# Virtual Biology Lab Simulation: Mitosis Group 11

*Steve Pak, Marvin Liu, Anthony Porturas, Brandon Lum,*
*Evan Chipps, Jeffrey Gillen*

Project Website[1]

[1]https://sites.google.com/site/virtualbiologylab/

# Individual Contributions

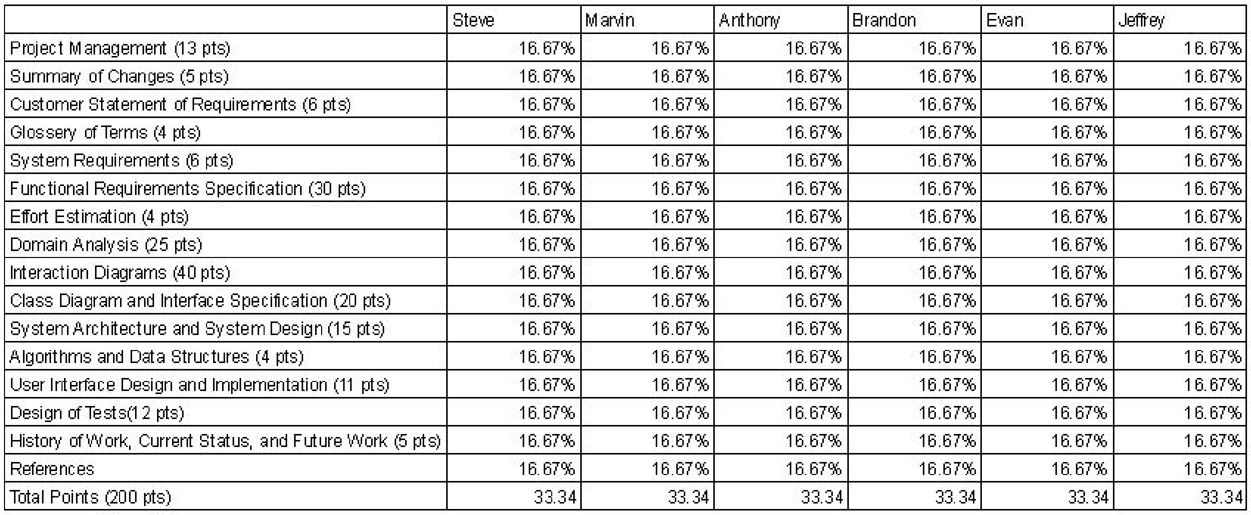
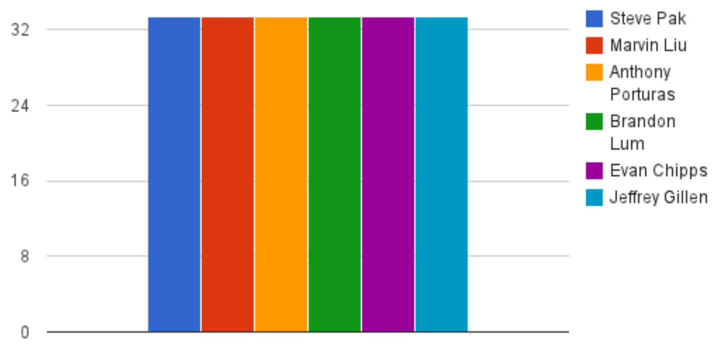


Figure 1: Responsibility Matrix

| | Steve | Marvin | Anthony | Brandon | Evan | Jeffrey |
|---|---|---|---|---|---|---|
| Project Management (13 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Summary of Changes (5 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Customer Statement of Requirements (6 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Glossery of Terms (4 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| System Requirements (6 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Functional Requirements Specification (30 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Effort Estimation (4 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Domain Analysis (25 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Interaction Diagrams (40 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Class Diagram and Interface Specification (20 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| System Architecture and System Design (15 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Algorithms and Data Structures (4 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| User Interface Design and Implementation (11 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Design of Tests(12 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| History of Work, Current Status, and Future Work (5 pts) | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| References | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% | 16.67% |
| Total Points (200 pts) | 33.34 | 33.34 | 33.34 | 33.34 | 33.34 | 33.34 |

Figure 2: Responsibility Allocation



**Steve Pak** 16.67%—33.34 points

**Marvin Liu** 16.67%—33.34 points

**Anthony Porturas** 16.67%—33.34 points

**Brandon Lum** 16.67%—33.34 points

**Evan Chipps** 16.67%—33.34 points

**Jeff Gillen** 16.67%—33.34 points

# Contents

## 1 Summary of Changes

- Integration diagrams
- Grading Scheme and Mathematical Model

Note: this document will refer to students and professors as male at times, in order to simplify writing, and to avoid reading "the student" multiple times within the same sentence or paragraph. We understand that students and professors can be female as well.

