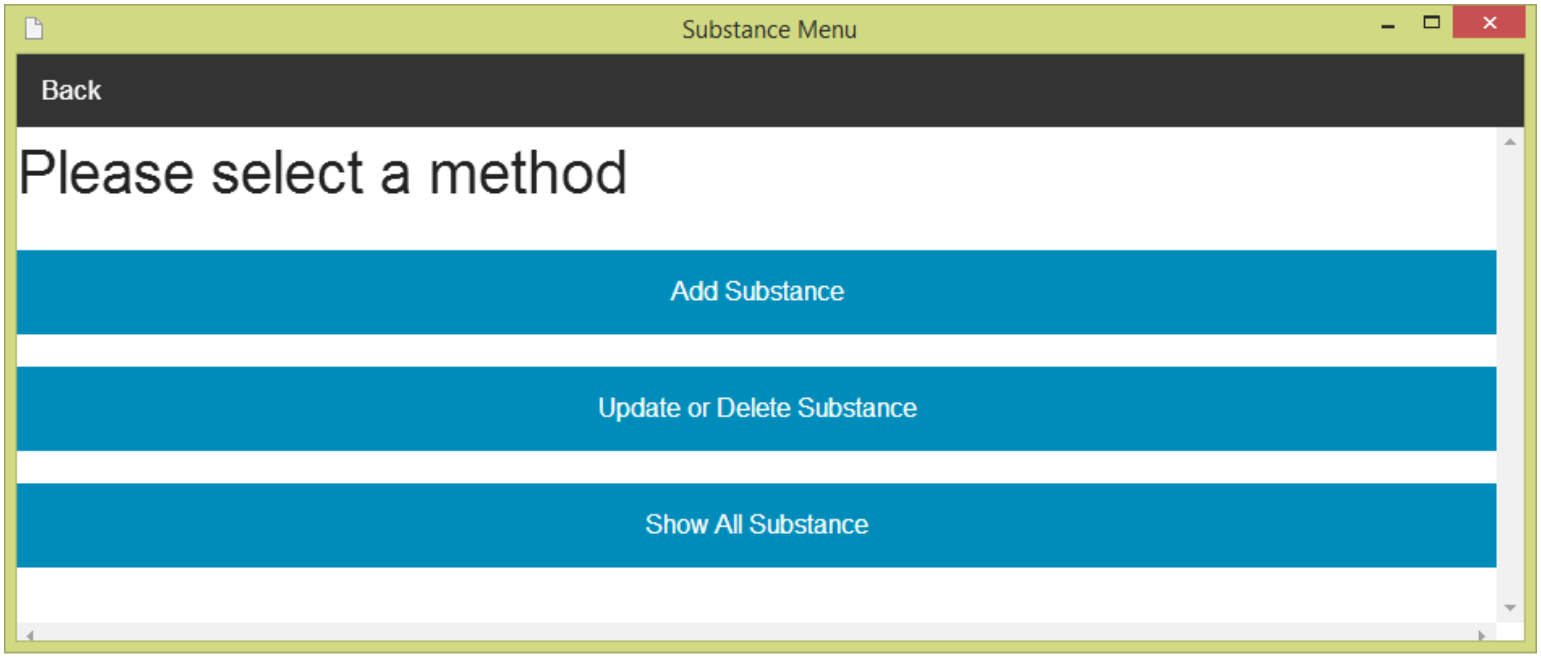


Figure 74 - UI-07: Substance management menu

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

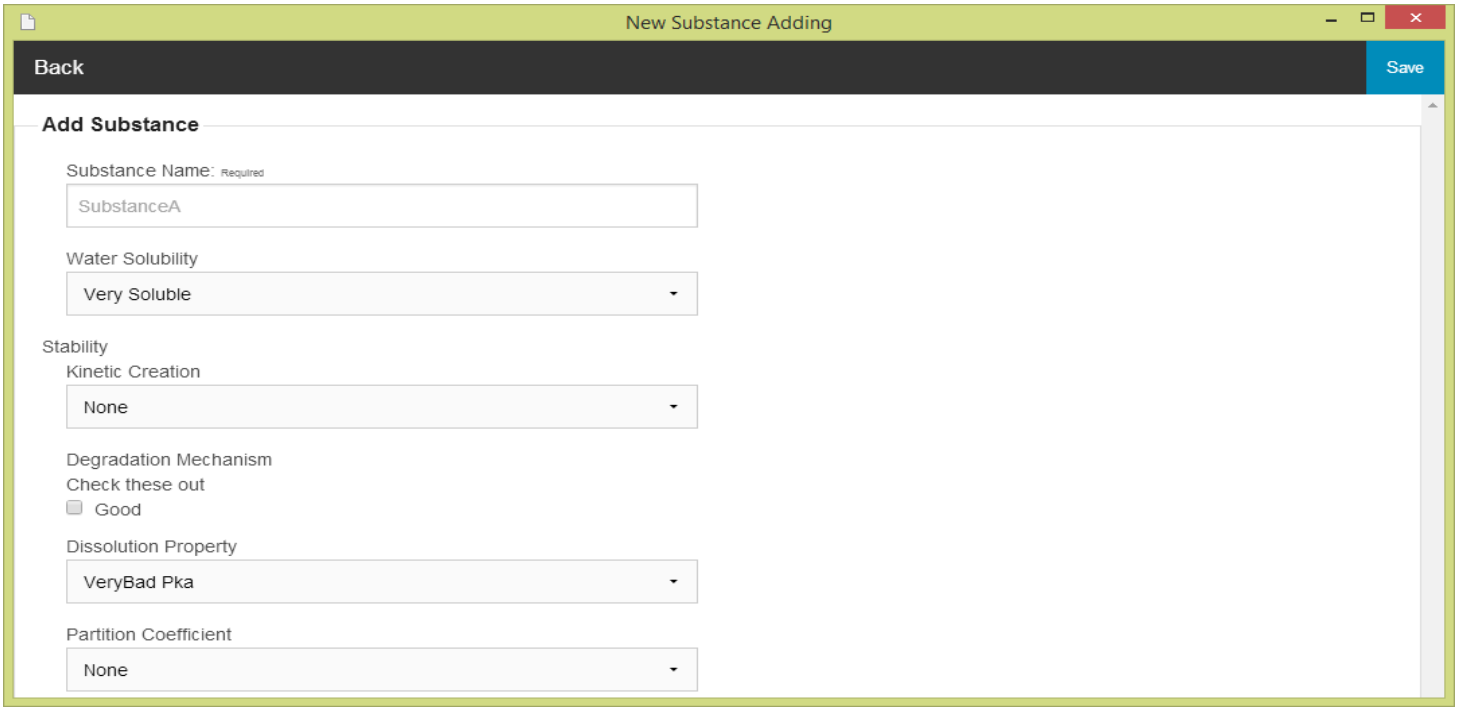


Figure 75 - UI-08: Substance adding form

