Software Engineering

Group Number Five

The Rutgers Virtual Biology Laboratory

Website: http://sjlu.github.com/Virtual-Biology-Lab

March 11, 2012

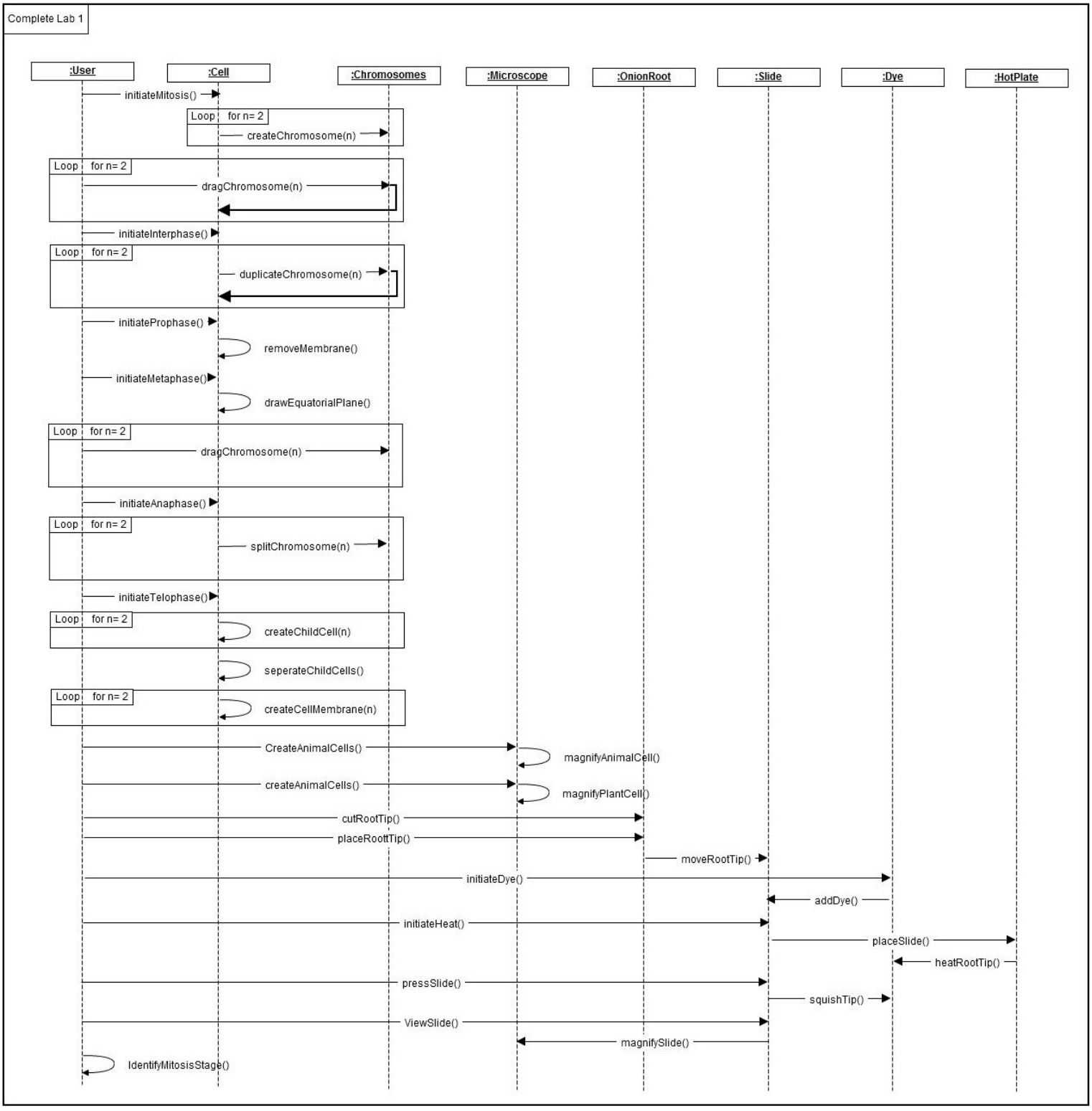
Ryan Cullinane, Michael DiLalo, Nicholas Guida, Steven Lu, Kevin Miller, Cady Motyka

Report Two

1. Interaction Diagrams

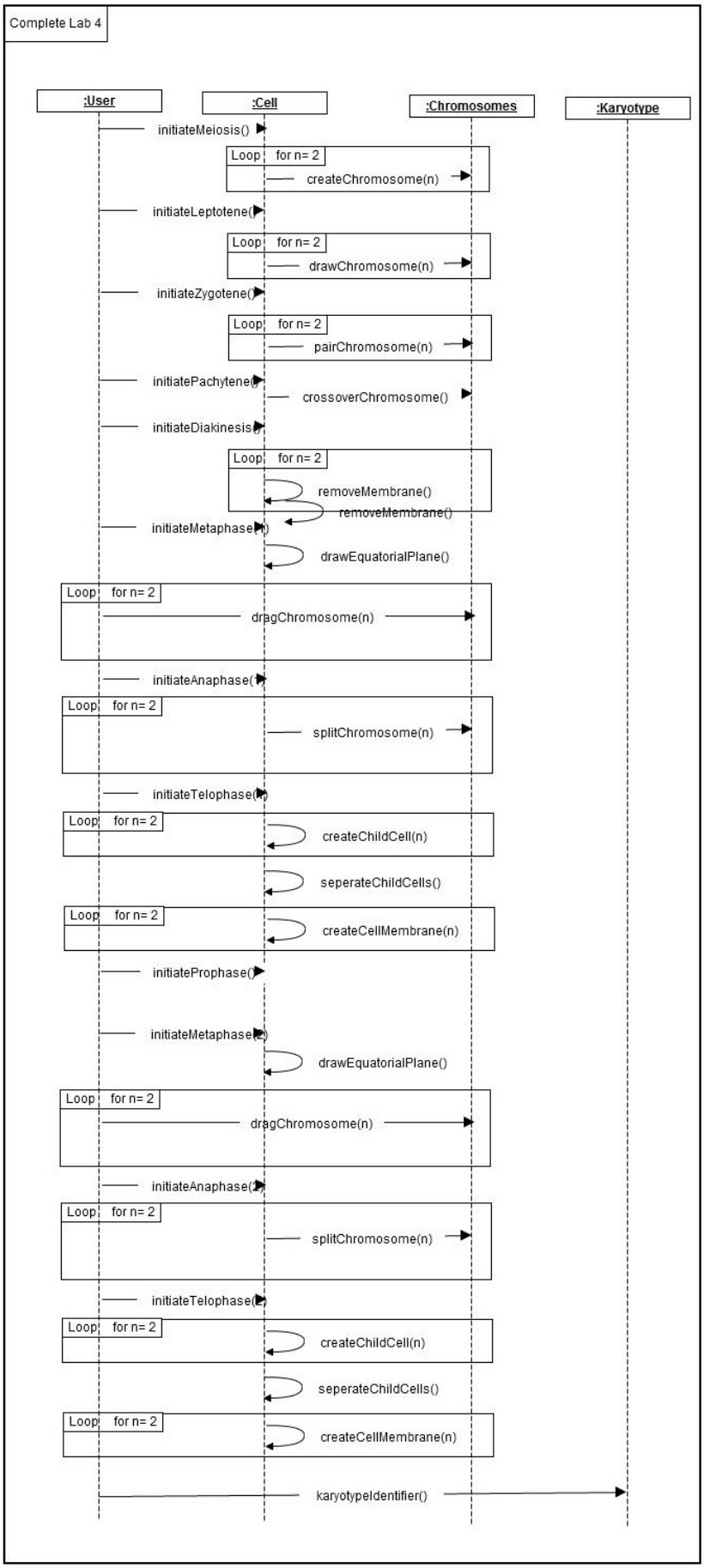
1.1 Interaction Diagrams

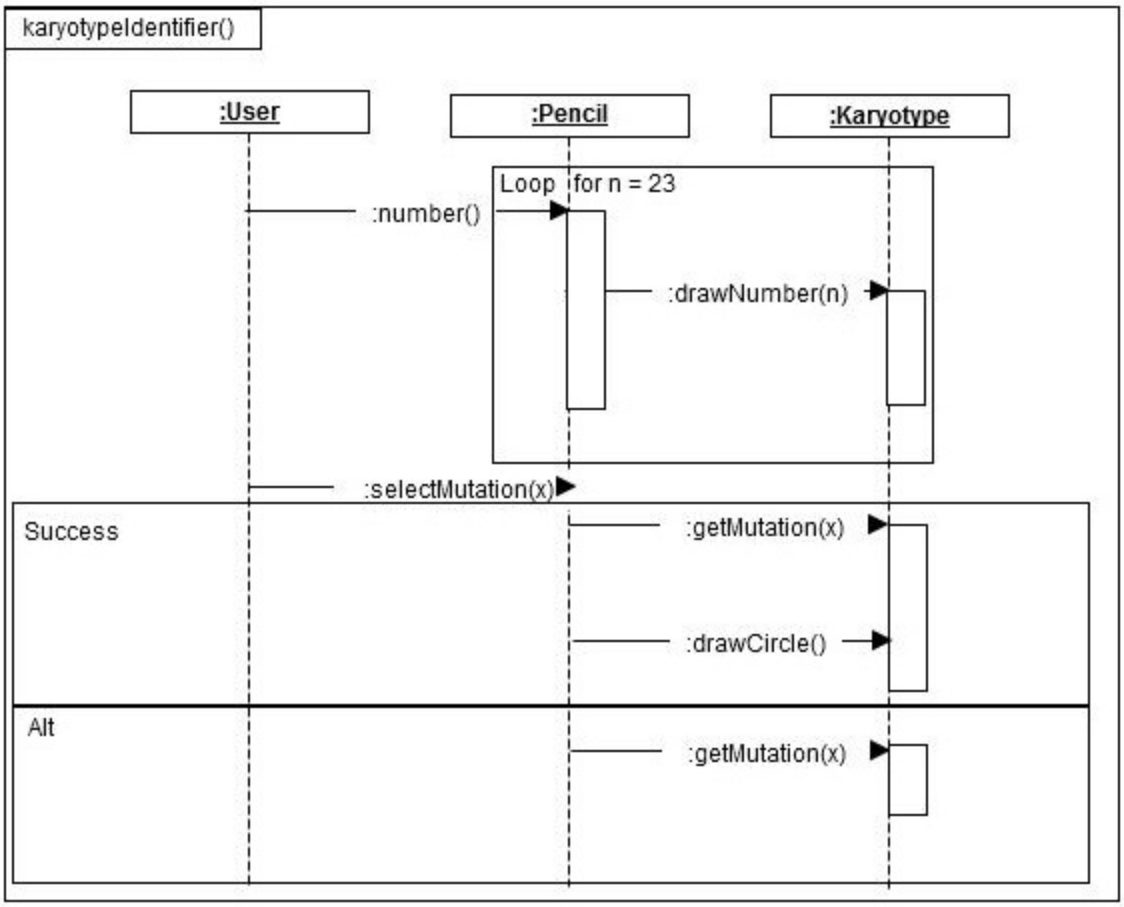
1.1.1 Laboratories 1 and 4



On the previous page is the fragment interaction diagram for the first laboratory of the Virtual Biology Laboratory. In this laboratory the student will prepare a slide of an onion root and observe the process of cell division by mitosis. This diagram shows that first the system draws the chromosomes and then has the student drag them into the correct space. Then the user must initiate each stage of mitosis. Once prophase is initiated, the nuclear membrane disappears. Once metaphase is initiated, the equator is drawn across the cell. Each individual step is shown above. After this is finished, the interaction diagram shows that the student would then move onto creating the slide of the onion root.

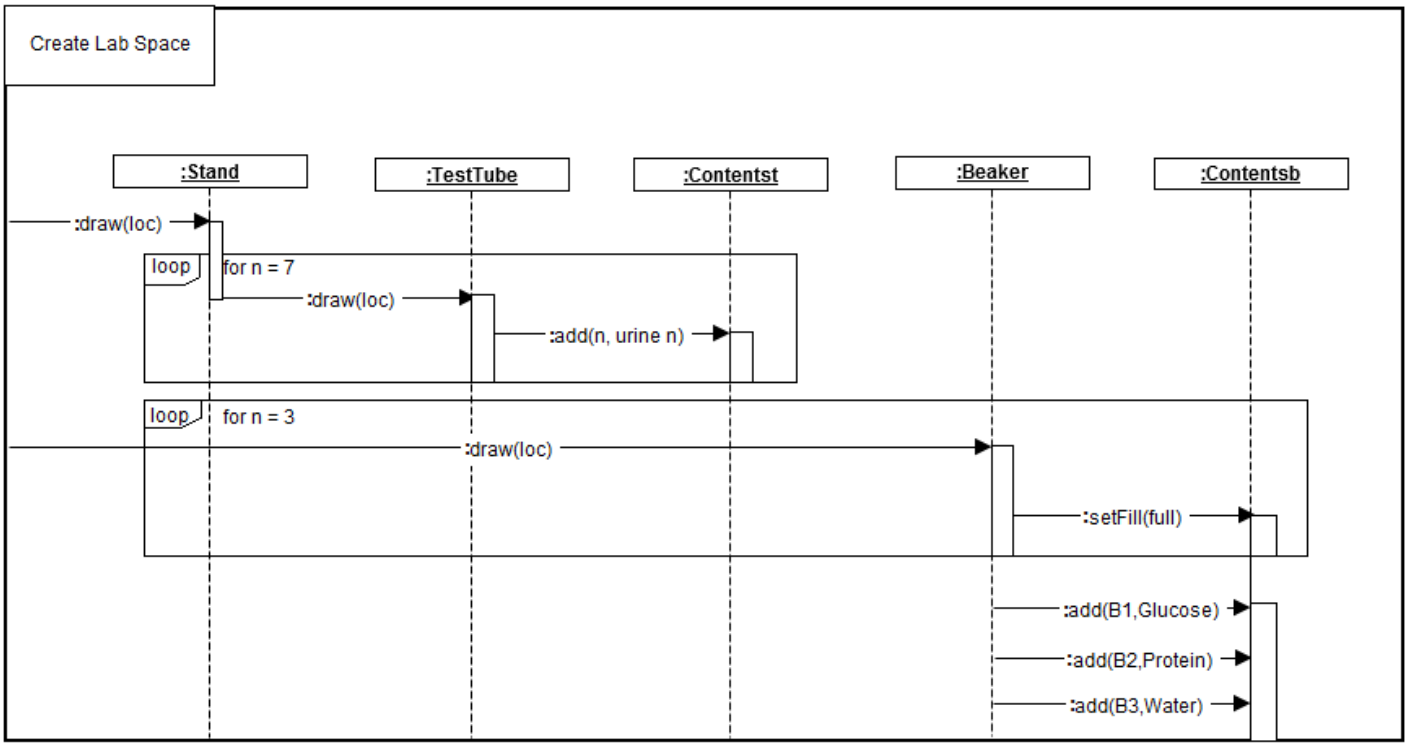
On the next page is the fragment interaction diagram for the fourth laboratory. The first part of this laboratory is to observe a different type of cell division called meiosis. The process of meiosis is similar to mitosis, but more of the steps are repeated in order to have the resulting cells have the correct amounts of genetic material. One of the differences between meiosis and mitosis, is that there are many more steps to prophase. Every interaction for the second part of lab four is shown in the karyotype intraction diagram fragment.


Complete Lab 4
:User
:Cell
:Chromosomes
:Karyotype
initiateMeiosis()
Loop for n= 2
createChromosome(n)
initiateLeptotene()
Loop for n= 2
drawChromosome(n)
initiateZygotene()
Loop for n= 2
pairChromosome(n)
initiatePachytene()
crossoverChromosome()
initiateDiakinesis()
Loop for n= 2
removeMembrane()
removeMembrane()
initiateMetaphase()
drawEquatorialPlane()
Loop for n= 2
dragChromosome(n)
initiateAnaphase()
Loop for n= 2
splitChromosome(n)
initiateTelophase()
Loop for n= 2
createChildCell(n)
seperateChildCells()
Loop for n= 2
createCellMembrane(n)
initiateProphase()
initiateMetaphase()
drawEquatorialPlane()
Loop for n= 2
dragChromosome(n)
initiateAnaphase()
Loop for n= 2
splitChromosome(n)
initiateTelophase()
Loop for n= 2
createChildCell(n)
seperateChildCells()
Loop for n= 2
createCellMembrane(n)
karyotypeIdentifier()

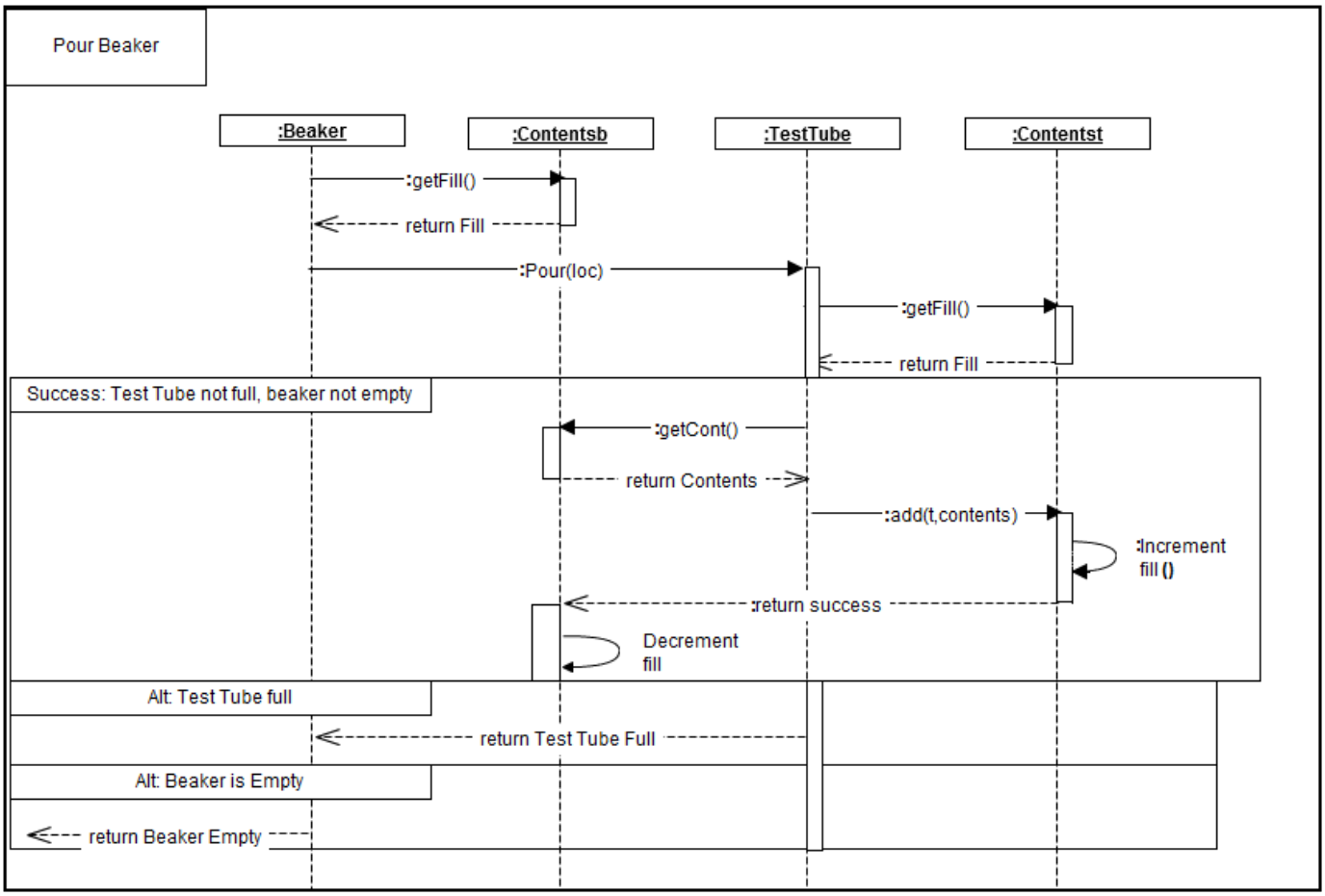


This interaction diagram fragment shows the last part of lab four. In this laboratory the student will be given a picture of a karyotype with human chromosomes. they will then number all of the chromosomes. After this has been shown, the student will have to select which chromosome has the mutation. If they select correctly the chromosome will appear circled. If they select incorrectly the system will return and error and ask them to select again.