# 2 Customer Statement of Requirements

## 2.1 Problem Statement

Laboratories are used every year by students worldwide to gain hands on experience simulating and observing the mysteries of science. Experiments are constructed from multiple variables, ranging from student and professor experience with materials to equipment familiarity and budget. When a new lab procedure is created, schools must acquire equipment, which can cost upwards of thousands of dollars. Depending on the schools budget, some schools may be limited to the quantity of physical devices acquired. With physical devices, maintenance becomes necessary due to the following: everyday use, age, and even misuse–which is expensive to repair or even replace. A full list of the problem domain is listed below:

1. Lab equipment is very expensive.
2. Lab equipment is limited in quantity
3. Class time is limited; as a result, the lab design may exclude some important concepts. Additionally, those unable to finish must return at another time to complete it, or risk obtaining a lower grade.
4. Some lab equipment is difficult to use without a tutorial or manual.
5. Time-consuming lab demonstrations are inexpedient.
6. Monitoring students become impossible due to teacher to student ratio, leads to a lot of wasted time, as a Lab TA can only help one group at a time, while the students that need help ended up waiting.
7. In cases of misuse, the repairing/replacing of lab equipment uses up time and money.
8. Its difficult to get an accurate lab assessment on performance in a real life experiment.
9. Interactivity is preferable for students because it engages them in learning.
10. Real life visuals may be too hard to analyze correctly for beginners.
11. Students should be able to learn at their own pace and run the lab simulation as much times as needed for better understanding..
12. Simplicity is preferred when introducing students to new subjects.
13. There is no consistent standard grading policy due to human error.
14. In regular lab assignments, students work in groups, which hinders individual learning. Some teammates may do the work while the others sit and watch.

Going to an actual laboratory can be inconvenient and cause numerous problems for both the school and students. Stress associated with these laboratory issues range from financial burdens to time constraints. Additionally, lab equipment is expensive, limited in supply, and complicated to learn for beginners. They can also become broken and misused, forcing the school to buy new equipment so other students can use them in the future.

Equipment such as microscopes, tools, and B & L Spectronic 20 spectrophotometers can cost thousands of dollars per device. The cost is significantly higher if one of each is required for each group of students in a class. Furthermore, some students will not be able to finish the lab because of a lack of supplies and equipment. They will end up waiting until another group finishes using their equipment, which wastes time. This causes stress and a flimsy knowledge of the material set as the learning objective. Due to limited time and resources, including availability of professors and Teaching Assistants, students may often receive inadequate feedback, inhibiting improvement in future labs.

To remedy these real life problems, a web-based program will be created that is capable of running the biology mitosis that is used by the General Biology (01:119:101) course at Rutgers University. A virtual biology lab provides the tools and a controlled environment for which students can perform their duties in the lab without the limitations that are generally associated with their real-world counterparts. This description of the lab experiment can be seen here:

- Lab 2: Cell Division[2]
  Objective:.

  1. Describe each stage of nuclear division (mitosis) and cytoplasmic division (cytokinesis) and identify each under the microscope
  2. Use a compound microscope and describe two stains used to facilitate observation
  3. Graph data correctly paying particular attention to the differences between independent and dependent variables

Virtual Biology labs solve the need for physical equipment which will greatly reduce the cost and time associated with maintaining such products. Along with such improvements, this solution will provide students with an analysis of their work as they progress through the lab in the form feedback preset by the TA and/or Professor.

The virtual biology lab would allow students to learn and execute the lab at a pace that they feel comfortable, allowing time for them to absorb and fully understand the material. Students that can grasp concepts more easily than others will no longer be hindered by those that need more time while those that need more time wont be pressured to keep up. The virtual solution will generate a more individualistic approach to the lab environment to prepare allow perform the lab to the best of their ability.

The web-based solution will consist of a simple yet aesthetically pleasing user interface consisting of some key features that make this product stand out. Some basic features included: a user interface for both students and professors, the actual lab simulation, and to observe student performance in the lab simulation, respectively. Some stand out features that are to be included with this solution: feedback preset by the TA/Professor and a strike system to provide warnings to students indicating points of error.

