SPECTROPHOTOMETER

Bio Lab Part #2

Bingbing Xu, Chao Han, Junwei Zhao, Xueyuan Song

Content

Contribution Breakdown

	Bingbing Xu	Chao Han	Junwei Zhao	Xueyuan Song	Total
Section 1 Interaction Diagrams	20	20	40	20	100
Section 2 Classes + Specs		100			100
Section 3 Sys Arch& Design	10		60	30	100
Section 4 User Interface	50			50	100
Section 5 Testing Design			100		100
Section 6 Project management		100			100
Reference	25	25	25	25	100
Team Management	25	25	25	25	100

1. Interaction Diagrams

1.1 ID1 SwitchON

In use case UC-1: SwitchOn, according to Expert Doer design principle, Interface should send TurnOnRequest and receive the state of switch sent by StudentOperationController. After that, it sends LigntOn and MeterRead to IndicateLamp and Meter, respectively, in order to make the device start to work.

Therefore, responsibility R1 for sending turn on request should be assigned to Interface. The object Interface must know the state of switch when responding, according to Expert Doer principle, the Interface should be assigned responsibility R2 responding results with IndicateLamp and R3 MeterRead. Moreover, R4 which is updating power state should be assigned to StudentOperationController, and R5 which is Archivedata should be assigned to DatabaseConnnection. Figure 1-1 is the interaction Diagram of Use Case SwithOn.

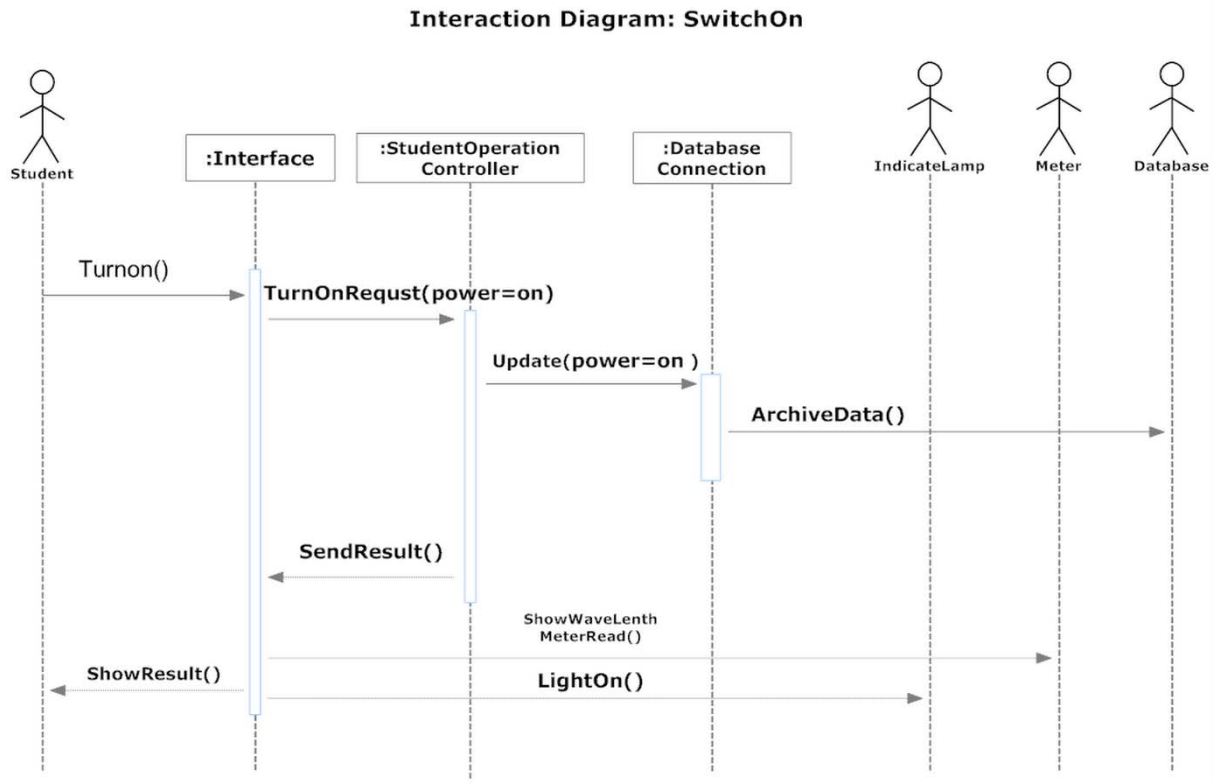

Figure 1-1 ID1 SwitchOn

1.2 ID2 AdjustBlank

Interaction Diagram of AdjustBlank is shown in Figure 1-2. Responsibility R1 (SendResult: MeterRead) can be assigned to ResultCalculator. In addition, there are two feasible alternatives as following:

1. ResultCalculator sends the result to Database

2. Database then returns the result to Interface

According to the High Cohesion principle, we assign this responsibility to ResultCalculator.

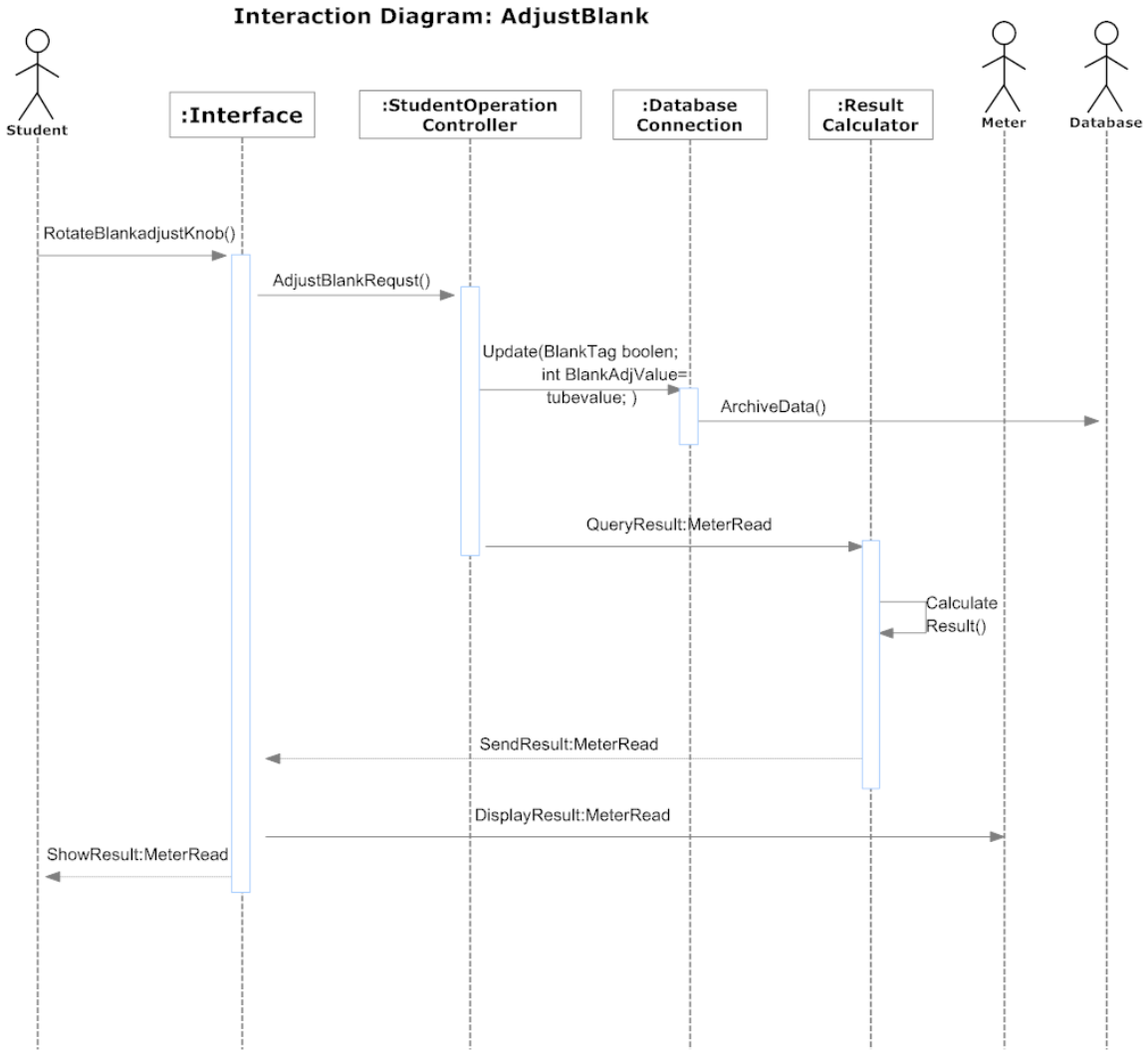

Figure 1-2 ID2 AdjustBlank

1.3 ID3 AdjustZero

Figure 1-3 is the interaction diagram of AdjustZero. In this case, responsibility R1 is assigned for sending the request to AdjustZero. The information request first is given to StudentOperationController, so by Expert Doer Principle, responsibility R1 should be assigned to the StudentOperationController. We assigned the responsibility R2 for updating the information of point in the database. According to Expert Doer Principle, the information first is given to Database connection, so the Database connection should be assigned for responsibility R2.

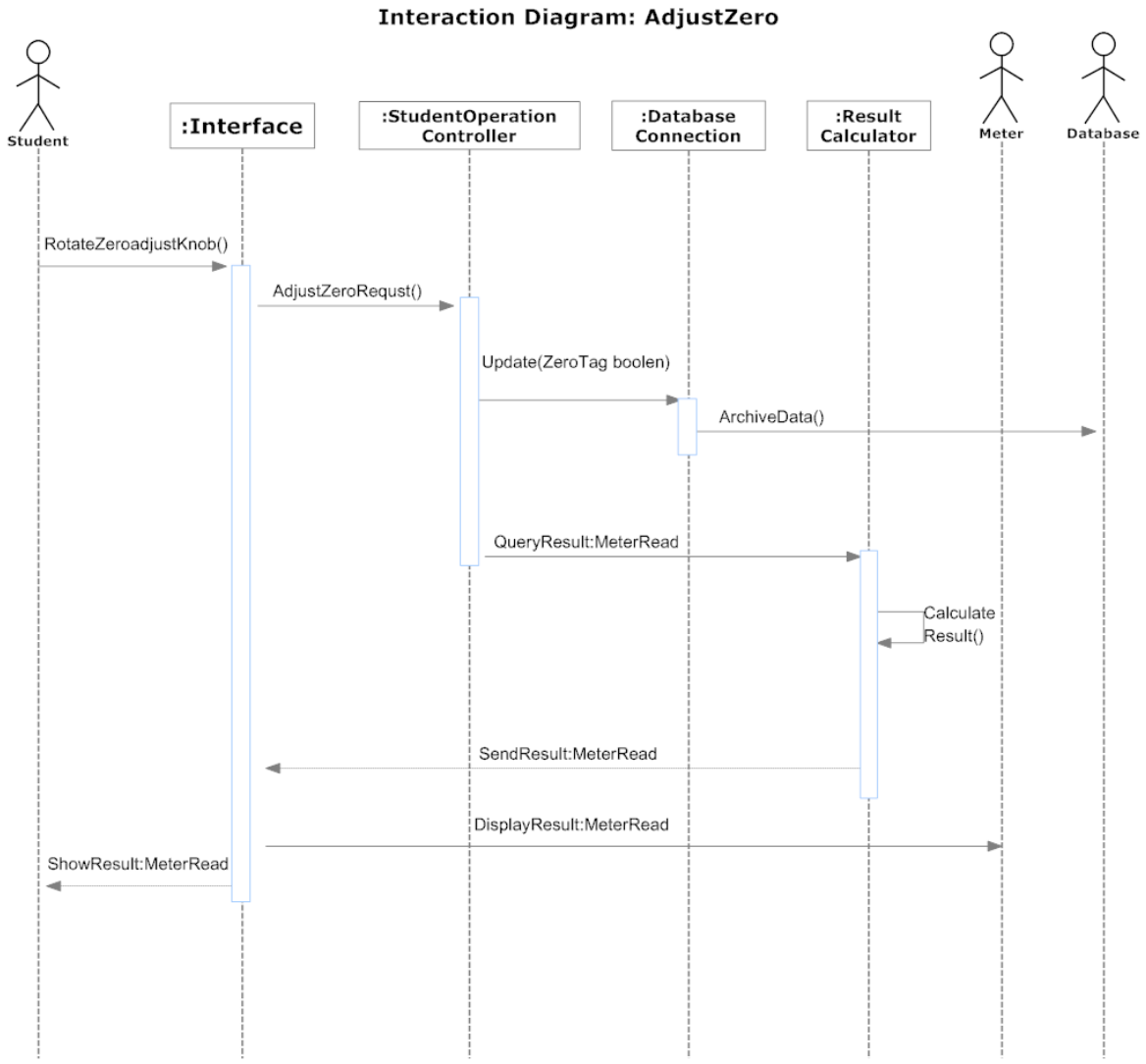

Figure 1-3 ID3 AdjustZero

As for responsibility R3 (calculate and update the meter information), there are alternative options:

1. Database Connection and Result Calculator, both can be the first to get hold of meter information.
2. Data calculator can directly compare the meter information and send MeterRead request to the interface.