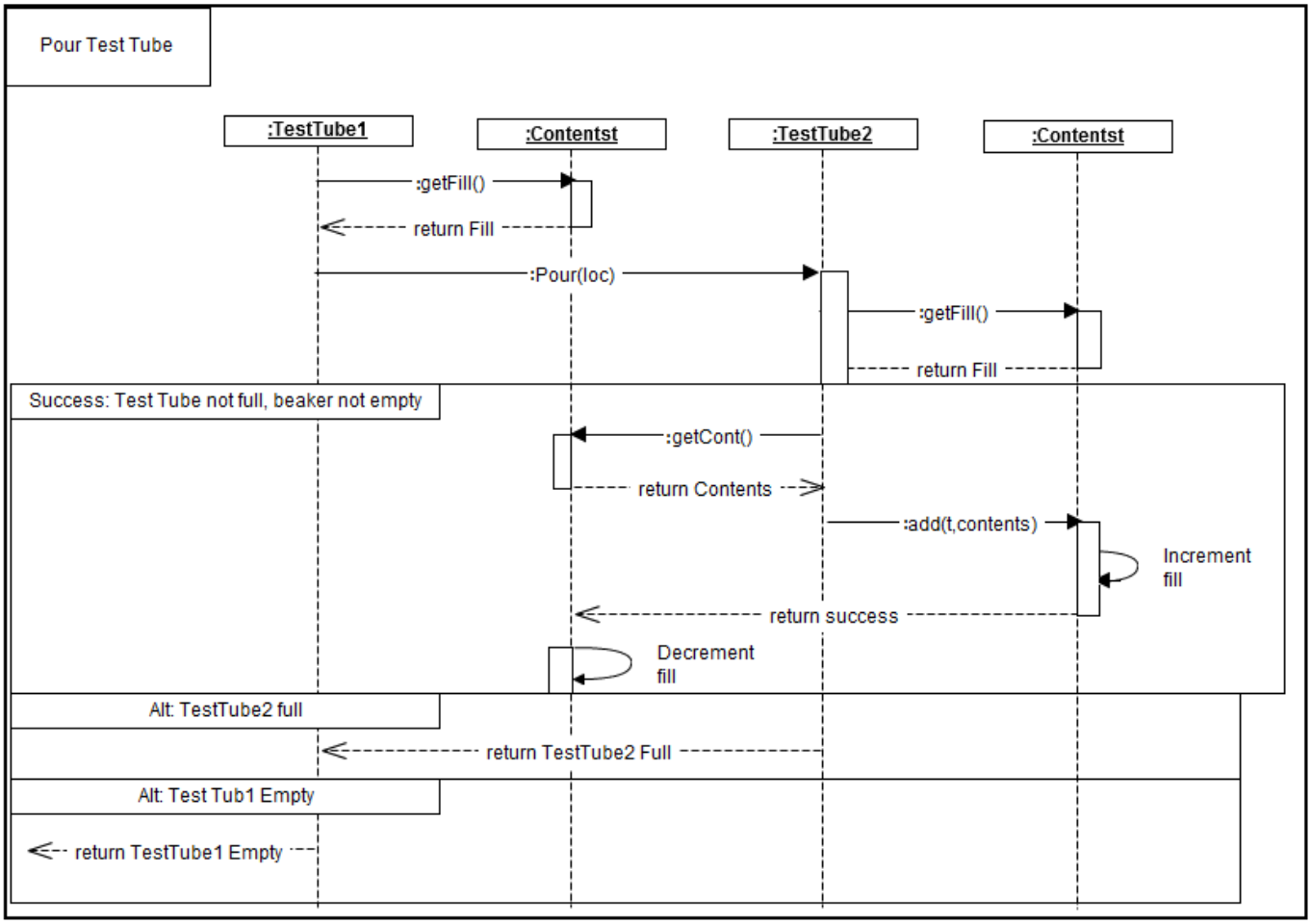
1.1.2 Laboratories 2 and 3



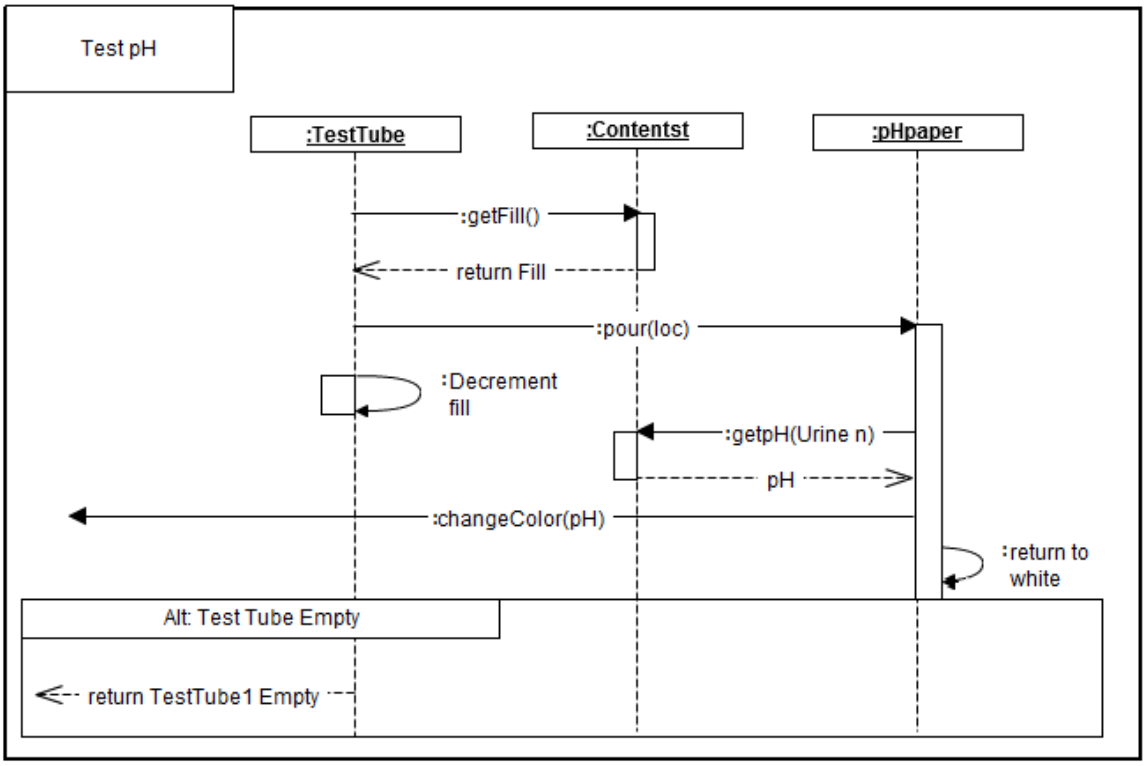
This is the interaction diagram fragment for creating the lab space for the second and third laboratory. In this part the system draws out the stand, test tubes and beakers, and then initializes the right test tubes with the correct liquids and initializes the beakers with the correct liquids.



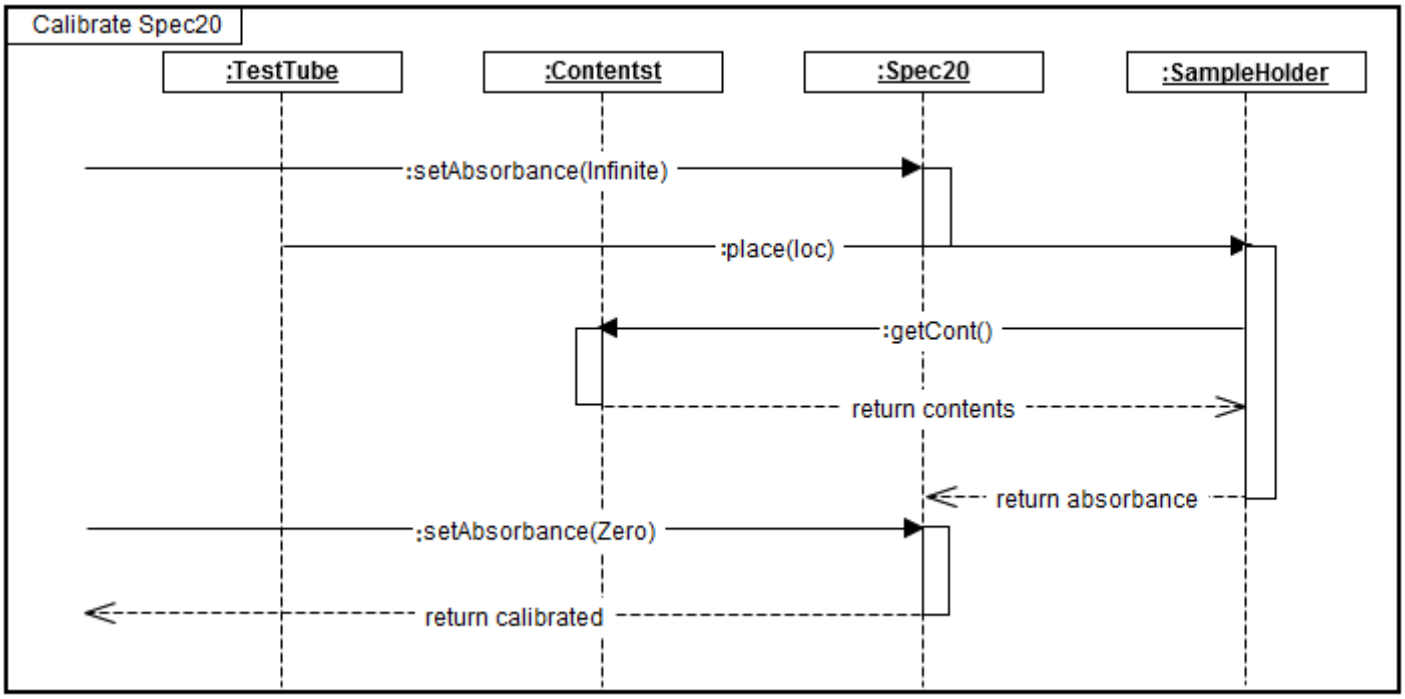
This interaction diagram fragment is the interactions of pouring a beaker into a test tube. It shows how the system check to see if either the beaker is empty or the test tube is full, and then how it transfers the contents from the beaker to the test tube, and then increments the fullness of the test tube and decrements the fullness of the beaker.



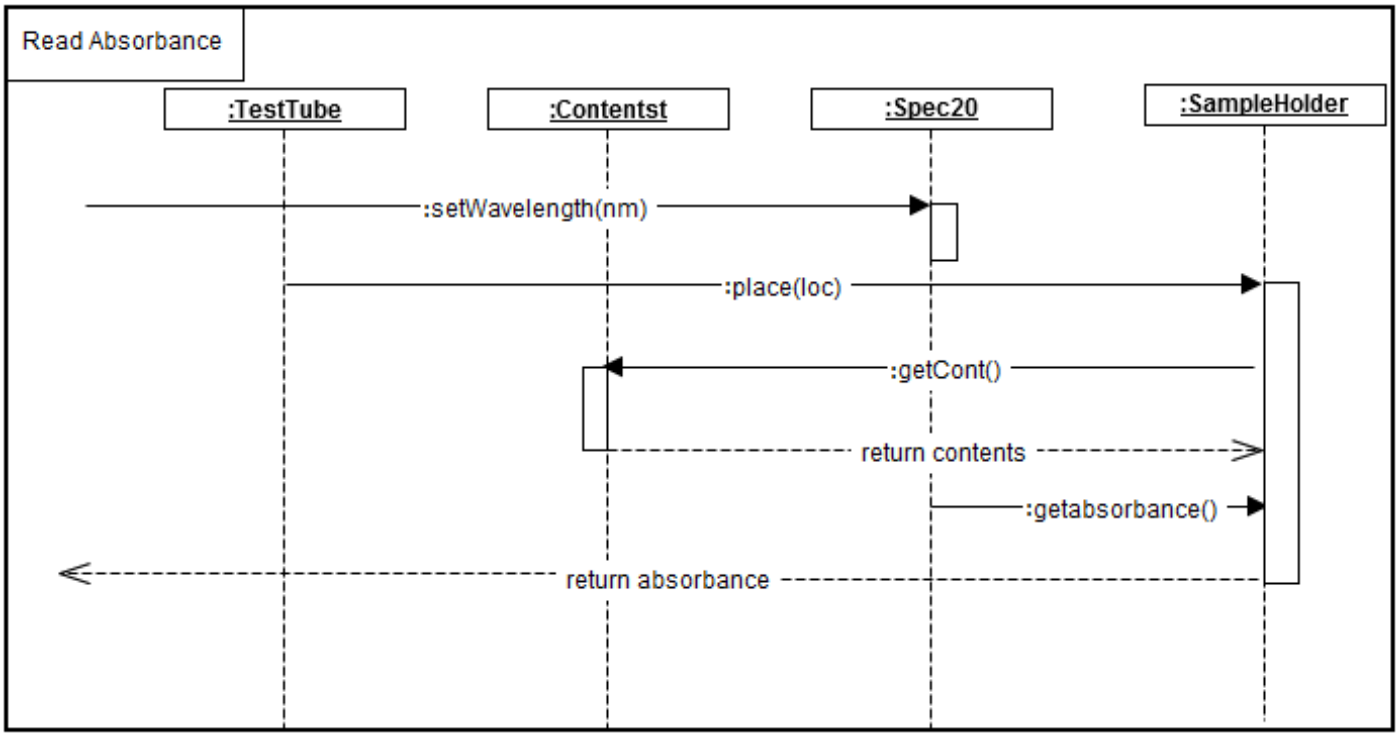
This is the interaction diagram fragment for pouring the test tube. In laboratory two, the test tube is poured onto the pH paper and in laboratory three, the test tube is poured into another test tube. this fragment shows how the system will check if either test tube is full or empty. Then once the contents of one test tube have been read by the other and added to its contents, the fullness of the second test tube is incremented and the fullness of the first test tube is decremented.



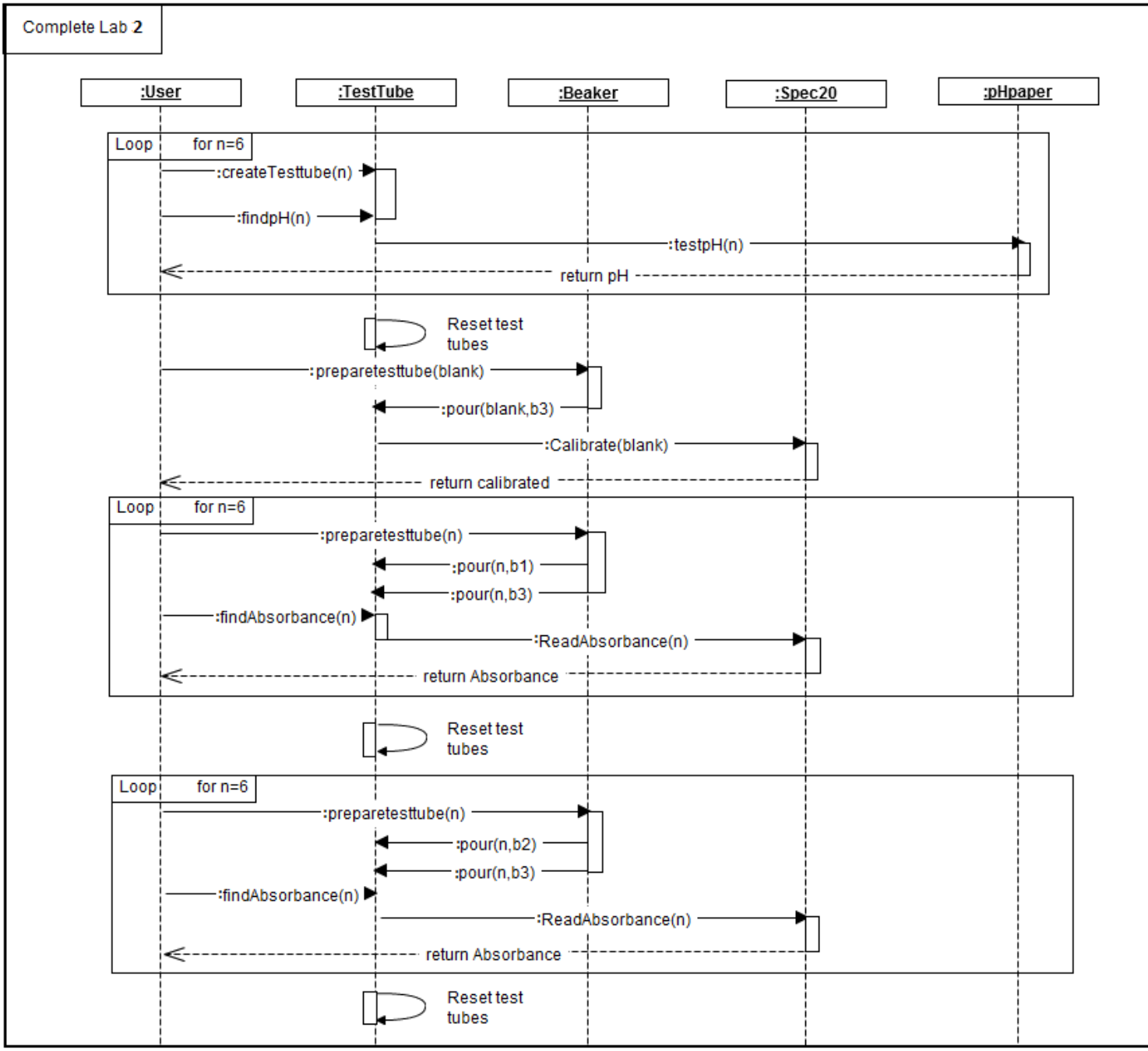
This interaction diagram fragment shows the interactions involved with testing the pH of the contents of a test tube. It hows how the test tube's contents are poured into the pH paper and then the pH paper accesses the number corresponding to the pH from the contents of the test tube. The pH paper then uses that number to change into the color corresponding to the pH number it received. The pH paper then returns to white.