One of the more exclusive features, feedback, will be dynamic. Depending on the students response to an activity or question, the system will provide feedback appropriate to the selection. Example: a student selects a piece of equipment from a list, if the student selects an incorrect tool, the simulation

[2]http://www.ece.rutgers.edu/~marsic/books/SE/projects/ViBE/biolab-1.pdf

will tell the student a mistake has been made and provide the name of said equipment and ask the student to try again. The system will keep track of the mistakes and provide the professor/grade with a list for reference.

Through this website, the professors will have access to a list of features to which they can choose to include with their corresponding labs. The professor will have access to a list of features including, but not limited to: feedback system, strike system, grades, and a summary system per each student.

Following the Mitosis Lab Manual from the description above, courtesy of Rutgers University, the simulation will run as follows. At first, the simulation will start off with a learning animation, where the steps of mitosis(interphase, prophase, metaphase, anaphase, telophase) are demonstrated. This serves as a quick overview and precursor to the actual lab experiment, analogous a TA giving a brief description and lesson prior to the experiment. Following the brief learning session, the interactive portion of the experiment will begin. It will start off with the student controlling a knife, cutting a piece of the onion root, and placing it on the slide. A stain will be used by the student to make observing the cell less difficult. The student will then place the slide under a microscope, and it will show the cell, possibly out of focus, forcing the student to focus the microscope. The students will then observe the cells dividing and the different phases of Mitosis. The steps above will be repeated for the whitefish, and prompt the student to note the differences. After observing the animal cell, the simulation will then go through an interactive tutorial on graphs and table, specifically the pie chart and bar graph. Data collection is an important part of any lab experiment, and without it, results of the experiments are hard to analyze. For this reason, the team decided that data collection is a crucial part of the experiment and of the lab itself, so it is included within the simulation.

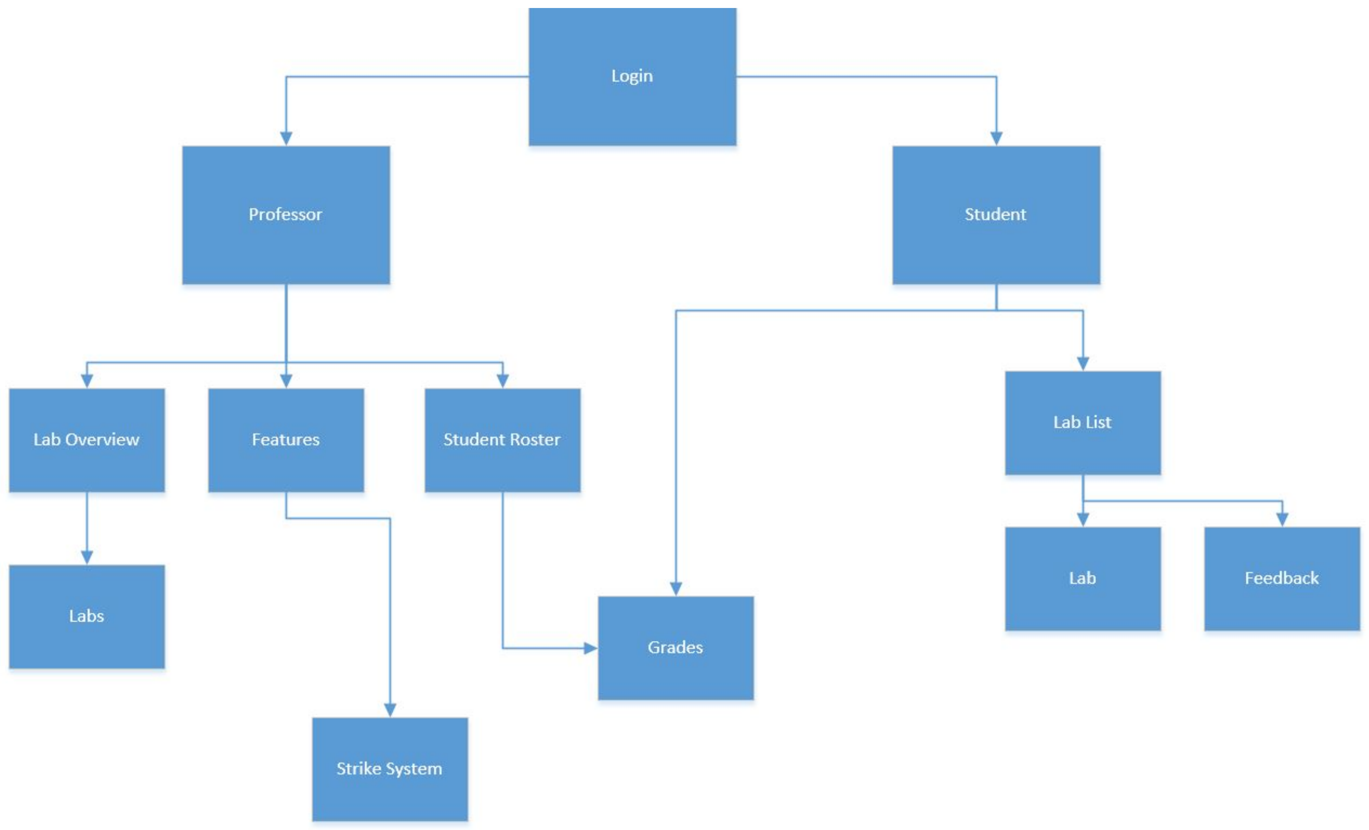
The lab manual describes its objectives below:

- describe each stage of nuclear division (mitosis) and cytoplasmic division (cytokinesis) and identify each under the microscope.
- use a compound microscope and describe two stains(acetocarmine or aceto-orcein) used to facilitate observation.
- graph data correctly, paying particular attention to the difference between independent and dependent variables.

The above simulation not only fulfills the above goals, but creates an improved solution alternative to a physical lab experiment in terms of cost and convenience.

From the virtual biology lab, users will gain the ability to perform the lab simulations from anywhere with Internet access. Since each student will be logging in using their Rutgers University netID, their grades and progress will be saved accordingly: instructors can see each students grades and mistakes. Additionally, students will be given immediate, real-time feedback if a mistake is made during the lab simulation. From this, they will be able to learn from their mistakes and better understand what changes are necessary.

Figure 3: Graphic of a diagram visualizing the functionality of the program to the different users



## 2.2 Glossary

| Term | Definition |
|---|---|
| Anaphase | The stage of mitosis in which the chromosomes move toward the poles of the spindle. |
| Apical Meristem | A region of active growth (in this case, of an onion root). |
| Centromere | The point or region on a chromosome to which the spindle attaches during mitosis. Generally found in the center of the chromosome. |
| Chromatid | One of the usually paired and parallel strands of a duplicated chromosome joined by a single centromere. |
| Chromosome | The part of the cell that contains the genes which control |
| Cytokinesis | The process of in which a single cell splits itself into two equal cells through cytoplasmic division. |
| Cytoplasm | The organized complex of inorganic and organic substances external to the nuclear membrane of a cell. |
| Diploid | A cell or an organism consisting of two sets of chromosomes (2n). |
| Dye Solution | A solution of acetocarmine or aceto-orcein used to increase ones ability to see small structures within the cell. |
| Equatorial Plane | The plane perpendicular to the spindle of a dividing cell and midway between the poles. |
| Error | An incorrect action within the simulation, that the system detects. |

| | |
|---|---|
| Haploid | A cell or an organism consisting of one set of chromosomes (n). |
| Homologous | Having the same relative position, value, or structure. |
| Hot Plate | A machine used to heat material. |
| Interphase | The interval between the end of one mitotic division and the beginning of another. |
| Meristematic Region | The region in which cell division is occurring. |
| Metaphase | The stage of mitosis in which the chromosomes become arranged in the equatorial plane of the spindle. |
| Metaphase Plate | Also known as equatorial plane. see definition of equatorial plane. |
| Microscope | A device used for producing a much larger view of very small objects so that they can be clearly seen. |
| Microscope Slide | A thin piece of glass that is used to hold material that will be inspected under a microscope. |
| Mitosis | A type of nuclear division that involves a sequence of events by which the nuclear material of one cell is distributed into two equal parts. |
| Navigation Portal | The main menu screen that both students and professors will see as soon as they log in. This serves as a central node to access different parts of the system. |
| Nuclear Envelope | A double layered membrane that envelops the nucleus of a cell, separating the contents of the nucleus from the cytoplasm. |
| Nucleolus (Nucleoli) | A round granular structure within the nucleus of a cell, composed of protein and RNA (ribonucleic acid). |
| Nucleus | The central part of most cells that contains genetic material and is enclosed in a membrane. |
| Pole | Two structures that exist on either side of the cell that is divided by the equator. |
| Prophase | The initial stage of mitosis characterized by the condensation of chromosomes consisting of two chromatids, disappearance of the nucleolus and nuclear membrane, and formation of mitotic spindle. |
| Root Cap | A mass of cells at the end of a root (in this case, of an onion) that cover and protect the growing cells. |
| Stain | Also known as dye solution. see definition of dye solution. |
| Somatic Cell | One of the cells of the body that make up the tissues, organs, and parts of that individual other than the germ cells. |
| Spindle Fiber | Any of a network of filaments that collectively form a mitotic spindle that is involved in moving and segregating the chromosomes. |
| Telophase | The final stage of mitosis in which the spindle disappears and the nucleus reforms around each set of chromosomes. |
| Whitefish Blastula | An early stage in the embryonic development of a whitefish. |