The Database connection has the responsibility to update the information in database, and by High Cohesion principle, it favors assigning the responsibility R3 to the Database connection. This solution violates the Low Coupling Principle, because Database connection acquires relatively large number of associations. On the other hand, by Expert Doer Principle, it favors assigning responsibility R3 for Data Calculator. So we assign the responsibility R3 to Data calculator.

At last, we define responsibility R4 for sending the result to interface. The result first is given to interface, so by Expert Doer Principle, the interface should be assigned responsibility R4.

1.4 ID4 SelectWaveLength

In the UC-4: SelectWavelength. In this case, as Figure 1-4 shows, responsibility R1 is assigned for sending the request to adjust wavelength. The information request first is given to StudentOperationController, so by Expert Doer Principle, the StudentOperation Controller should be assigned responsibility R1. The responsibility R2 is updating the wavelength status, and still by Expert Doer principle, the information is first given to DatabaseConnection. And by High Cohesion principle, it favors assigning the responsibility R2 to the Database connection. Responsibility R2 was assigned for DatabaseConnection. Then we define responsibility R3 for sending the result to interface. The result first is given to interface, so by Expert Doer Principle, the interface should be assigned responsibility R3.

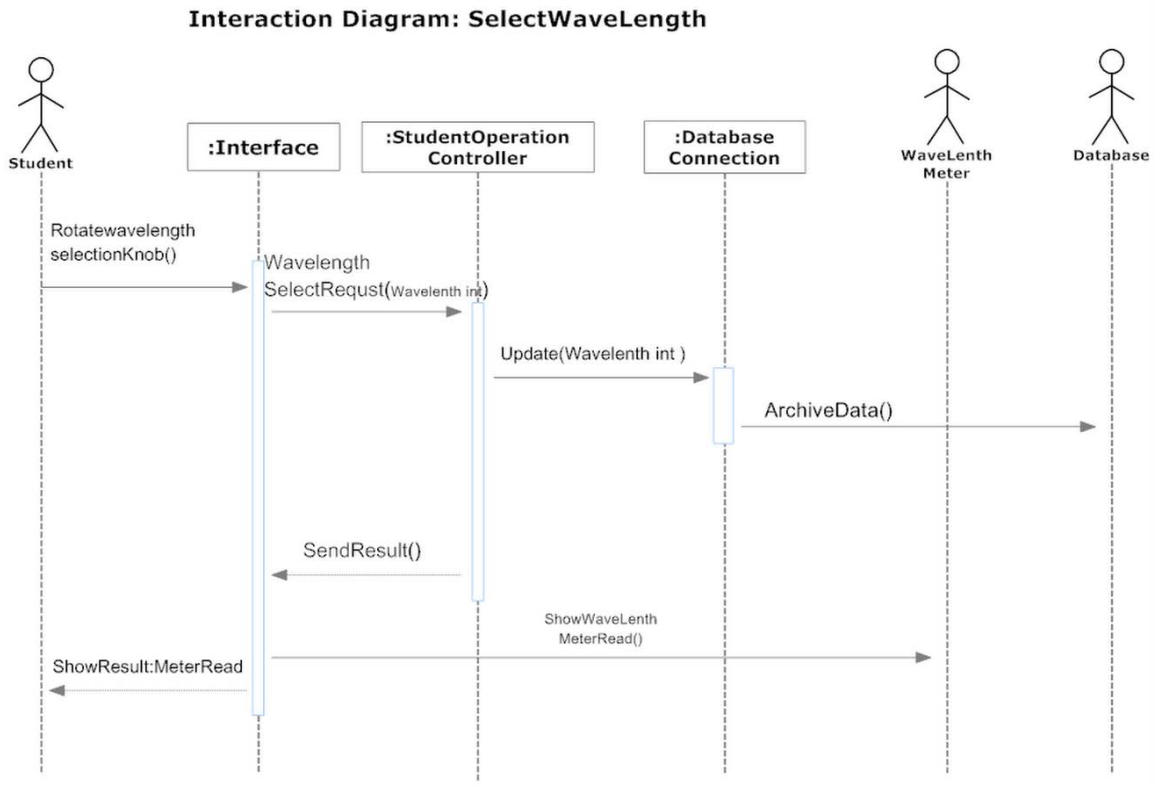

Figure 1-4 ID4 SelectWaveLength

1.5 ID5 UpdateInfoBoard

In the UC-5: UpdataInfoBoard. In this case, as Figure 1-5 shows,responsibility R1 is assigned for sending the request to update information board. By Expert Doer Principle, the request is first send to TeacherOperationController. By High Cohesion principle, the TeacherOperationController has the responsibility to update the information board. So both

principles favor assigning responsibility R1 for TeacherOperationController. Then we assigned the responsibility R2 for updating the information board in the database. By Expert Doer Principle, the information first is given to Database connection, and by High Cohesion Principle, it favors assigning the responsibility R2 to the Database connection. So the Database connection should be assigned for responsibility R2. We define responsibility R3 for sending the result to interface. The result first is given to interface, so by Expert Doer Principle, the interface should be assigned responsibility R4.

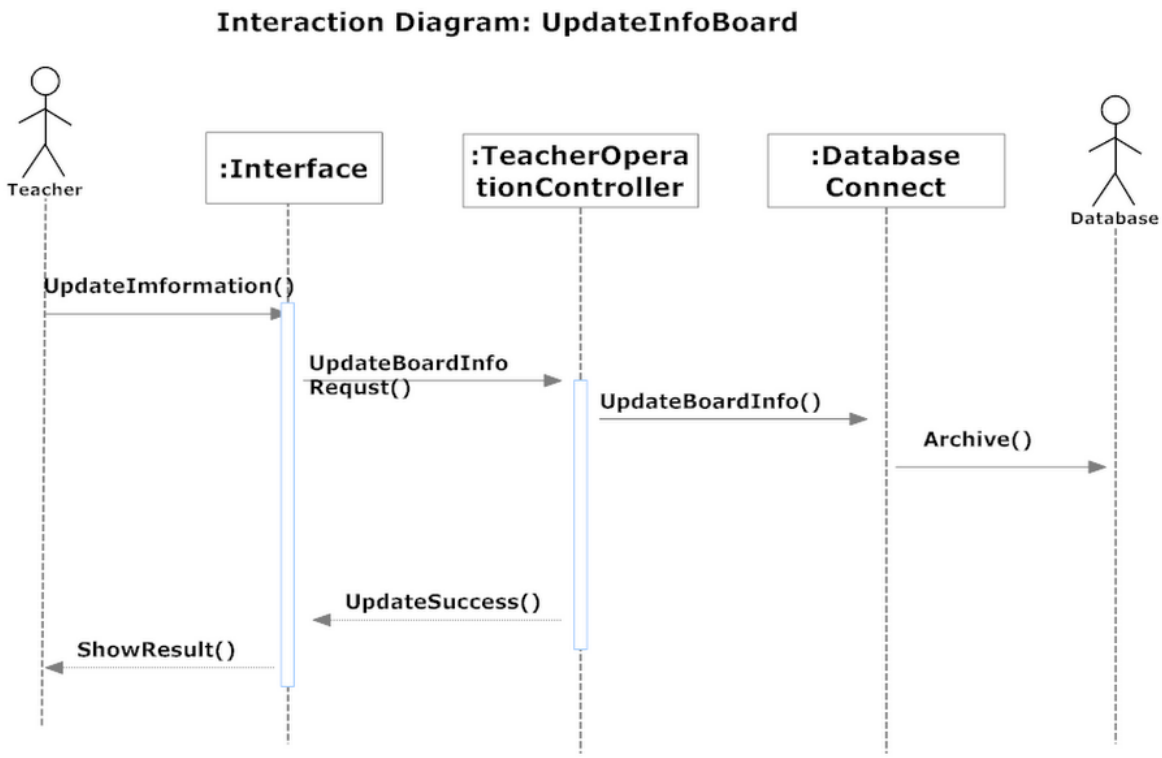

Figure 1-5 ID5 UpdateInfoBoard

1.6 ID6 OpenLid

In use case UC-6: OpenLid, assigning responsibilities R1 (get(query: LidTag)), R2(retrieve data), and R9(send back data) to StudentOperationController and DatabaseConnection, and Database is straightforward, as Figure 1-6 shows. ResultCalculator provides a data comparing and computing function. So after processing the data from Database, it will be assigned with responsibility R3 (send the decision). Then SudentOperationController will act its responsibility R4 (open the lid). These assignments all follow Expert Doer design principle.

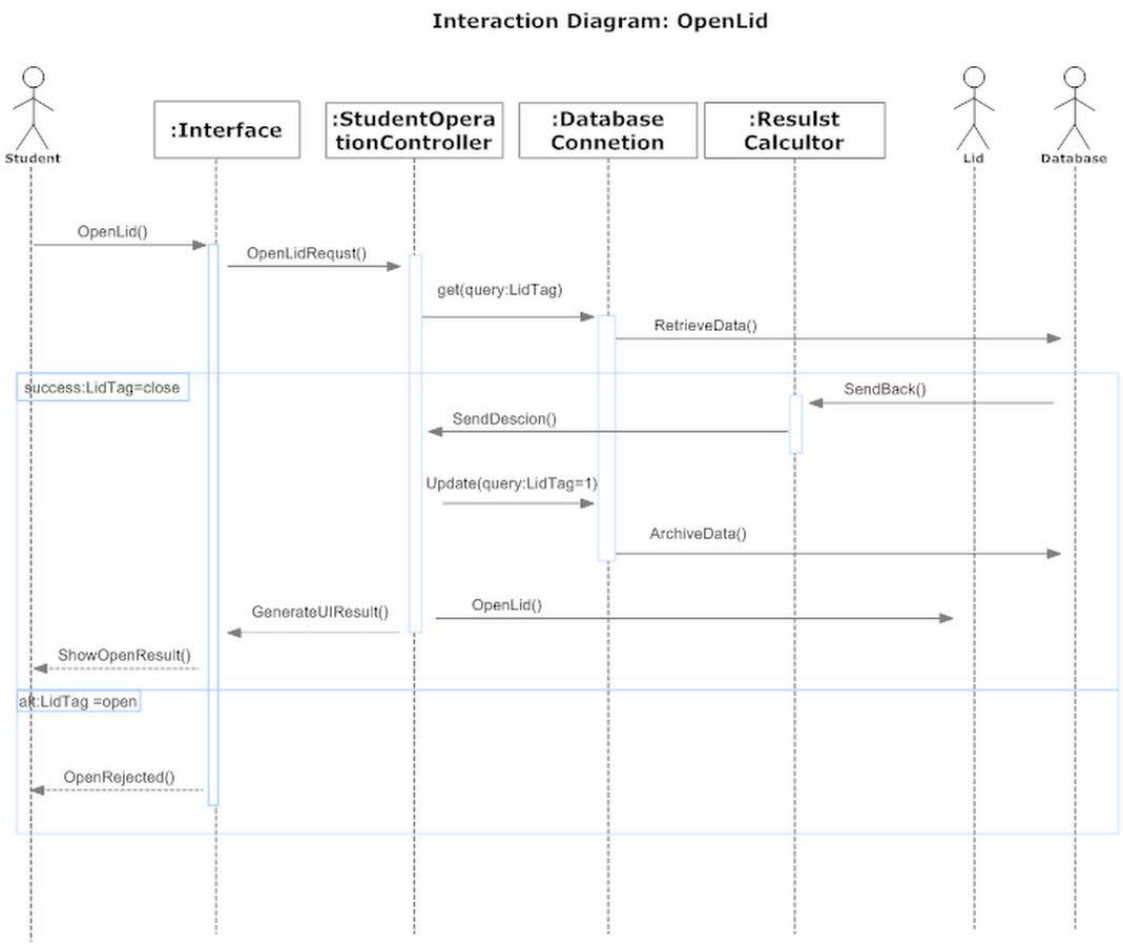

Figure 1-6 ID6 OpenLid

1.7 ID7 CloseLid

The UC-7: CloseLid is very similar to UC-6. R1, R2, R3, and R4 are the same with UC-6. After updating, the StudentOperationController will act responsibility R5 (open the lid). Figure 1-7 shows the interaction diagram of it.