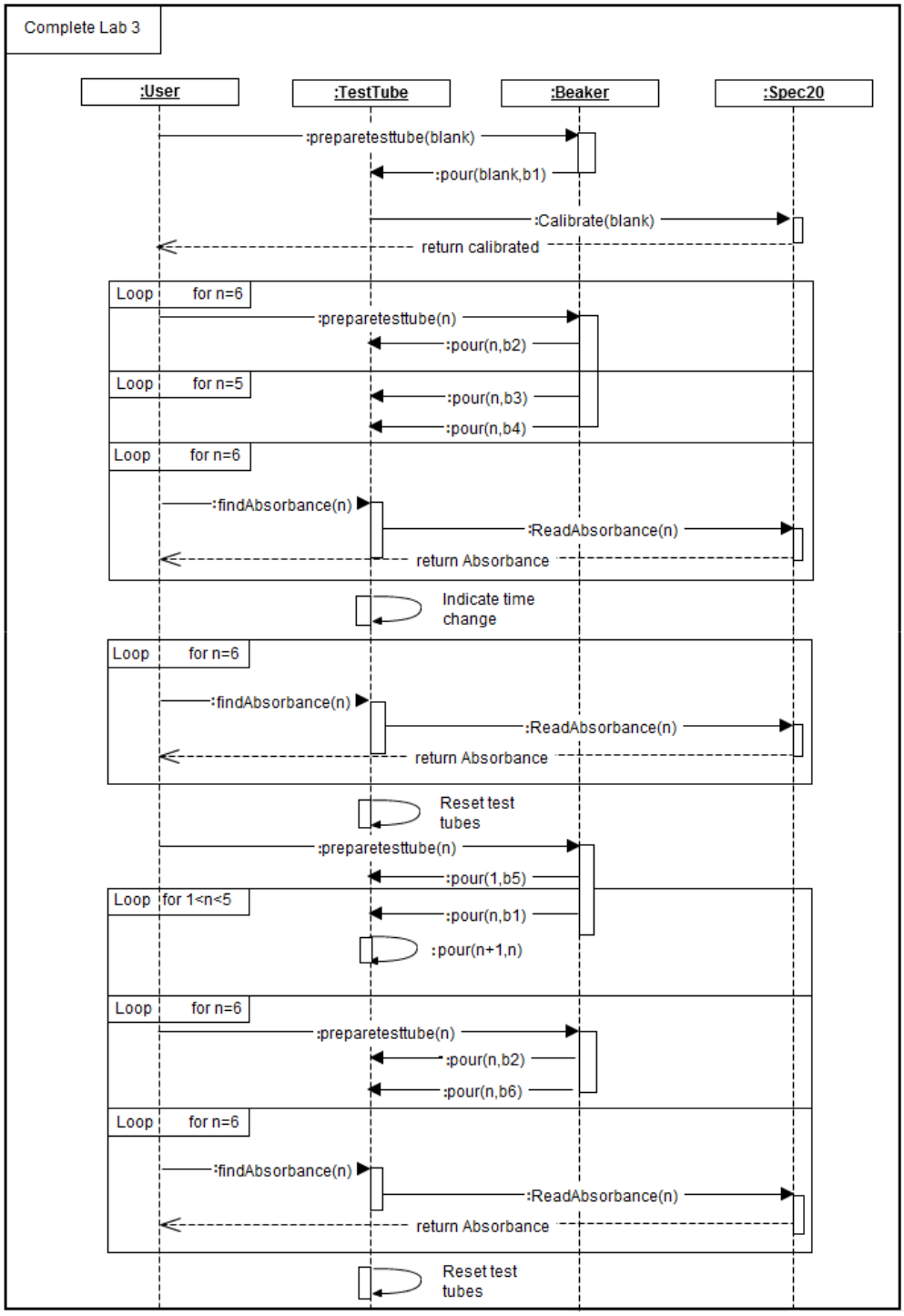
This interaction diagram fragment shows the interactions involved with calibrating the spec20 spectrophotometer. It involved the user setting the infinite absorbance, and placing the bank test tube within the sample holder and checking the absorbance. Then the user must set the absorbance to zero and run it again, the machine then would return that it is calibrated.



This is the interaction diagram fragment that shows how the user uses the system to read the absorbance of a test tube. If the Spec20 is calibrated, then the user can place a test tube within its sample holder. The spec20 would then get the contents of that test tube and look for its absorbance and then return that number that it found.



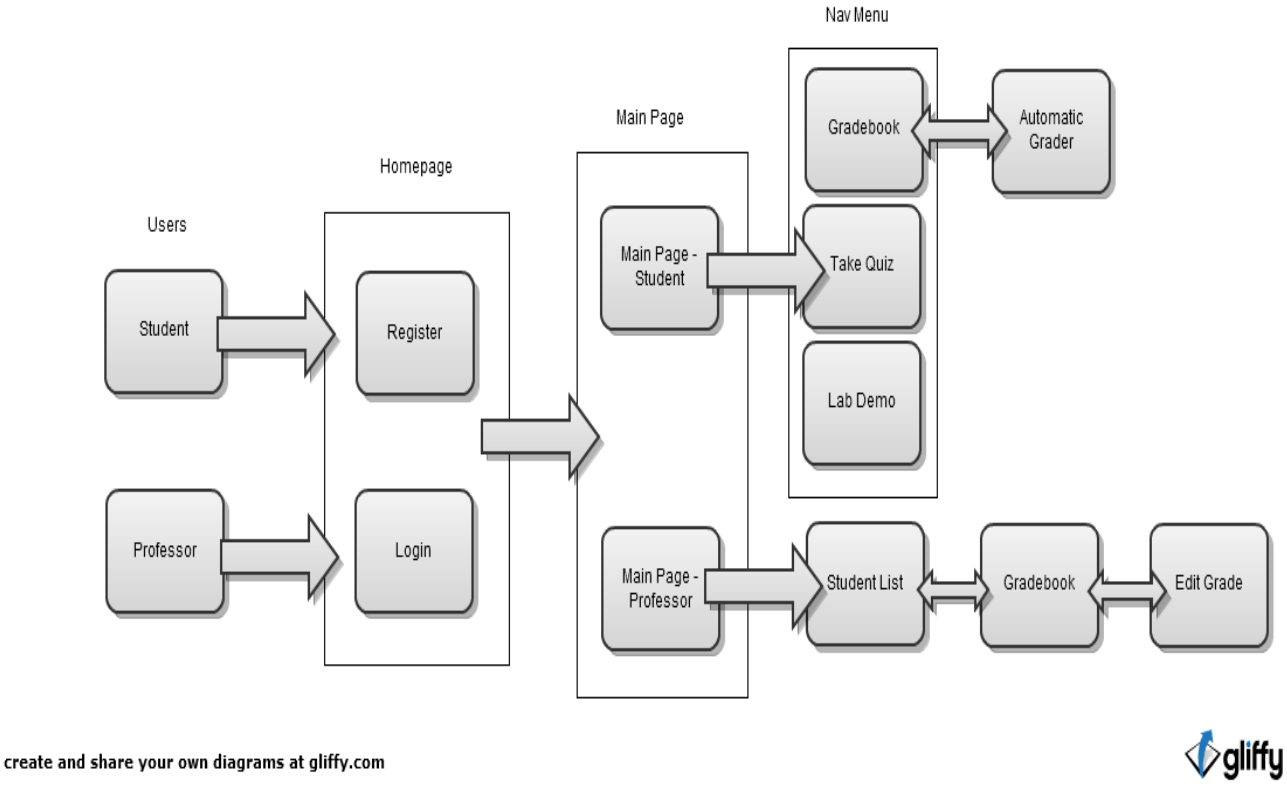
[REF: Create Lab Space, Pour Beaker, Test pH, Calibrate Spec20 and find Absorbance]
This is the full interaction diagram for the second laboratory: biological molecules. The user is adding to the test tubes, checking their pH and then checking for their absorbance. This interaction diagram uses the fragments shown above like creating the lab space, pouring the beaker, testing pH, calibrating the Spec20 and then finding the absorbance. If the user tried to do any of these steps in the wrong order, the system would return an error saying that the previous step had not been completed correctly.



[REF: Create Lab Space, Pour Beaker, Pour Test Tube Test pH, Calibrate Spec20, find Absorbance.]

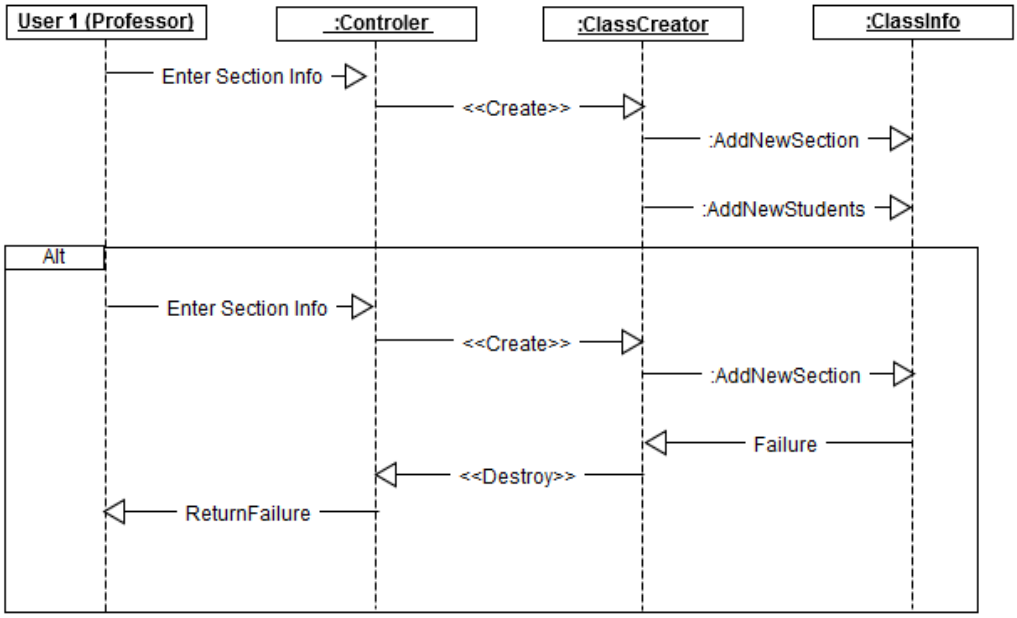
This is the interaction diagram for completing the third laboratory: Enzyme activity. It also uses the interaction diagram fragments as laboratory two, but there are some changes made in what the contents of the beakers and test tubes are. Just like in that laboratory though, the interactions between test tubes, beakers and the spec20 are the same.

1.2 Non functional sequence diagrams



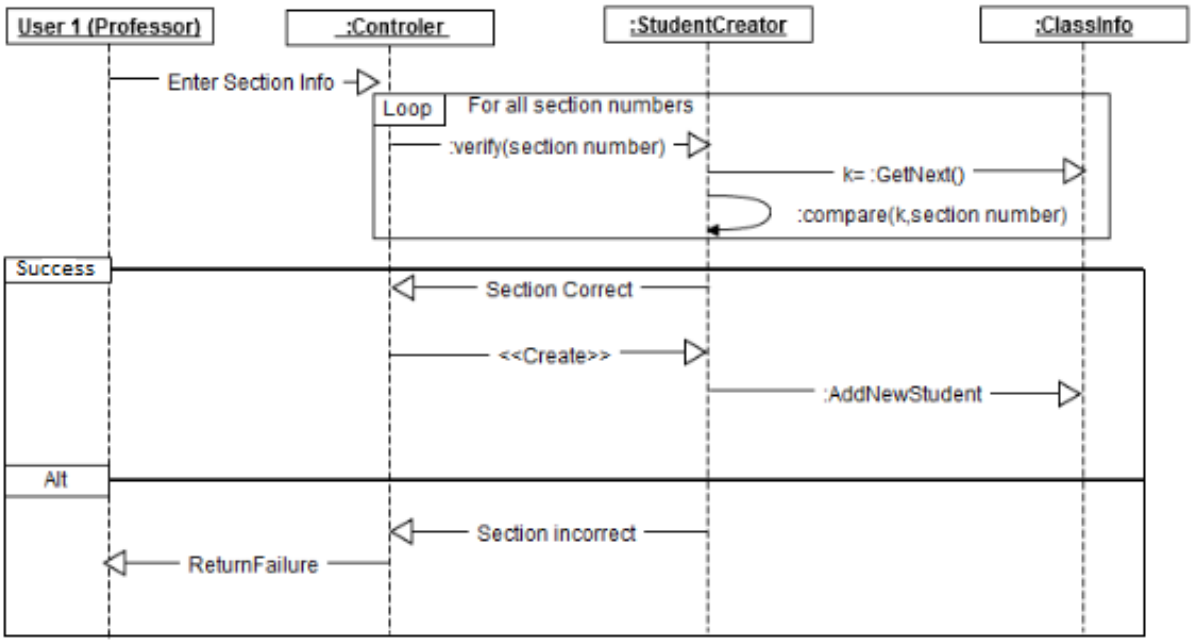
The interaction diagram displayed here shows the flow between the UI elements that are visible in the system. There are several grouped, main functions. These include, the users, the homepage, the main page and the navigation menu. Each of these elements have sub-elements in which are distinguished differently between the students and the professors.

Sequence Diagram Register Class

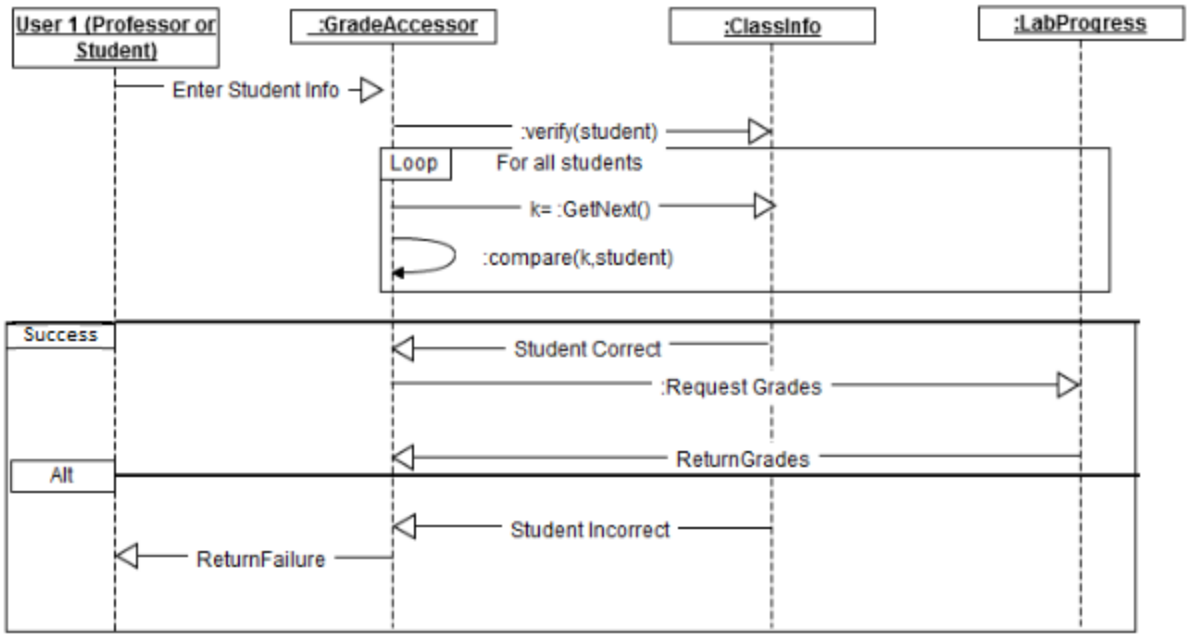


This diagram shows the reaction the system has to the use case RegisterClass.