Table 1: Glossary of Terms

Figure 4: Cell Structure: Plant Vs Animal

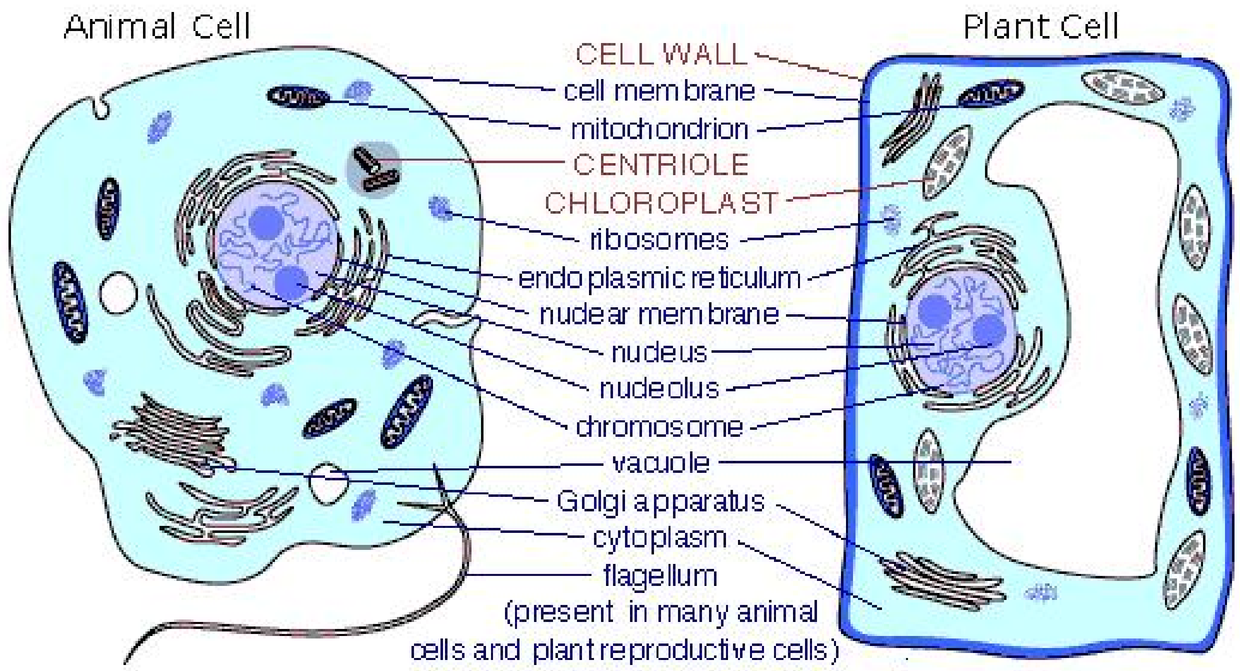
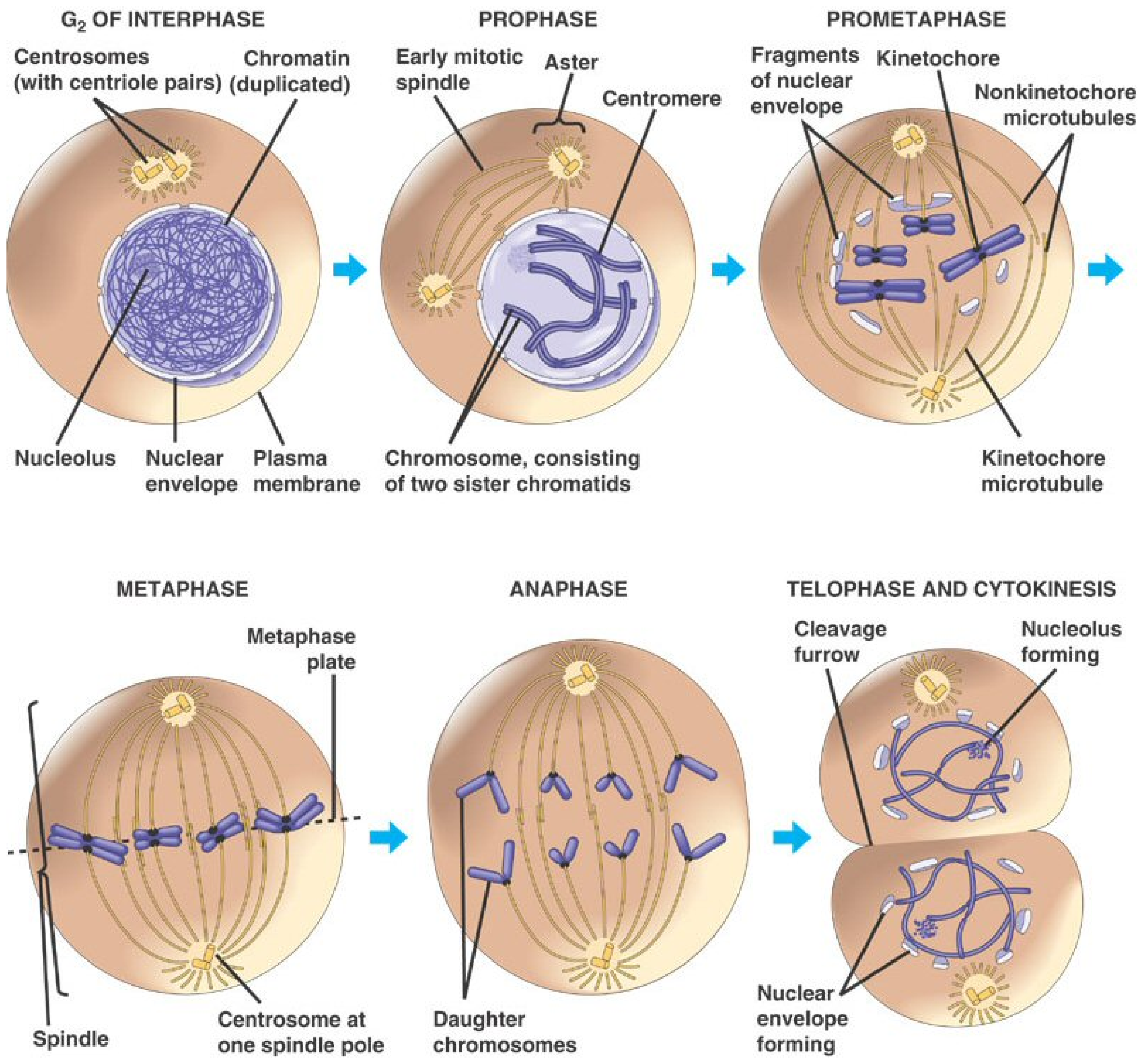


Figure 5: Stages of Mitosis



# 3 System Requirements

The virtual biology lab is extremely complex, as opposed to being complicated. While it seems very difficult to implement such a system, in reality, the system is essentially made up of many smaller and simpler features. The feature list of this system is both numerous and unique in nature. To simplify the need to specify which requirements are necessary and higher priority compared to other features which are optional and low priority, 2 enumerated requirement lists were created, one to specify the Functional Requirements, the other to specify the Non-Functional Requirements. Lastly, a table specifying the On-Screen Appearance Requirements is added below to explain what graphics are needed.