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.

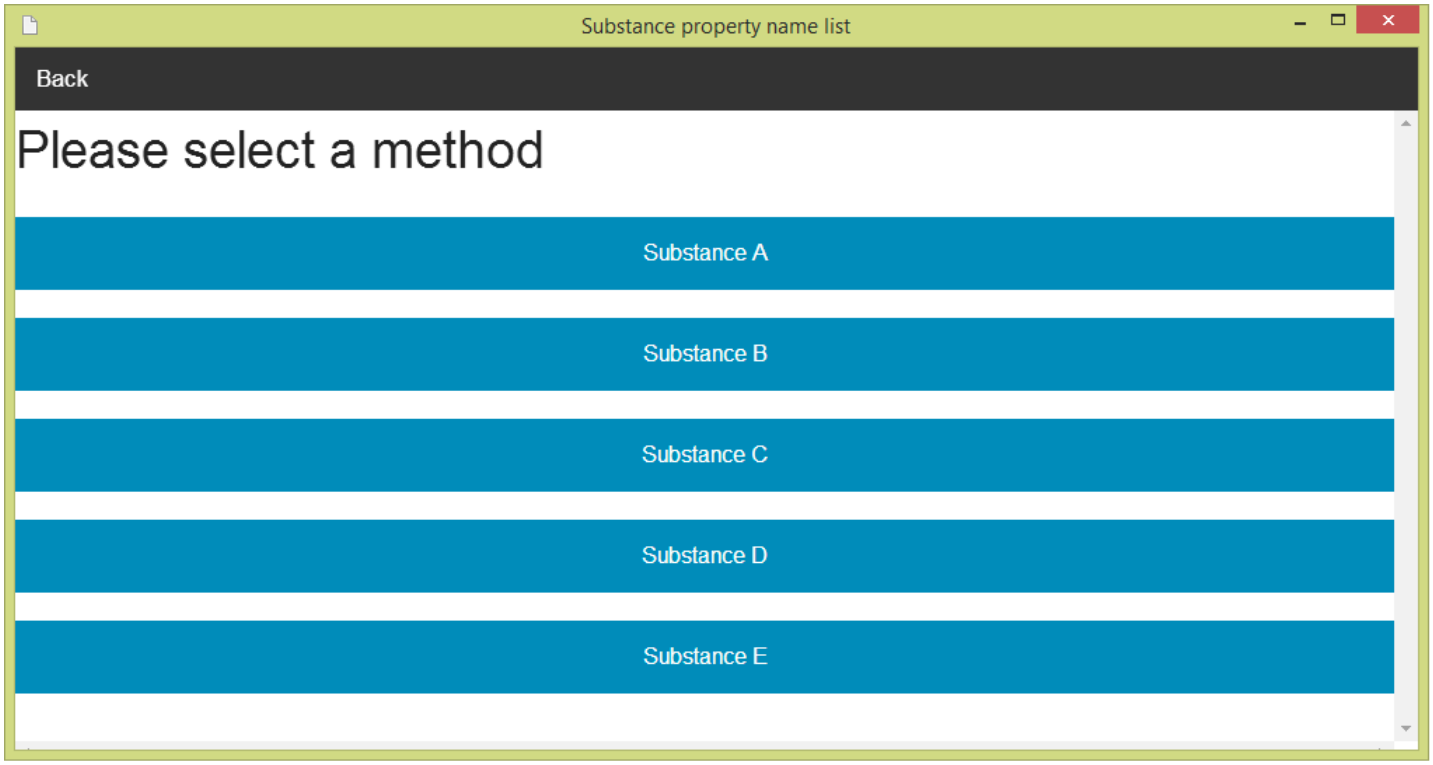


Figure 76 - UI-09: Substance name list

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.