Sequence Diagram Register Student

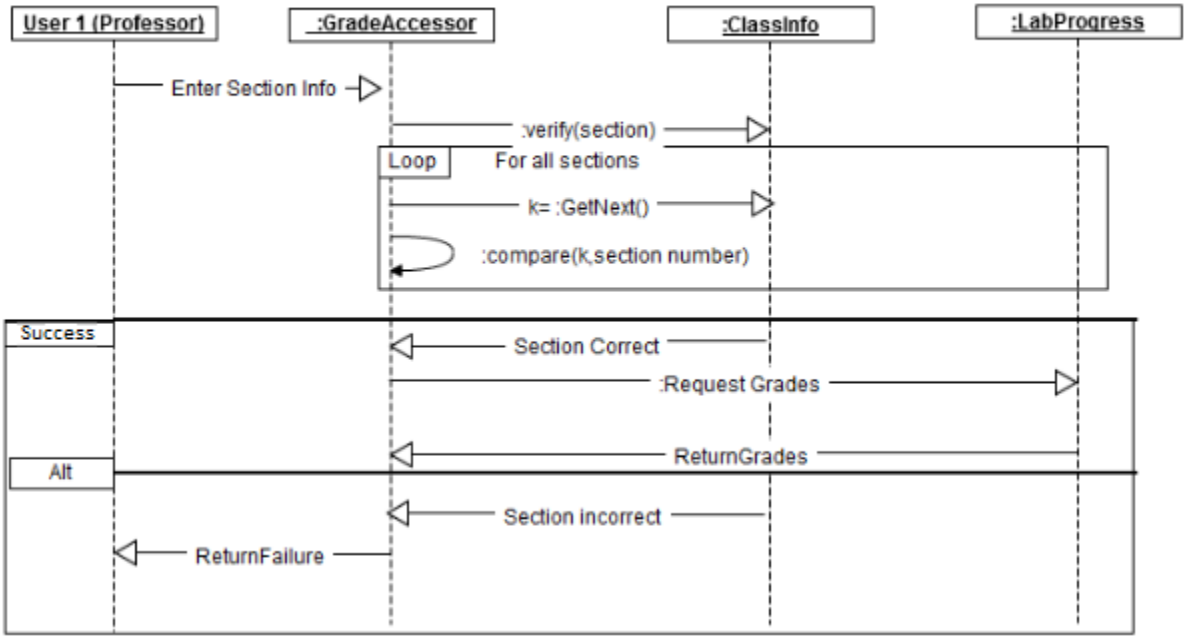


Sequence Diagram View Student Result



This diagram shows the reaction the system has to the use case ViewSudentResult

Sequence Diagram View Class Result

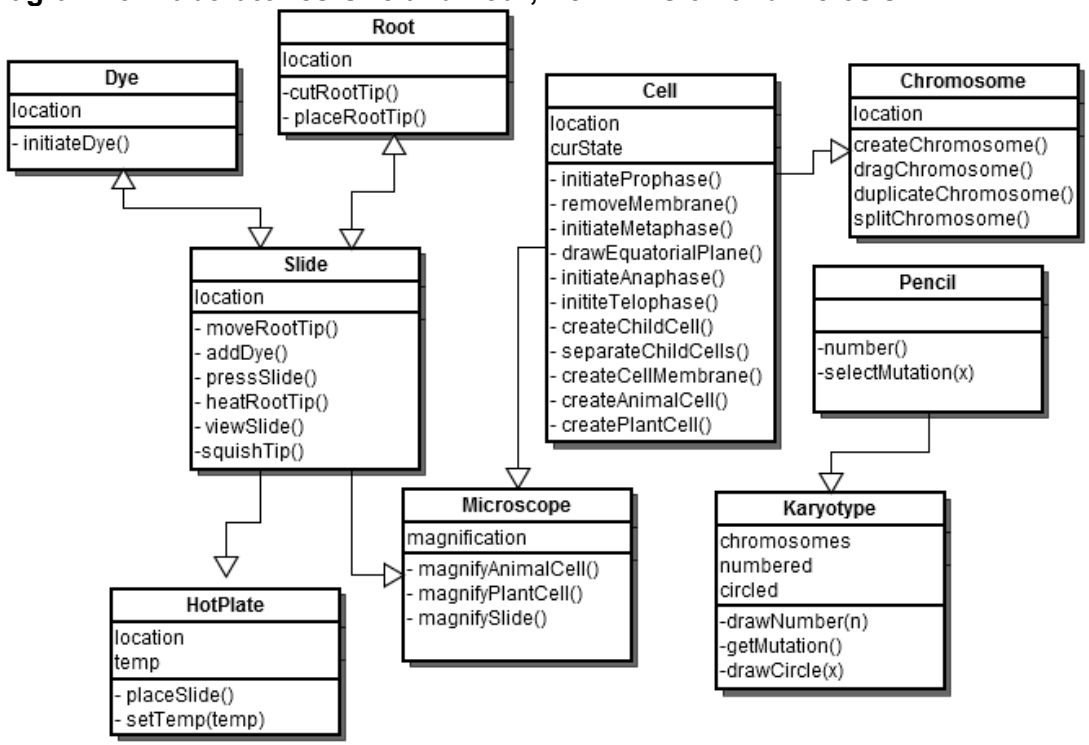


This diagram shows the reaction the system has to the use case ViewClassResult

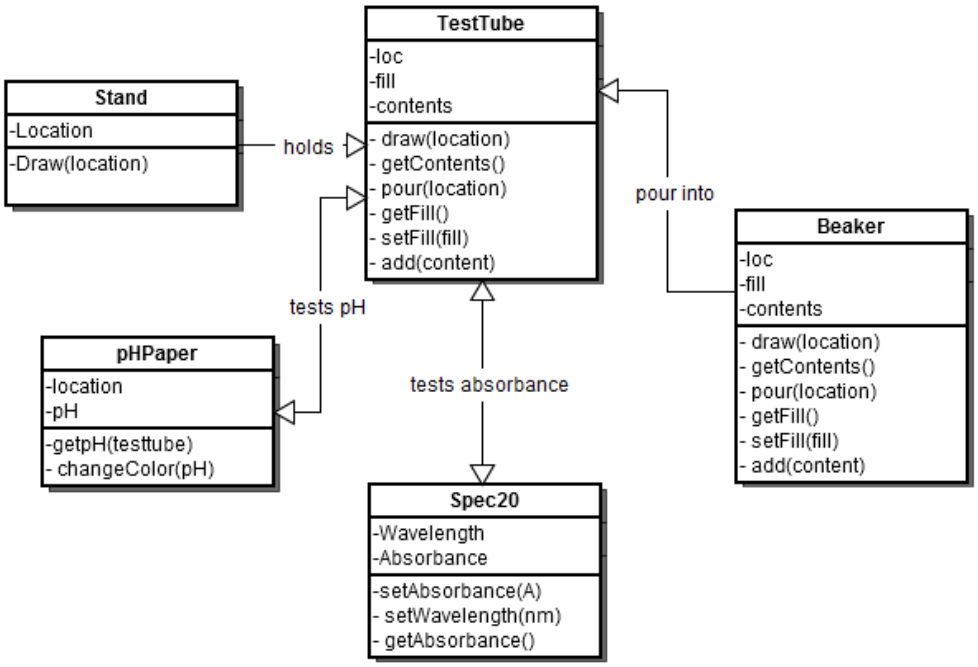
2. Class Diagram and Interface Specification

2.1 Class Diagram

Class Diagram for Laboratories One and Four, Cell Division and Meiosis



Class Diagram for Laboratories Two and Three, Biological Molecules and Enzyme Activity



2.2 Data Types and Operation Signatures

	Laboratories 1 and 4, Cell Division and Meiosis
Cell	• Location, integer • State, represents which state of mitosis the cell is in • initiateProphase() begins the process of prohpase • removeMembrane() takes away the image of the circle around the nuclear material • initiateMetaphase() begins the process of metaphase • drawEquatorialPlane() draws a line at the point of the equatorial plane in the cell • initiateAnaphase() begins the process of anaphase • inititeTelophase() begins the process of telophase • createChildCell() creates a second cell that looks the same as the original • separateChildCells() moves the child cell from the original • createCellMembrane() redraws the cell membrane arounf the original and the child cell • createAnimalCell() creates a representation of an animal cell • createPlantCell() creates a representation of a plant cell
Chromosome	• createChromosome() draws out a single chromosome within the cell • dragChromosome() allows the user to drage the chromosome to a new location • duplicateChromosome() draws a second chromosome that is the same as the first • splitChromosome() splits an existing chromosome in half
Pencil	• umber() numbers each of the chromosomes on the karyotype • selectMutation(x) user selects which chromosome has a mutation in it
Karyotype	• numbered, integer keeps track of how many chromosomes are numbered • circled, integer, keeps track of how many chromosomes are circled • drawNumber(n) draws the number under the chromosome • getMutation() finds if the user selected the correct location for the mutation • drawCircle(x) draws circle around the chromosome with the mutation
Microscope	• Magnification, integer, represents how zoomed in the microscope is • magnifyAnimalCell(), magnifies the animal cell • magnifyPlantCell(), magnifies the plant cell • magnifySlide() magnifies whatever is in the slide
Hot Plate	• Temp, integer, sets the temperature of the hot plate • placeSlide(), moves the slide to the hot plate • setTemp(temp) sets the temperature of the hot plate

Slide	• Location, integer, represents the loation of the slide • moveRootTip() moves the onion root tip to the slide • addDye() adds dye to the contents of the slide • pressSlide() presses the slip cover onto the slide • heatRootTip() heats up the root tip when contained in the slide and placed on the hot plate • viewSlide() views the contents of the slide when it is placed on the microscope • squishTip() crushes the contents of the slide in order to have the slide cover lay flat.
Dye	• initiateDye() initiates the change in color of the object that the user as added the dye to
Root	• location, integer, shows the location of the root • cutRootTip() removes one part of the root for the user to move • placeRootTip() allows user to move the root from one space to another
	Laboratories 2 and 3, Biological Molecules and Enzyme Activity
Test Tube	• location (integer) • contents, • fill (integer) • draw(location) draws test tube in a specified location • getContents() returns what is contained in the test tube • pour(location) adds the contents of the test tube to the location indicated, either pH paper or beaker • getFill() returns if the test tube is empty or full • setFill() sets the amount of the test tube that is full, incremented whenever something is poured in or out • add(content) adds a content to the list of what is in the test tube
Beaker	• location (integer) • contents, • fill (integer) • draw(location) draws beaker in a specified location • getContents() returns what is contained in the beaker • pour(location) adds the contents of the beaker to the location indicated, into test tube • getFill() returns if the beaker is empty or full • setFill() sets the amount of the beaker that is full, decremented whenever something is poured out • add(content) adds a content to the list of what is in the test tube
	• location (integer) • pH (integer 0 through 7)

	• getpH(testtube) returns the pH number to the contents of what is in the test tube • changeColor(pH) changed the colopHPaperr of the pH paper to show whichever level pH it is set to. Returns to white when set to 0
Spec20	• Absorbance (integer) • Wavelength (integer) • setAbsorbace(A) sets the light and control settings for the spec20, set to 0 for the zero calibration and 1 for the infinite calibration • setWavelength(nm) set the wavelength of the wave that the spec20 is shining through the test tube in order to find its absorbance, in nanometers • getAbsorbance() returns the absorbance of the contents of the test tube being tested
Stand	• Location (integer) • Draw(location) draws out the stand and prompts the drawing of 7 test tubes

Content	What items can be contained in either a test tube, beaker, pH paper or the sample holder in the spec 20. Listed below are also certain properties of these contents that either the pH paper or the Spec 20 would read • Urine from patients 1 through 6 (all pH 7 except for pH of 5 is 4 and the pH of 6 is 8) • Distilled water • Glucose • DSN (makes absorbance of each persons urine as follows: 1=.781, 2=.121, 3=2, 4 = .16, 5= .19 and 6 = .089) • Protein (makes absorbance of each persons urine as follows: 1=0, 2= 0, 3= .02, 4= .2, 5= 0.0222 and 6=0.17) • Buffer • pNPP • Sodium • Phosphate • Enzyme (At concentration 1= initial A= .04, after 5 minutes A5=0.42, concentration ½ A=.044, A5=.067, concentration ¼ A=.066 and A5 = .15, at concentration 1/16 A = .66 and A5= .054 at concentration 1/32 A=.051 and A5=.056, concentration 0 A=.065 and A5=.051

2.3 Traceability

The First laboratory only had two domain concepts listed that had to do with a cell, the animal cell and the plant cell. Rather than just having a class for the cell, each component of the cell is given its own class. This means that the domain concept of a cell was split into cell, nucleus, chromosome, pole and equator. This allows for the system to have specific interactions between the user and each specific component of the cell. The other domain concepts, the slide, cover slip, dye solution and onion roots, have all been given their own classes also and interact in the same way that is described in the domain concepts relation chart.

The Second laboratory and all of the use cases and domain concepts involved in it have been covered by the following classes: Stand, Spec20, Test tube, Beaker and pH paper. The main domain concepts that needed to be covered here were the Spec20 machine, test tubes, the pH paper and all of the contents of the test tube. The spec20 and pH paper both act as a tool to get information about these liquids and return to the user, in an animation, what the absorbance or the pH of those liquids are. The class for the test tube matches exactly the domain concept of the test tube and the beaker class was added in order to simplify the functions. Rather than creating a class for each of the domain concepts that were contained in another object, like distilled water or protein, they became the items that were contained in the test tubes and or beaker.

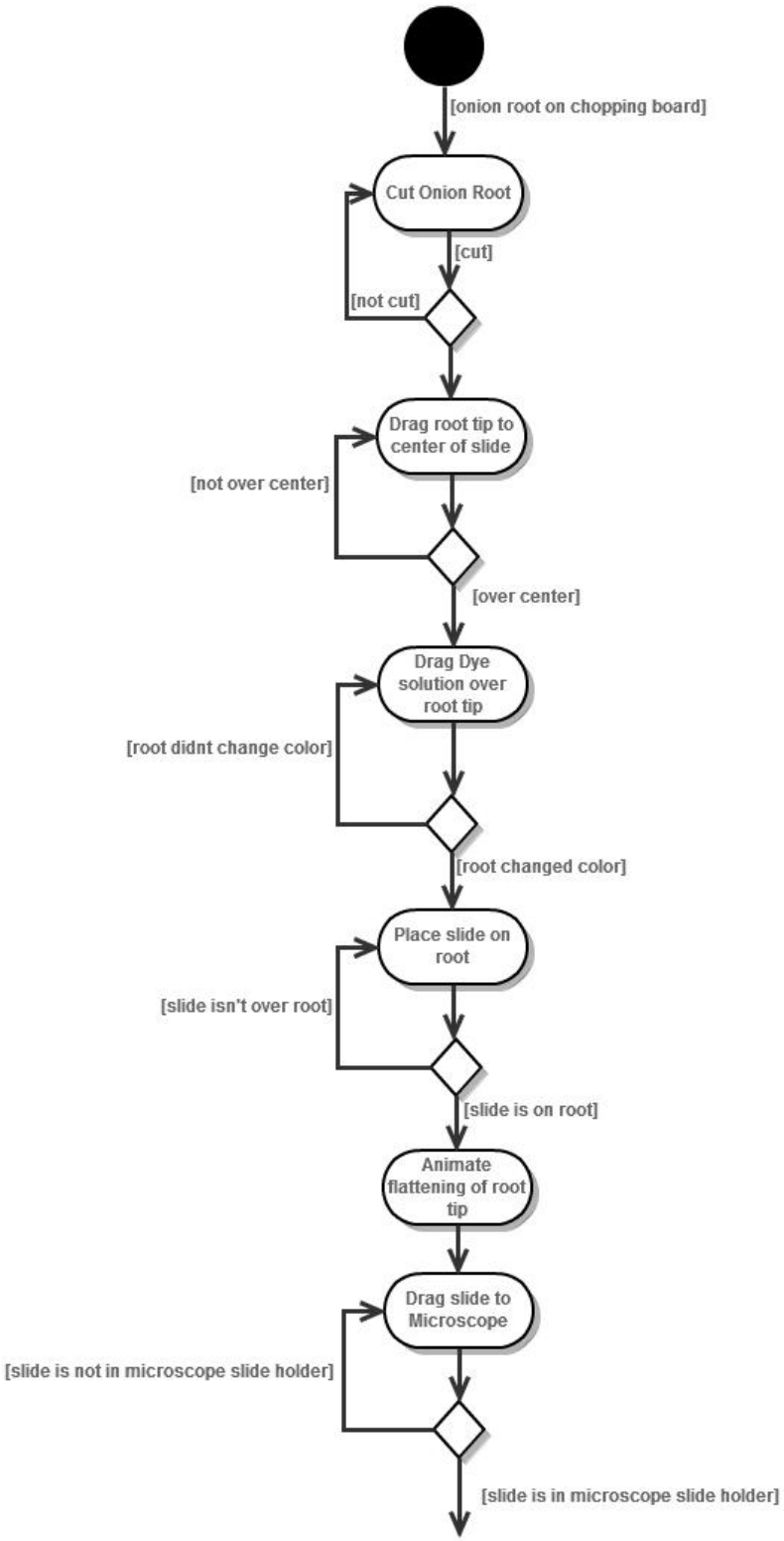
The Third laboratory had almost the exact same domain concepts as the second laboratory so their class diagram is the same. The only thing that needed to be added was a function that allowed the user to pour the contents of a test tube into another test tube. This lab also needed different items contained within the test tubes and beakers. Aside from the addition of those few things, the two laboratories use the Spec20 exactly the same way.

The fourth laboratory has the same class diagram as the second because they both involve very similar domain concepts. All of the parts of the cells were not named in the concept domain but needed to be made classes o account for all of the interactions that the user has to have with them. So the domain concept “slide” was split into cell, nucleus, chromosome, equator and pole classes and all of these interact in a way to show the user the process of meiosis. The other domain concept, the karyotype, directly translated to a class that allows the user to number all of the chromosomes shown in it and circle where the mutation is.

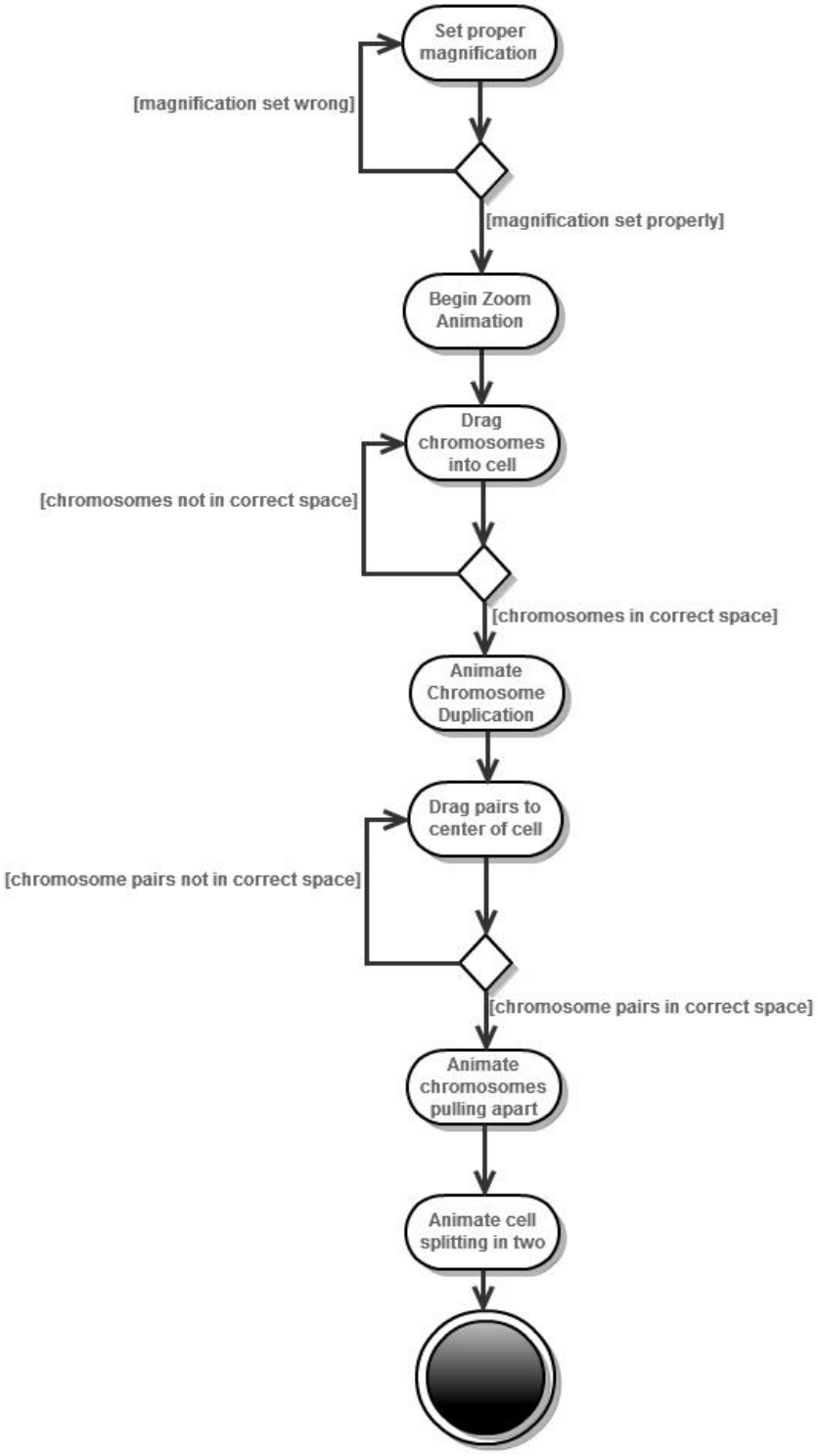
All of the names of the classes are derived from either the exact name of the domain concept that it is based off of, or the actual name of the component of the cell that they are named after. For example, the few classes that were created when dividing the domain concept of a cell, were all named for the specific component of the cell that they represent, like the chromosome or the equator of the cell.