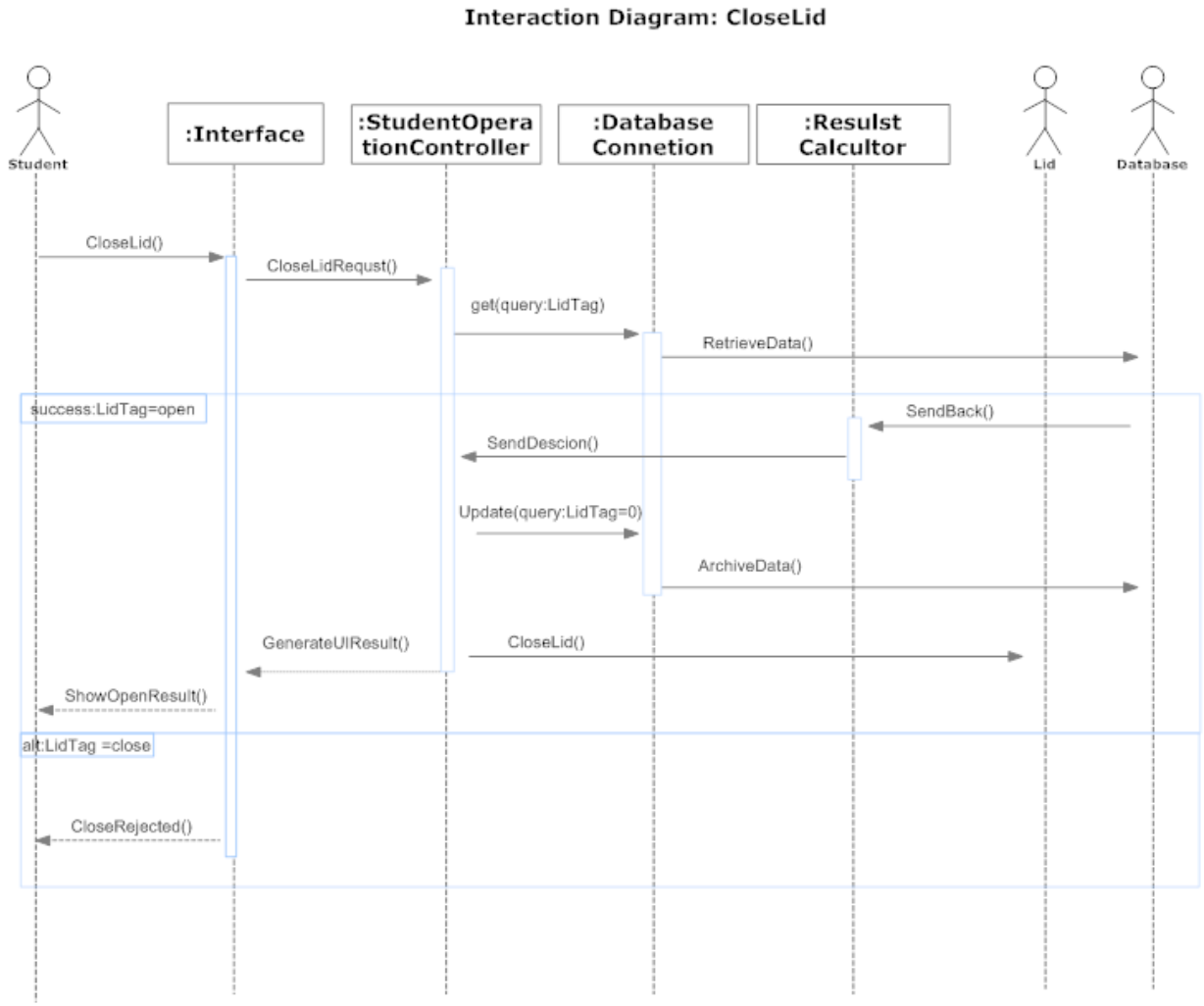

Figure 1-7 ID7 Interaction Diagram

1.8 ID8 SelectTestTube

When designing this interaction diagram (Figure 1-8), we apply Expert Doer Principal to Interface object. Interface takes responsibility of accept user SelectTestTube request, send it to StudentOperationController, receive the result and display it on screen (TestTube) to student.

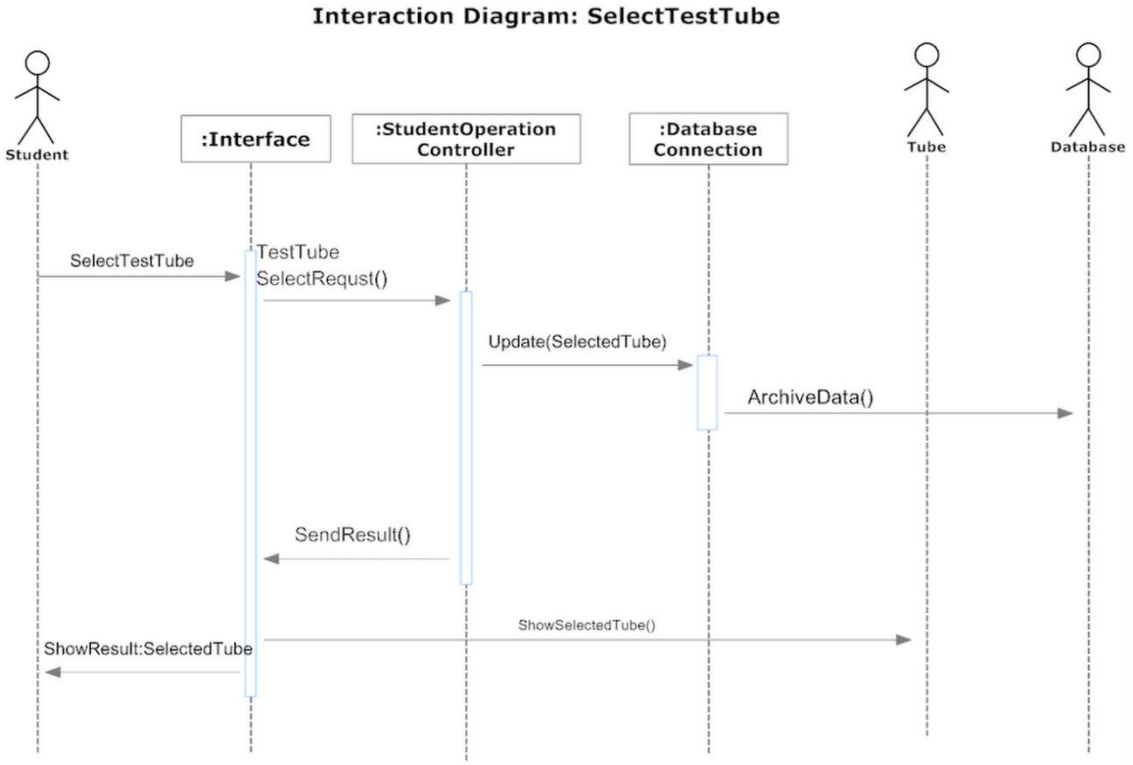

Figure 1-8 ID8 SelectTestTube

Considering the low coupling and high cohesion principal, we derive two objects StudentOperationController and DatabaseConnection. StudentOperationController takes responsibility of respond to SelectTestTube request, reference DatabaseConnection calls for Database operations (here it is update), and reference ResultCalculator (not used in this diagram). At last, it generates a result and sends it back to Interface.

Communication Responsibilities are shown in Table 1-1, as following

Responsibility Description
Send message to StudentOperationController to notify the operation type
Send message to DatabaseConnection to require an update request.
Send message to Interface to Deliver the result.

Table 1-1 Communication Responsibilies of ID8

1.9 ID9 InsertTestTube

In this interaction diagram (Figure 1-9), we also use Expert Doer Principal to Interface object. Interface takes responsibility of accept user Insert request, send it to StudentOperationController, receive the result and display it on screen (Meter) to student.

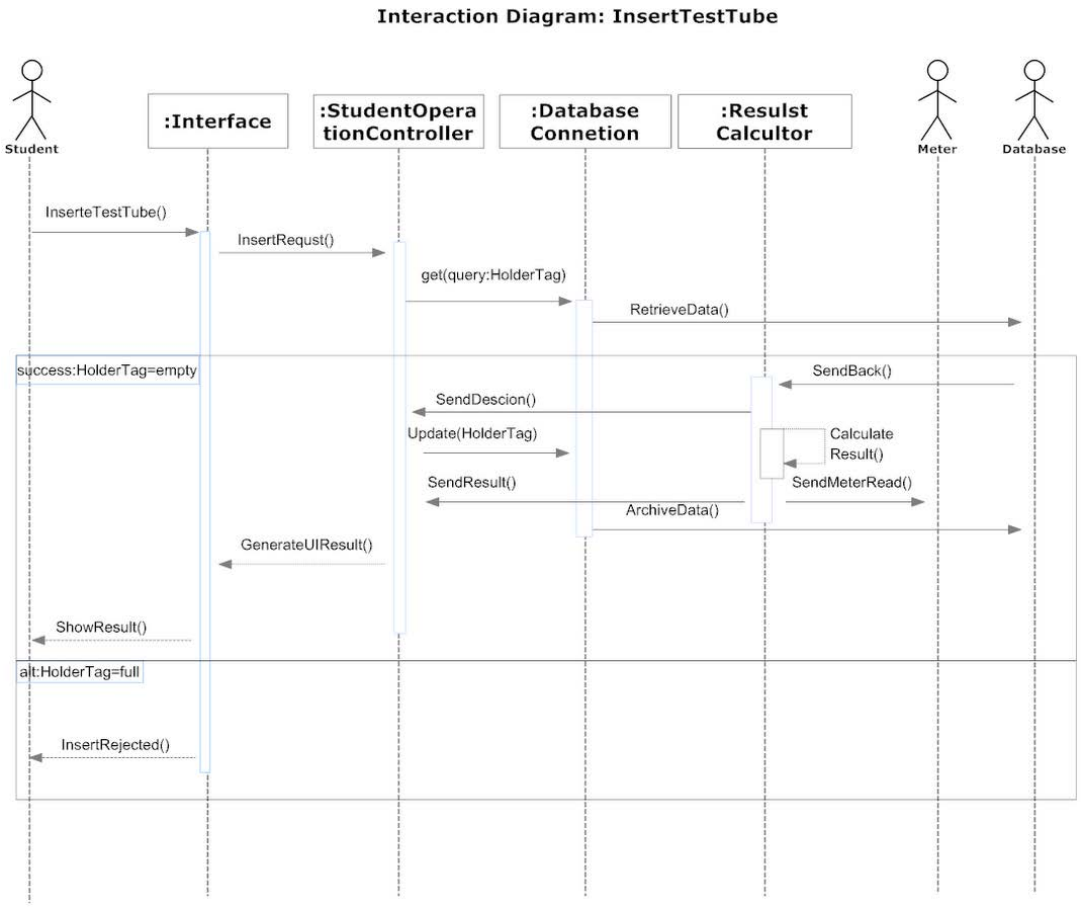

Figure 1-9 ID9 InsertTestTube

Considering the low coupling and high cohesion principal, we derive three objects StudentOperationController, DatabaseConnection and ResultCalculator. StudentOperationController takes responsibility of respond to InsertTestTube request, reference DatabaseConnection call for Database operations (here it is update), and reference ResultCalculator, respectively. At last, it generates a result and sends it back to Interface. ResultCalculator will first of all use the HolderTag retrieved from Database to make a decision (pass or reject). If the Insert operation is valid (which means the HolderTag=empty), it will proceed to calculate and send back the result. At the same time StudentOperationController will reference DatabaseConnection to update the HolderTag in Database.

Communication Responsibilities are shown in Table1-2

Responsibility Description
Send message to StudentOperationController to notify the operation type
Send message to DatabaseConnection to query HolderTag.
Send message to StudentOperationController to inform the operation decision.
Send message to DatabaseConnection to require an update request.
Send message to Interface to Deliver the result.