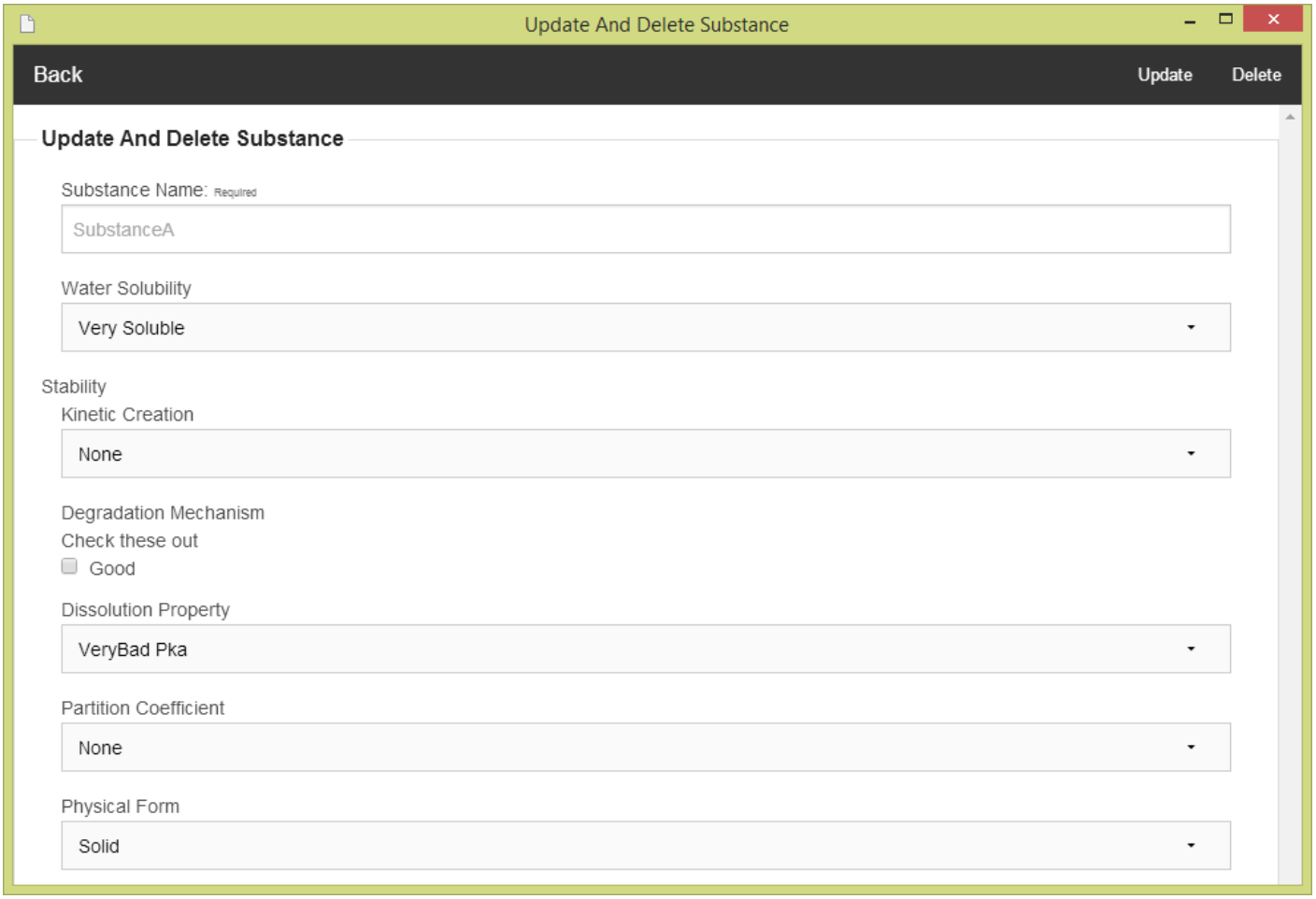


Figure 77 - UI-10: Substance updating and deleting form

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.