2.4 Activity Diagrams

Laboratory One Cell Division Activity Diagram



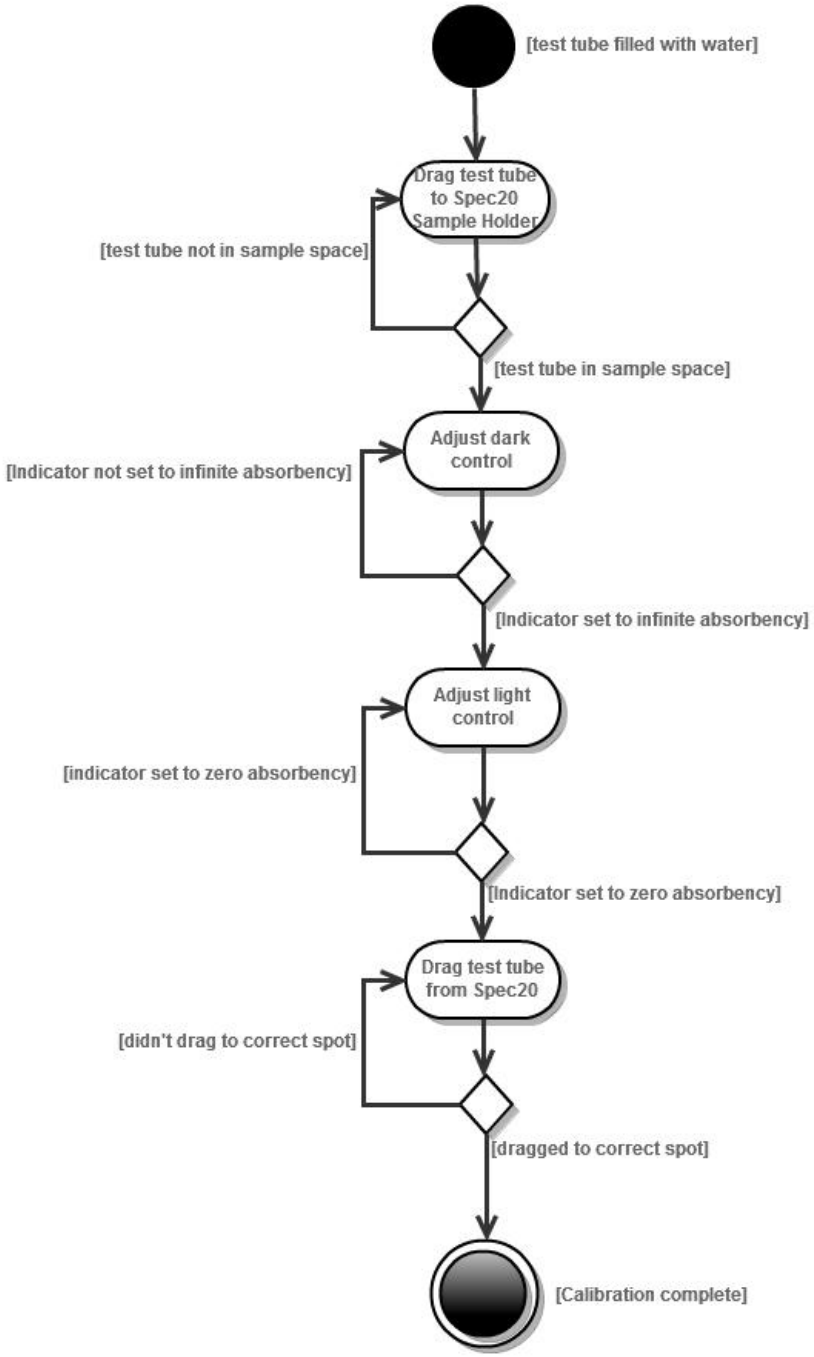
Continued from previous page



This diagram shows all of the steps involved in the first laboratory, it consists of two parts,

first the user using the virtual laboratory to prepare a slide, place it in the microscope, setting the magnification and then viewing the slide. While viewing the slide the user will be able to interact with the chromosomes and view the animations of a cell going through all of the different stages of cell division.

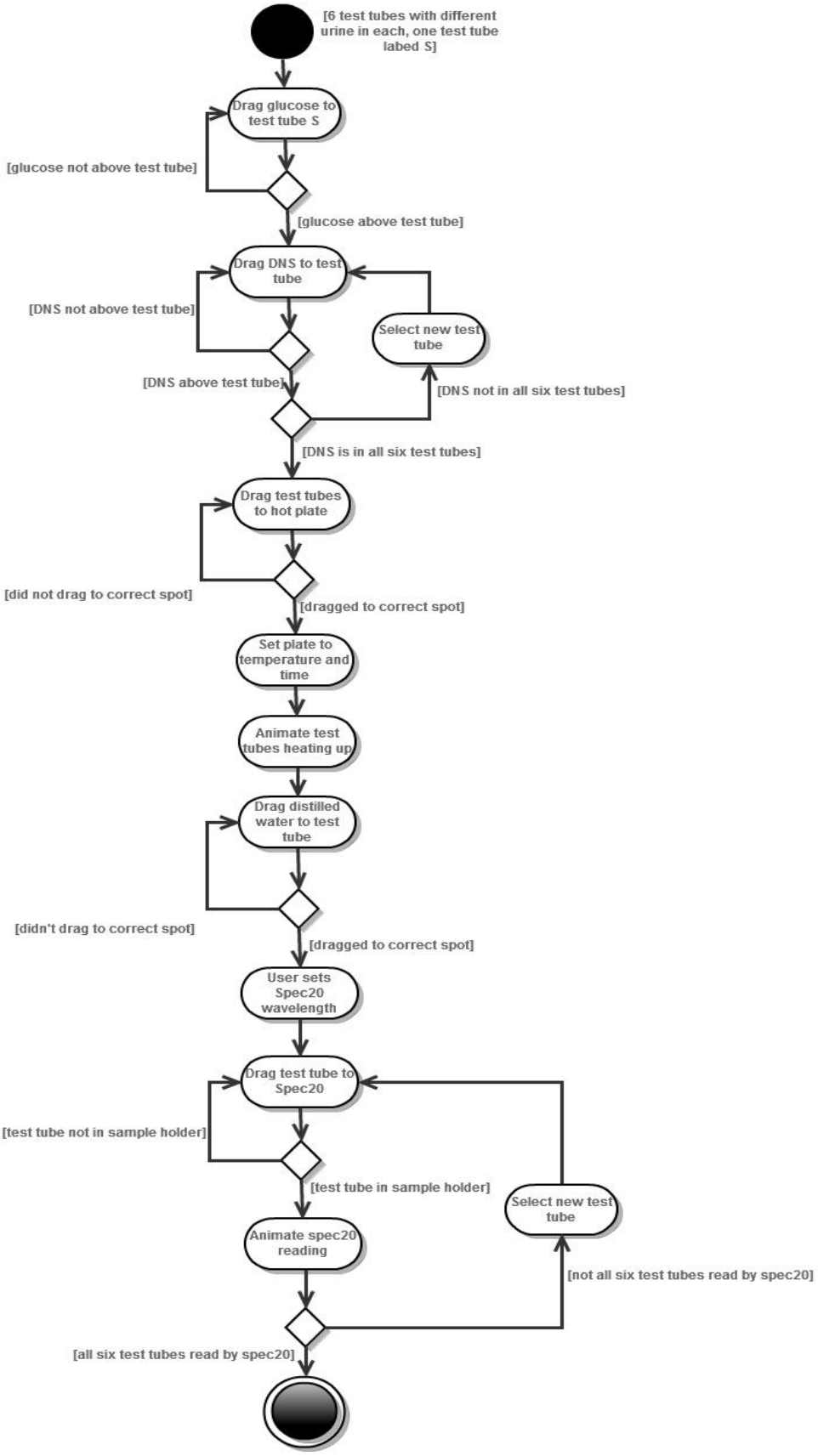
Laboratory Two Biological Molecules Activity Diagram



The activity diagram shown above explains the states that are gone through when calibrating the Spec20 Spectrophotometer. This process will be used at the beginning of the second and third

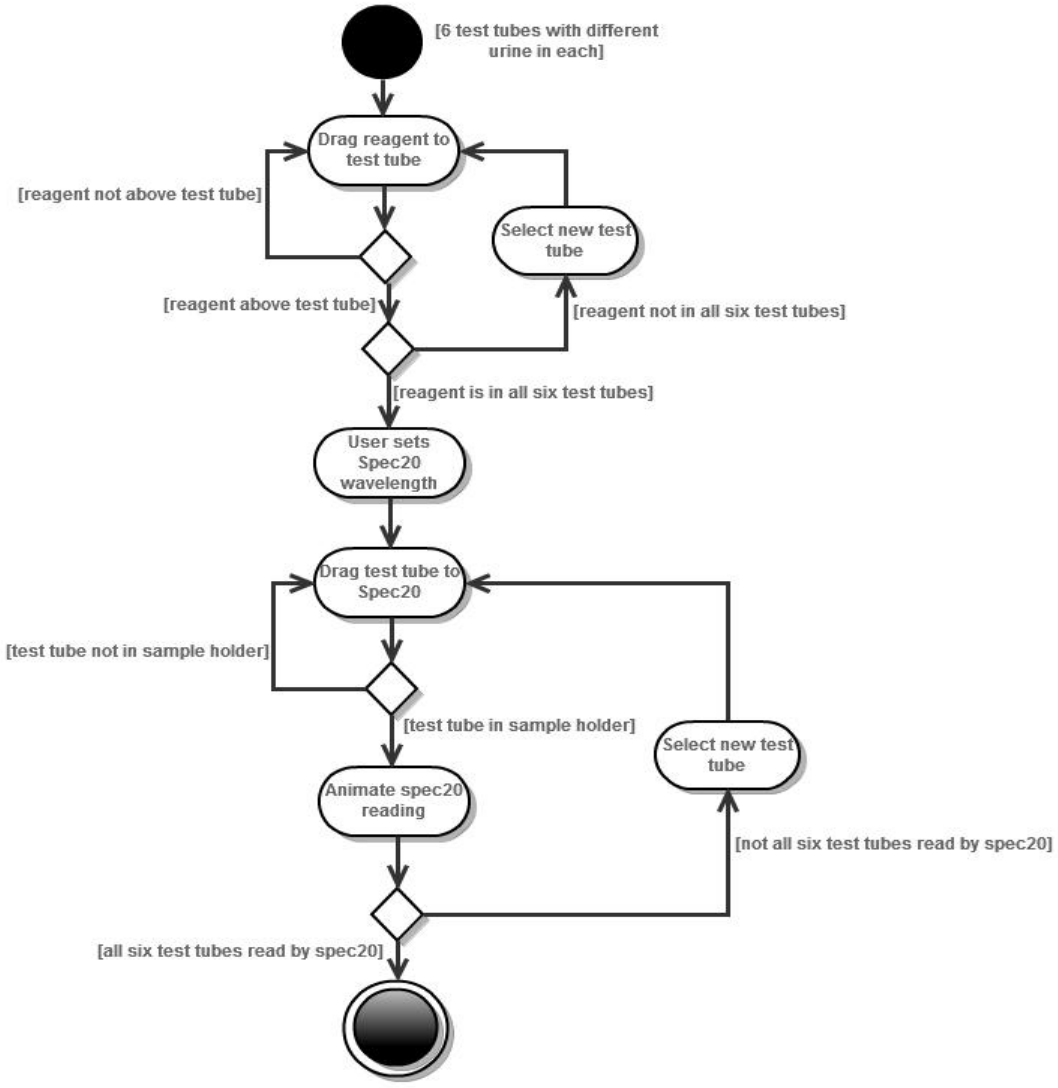
laboratories to set the Spec20 to the correct zero and infinite absorbancies.

The parts to this lab are shown in separate state diagrams to make the flow of the laboratory more easy to understand. Each part below will be repeated for all 6 test tubes but the state diagrams are only shown once. Since the goal is to compare the properties of the components of each test tube, as the lab progresses, a table will be filled in with the observed results so that the student does not need to keep track. Below, in the first part is the state diagram for testing the pH of each patients urine. This involves just the student dragging each test tube to the pH paper and then the animation of the pH paper changing color to indicate the pH of the urine.

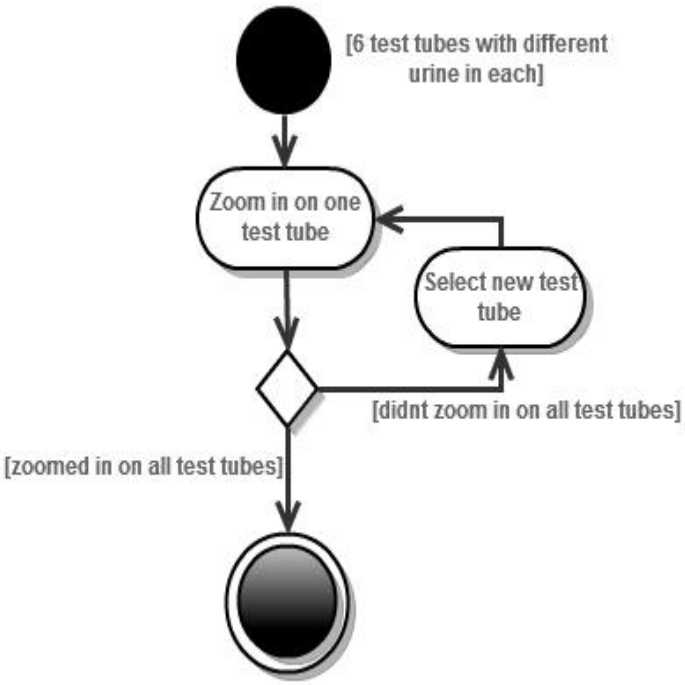

[6 test tubes with different urine in each, one test tube labed S]
Drag glucose to test tube S
[glucose not above test tube]
[glucose above test tube]
Drag DNS to test tube
[DNS not above test tube]
Select new test tube
[DNS above test tube]
[DNS not in all six test tubes]
[DNS is in all six test tubes]
Drag test tubes to hot plate
[did not drag to correct spot]
[dragged to correct spot]
Set plate to temperature and time
Animate test tubes heating up
Drag distilled water to test tube
[didn't drag to correct spot]
[dragged to correct spot]
User sets Spec20 wavelength
Drag test tube to Spec20
[test tube not in sample holder]
[test tube in sample holder]
Select new test tube
Animate spec20 reading
[not all six test tubes read by spec20]
[all six test tubes read by spec20]

This diagram on the previous page shows the second part to the laboratory where the user will add liquids to the test tubes, place them on the hot plate to heat up and then use the spec 20 to read each test tubes absorbance. Since not every step must be completed 6 times, there is a not on the side as to which items must be repeated by the user

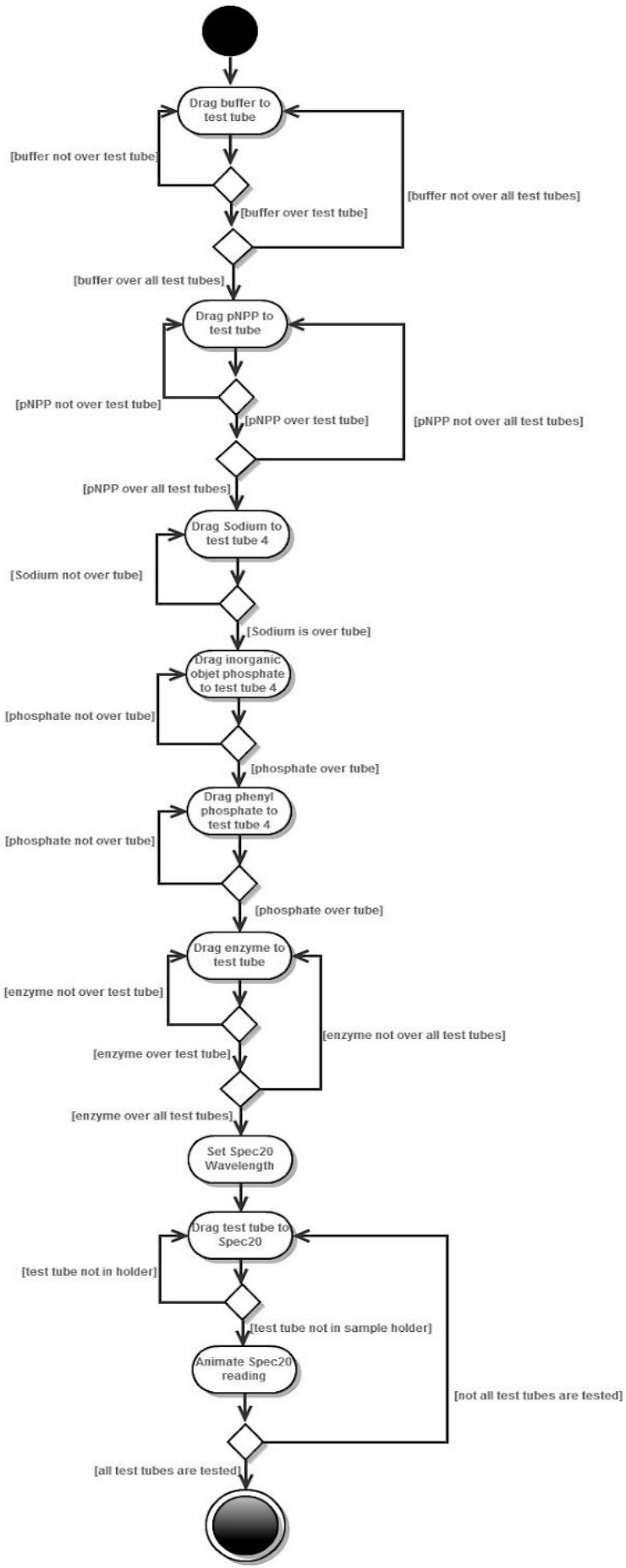
The state diagram below shows the third interaction between the user and the Spectronic 20 machine, this time testing for the level of protein in the urine. It is very similar to the previous section of the second laboratory.