Table 1-2 Communication Responsibility of ID9

1.10 ID10 RemoveTestTube

When designing this interaction diagram (Figure 1-10), we apply Expert Doer Principal to Interface object. Interface takes responsibility of accept user Remove request, send it to StudentOperationController, receive the result and display it on screen (Meter) to student.

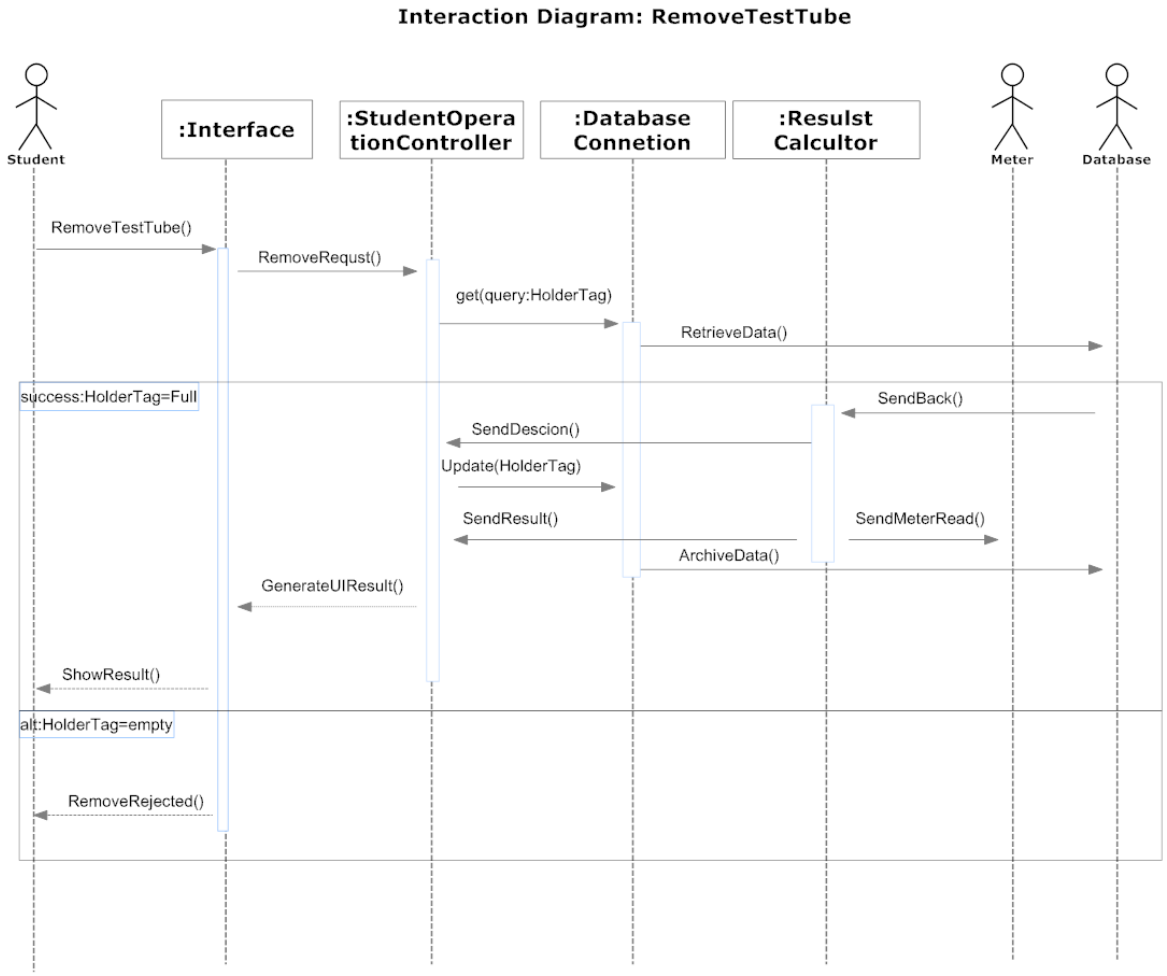

Figure 1-10 ID10 RemvoveTestTube

Considering the low coupling and high cohesion principal, we derive three objects StudentOperationController, DatabaseConnection and ResultCalculator. StudentOperationController takes responsibility of respond to RemoveTestTube request, reference DatabaseConnection call for Database operations (here it is update), and reference ResultCalculator. At last, it generates a result and sends it back to Interface. ResultCalculator will first of all use the HolderTag retrieved from Database to make a decision (pass or reject). If the Insert operation is valid (which means the HolderTag=full), it will proceed to send back the result. At the same time StudentOperationController will reference DatabaseConnection to update the HolderTag in Database.

Communication Responsibilities are shown in Table 1-3

Responsibility Description
Send message to StudentOperationController to notify the operation type
Send message to DatabaseConnection to query HolderTag.
Send message to StudentOperationController to inform the operation decision.
Send message to DatabaseConnection to require an update request.
Send message to Interface to Deliver the result.

Table 1-3 Communication Responsibilities of ID 10

1.11 ID11 SetSample

Figure 1-11 is the interaction diagram that shows the how teachers set Sample's value. Teachers send SetSample Requst() through interface. After inputting SampleValue at TecherOpeartionController, the SampleValue is sent to the Database Connection, and then it is saved in database. At the same time, TeacherOperationController send a SetSuccess() message to Interface.

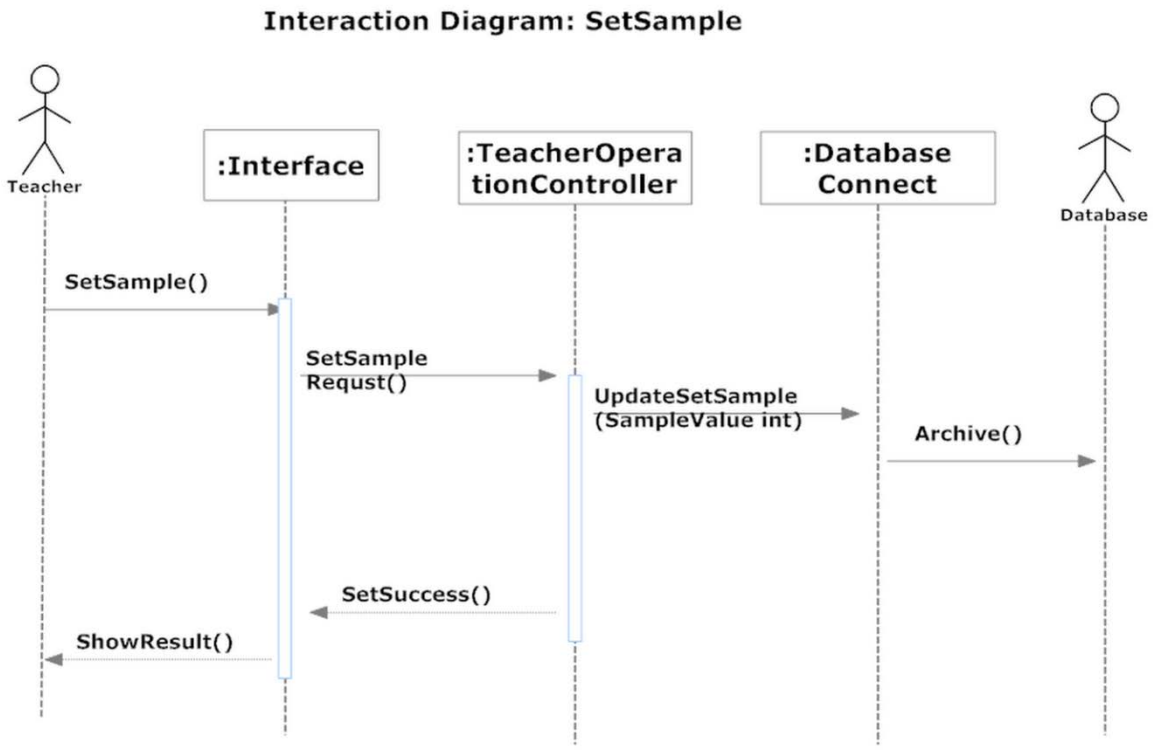

Figure 1-11 ID11 SetSample

1.12 ID12 Login

Figure 1-12 shows how the system authorize users' login. The user send LoginRequst() in the interface, then the interface send the LoginRequst() to the database. In the database, there is a list which stores information of all user. By camparing with the information sent by interface, the database gives appvoral or rejection to the users' request.

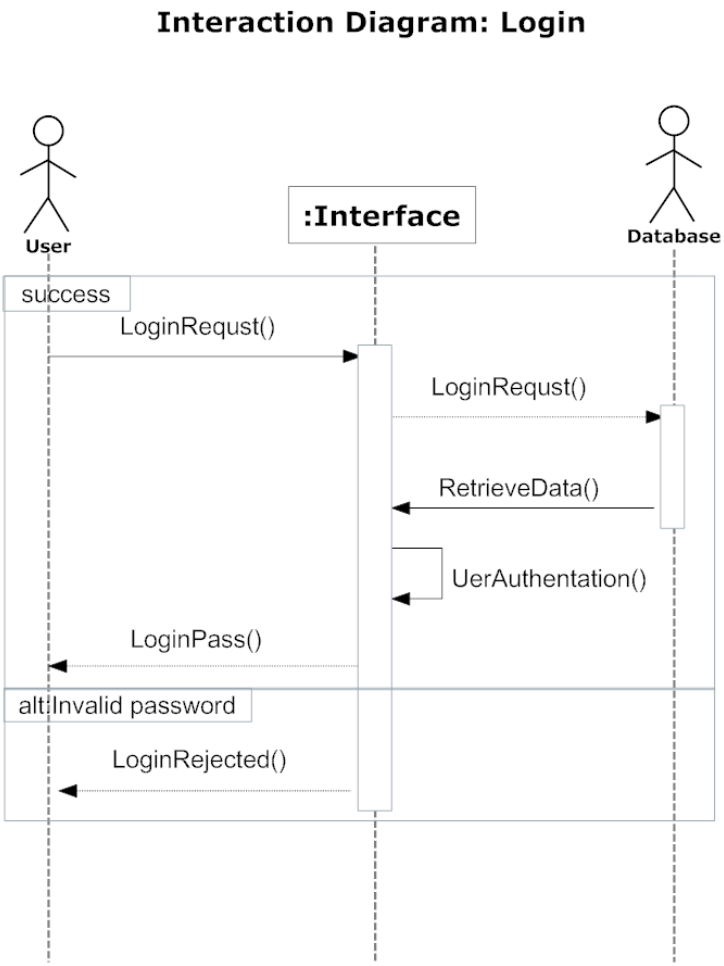

Figure 1-12 ID12 Login

2 Class Diagram and Interface Specification

2.1 Class Diagram

The following diagrams are class diagrams of this software.

Figure 2-1 shows the login mechanism, and classes related to enter teachers' and students' interface.