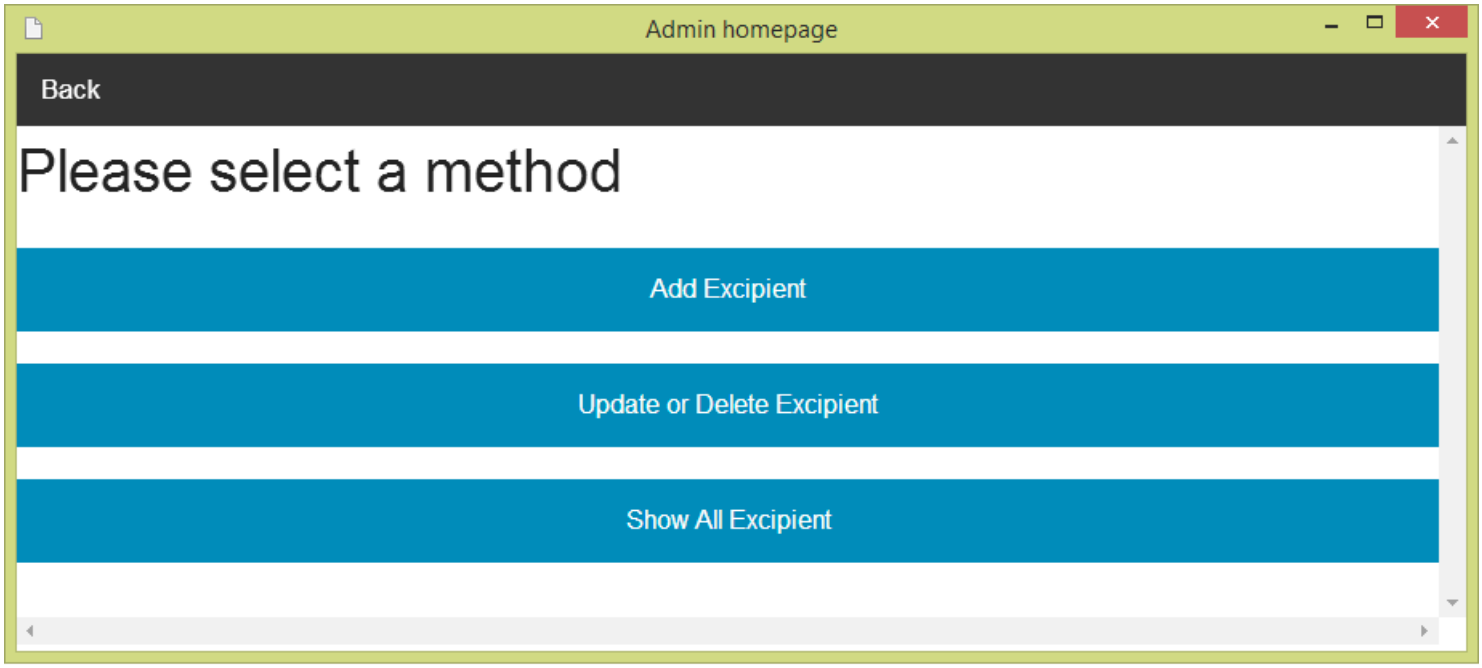


Figure 78 - UI-11: Excipient management menu

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.