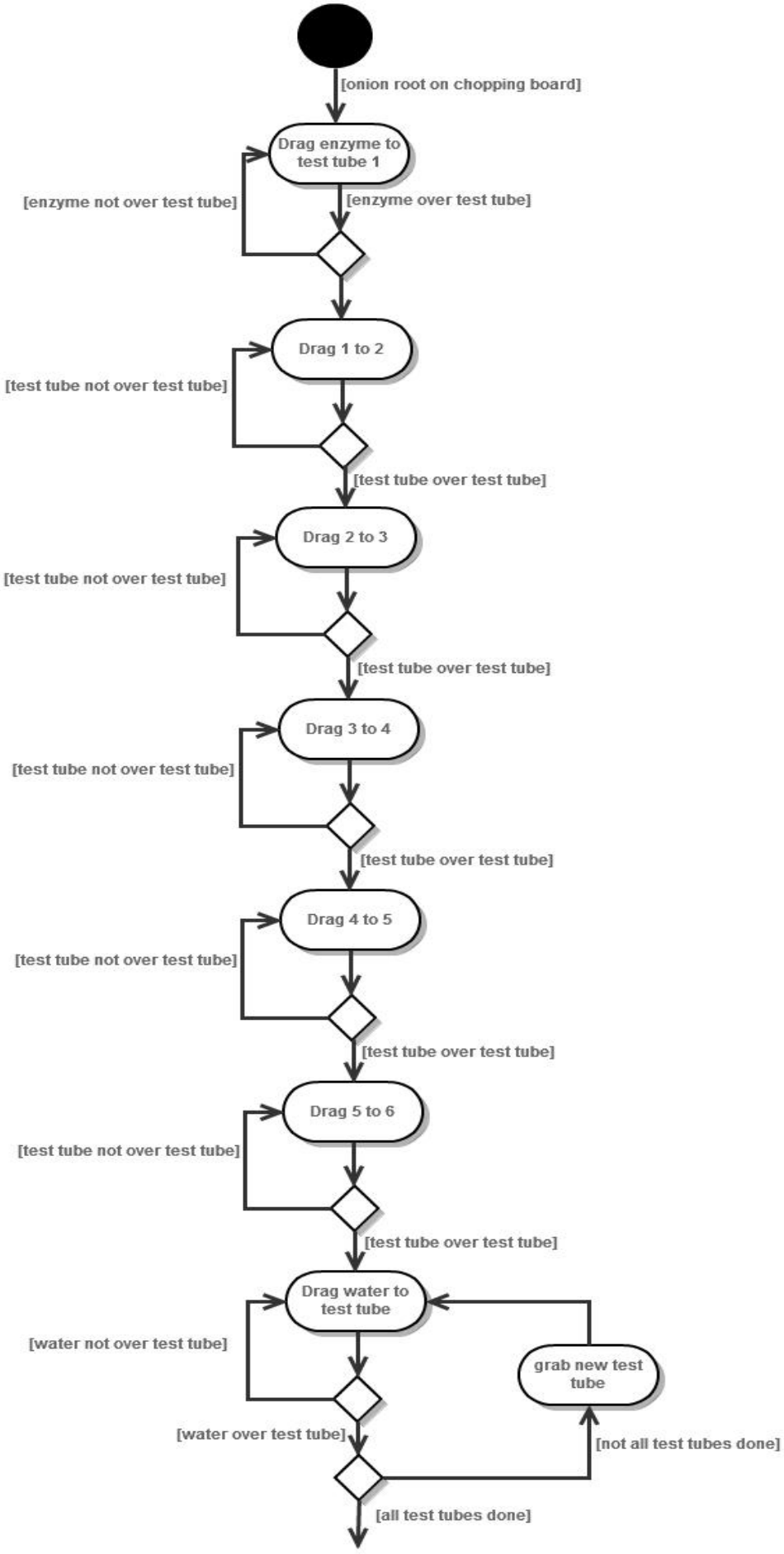
Below is the last activity diagram for the second laboratory, in this the system will just show the user each test tube and have them select which color it appears to be.



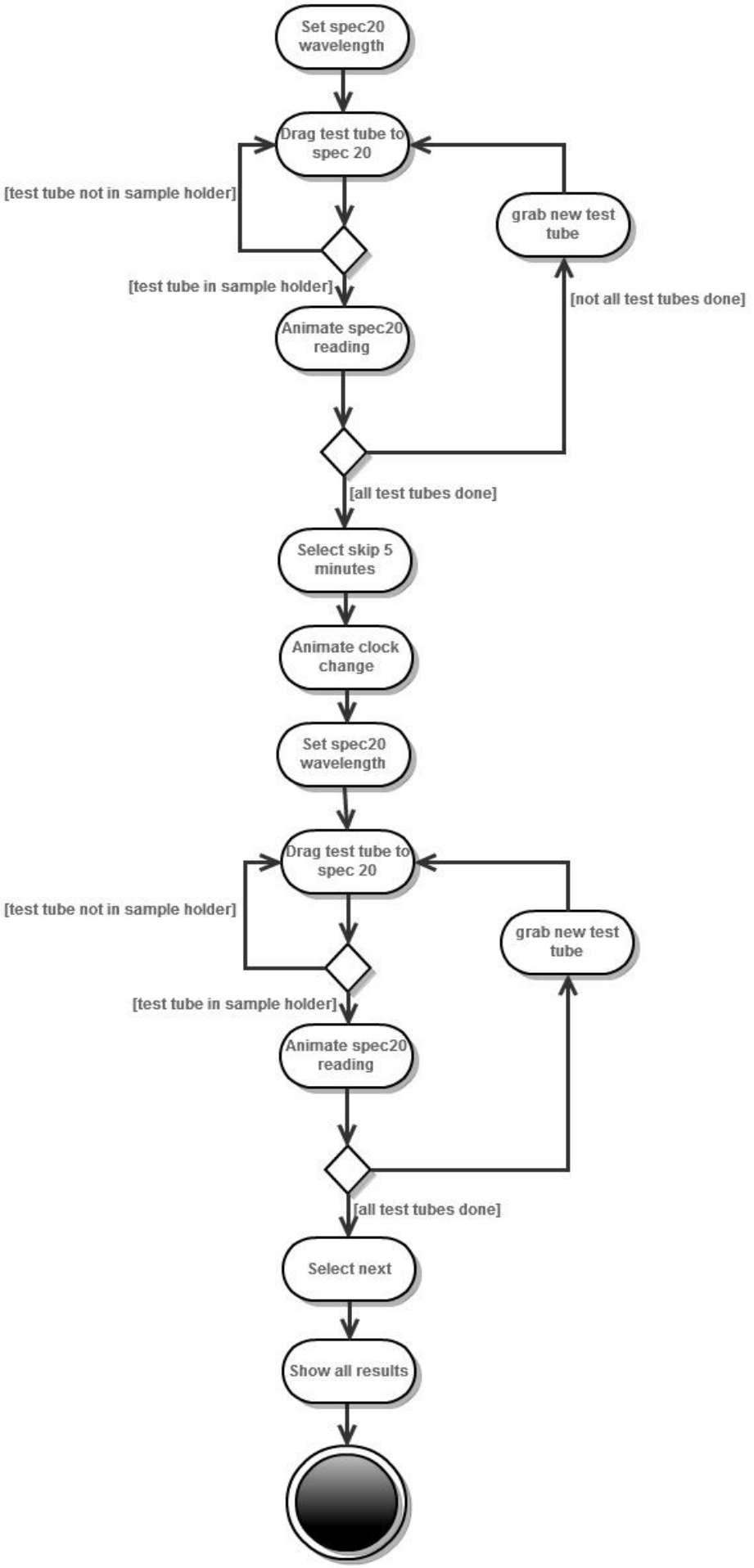
Laboratory 3 Enzyme State Diagram (next page)


Drag buffer to test tube
[buffer not over test tube]
[buffer over test tube]
[buffer not over all test tubes]
[buffer over all test tubes]
Drag pNPP to test tube
[pNPP not over test tube]
[pNPP over test tube]
[pNPP not over all test tubes]
[pNPP over all test tubes]
Drag Sodium to test tube 4
[Sodium not over tube]
[Sodium is over tube]
Drag inorganic objet phosphate to test tube 4
[phosphate not over tube]
[phosphate over tube]
Drag phenyl phosphate to test tube 4
[phosphate not over tube]
[phosphate over tube]
Drag enzyme to test tube
[enzyme not over test tube]
[enzyme over test tube]
[enzyme not over all test tubes]
[enzyme over all test tubes]
Set Spec20 Wavelength
Drag test tube to Spec20
[test tube not in holder]
[test tube not in sample holder]
Animate Spec20 reading
[not all test tubes are tested]
[all test tubes are tested]

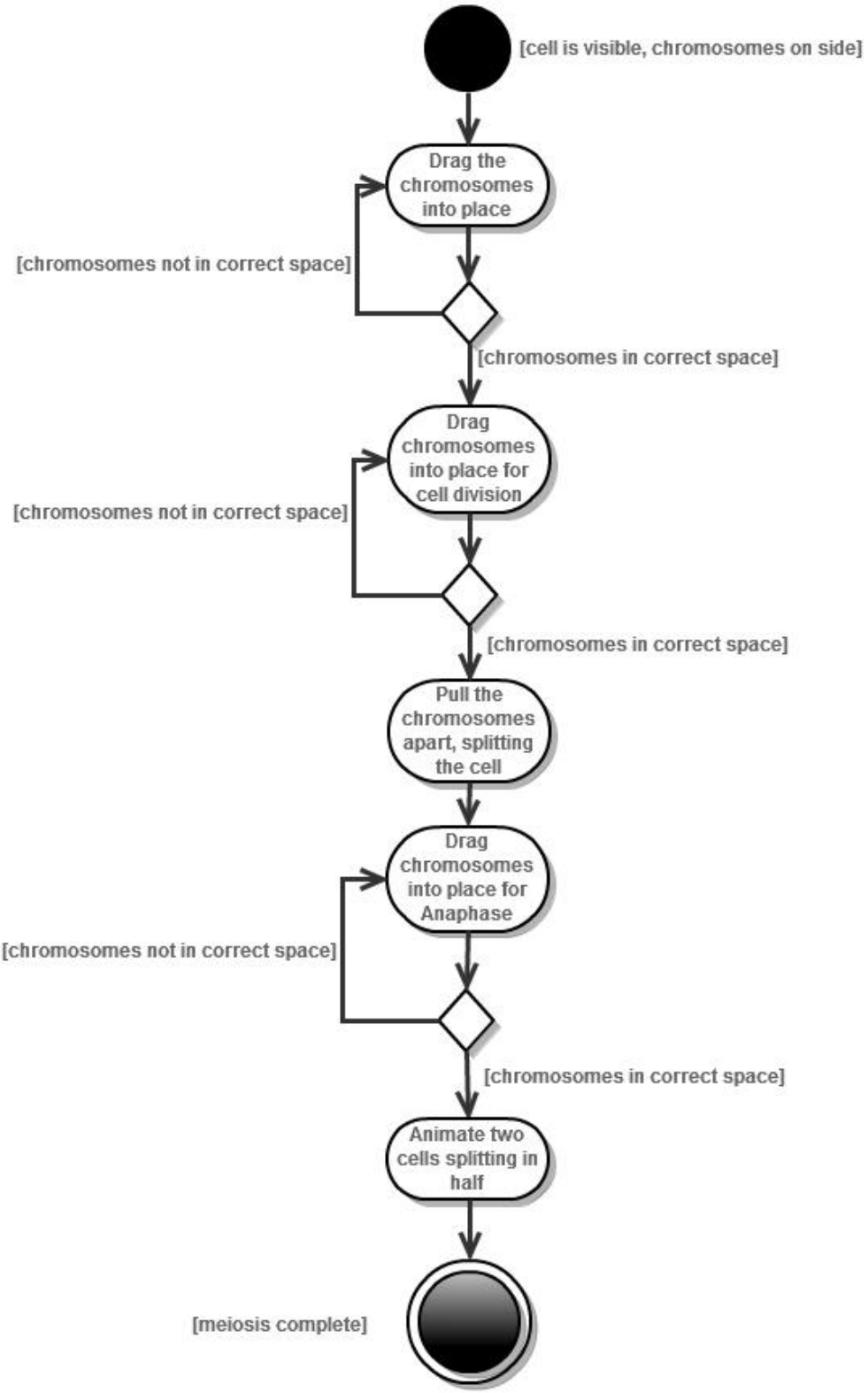
The above diagram is the state diagram for the first part of the third laboratory. In this activity the user must use different liquids to fill each of the 6 test tubes, after this they must drag each test tube to the spec20 to get its absorbance and time before it reaches an absorbance of 3 or 4.


[onion root on chopping board]
Drag enzyme to test tube 1
[enzyme not over test tube]
[enzyme over test tube]
Drag 1 to 2
[test tube not over test tube]
[test tube over test tube]
Drag 2 to 3
[test tube not over test tube]
[test tube over test tube]
Drag 3 to 4
[test tube not over test tube]
[test tube over test tube]
Drag 4 to 5
[test tube not over test tube]
[test tube over test tube]
Drag 5 to 6
[test tube not over test tube]
[test tube over test tube]
Drag water to test tube
[water not over test tube]
grab new test tube
[water over test tube]
[not all test tubes done]
[all test tubes done]

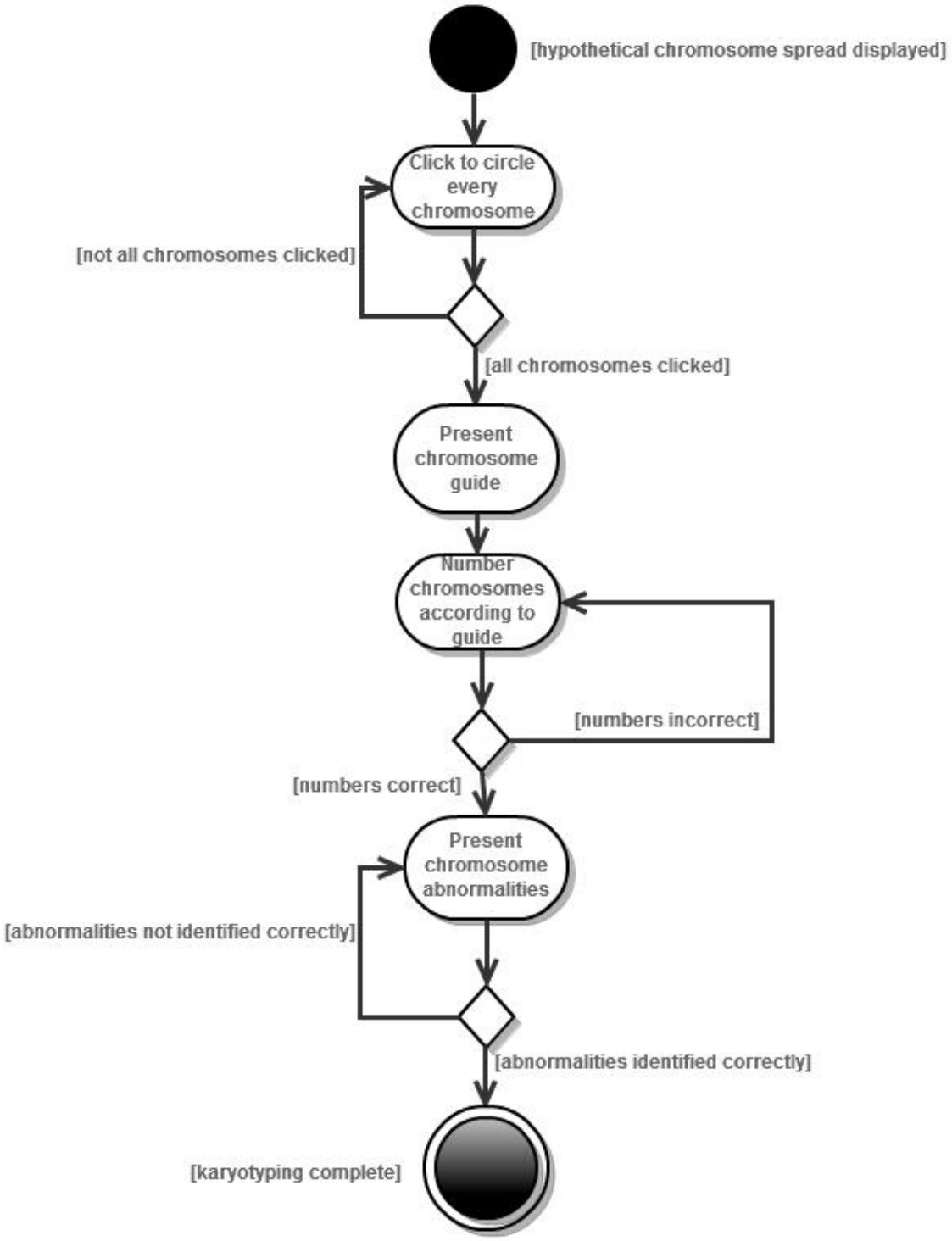
The first part to this section of the laboratory involved diluting an enzyme so that the first test tube contains the most concentrated mixture of water and enzyme and the last test tube holds the least concentrated mixture of water and the enzyme. The user will accomplish this in the virtual biology lab by dragging each test tube to the next and emptying half of its contents into the next test tube. After this is complete the user must drag the distilled water to fill each of the test tubes. Below is the state diagram for the rest of this laboratory that involves entering each test tube into the Spec20 and reading the absorbancey, selecting to skip ahead 5 minutes and then re reading all of the absorbancies.


Set spec20 wavelength
Drag test tube to spec 20
[test tube not in sample holder]
grab new test tube
[test tube in sample holder]
[not all test tubes done]
Animate spec20 reading
[all test tubes done]
Select skip 5 minutes
Animate clock change
Set spec20 wavelength
Drag test tube to spec 20
[test tube not in sample holder]
grab new test tube
[test tube in sample holder]
Animate spec20 reading
[all test tubes done]
Select next
Show all results

Laboratory 4 Meiosis Activity



Diagram



These two activity diagrams show the fourth laboratory, on meiosis. This lab has two parts, the first is very similar to the first laboratory because it shows the interactions between the user and the chromosomes within a cell in order to show the type of cell division called meiosis. In the second part is shows the interactions between the user and the Karyotype picture and how the system aids the user in finding and circling the chromosome defect in the picture.