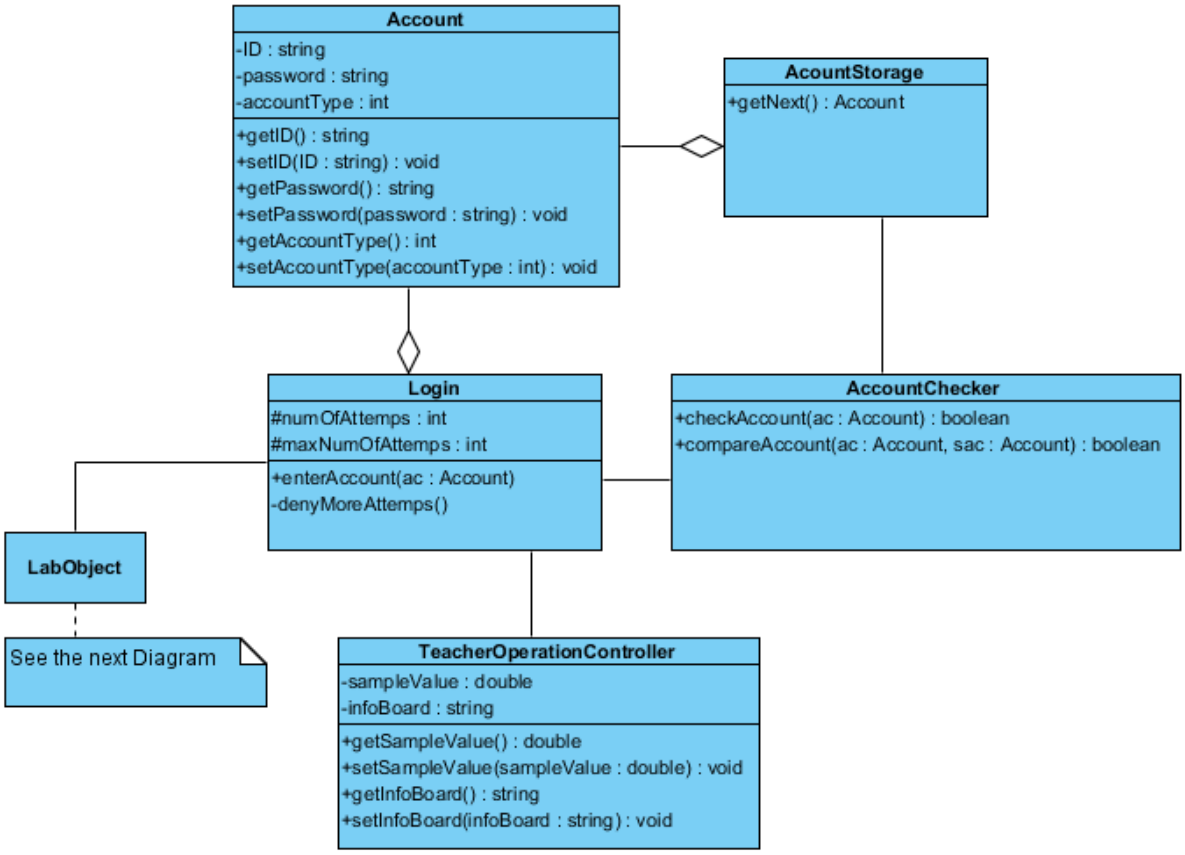

Figure 2-1 class diagram #1

Figure 2-2 shows the major components of spectrophotometer, which contain all classes of operations, and the relationship of all objects of the spectrophotometer.

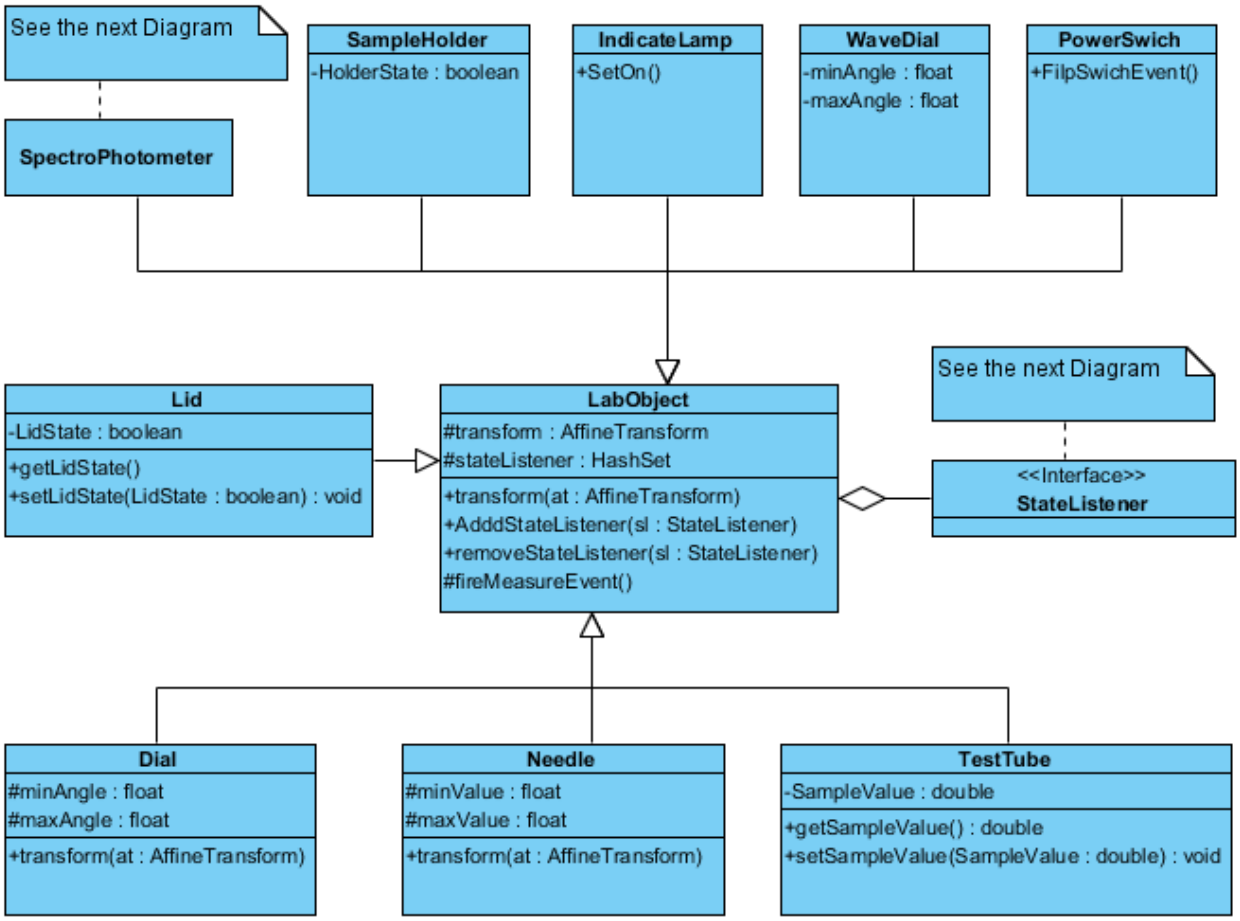

Figure 2-2 class diagram #2

Figure 2-3 is the internal structure of spectrophotometer. It includes the part that store states of components. And by using these states, the software can compute the result of the experiment.

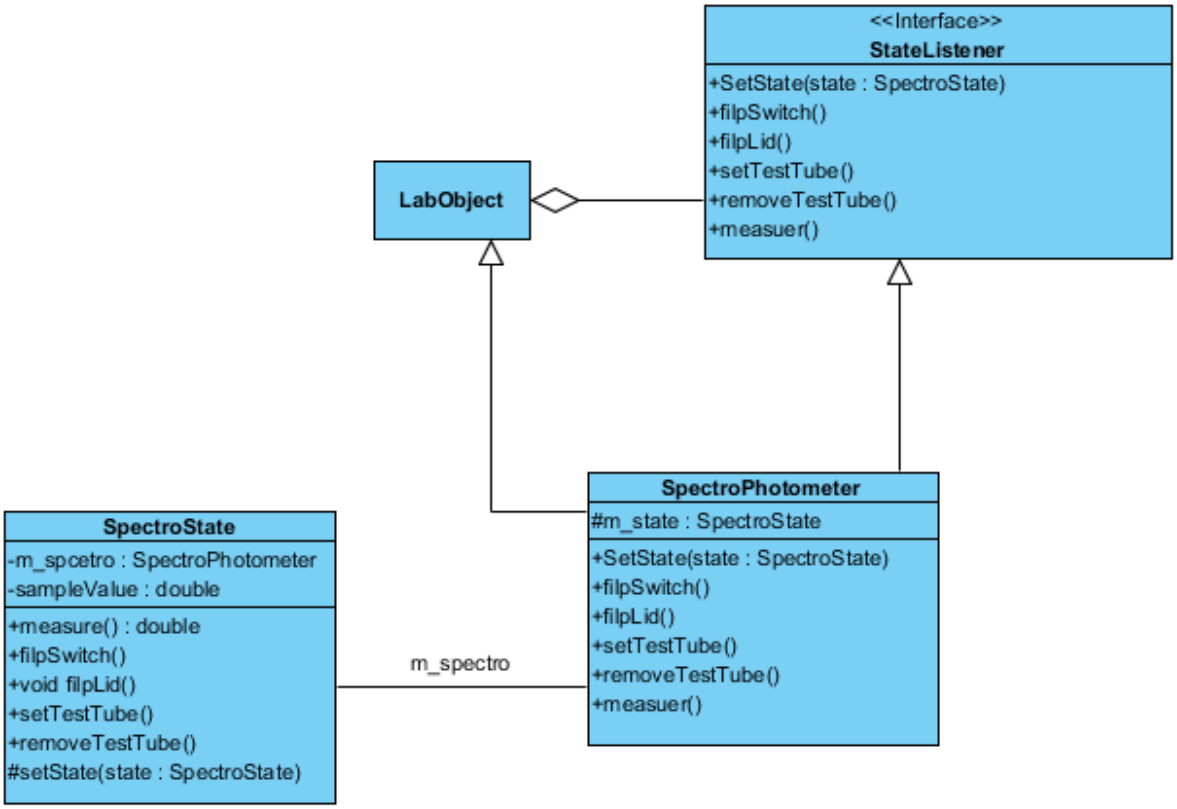

Figure 2-3 class diagram #3

2.2 Data Types and Operation Signatures

2.2.1 Account

An Account class (as Figure 2-1 shows) contains three parts, an ID, a password and an accountType. Because all attributes are private, the operations of this class are all Getters and Setters which are used to get and modify these attributes, respectively.

Class Name:	Account	
Attributes:	-ID: string	Account ID
	-password: string	password related to Account ID
	-accountType: int	types of account
Operations:	+getID():string	Getters and Setters
	+setID(ID): void	
	+getPassword():string	
	+setPassword(Password: string)void	
	+getAccountType():int	
	+setAccountType(accountType: int):void	

2.2.2 Login

This class (as Figure 2-1 shows) is used for login.

Class Name:	Login	
Attributes:	#numOfAttempts: int	Number of login Attempts
	#maxNumOfAttempts:int	the max number of login Attempts
Operations:	+enterAccount(ac: Account)	input ID, password and select account type
	-denyMoreAttempts()	system refuses account that reaches max attempts to login

2.2.3 AccountStorage

This class (as Figure 2-1 shows) is used to authorize account. It returns the next Account in the List of Account.

Class Name:	AccountStorage	
Attributes:		
Operations:	+getNext(): Account	move to the next Account and return it

2.2.4 AccountChecker

It is the class (as Figure 2-1 shows) that checks for the account validity. The mechanism of authorization is compare the Account input by user with each Account stored in the List. If there is an Account exists in the AccountStorage, it returns 1, otherwise, it returns 0.

Class Name:	AccountChecker	
Attributes:		
Operations:	+checkAccount(ac:Account):boolean	check the format of the Account input
	+compareAcccount(ac:Account,sac:Account):boolean	compare the Account user input with Account got from AccountStorage

2.2.5 TeacherOperationController

This class (as Figure 2-1 shows) is designed teachers who can modify some value and information of the system.

Class Name:	TeacherOperationController	
Attributes:	-sampleValue: double	The sample's concentration value
	-infoBoard:string	information that will be posted on the infoBoard
Operations:	+getSampleValue():double	Getter and Setter
	+setSampleValue(sampleValue:double):void	
	+getInfoBoard():string	
	+setInfoBoard(infoBoard:string):void	

2.2.6 LabObject

This is an abstract base class for all parts of the spectrophotometer (shown as Figure 2-2). This is a graphical figure base class that can be manipulated by mouse pointer. The manipulations include click and drag. A dragging manipulation is converted into an affine transformation that will be applied to the corresponding figure.

Class Name:	LabObject	
Attributes:	# transform : AffineTransform	Represents the current affine transformation of the object, relative to its non-transformed prototype.
	# stateListeners : HashSet	Listener objects interested in state changes of this lab object.
Operations:	+ transform(at : AffineTransform)	Applies an affine transformation to this object.
	+ addStateListener(sl : StateListener)	Adds the listener `sl` to the list of listeners.
	+ removeStateListener(sl : StateListener)	Removes the listener `sl`.
	# fireMeasureEvent( )	Calls the method `measure()` on all listener objects.

All objects are in a non-transformed prototype, which means zero translation, zero rotation, and scale is 1 for both *x* and *y* dimension (i.e., resized to 100%). To change the object's translation, rotation, or scale (resizing), just call the method `transform()` with the appropriate transformation parameter.

```
protected transient HashSet stateListeners;

protected void fireMeasureEvent()

{

        HashSet listeners = (HashSet) stateListeners_.clone();

        for (Iterator i = listeners.iterator(); i.hasNext(); ) {

                ((StateListener)i.next()).measure();

        }

}
```

2.2.7 PowerSwitch

Extends LabObject

This object is on/off switch (shown as Figure 2-2). It turns on or off the IndicateLamp when the spectrophotometer is turned on or off (see section 2.2.8). When the switch is "OFF", the device cannot work, which means that the indicateLamp is off, and, in any case, the needle on the meter cannot move.