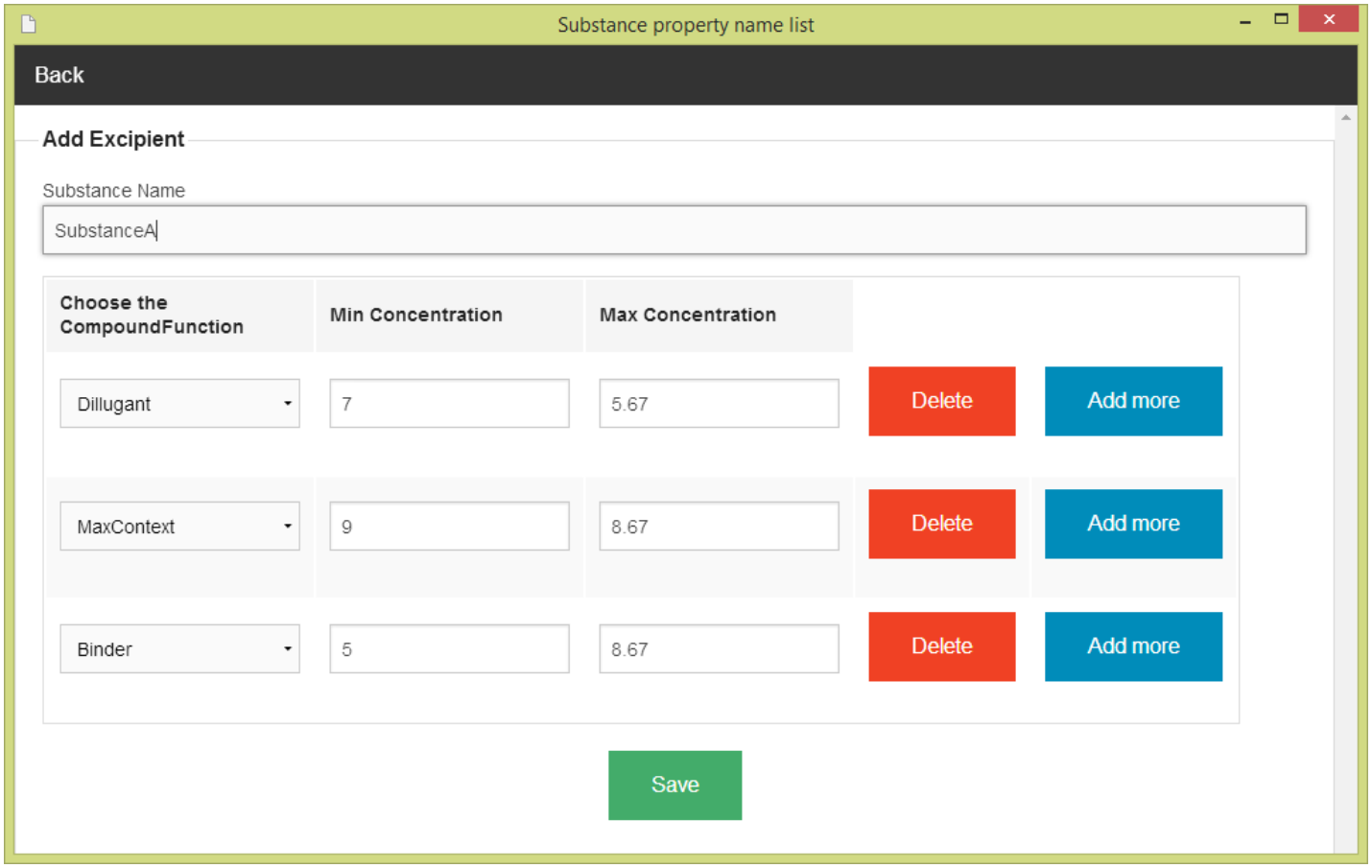


Figure 79 - UI-12: Excipient adding form

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

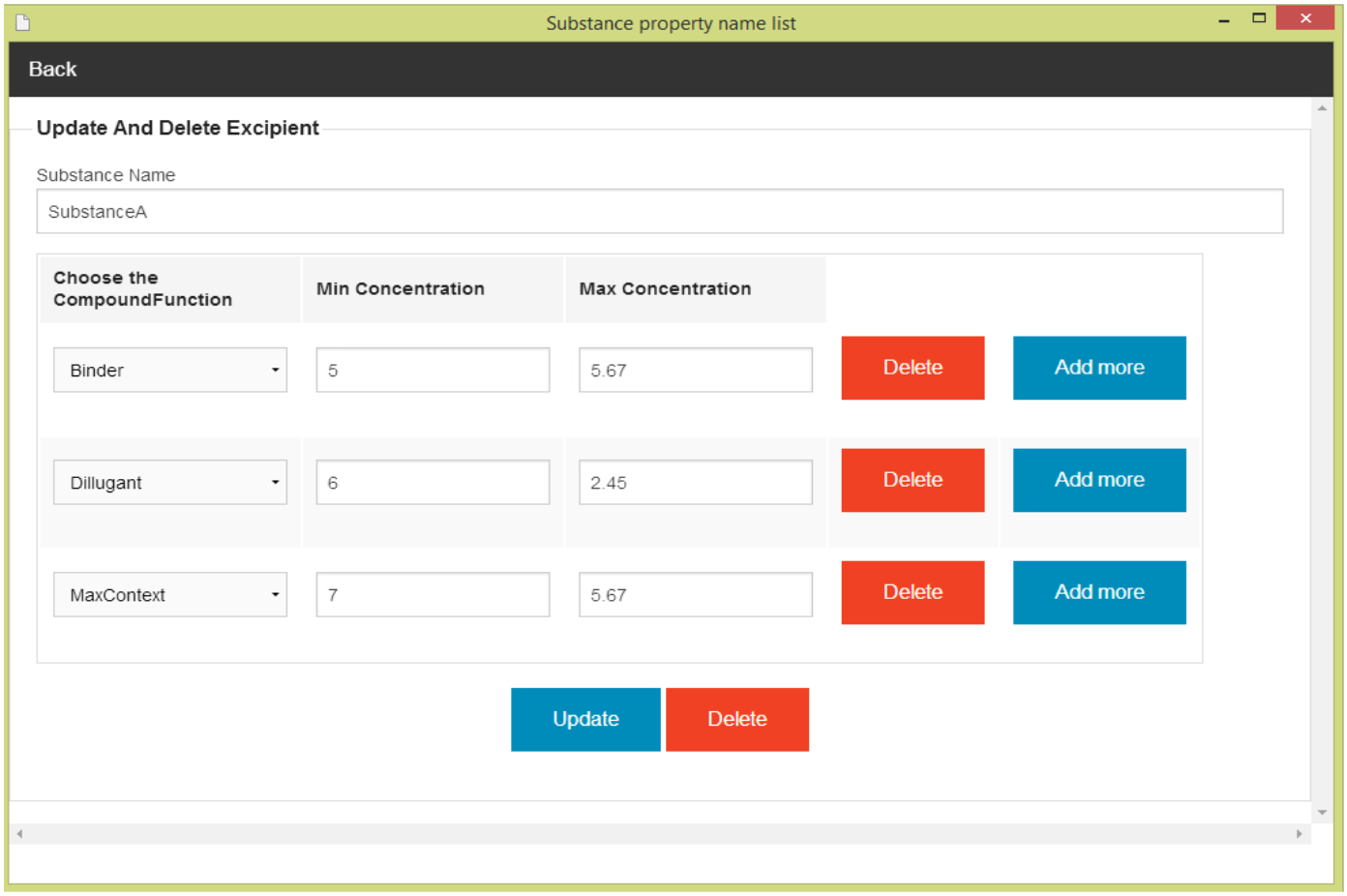


Figure 80 - UI-13: Excipient updating and deleting form

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.