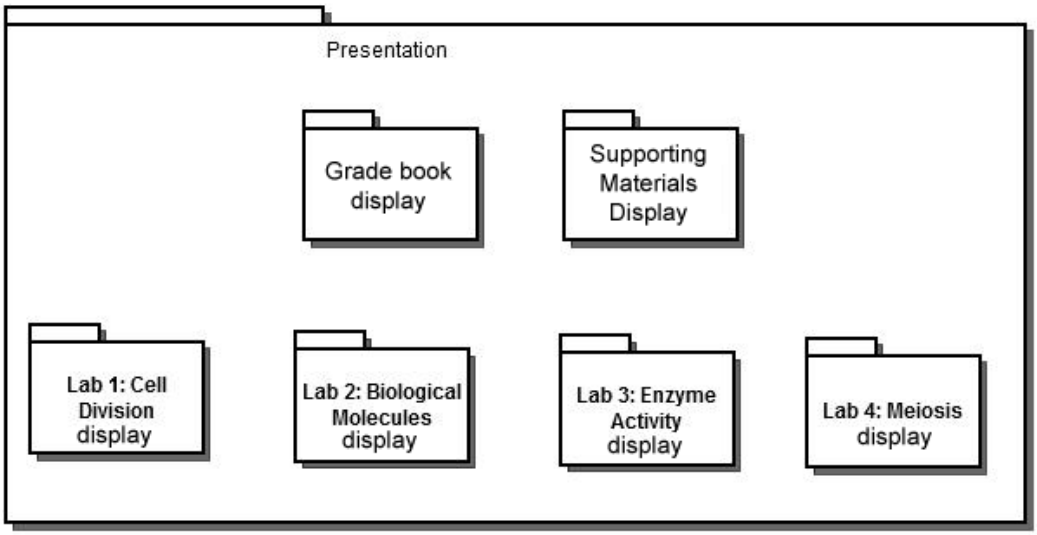
3. System Architecture and System Design

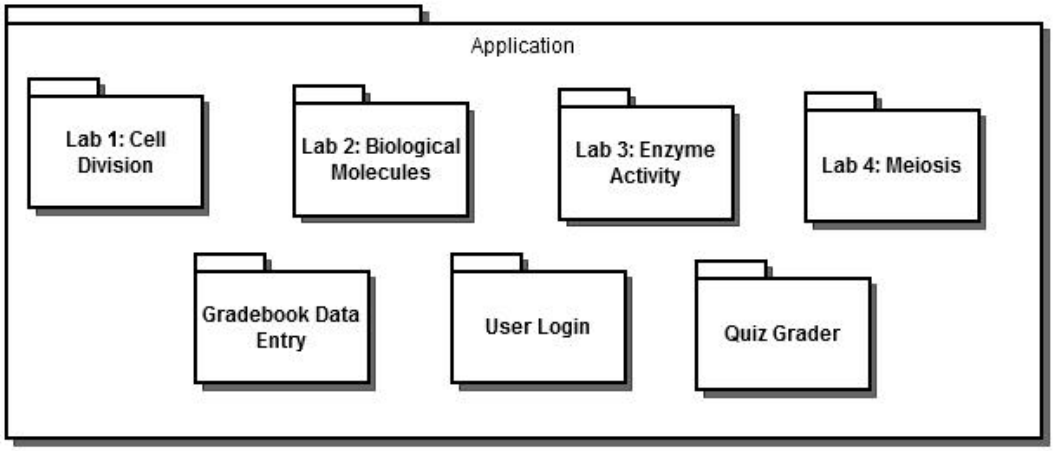
3.1 Architecture Styles

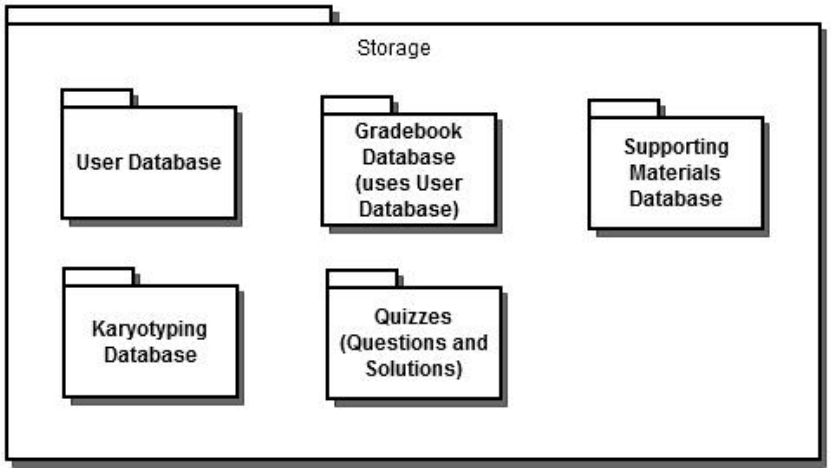
There are two major components and architectures used within our application. The first component being the website. We will be programming in a PHP environment built on top of a Linux server running a Apache, MySQL, and PHP (also typically known as a LAMP stack). We will be developing in a Model, View, Controller (also known as MVC) to help the structure and development of PHP code. Note that this will follow the standard server-client architecture where the server will distribute the necessary content to the user provided their input. Note that the server will also require a database architecture where it must input grade results and user information into. We will be relying on this to provide dynamic data.

The second component involves animation and the actual labs themselves. Here, we will use a standard UI architecture, using Flash, to display and show educational and hands-on material to the user which was first provided to them by the server-client architecture.

3.2 Identifying Subsystems







We have divided our subsystems into three different levels each with different purpose for the individual components. In our Presentation level, we handle purely the visual aspects of the system and how they will be displayed to the user. For example, we have our Flash Animations, which consists of the four lab reports and the corresponding quizzes.

In our second level, the Application level, our subsystems will be described in terms of what they will be actually doing as a part of the system. Here we have descriptions for each of the labs and how they will react and respond to user input. A more specific example of an item from this subsystem would be in Lab 2: Biological Molecules, where the user deals with the spec20 instrument. This subsystem will layout how this instrument behaves to certain conditions and what it should inform the system the output is. Our application level also mentions quizzes which on the presentation level will be handled by the “Flash Animation”.

Finally we have the Storage level which contains all of our databases and static information about items such as the lab methods and data, the correct solutions to quizzes, any content that a professor has uploaded for supporting materials, and where our user’s information will be dealt with. The user data is by far the most important in this section because this will contain their progress in labs, which will be passed to the gradebook when certain checkpoints/requirements are met, and all of their identifying information so that they will be able to continue their work from where they left off each time they login.

3.3 Mapping Subsystem Hardware

Our application requires both a server environment and a client environment. The server environment will pre-process hypertext using PHP given input from the user and information from the database. The server will be responsible for grading quizzes, showing and calculating grades and displaying the correct labs.

The client will require a web browser that is compatible with the latest standards of HTML, CSS, Javascript, and Flash. Preferably browsers such as Chrome, Firefox or Safari. The client will be in charge of displaying content served by the server properly, meaning, interpreting the HTML, CSS, Javascript and Flash data.

3.4 Persistent Data Storage

Our application will require to store session data onto the client’s system and browser. This is better known as cookie data, which helps the server interpret who the request belongs to and where to grab proper information from the database. As mentioned, we will also require a database which will store user information such as name, grades, passwords, emails and more that are required for dynamic data computing for each individual user.

3.5 Network Protocol

The server and client will require standard HTTP ports and communication. This communication requires standard socket transfer, POST and GET request support. This type of data has been chosen due the nature of standard HTTP servers and Apache’s schema.

3.6 Global Control Flow

3.6.1 Execution Orderness

Our system is an event-driven system where a user can execute and perform labs in orders they wish to choose. However, there is a linear path in which the user must complete the labs. For

example, a user can choose to browse a lab and choose to view their grades before taking a quiz.

3.6.2 Concurrency

Yes, our application will be using multiple threads on the server-side level. Apache, the utility that serves HTTP and PHP requests, handles concurrent threads by spawning a new child process for each unique client request. Apache will also spawn its on PHP thread for each child process if required.

3.7 Hardware Requirements

Our system will require a server with standard resources. Of minimum a virtual machine with root access. It should at least have 512 megabytes of usable RAM and a minimum of 4GB disk drive. We will require Apache, PHP and MySQL installed on this server.

Our application will require clients to have a computer which can meet the minimum standards of running a web browser and the Flash plugin. We will require at least a screen that can run a resolution of 1024 pixels wide.

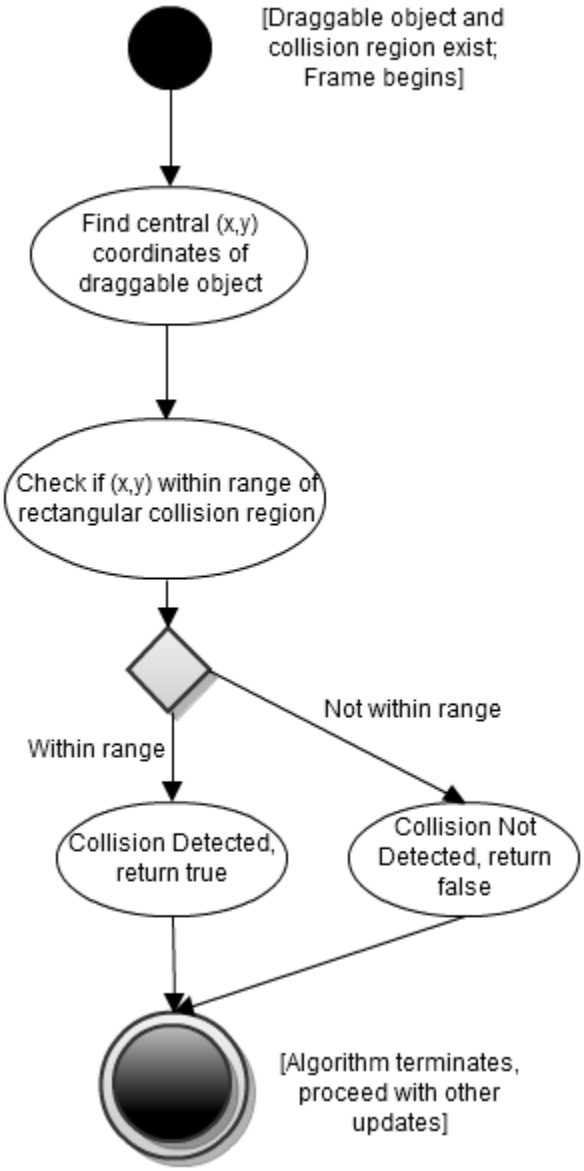
4. Algorithms and Data Structures

4.1 Algorithms

Although the goal of the virtual biology lab is to simulate real lab situations and phenomena, we use a very basic modeling scheme to present this to the user. Interaction with the lab demos is guided, meaning the system will wait for proper user input before moving to the next state. Actual biological and physical phenomena will not be mathematically calculated; instead there will be discrete changes in the properties of the system that simulate results you would expect in an optimal lab setting.

However, the most prominent example of algorithm use in our project would be our collision detection algorithm. Many of the labs require the user to move objects to certain rectangular regions, and these regions will be hard coded into the system by inputting the coordinates of their 4 corners. For example, in lab 1, the user is required to move two chromosomes into their proper positions in the cell. On every frame update, we will run a method that computes whether the chromosome is in the proper region of the cell.

First, we would need to calculate the central x and y coordinates of the chromosome object (which is being dragged by the user's mouse). This is preferable to the (x,y) coordinates of the mouse itself because it gives a more accurate reading on where the object actually is. We then use these values and check if they within a certain range (within the rectangular region), and if so, the system should detect the collision and proceed to the next step if applicable. This basic algorithm can be expanded to any object and any region we desire. The activity diagram is shown below:



4.2 Data Structures

For calculating grades, obtaining class rosters and so on, we will be obtaining the data that has been associated from the database into an associative array. This associative array acts similarly to a hash-table, allocating all spaces and keys into memory, allowing us to handle performance in constant time. With these associative arrays, we are able to calculate averages, means and much more analytics for grades. We will also be using associative arrays into grading quizzes and so forth as defined from a POST operation from the web-browser.

All other items, such as the labs and animations will not require any sort of complex data structures as they are formed more as a “movie” that runs in linearity. Also, we are unable to understand the technology of Flash as it is proprietary technology.

5. User Interface Design and Implementation

Since our first report, our initial screen mock-ups have not changed drastically. One of the main changes is that some of the artwork has been improved from simple MS Paint sketches, to full fledged final presentation material. This can be seen in the demo of the first lab, which can be found at the end of Section 7.4.

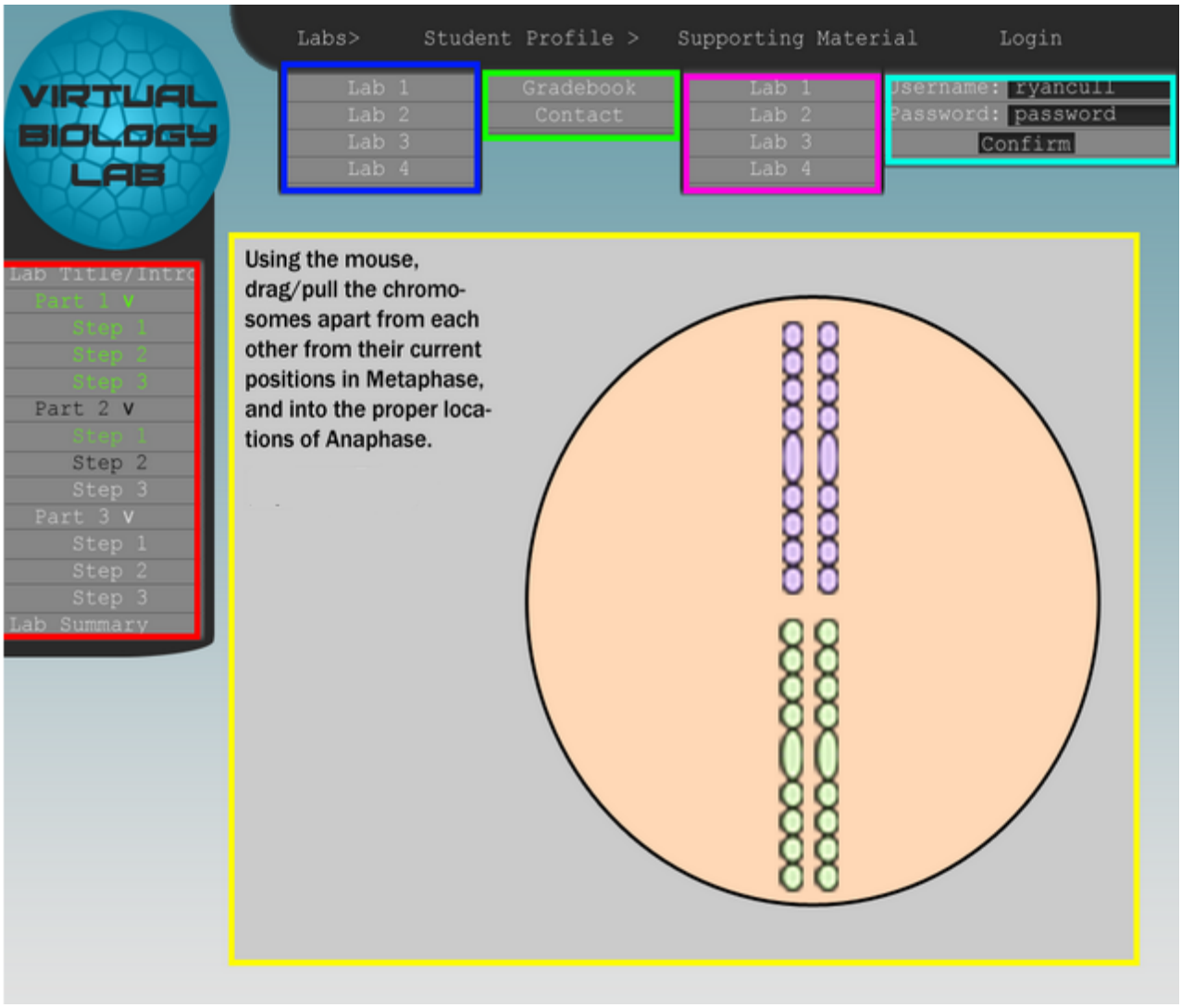
In our demo you can see that there are almost two panels in our design. The top panel which has a lighter gray background is the directive panel. This area displays information to the user about the current stage of the lab and what the user should be doing in order to interface with the system. For example, our first “slide” of the animation instructs the user to “Drag and drop the chromosomes into place so they look like this:” accompanied by a diagram of the cell that the user needs to duplicate. Below this is a darker gray panel, which is the interactive portion of the interface. Here we render all the items that the user needs to complete the given instructions and allows them to follow the required steps. In our first slide, we can see the user is able to drag and drop the chromosomes to any location they desire in the frame (including the directive panel). Once the user lets go, the item being dragged will remain there and remain movable until the user has placed it correctly.

Due to precision issues, the user cannot be expected to place the chromosomes in a pixel-perfect position, so there is a margin of leniency where the system will automatically reposition the chromosome to the “accepted” position if the user is close. From here the user will continue to perform the proper actions until the system is satisfied with the results, at which point a the system will prompt the user to continue to the next slide in the lab. In which a similar user interaction will be repeated.

Our lab demonstrations will also be incorporated into a website with a fully functioning back-end that maintains user progress. This system allows for users to select which lab they would like to work on, look up what sections of each lab they have completed, and even find supporting materials for the lab assignments. These features are all extremely easy to access requiring only a handful of clicks in order to navigate to each menu. Each of these features is displayed directly on each page of the site, which grants this instant access and eliminates an immeasurable amounts of wasted time due to searching for links to pages.

Overall based on these factors, our system is proven to have a very high ease-of-use, while still maintaining a visually appealing and stimulating interface, making the user experience more enjoyable. This ease-of-access will be extremely easy to see in our first demo, which will include the web page that includes the individual lab simulations.

For each of the requirements we have listed I have provided the following picture with labels and below will describe each label and the means it will take to access each portion.



Red: Here in the red section we have a listing of the current sections available in the lab that the user is working on. This information will very depending on what section the user has entered. For the “Labs” the system will display information similar to the current information, where every item with a “v” next to the name represents a drop down. This action requires 1 click to either expand the content below, or hide the content in that subsection. The other operation this navigation can support requires 1 click, where the system will refresh the page and bring the user to the requested section of the lab. For our other sections such as “Supporting Materials” we will present the main subsections for each lab, which will interact the same as the other drop downs (1 click) and a listing of the content that the professor shares (also 1 click). Gradebook will provide the same functionality of a single click interface, but each item will refresh the content showing the grades for the selected assignment.

Blue, Green, Pink, Cyan: These sections all have the same base functionalities. When the corresponding header is clicked, the content shown in the drop downs will appear. This means that each of these require a single click to access the content. To further access each sections content, the user will only be required one more click to refresh the page with the requested content on the page.

Cyan: While accessed by the same method as the other 3 sections in the previous paragraph,

this section requires more interaction. As you can see in the linked animation here (http://www.mikedilalo.com/VirtualBiologyLab/Report_2/Animation.gif), I have shown the way to access the each header and view the drop down. This animation goes further by showing the instructions and the remainder of the animation. The amount of user input in this section will vary based on each user. All users will need to input their user name (generally ranging from 4-12 characters) and a password (minimum requirement of 6 characters, ranging from 6 - 12 keypresses). Then one additional input will be required, either a key press of the return key to confirm the information, or a click on the confirm button.

Yellow: This is the section that will require the most user interaction. While the user is performing the lab assignment, they will be instructed to perform multiple tasks involving user inputs. For example in our first lab demo, we will go slide by slide and quantify the minimal number of interactions required to complete the lab so far (<<< PLEASE PUT THE LAB DEMO HERE>>>). First the user will be required to move the two chromosomes into the correct locations, (minimum 2 mouse clicks and dragging to the proper position and releasing). To proceed to the next step the user will need to click the “Next” button (total of 3 mouse clicks). On our next slide, the user will see an animation and be asked to click “Next” in order to proceed to the next step (total: 4 clicks). Again the user will need to drag two items into place (minimum 2 mouse clicks and dragging to the proper position and releasing) and continuing by clicking “Next” (total: 6 mouse clicks). Once again an animation will be played and the user will be asked to click next. Therefore in the lab so far, the user needs seven mouse clicks to properly interface with the lab, which is fairly simple given the subject matter being covered.