Class Name:	PowerSwitch	
Attributes:		
Operations:	on mouse click	When mouse pointer is clicked on this object, this method calls `fireFlipSwitchEvent()`.
	# fireFlipSwitchEvent( )	Calls the method `flipSwitch()` on all listener objects.

The list of listeners used in the method `fireFlipSwitchEvent()` is inherited from the base LabObject.

2.2.8 IndicateLamp

Extends LabObject, shown as Figure 2-2.

The indicateLamp (also known as ON/OFF indicator) indicates when the spectrophotometer instrument is turned on (using the on/off switch). It is a part of the absorbance meter. The light is shown as a green oval in the upper right corner of the absorbance meter scale.

Class Name:	indicateLamp	
Attributes:		
Operations:	+ setOn( )	Turns the green light ON or OFF.

2.2.9 Dial

Extends LabObject, shown as Figure 2-2.

There are two dial knobs on the front side of spectrophotometer:

(a) The zero-control dial (also known as “AdjustZero”).
(b) The light control dial (also known as “AdjustBlank”).

Class Name:	Dial	
Attributes:	# minAngle : float # maxAngle : float	Dial's minimum rotation angle. Dial's maximum rotation angle.
Operations:	+ transform(at : AffineTransform)	Applies a rotation transformation.

A knob is shown graphically as a figure consisting of a circle and a line, which represent the knob and the reference mark.

The operation `transform()` allows the user to rotate the knob and set it in the desired position. This operation first calls the operation `transform()` on LabObject (its superclass), and then it informs all the listeners that a measurement should be performed for the new position of the dial. The operation `fireMeasureEvent()` is described in Section 2.2.6.

The code transform(at : AffineTransform):

```
{
   super.transform(at);
   fireMeasureEvent();
}
```

Behavior "measuring" performs the calculations based on the solution's density and the light wavelength and calls the `transform()` method on the instrument needle (described in Section 2.2.12) to display the wavelength. It also turns on or off the pilot lamp when the spectrophotometer is turned on or off. The knob "turning" behavior causes the "measuring" behavior to redo the measurement when a dial is rotated. Similarly, the lid "opening" behavior causes the "measuring" behavior to redo the measurement when the sample holder's lid is opened or closed (described in Section 2.2.13).

2.2.10 TestTube

Extends LabObject

It has an attribute named SampleValue (shown as Figure 2-2), which can be modified by teacher in teachers interface.

Class Name:	TestTube	
Attributes:	-SampleValue:double	Concentration of the sample in the test tube
Operations:	+getSampleValue():double	Getter and Setter
	+setSampleValue(SampleValue: double):void	

2.2.11 WaveDial

Extends LabObject

This object (shown as Figure 2-2) is the wavelength control dial for setting the color of the illumination light. Because this object is normally seen in a side projection but the main view is from the top, there will also be a magnified view of the wavelength dial (shown as top view).

Class Name:	WaveDial	
Attributes:	# minAngle : float # maxAngle : float	Dial's minimum rotation angle. Dial's maximum rotation angle.
Operations:		

2.2.12 Needle

Extends LabObject

This class is the object that displays result in term of spinning the needle by calculation, shown as Figure 2-2.

Class Name:	Needle	
Attributes:	-minAngle:float	shows the smallest value on the meter
	-maxAngle:float	shows the largest value on the meter
Operations:	+transform(at: AffineTransform)	spin the needle

2.2.13 Lid

Extends LabObject

The Lid has two states open and close. The state is highly related to the correctness of result. When the lid is open, measuring the concentration will cause the incorrectness because some day light may leak in to the sample holder. The states of the lid are changed between open and close by clicking the lid.

Class Name:	Lid	
Attributes:	-LidState: boolean	the lid is open or close
Operations:	+getLidState():boolean	Getter and Setter
	+setLidState(LidState: boolean):void	

2.2.14 SampleHolder

Extends LabObject

This object is the sample holder for the test tube. It is a hole where the test tube is inserted for measuring the light absorbed by the substance contained in the test tube. There is also a lid on top of the sample holder (See Section 2.2.13). Several facts should be observed:

- The sample holder can be either empty or hold one test tube
- The holder's lid can be open (lifted) or closed
- The test tube cannot be inserted or removed when the lid is closed
- The measurement should not be correct while the lid is open, because the external light will interfere with the measurement light that illuminates the test tube.

Class Name:	SampleHolder	
Attributes:	-HolderState:boolean	the states of the sample holder, open or close
Operations:		

2.2.15 SpectroState

The following events can be identified for user interaction with the spectrophotometer:

1. Flip the power switch ON or OFF
   Event: "flipSwitch"
2. Flip the lid of the sample holder OPEN or COSED
   Event: "flipLid"
3. Rotate the dial knob

Event “measure”

4. INSERT or REMOVE the test tube to/from the sample holder
   Events: “setTestTube” and “removeTestTube”

There are three state variables that define the spectrophotometer state:

1. Power switch value: ON or OFF
2. Sample holder’s lid value: OPEN or CLOSED
3. Sample holder’s occupancy value: OCCUPIED or EMPTY

Based on these state variables, we define the following states (Table 2-1) of the spectrophotometer:

State Name	State Variable Values	Java Class
OnEmptyClosed	{Switch=ON, Sample-holder=EMPTY, Lid=CLOSED}	OnEmptyClosed.java
OnEmptyOpen	{Switch=ON, Sample-holder=EMPTY, Lid=OPEN}	OnEmptyOpen.java
OffOccupiedClosed	{Switch=OFF, Sample-holder=OCCUPIED, Lid=CLOSED}	OffOccupiedClosed.java
OffOccupiedOpen	{Switch=OFF, Sample-holder=OCCUPIED, Lid=OPEN}	OffOccupiedOpen.java
OffEmptyClosed	{Switch=OFF, Sample-holder=EMPTY, Lid=CLOSED}	OffEmptyClosed.java
OffEmptyOpen	{Switch=OFF, Sample-holder=EMPTY, Lid=OPEN}	OffEmptyOpen.java
OnOccupiedClosed	{Switch=ON, Sample-holder=OCCUPIED, Lid=CLOSED}	OnOccupiedClosed.java
OnOccupiedOpen	{Switch=ON, Sample-holder=OCCUPIED, Lid=OPEN}	OnOccupiedOpen.java

Table 2-1 States of Spectrophotometer

Class Name:	SpectroState	
Attributes:	m_spectro : SpectroPhotometer number: double	 The number shown by default is 5.67

Operations:	+ measure( ) : double
	+ flipSwitch( ) // abstract
	+ flipLid( ) // abstract
	+ setTestTube( ) // abstract
	+ removeTestTube( ) // abstract
	# setState(state : SpectroState)
		Returns the current value of "number".

Note: this is an abstract class. Any specific state has a class that extends the SpectroState in order to get the result.

```
protected  void setState (SpectroState state)
{
        m_spectro.setState(state);
        m_spectro.measure();
}
```

2.3 Traceability Matrix

3 System Architecture and System Design

3.1 Architectural Styles

The main components in our application are the interface and the data processing part. The interface is designed for displaying a virtual Spectrophotometer, providing all the operation dials and returning results of the experiment. While the data processing part is responsible for the user's commands and running the right subsystem model. The architectural styles of our application use service-oriented architectures. Figure 3-1 gives the detailed illustration.

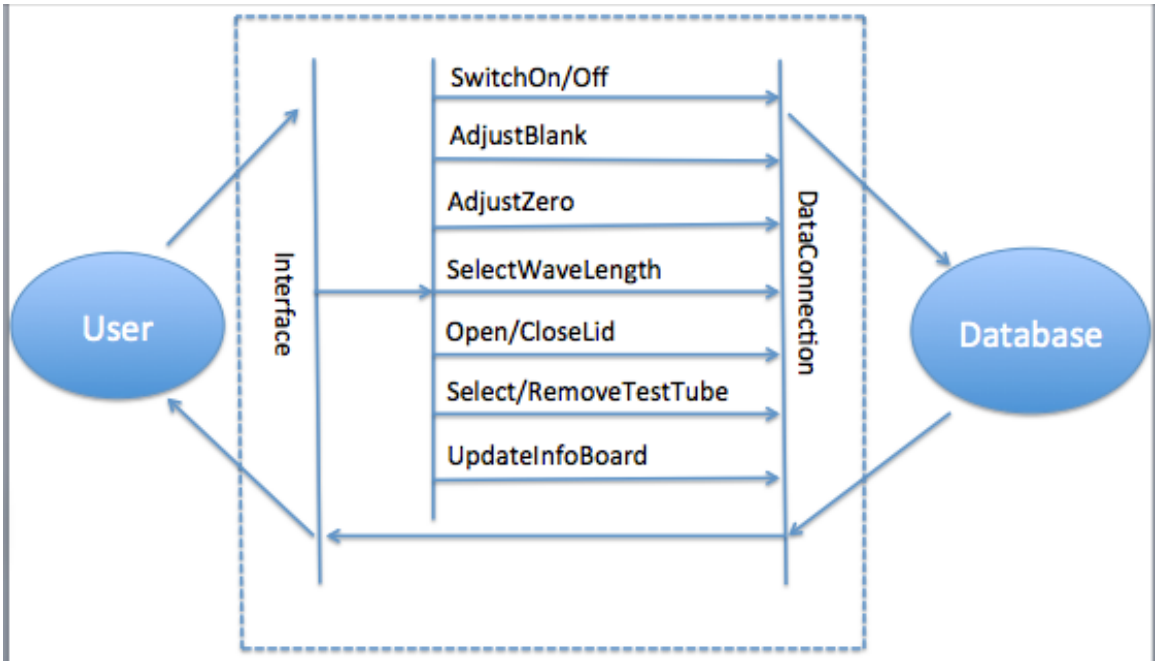

Figure 3-1 Service-Oriented Architectural Styles of Virtual Spectrophotometer

3.2 Identifying Subsystems

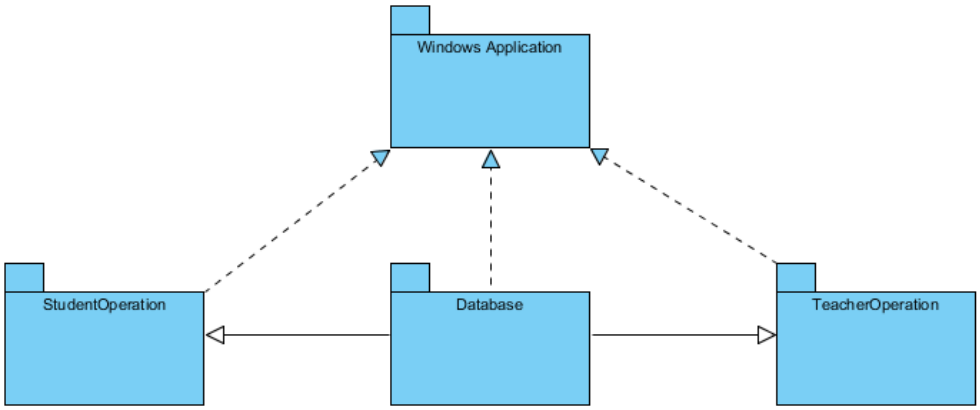

Figure 3-2 Subsystems

Figure 3-1 shows the subsystems in our project. They are StudentOperation, Databse and TeacherOperation.

The first subsystem is StudentOperation Subysystem. It presents a real spectrum like interface to students. Aiming at make student become more familiar with spectrum, this subsystem allows students have operations like switch on, adjust to blank, insert test tube and so on. As for its purpose, it is not a linear process system. It allows students make mistakes which will result in inaccurate result of test sample.

The second subsystem is named Database. As its name, its main function is to maintain the data interaction between interface and database. When students behave like (open/close lid, insert/remove test tube), the information (lid status tag, tube status tag and so on) will be updated in database via this subsystem. Furthermore, this subsystem deliver teacher updated information (like information board and sample value) as well.

The third subsystem is TeacherOpeartion. It allows teacher to update information board which displays tips for students. Also teacher can set up sample value in this subsystem, so that they can see whether student can use this virtual device in a correct way.

3.3 Mapping Subsystems to Hardware

Our system need to store data into database in order to ensure the normal interaction between users and interface. The database can be divided into two parts which are login database, experiment database.