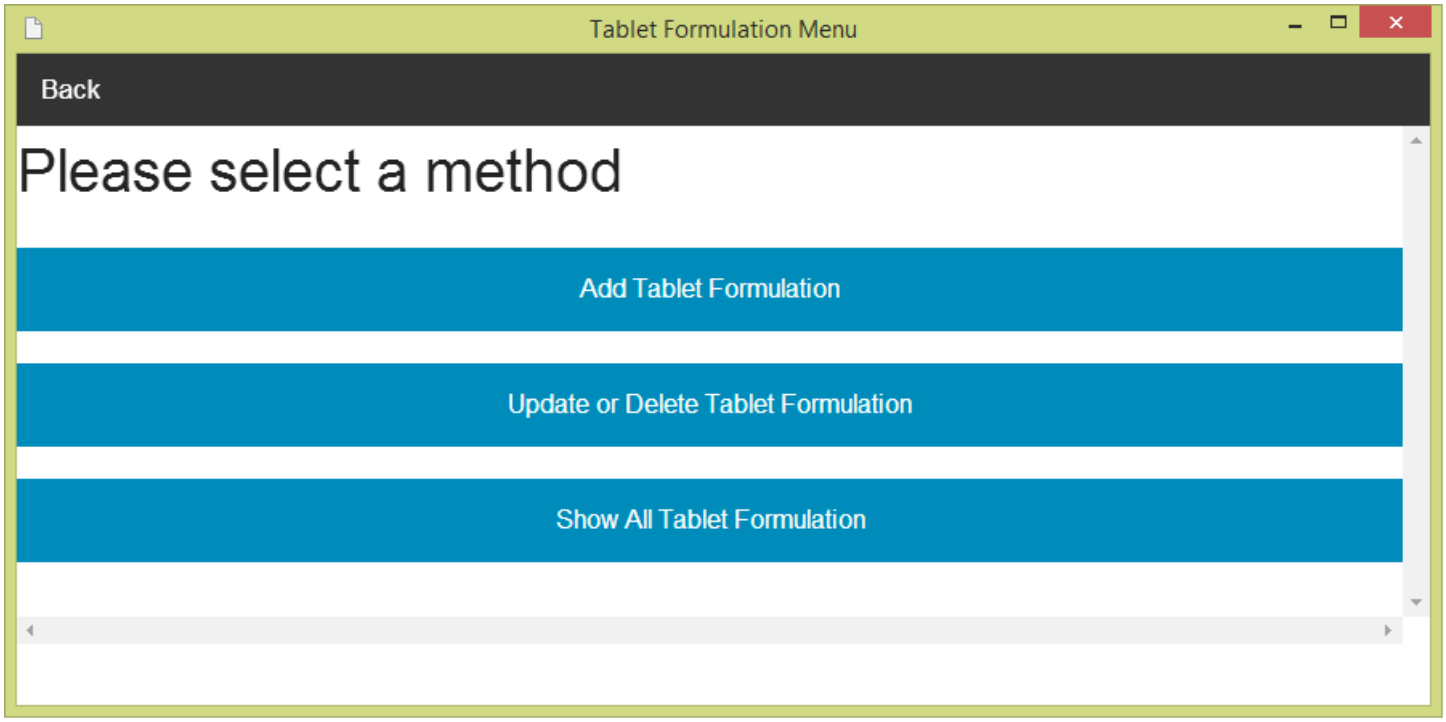


Figure 81 - UI-14: Tablet Formulation management menu

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.