6. Design of Tests

The tests we will be describing in this section relate to what we feel are the most important and ubiquitous functions inherent to our software which are pivotal for the main successes of our use cases. Some of these test cases were enumerated on our first report, and will be further expanded upon in this report.

6.1 Unit Testing Test Cases

All tests will be initiated using a systematic approach, wherein multiple unique input values will be selected with each value engendering a different response from the software. This way, we will clearly be able to decipher how our software operates under normal (pass) and faulty (fail) input operation. I will list every important use case/submodule in this section, and then elaborate on the test to be used or elucidate as to why no test is needed.

PourBeaker/TestTube - This test will consist of affirming that moving a beaker or test tube over an appropriate object will empty the contents of the beaker/test tube onto or into the appropriate object.

Input	Action	Output

-Beaker/test tube full	Move beaker/test tube over appropriate object	Success: object now contains contents of beaker/test tube
-Beaker/test tube not full	Move beaker/test tube over appropriate object	Failure: beaker /test tube does not have any contents
-Beaker/test tube full	Move beaker/test tube over inappropriate object	Failure: Object does not interact with beaker/test tube

testpH - This test will consist of affirming that moving a piece of pH paper into an associated liquid will result in a change of color for the pH paper and be an accurate representation of the liquid being tested.

Input	Action	Output
-clean pH paper	Move pH paper over acid or base	Success: pH paper changes color reflecting acid or base
-already used pH paper	move pH paper over acid/ base	Failure: pH paper already used
-clean pH paper	move pH paper over non liquid	Failure: pH paper needs to be testing in an associated liquid

calibrateSpec20 - This doesn't need testing, as it only requires setting the settings of the Spec20. Obviously the output of the Spec20 needs to match the input, but that's the only area of error which exists and can easily be identified/fixed during the lab.

ReadAbsorbance - This test will consist of confirming that the contents being tested in the spec20 are reading the correct absorbency.

Input	Action	Output
-full test tube in spec20	Read absorbancy	Success: Spec20 reads off correct absorbancy for chemical in test tube
-improper test tube/ test tube not full	Read absorbancy	Failure: test tube does not have any contents and/or invalid test tube
-full test tube in spec20, wrong settings	Read absorbancy	Failure: incorrect or no absorbancy due to bad parameters

Create LabSpace - This does not need a test, again because of it's simplicity in the output for the given input. When a user requests for a new lab, the createLabSpace should always give him the correct lab with the correct materials. Failure of this will be seen immediately

and able to be fixed without any extra testing needed.

Karyotype Identifier - This test will consist of confirming that the shown chromosome mutations have the appropriate disease name attached to it.

Input	Action	Output
-Chromosome mutation	Match up appropriate mutation name	Success: Correct chromosome mutation name
-Chromosome mutation	Match up incorrect mutation name	Failure: Incorrect chromosome mutation

RegisterClass - Test will consist of multiple different inputs and actions to the RegisterClass function to ensure proper creation. Possible tests with outcomes include:

Input	Action	Output
-Instructor login	Register new Class	Success: new class registered
-Student login	Register new Class	Failure: User doesn't have proper credentials
-Instructor login	Register existing Class	Failure: Class already exists

RegisterStudent - Test will consist of different attempts of registering students to a Bio section. Possible tests with outcomes include:

ASSUMED: STUDENT LOGIN

Input	Action	Output
-Correct section, non-full section, and NetID	Register Student	Success: You are now registered
-nonexisting section and NetID	Register Student	Failure: Section does not exist
- Correct section, but improper NetID	Register Student	Failure: Student is not in course
-Correct section and NetID, full section	Register Student	Failure: Section is full

ViewStudentResult - This function doesn't need a test, as it's behavior is very one dimensional and all that's important is that the information comes up when requested. Any fail case for this function would be a graphical bug in the interface/program which wouldn't need external testing.

ViewClassResult - Similar to ViewStudentResult, this function is very one dimensional and should only be viewable to the instructor. If the ViewClassResult graphical interface button were to appear for a student or the information were not to come up, then it would be a bug with the graphical interface which wouldn't require necessary external testing.

6.2 Test Coverage

The test coverage for this software is two fold: (1) Ensure that invalid inputs are dealt with quickly and gracefully, and (2) Ensure that there are no software coding faults which may present an invalid output or which may fail to execute correctly. In order to be certain that our software does what is expected of it, we have created a litany of tests which should encapsulate the majority of problems with our program. Virtually every method being tested has a combination of incorrect and correct inputs along with another combination of incorrect and correct actions to exhume any and all outputs available. We feel that these tests are sufficient (although could potentially be revised) to ensure proper operation under all inclement loads. As a brief example, let's quickly look at testpH and registerClass to discuss what is exactly covered under each.

testpH -
(1) The first of these tests is the one which leads to the success case, in which a blank piece of pH paper is used on the appropriate liquid (either acid or base) and the pH paper turns the correct color. This is the easiest and most obvious case, where everything goes smoothly and correctly
(2) The second test is a fail case, where we try to re-use a piece of pH paper on a new liquid (either acid or base). In this instance, the pH paper should maintain the same color and not change or morph colors due to the new liquid being exposed to it. This way, we guarantee that one cannot accidentally mix up the chemicals' resultant pH paper.
(3) The third test is a fail case, where we try to move a clean piece of pH paper over an inappropriate object (e.g empty test tube/beaker, spec20, microscope, heat plate, etc). In this instance, the pH paper should maintain it's color and form until properly exposed to the correct object. However, what this test does not correct is that if a user wanted to test test tube 1, but accidentally goes over test tube 2, the pH paper WILL change color reflecting the

contents of test tube 2.

registerClass -
(1) the first of these tests is a pass case, where the proper credentials are presented (instructor login) and the instructor attempts to register a new section which does not already exist. Although there will be a software limitation on the number of classes which can be created, we didn't feel it to be necessary to test the cap due to realistic uses of the software which wouldn't lend itself to testing such superfluous things.
(2) the second of these tests is a fail case, where the improper credentials are presented (student login) and the button to register a class is unavailable. Obviously, a student should not possess power to create a new section, and the button should be unavailable to every individual who logs in with non-instructor credentials. This is pivotal to proper operation of our software.
(3) the third test is also a fail case, wherein by a user who logs in with the proper credentials (instructor login) attempts to create a new section which already exists. The software will not allow an action, and present the user with an error stating "error: section already exists". This is to ensure that there are no duplicate sections being created/overwritten.

These are just two examples of the dozen or so methods which were tested, yet represent the logic and thought process which went into each method. Basically, we want strict adherence to software functionality and handling when presented with improper inputs. With those problems sorted, our software becomes much more stalwart and user friendly.

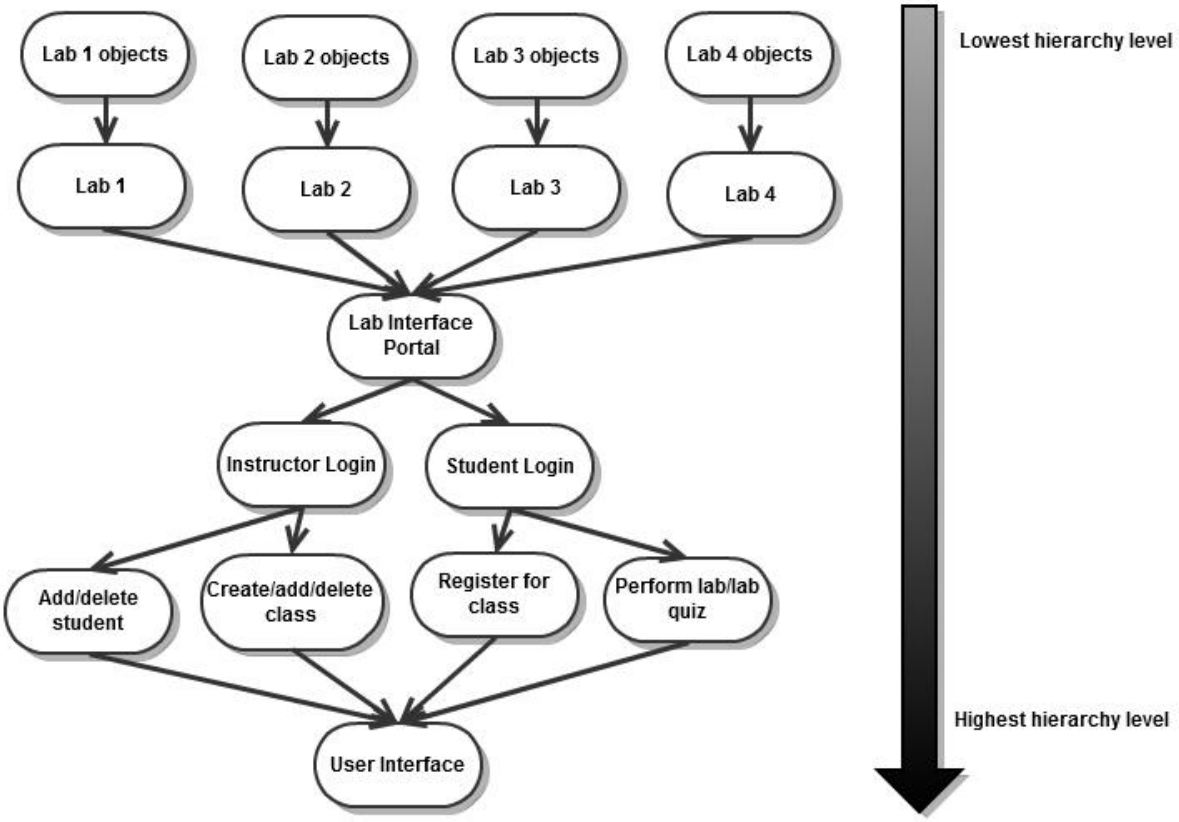
6.3 Integration Testing Strategy

Our integration testing strategy will implement the Bottom-Up integration path. This integration path requires the testing of low level, independent pieces on the hierarchy and working your way up until you reach the top piece, which interconnects all bottom pieces.

For us, the low level pieces will be the labs and the contents of the labs. We will work on each lab independently, ensuring that proper operation exists independent of everything else. Then, once the labs work sufficiently, we will begin to integrate them into our user interface, making sure that the user interface produces the appropriate lab, etc. From there, we will begin to add other user interface functionality, including the ability for students and instructors to create/register for sections and view given labs/quizzes/grades. Finally, all of it will be brought together with the user interface having the correct buttons and display for every individual part and allowing quick navigation between any two parts (e.g labs and grades).

We feel that this testing strategy is the most appropriate for our design, as lab

functionality is a huge - yet independent - portion of our software. Therefore, although our strategy is somewhat bottom heavy, we can easily work our way bottom to top knowing that the lower pieces on the hierarchy are independent and correct. Here is a brief diagram summarizing everything stated:



7. Project Management and Plan of Work

7.1 Merging the Contributions from Individual Team Members

To complete this project, we all created a google Document that allowed for everyone to update the same final report simultaneously. With this web document we were able to format and complete each section, upload all of the images of the use case diagrams, and leave comments on the document for any one that was reading it. This meant that every persons contribution to our report was in the same place from the beginning, and only a few formatting changes needed to be made before submitting the report just to ensure that everything was uniform and consistent.

7.2 Project Coordination and Progress Report.

The week of February 20th was devoted to the writing of the second report, starting with the week of February 27th we will begin implementing all of the use cases and creating the images and animations for the actual laboratories. We are going to have another group meeting to assign jobs and create a schedule of sub meetings for two or more people to meet and work on their contribution to the demo. By the submission of the final part of the second report we will have an outline of each sub group and when they worked on each part of their responsibilities.

Each part of the report was the responsibility of a specific person to complete and they were in charge of managing that sections. That person was in charge of either completing the work or delegating it to other group members. They were also in charge of making sure their section was finished early enough so that if any other section depended on that information, there would also be sufficient time to complete the second section.

As of the week of March 4th, a significant amount of work has been put into creating the first demo. The week of March 11th is going to be devoted to fixing mistakes made in the first and second report. During this time our group needs to make changes to the use cases and other important parts of the first report in order to better reflect the priority weight of all of the system requirements.

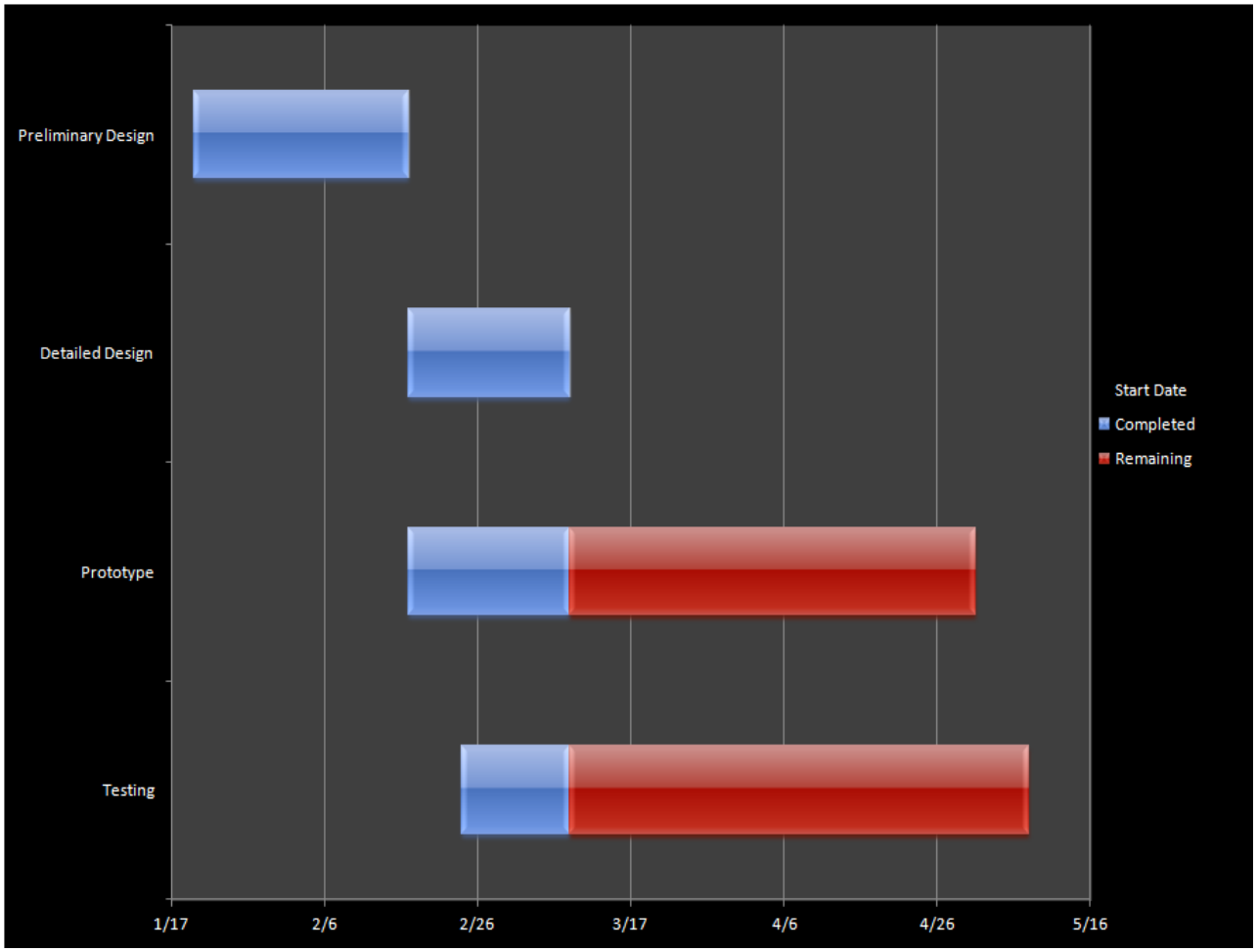
7.3 Plan of Work

With our first report we submitted the schedule shown below with our plan or work, in this schedule we listed all of the due dates and items we hoped to finish before that. The milestones that we have to complete to accomplish each report and demo are implementing and testing each of the non functional use cases, for example the register student and view class results use cases.
Most of the emphasis will be placed on the Preform Lab use case and the following use cases:

- Laboratory One Cell Division
- Laboratory Two Biology Molecules
- Laboratory Three Enzyme Activity
- Laboratory Four Meiosis

In order to complete these labs we must also implement the following:

- A laboratory space
- Microscope
- Spectronic 20 Spectrophotometer
- Karyotyping generator



7.4 Breakdown of Responsibilities

Team Member	Class and Module Responsibilities
Ryan Cullinane	Microscope, Prepare Slide (Register Class)
Michael DiLalo	Laboratory One Cell Division (View Student Result, View Class Result)
Nicholas Guida	Spectronic 20 Spectrophotometer (Register Student)
Steven Lu	Laboratory Two Biology Molecules, (Add Student)
Kevin Miller	Laboratory Three Enzyme Activity
Cady Motyka	Laboratory Four Meiosis (Karyotyping generator, Take Lab Quiz)

Responsible for Coordinating Integration	**Steven Lu**

Responsible for Coordinating Testing	**Ryan Cullinane**

Contributions to Second Report:

Responsibilities	Ryan Cullinane	Mike Dilalo	Nick Guida	Steven Lu	Kevin Miller	Cady Moytka	Totals	Possible Points
Totals:	**16.9**	**19.4**	**15.5**	**15.8**	**7.7**	**24.7**	**100%**	**100**
Project Managment	20%	25%	10%	10%	10%	25%	**100%**	**20**
1.1 Interaction Diagrams lab14	0%	70%	0%	0%	10%	20%	**100%**	**13**
1.1 Interaction Diagrams lab23	10%	0%	0%	0%	10%	80%	**100%**	**12**
2.1 Class Diagrams and Specs	20%	20%	0%	0%	0%	60%	**100%**	**10**
2.2 Activity Diagrams	0%	0%	50%	0%	0%	50%	**100%**	**3**
2.3 System arch and design	0%	0%	0%	100%	0%	0%	**100%**	**13**
3.1 subsystems	100%	0%	0%	0%	0%	0%	**100%**	**2**
3.2 Algorithms and Data	0%	0%	0%	20%	80%	0%	**100%**	**4**
3.3 User Interface	70%	30%	0%	0%	0%	0%	**100%**	**11**
3.4 Design of Tests	0%	0%	100%	0%	0%	0%	**100%**	**12**

Contributions to First Demo:

First Laboratory	Mike
Back end	Steven

8. References

Campbell, Neil, Jane Reece, Lisa Urry, Michael Cain, Steven Wasserman, Peter Minorsky, and Robert Jackson. *Biology*. 2nd Edition. Volume 1. San Francisco: Pearson, 2008.

General Biology 01:119:101- 2011 Laboratory Manual

Marsic, Ivan. "Software Engineering.", 16/01/2012. 1 Feb 2012. <http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf>.

Solomon, Eldra, Linda Berg, and Diana Martin. *Biology*. 8th Edition. Volume 2. Belmont: Thomson Brooks/Cole, 2008.