Login database provide a way to valid the username and password of both student and teacher, and update the system status which decide next interface will be students or teachers interface.

When students log into the students interface, they will see our virtual device and do the experiment steps. Then these steps will generate a corresponding data, after that these data will be sent to the experiment database and update the database value. DataCalculator will get this data from experiment database. By comparing the data table, the DataCalculator will find the final result which was influence by sequence of operation. As a teacher, when he login our system, he will see the administration interface. On this interface he can change the value of our test tube meanwhile he can update the database immediately.

We choose flat (table) model as our database model. The flat (or table) model consists of a single, two-dimensional array of data elements, where all members of a given column are assumed to be similar values, and all members of a row are assumed to be related to one another. In our login database Columns are for account type, ID and password (shown as table 3-1). Each row would have the specific password associated with an individual user. Columns of the table often have a type associated with them, defining them as character data, date or time information, integers, or floating point numbers.

Account type	ID	password
1	00001	aaaa
1	00002	aaaa
……	……	……
2	147001234	abcd
2	147003456	Abcd1234

Table 3-1 account

In database for experiment columns represent for adjustZero, adjustWavelength, adjustBlank, lipStatus and tubeValue. Each row will have the specific value associated with different operation. Table 3-2 illustrates data model.

	adjustZero	adjustWavelength	adjustBlank	lipStatus	tubeValue
147001234	1	380	1	1	580
147007864	0	450	1	1	390
147004532	1	350	0	0	750
……	……	……	……	……	……

Table 3-2 Datebase structure

3.4 Persistent Data Storage

We design our system as an event-driven system that user can do whatever they want in the real lab. Each event will change the associated value in the database as we mentioned before. In real lab, the sequence of operations has great influence on experiment result. Identifying these different sequences and responding reasonable results are big issues when considering the execution orders. Each event may change the value in database. However, the value will be not only depended on the kind of event, but also on the sequence of the event. For instance, when you first adjusting zero then selecting the tube and first selecting the tube then adjusting zero,

both methods can generate the value updating in database. But the value is totally different, because the second sequence will lead to a correct result and the first sequence will not.

3.5 Hardware Requirements

Since this is a small application developed by using Java, a computer that supports JRE can run this application. On the official website of Java, we found requirement of java are described as "*Intel and 100% compatible processors are supported. A Pentium 166MHz or faster processor with at least 64MB of physical RAM is recommended. You will also need a minimum of 98MB of free disk space"*. Therefore, we take this as the Hardware Requirements for this application.

4 User Interface Design and Implementation

4.1 GUI overview

As we mentioned before, three interfaces compose our virtual device, and they are login interface, and student interface and teacher interface.

Figure 4-1 loginGUI

The login interface consists three parts as it is showed in the figure1.1. The first part is the blanks of the username and password as it is usually be. Account selection and two buttons are the second part and last part. We design the login interface as simple as it could be in order to reduce the user effort. In the second demo we will add background on the interface and make it more beautiful.

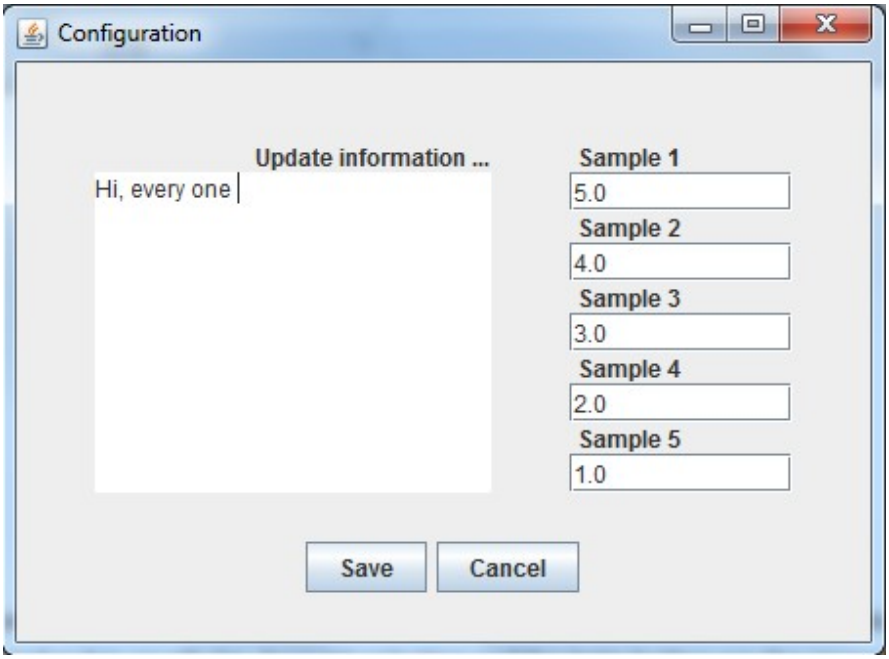

Figure 4-2 TecherGUI

When entering the teacher interface, you will see the teacher interface. This interface is used for updating the information board and setting the test value. Compared with the teacher interface we designed before, the new one changes the distribution of blanks and make it more friendly and accessible. We change the button 'reset' with the button cancel. Because by pressing backspace you can easily delete the word you just typed in. This allows you to retype in a word instead of deleting a sentence.

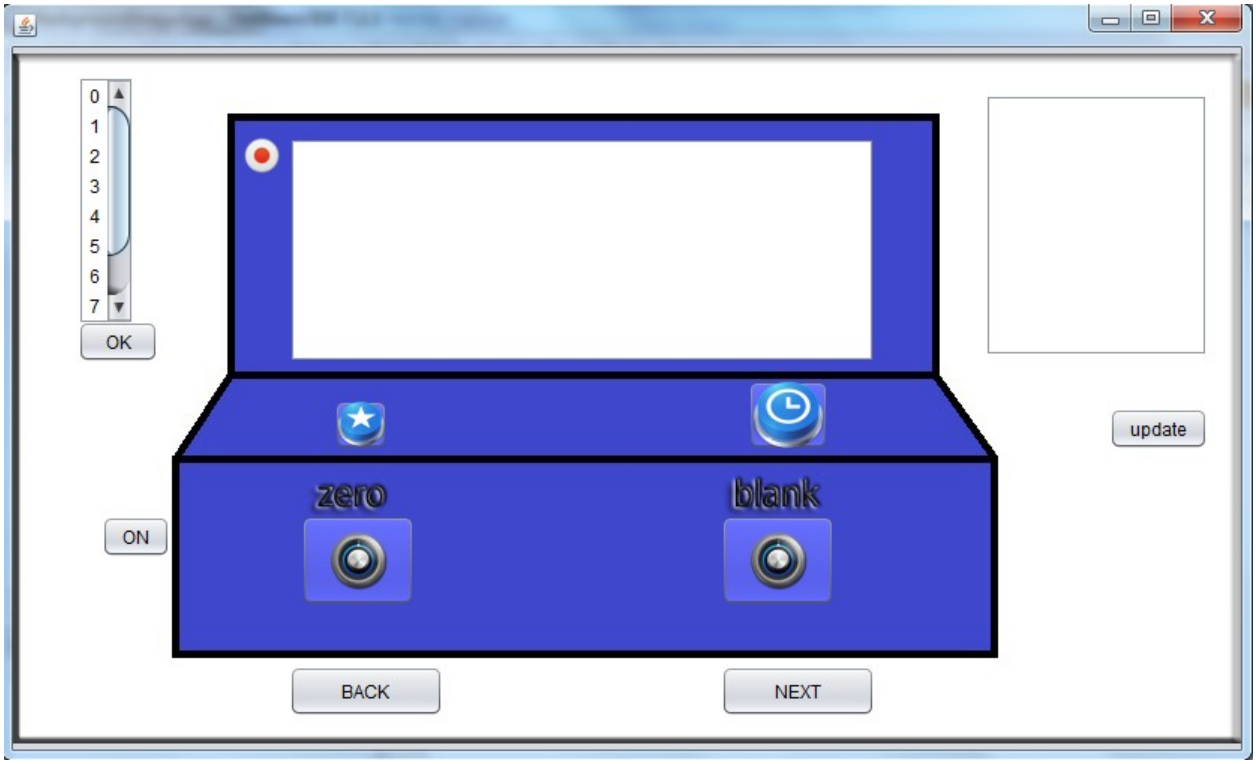

Figure 4-3 TeacherGUI

Student interface design is the most difficulty and importance among the three interfaces. According to our design principle, we will design our virtual machine as students see in the real lab. Unfortunately, in the first demo, we change the meter with a screen and the tube area with a scroll pane because of the time limits. We has realize the first step that trying to realize what students do in the real lab, and next we will focus on realizing what students see in the real lab. These will include the knob animation and the meter design. We haven't reduced the user effort, since we design our user effort according to what they do in the real lab.

4.2 GUI Design

4.2.1 The use of cross-platform color

The number of colors on our system should be considered before GUI design: 8 bits (256 colors), 16 bits (thousands of colors); and 24 bits (millions of colors). There might be dithering each time our java application is run. In order to get the best results, we will optimize our graphics on platforms as many as possible.

4.2.2 The design of application graphics

As for graphic file format choosing, we chose PNG (Portable Network Graphic) format. Compared with GIF format, PNG gives a much wider range of transparency options and color depths and it is usually used in java programs.

Application graphics that we design fall into three broad categories: Icons, Button graphics and Symbols. In icons design, there are several principles: a) Icons should be designed to identify clearly the object or concepts they represent; b) Make sure that large and small icons have similar shape, color, and detail if they represent the same object; c)Specify tool tips for each icon. In button graphics design, there are two principles: a) Use tool tips to help clarify meaning of toolbar buttons; b) Clearly show the action, state, or mode that the button initiates. Symbols include any graphic that stands for a state or a concept. In our application, they appear in alert box, question box, etc. Another important factor is all the graphics should use the same design style.

Some application graphics are not shown in our first demo, which will be shown in second demo.

4.2.3 The use of graphics for product identity.

The graphics for product identity include visual identifier of our application, product logo, information box about the application, etc. We haven't done this work until now but we will design these graphics after finishing all the functions of the application.

4.3 Ease-of-use Design

As we mention before, we will design our virtual device as they do in the real lab. That's also the principle when we considering the user effort. On the login interface, we design this interface as it is common being. We make the teacher's inter face easier to understand and operate, and as a teacher, you can just type in the information on the corresponding place. We change the button 'reset' with the button 'cancel' for the reason the reset function can be realized by pressing backspace on the keyboard. In the student's interface also called device interface, we design our user effort strongly in accord with operation steps which will help students be familiar with the real operation. The only user effort we add is the button 'update', we still design this button on the screen because by clicking this button, we will ensure students will see and pay attention to the information board.

5 Design of test

5.1 Use Cases will be tested

Test-case Identifier: TC-1a **Use-case Tested: UC-1 SwitchOn , main success scenario** **Complete/Fail Criteria: Everything works**	
Test Procedure	**Expected Result**
Step 1 Select student check box and login Step2 Turn on the switch	System display student interface Everything (zero adjust knob, blank adjust knob, wavelength select knob)on virtual spectrophotometer works

Test-case Identifier: TC-1b **Use-case Tested: UC-1 SwitchOn , main fail scenario** **Complete/Fail Criteria: Everything does not work**	
Test Procedure	**Expected Result**
Step 1 Select student check box and login Step2 With the switch not be turned on	System display student interface Everything (zero adjust knob, blank adjust knob, wavelength select knob)on virtual spectrophotometer does not work

Test-case Identifier:TC-2 **Use-case Tested:UC-2 Adjust Blank, main success scenario** **Complete/Fail Criteria:** The meter display the largest scale	
Test Procedure	**Expected Result**
Step 1 Select student check box and login Step2 Turn on the switch Step3 Click lid	System display student interface Machine starts work Lid opened

Step4 Insert blank tube (0) into sample holder	Tube inserted
Step5 Rotate blank adjust knob	The meter display the largest scale

Test-case Identifier: TC-3 **Use-case Tested:UC-3 AdjustZero, main success scenario** **Complete/Fail Criteria:** The meter display the smallest scale	
Test Procedure	**Expected Result**
Step 1 Select student check box and login	System display student interface
Step2 Turn on the switch	Machine starts work
Step5 Rotate zero adjust knob	The meter display the smallest scale

Test-case Identifier:TC-4 **Use-case Tested: UC-4 SelectWavelength, main success scenario** **Complete/Fail Criteria: Meter displays corresponding selected wavelength**	
Test Procedure	**Expected Result**
Step 1 Select student check box and login	System display student interface
Step2 Turn on the switch	Machine starts work
Step3 Rotate the wavelength select knob	Meter displays corresponding selected wavelength

Test-case Identifier:TC-5 **Use-case Tested: UC-5 UpdateInfoBoard main success scenario** **Complete/Fail Criteria: get the newest information**	

Input Data: Information stored in system	
Test Procedure	**Expected Result**
Step 1 Select student check box and login Step 2 Click update button on the right top of the screen	System display student interface The text field will display latest information

Test-case Identifier: TC-6 **Use-case Tested: UC-6, Open lid main success scenario** **Complete/Fail Criteria: The lid is opened**	
Test Procedure	**Expected Result**
Step 1 Select student check box and login Step 2 Click the lid of sample holder	System display student interface The lid is opened

Test-case Identifier: TC-7-a **Use-case Tested: UC-9 Insert test tube, main success scenario** **Complete/Fail Criteria: test tube be inserted successfully**	
Test Procedure	**Expected Result**
Step 1 Select student check box and login Step 2 Click the lid of sample holder Step 3 Choice the number of test tube, and click on OK button	System display student interface The lid is opened The test tube is inserted into the sample holder

Test-case Identifier: TC-7-b **Use-case Tested: UC-9 Insert test tube, main fail scenario** **Complete/Fail Criteria: test tube can not be inserted**	
Test Procedure	**Expected Result**
Step 1 Select student check box and login	System display student interface

Step 2 Choice the number of test tube, and click on OK button	The test tube can not be inserted into the sample holder.