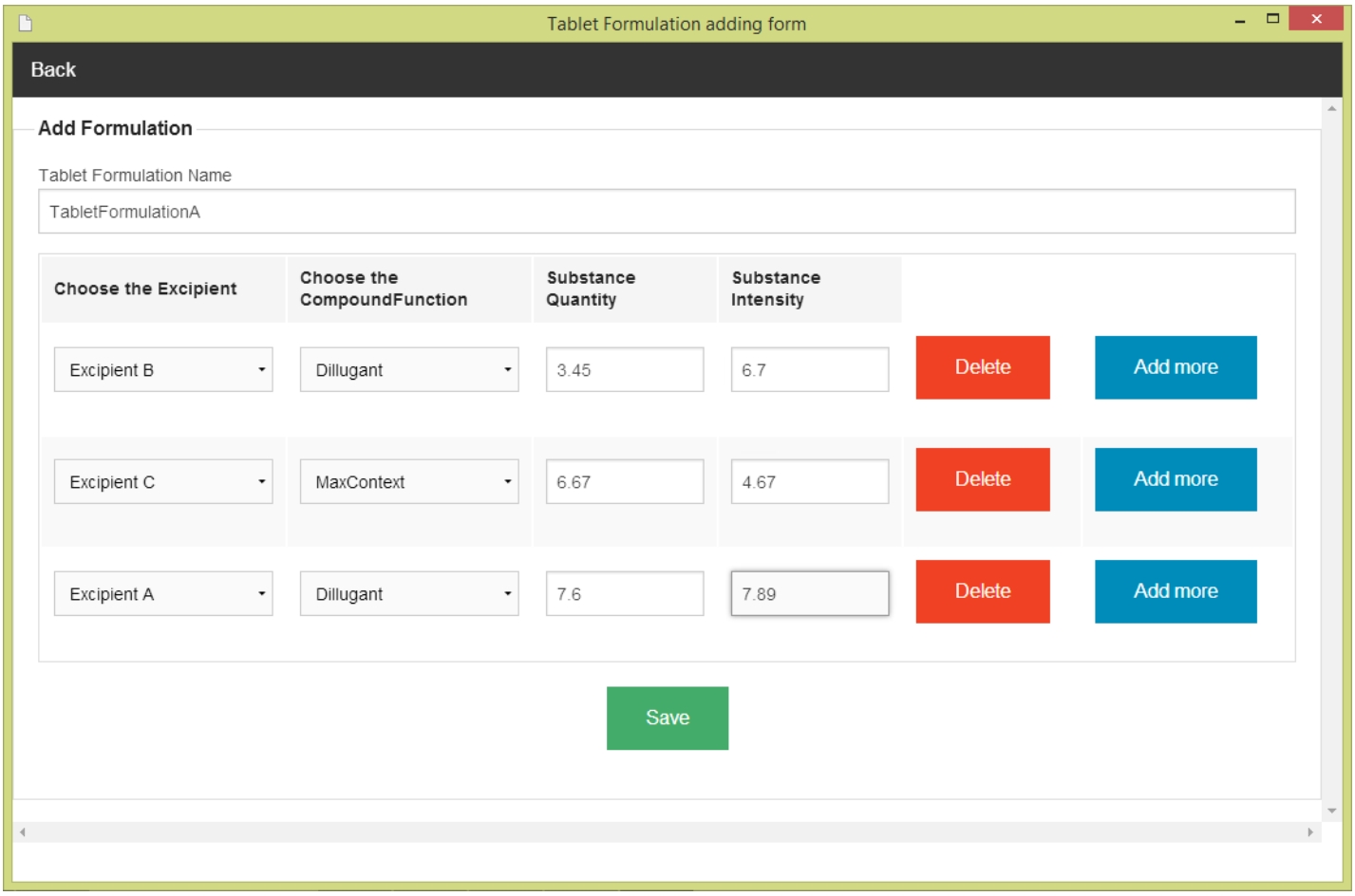


Figure 82 - UI-15: Tablet Formulation adding form

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.