Test-case Identifier: TC-7-a **Use-case Tested: UC-13 Login, main success scenario** **Complete/Fail Criteria: log into the system** **Input Data: valid user name and password**	
Test Procedure	**Expected Result**
Step1 Type in the user name and password and select corresponding check box	Log into the system successfully.

Test-case Identifier: TC-7-b **Use-case Tested: UC-13 Login, main failure scenario** **Complete/Fail Criteria: login failed** **Input Data: invalid user name and password** **Valid user name and password/wrong type**	
Test Procedure	**Expected Result**
Step1 Type in the invalid user name and password and select corresponding check box Step2 Type in the valid user name and password but choice the wrong checkbox (if it is a student account choose the teacher checkbox).	Login failed. Login failed

Test-case Identifier: TC-8 **Use-case Tested: UC-12 SetSampleValue, main success scenario** **Complete/Fail Criteria: The value stored in system is updated by teacher** **Input Data: Sample Value**	
Test Procedure	**Expected Result**
Step 1 Select teacher check box and login Step 2 Type in the sample value in corresponding field	Teacher login success. The sample value stored in system is updated.

Test-case Identifier: TC-9 **Use-case Tested: UC-11 SetInfoBoard** **Complete/Fail Criteria: The InfoBoard is updated** **Input Data: board information**	
Test Procedure	**Expected Result**
Step 1 Select teacher check box and login Step 2 Type in the information in corresponding field	Teacher login success. The infoboard information stored in system is updated.

5.2 Unit Testing test cases

In this section, 3 important objects will be tested.

StudentGUI

Input	Action	Result
Valid student username and password	Login	Login Success
Invalid student username and password	Login	Login Fail

TeacherGUI

Input	Action	Result
Valid teacher username and password	Login	Login Success
Invalid acher username and password	Login	Login Fail

Select TestTube

Input	Action	Result
click on test tube#1	select	select value=1
click on test tube#2	select	select value=2
click on test tube#3	select	select value=3
click on test tube#4	select	select value=4
click on test tube#5	select	select value=5
click on test tube#6	select	select value=6
click on test tube#7	select	select value=7
click on test tube#8	select	select value=8

Insert TestTube

Input	Action	Result
lid = open	insert test tube	Insert Success
lid = closed	insert test tube	Insert Fail

Blank Adjust

Input	Action	Result
switch=off	rotate blank adjust knob	meter displays nothing
switch=on holder = 0 tube	rotate blank adjust knob	meter displays largest scale
switch=on holder = { 1,2,3,4,5,6,7,8 tube}	rotate blank adjust knob	system runs inaccurateresult()

Zero Adjust

Input	Action	Result
switch=on	rotate zero adjust knob	meter displays smallest scale
switch=off	rotate zero adjust knob	meter displays nothing

Wavelength Select

Input	Action	Result
switch=on	rotate wavelength knob	meter displays the selected wavelength
switch=off	rotate wavelength knob	meter displays nothing

5.3 Test Coverage

These 9 .java files listed below will be covered. They possess about 85% of our whole project.

Unit 1 Login in test

In this unit, a Login program is used to test the login in function. In this program, user could choose to login in as student or teacher. The user need to input ID and password which are stored in the database. If the ID and password are not match with the database, the user cannot login in. When the ID, password and teacher/student are all right, it will display “Teacher login in” or “Student login in”, otherwise it will display ”invalid ID or password”.

Unit 2 Adjust Zero

In this unit, a adjustZero program is used to test the adjust zero function. When running this program, the user could set the initial value with zero.

Unit 3 Select Wavalength

In this unit, a selectWave program is used to test selectwavelength function. User could set the wanted wavelength.

Unit 4 Adjust Blank

In this unit, a adjust blank program is used to test adjust blank function. User could set blank state with the given sample solution.

Unit 5 Open/Close Lid

User could open or close the lid by clicking the cover with mouse in this unit. And the program could return a state of the lid.

Unit 6 Change Sample Value

User could set the sample solution with different values. And these values will be saved by clicking ‘’save’’ button.

Unit 7 Update Information board

User can input text in the text box. And by clicking save button, the information can be updated.

5.4 Integration Test Strategy

Our integration test will apply **top-down** integration path. As our project is a interactive system between human and system, every operation will result in different results. It is very common to a class call another class. Therefore the whole programing logic in our project is very crucial. Therefore, by implementing top-down integration test path, we could find faults easier than to separate them into small pieces.

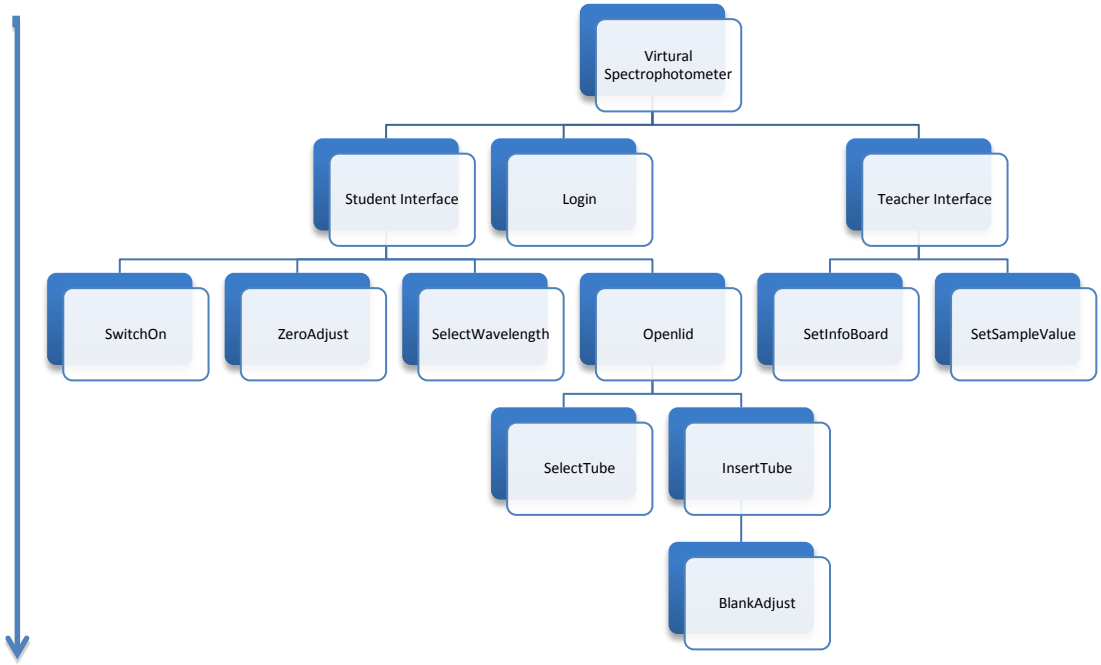

Figure 5-1 Interaction

6 Management and Plan of Work

6.1 Problems and Progress Report

Our project has three parts, LoginGUI, TeacherGUI, and StudentGUI(main part). Our group is divided into two subgroups, one is responsible for coding and the other is responsible for debugging and testing. The first two parts has been finished. Users, teacher and student, can log in their system, respectively, by inputting correct ID, password and account type, which is saved in the AccountStorage file. Teacher can change the information board and save it in a file named infoboard.txt. Student can see what teacher input after they enter their GUI. Also teacher can change the values of samples in the tubes. We are now concentrated on implementing the main part of our project, student GUI. By now, our Demo1 has been finished. We meet some problems, and we are trying our best to fix it. Since our project is based on a previous unfinished work, we have a part of source code of this program. However, we find that some source file is missing. For some classes, we only have .class file but not .java file, which makes us hard to continue to work. We are now facing dilemma that using plug-in to decompile the original file and re-write it, or design a new framework by ourselves.

6.2 Plan of Work

The next step of our work is to complete StudentGUI. Firstly, we are planning to divide our group into two directions, one is to study on the previous works and figure out solution of the problems mentioned above, and the other one is to do some research in order to make our Virtual Device works as a real one. Secondly, we will keep doing programming and testing work. We will try our best to implement all functions of Demo 2 before Dec. 7. We will collect all documents related to this project and finish the third report by Dec. 12 (as figure 5-1 shows).

Task Name	Duration	Start	Finish
Source code analysis	5 days	Mon 11/12/12	Fri 11/16/12
Scientific research	4 days	Wed 11/14/12	Mon 11/19/12
Programming	15 days	Wed 11/14/12	Tue 12/4/12
testing	11 days	Sun 11/25/12	Fri 12/7/12
Demo 2 deadline	0 days	Fri 12/7/12	Fri 12/7/12
finishing Demo 2	3 days	Wed 12/5/12	Fri 12/7/12
Report #3 Deadline	0 days	Wed 12/12/12	Wed 12/12/12
finishing Report #3	4 days	Sat 12/8/12	Wed 12/12/12
finishing Project	2 days	Thu 12/13/12	Fri 12/14/12

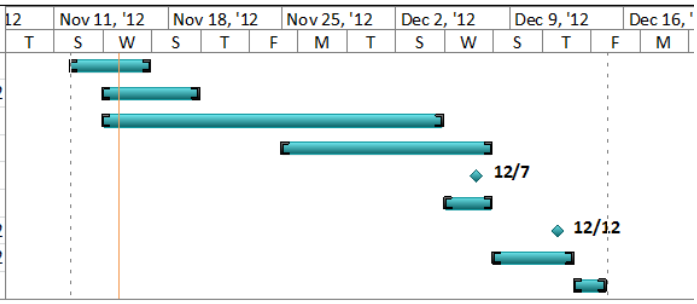

Figure 6-1 Plan of work

6.3 Breakdown of Responsibilities

There are four team members in our team. Their responsibilities is shown as following.

Bingbing Xu : scientific research, testing code and debugging.

Chao Han: Programming (Interaction) and Integration.

Junwei Zhao: Programming(Animation) and Testing.

Xueyuan Song: Programming (GUI).

Reference

[1] Marsic, Ivan. *Software Engineering*, Sep. 10, 2012. URL:http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

[2] Rajaram Subramanian and Ivan Marsic, *ViBE: Virtual Biology Experiments*

[3] UML Sequence Diagram Tutorial. http://www.sequencediagrameditor.com/uml/sequence-diagram.htm

[4] UML Sequence Diagrams: Guidelines. http://msdn.microsoft.com/en-us/library/dd409389.aspx

[5] Russ Miles and Kim Hamilton *Learning UML 2.0*

[6] Java look and feel design guidelinesURL: http://www.oracle.com/technetwork/java/index-136139.html

[7] User interface design URL: http://en.wikipedia.org/wiki/User_interface_design

[8]Java Testing and DesignURL: http://pttdownloads.s3.amazonaws.com/JavaTestAndDesign_Cohen.pdf

[9]Usability URL: http://en.wikipedia.org/wiki/Usability