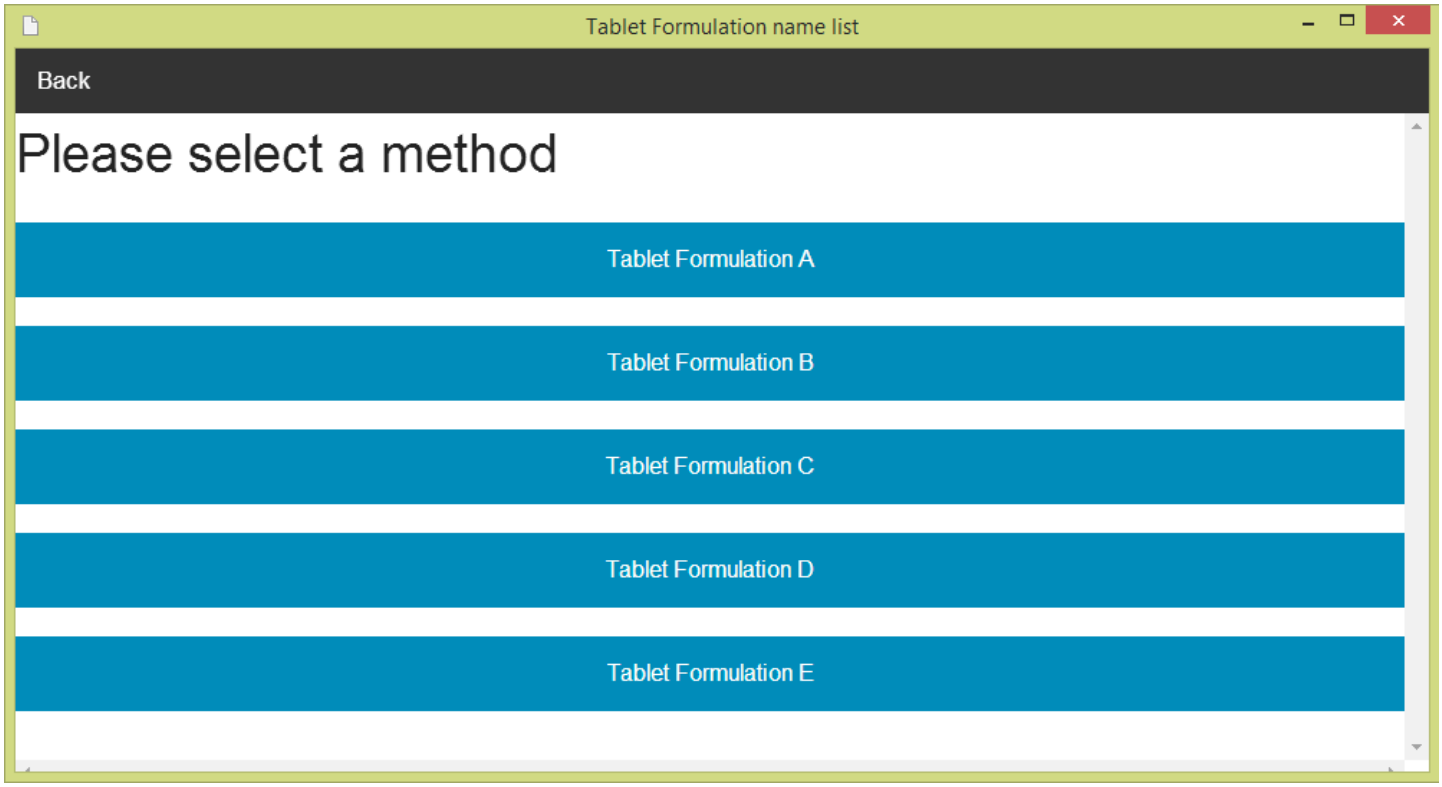


Figure 83 - UI-16: The list of tablet formulation name

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

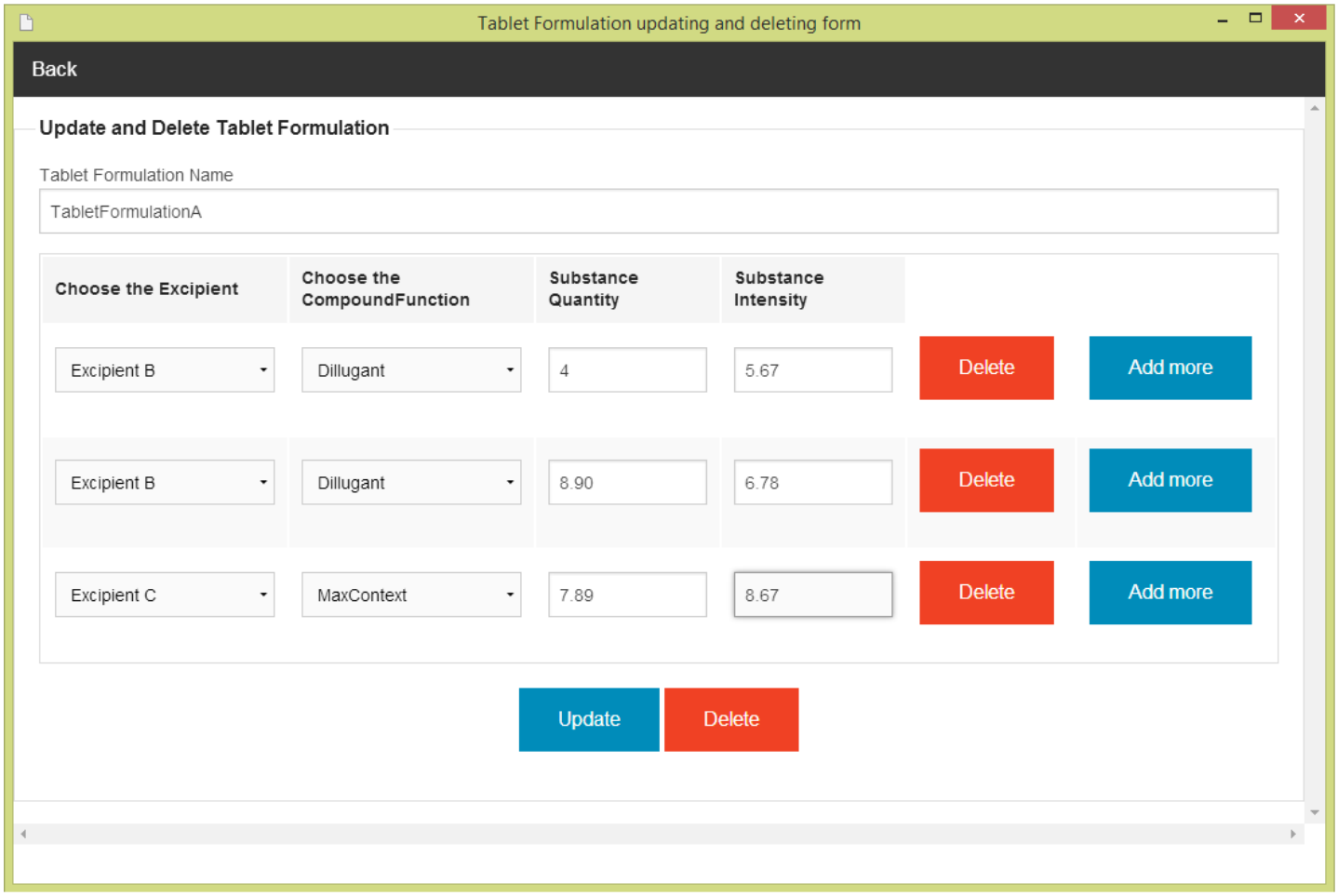


Figure 84 - UI-17: TabletFormulation updating and deleting form

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.