## 3.1 Functional Requirements

The team defined the functional requirements as any feature of the system that is mechanical in nature. For example, does it allow the student to run the simulation? These requirements are easy to test for, and many can be tested by simply running the program and seeing if it does what it says it does. The list is shown here: .

| Identifier | Priority Weight | Requirement |
|---|---|---|
| REQ-1 | 5 | The system shall run the entirety of the mitosis lab simulation when accessed by the user. |
| REQ-2 | 4 | The system shall allow the simulation to be interactive and complex so it has many options available. The options include: 1. A wide variety of incorrect choices shall be available, along with the correct answers, to allow the students to make errors. 2. The system shall allow the students to change microscope settings such as zooming in and out, to determine the best view of the slides. 3. The system shall allow the user to create graphs and tables discussing the time spent in each stage of mitosis. 4. The system shall allow the user to illustrate each stage of mitosis. 5. The system shall have an introduction simulation slideshow to the mitosis lab. |

| | | |
|---|---|---|
| REQ-3 | 4 | The system shall track the students actions and do the following:<br>1. The system shall send a message to the student as a feedback system, showing the students answer as either correct or incorrect.<br>2. If the action was an error, the system will keep count of the errors, as well as record where in the simulation it occurred. The feedback system mentioned previously shall elaborate on what the student did incorrectly.<br>3. At the end of the simulation, a record of the student's performance shall be saved in a database. |
| REQ-4 | 3 | The system shall make an action when the error count reaches the amount specified by the professor:<br>1. Points shall be deducted from the lab grade.<br>2. The student shall be forced to repeat the simulation. |
| REQ-5 | 2 | The system shall allow the professor to access and edit grades associated with students that are registered for the course, accessed through the database. |
| REQ-6 | 2 | The system shall allow a user to log in as professor or student. |
| REQ-7 | 2 | The system shall allow new users to register for the class. |
| REQ-8 | 1 | The system shall allow the student to view his or her grade, accessed through the database. |

Table 2: Functional Requirements

Obviously, REQ-1 is the highest priority. The simulation is the bread and butter of this project, without it, this project has no base. The key challenges within the simulation that the team decided to address are as follows: realistic simulation of the mitosis lab, and real-time feedback on the interactions of the student. The above table exemplifies this in REQ-2 and REQ-3. The team discussed the possibilities of the simulation and decided that the lab should not be linear. Instead, there are options for the student as a user to branch out from what the lab manual deems a correct response. Another way to interact with the system, and is necessary for any lab experiment, is data collection. The system should allow the student to create graphs and tables to collect the results of the experiment. Lastly, the student should be able to use tools effectively. In this experiment, the microscope is the most widely used tool, as well as equipment relating to the microscope. The system shall have a realistic simulation of the focusing of microscopes. These features are extremely important, because, sometimes the best way to learn is to learn from your mistakes. That is why the weight of REQ-2 is high(4).

A student should have the right to know whether a mistake was made. The system lets the student know through the feedback system described in REQ-3. The system will message the student, letting them know whether they made the correct action, or if they commited an error. If the student made the correct action, they will be allowed to move on to the next step. If they commited an error, the student will have to try again until they make the correct response. The feedback system will also be responsible for keeping track of the student errors. There is no need to keep track of the student's correct actions as the system will know the path the student took just by knowing which errors he made and which stage of the simulation he is at.

This project is heavily designed for users that are students taking biology. However, professors teaching the course should have some power and control over this system. REQ-4 and REQ-5 are directed towards the professor. The professor needs to keep track of the list of students registered for his course. Therefore, the professor should have access to a system feature that allows this, exemplified through REQ-4. The system will pull up a list of students and their ID numbers as well as grades for the lab, when the professor requests it. The professor should allow changes to be made, if need be.

A static universal grading scheme designed into the system is decidedly a bad idea. Different professors may have different policies on grading. This system will allow the professor to set options based on his policy and preference. Two over-lying options exist: Deduct points and force the student to redo the lab. A combination of the two is possible. The professor will then choose a number which represents the number of errors the student can make before a penalty ensues.

### 3.1.1 Acceptance Test

The acceptance test for the requirements can be seen below. The acceptance test is a way to measure the success of the project based on a specific requirement. The customer will run this test, if it passes, the requirement has been fulfilled. If the test fails, the requirement has not been fulfilled, and the project can be deemed a failure by the customer. The acceptance test cases will be labeled as ATCX.YY, where X is the requirement number and maps to REQ-X, and YY is the test case number, the first test case labeled as 1.

| | |
|---|---|
| ATC1.01 | Ensure that the computer can access the simulation. Run the simulation from start to finish, making only correct actions, to allow the simulation to move to the next step. If user can reach the end, the test is successful. |
| ATC2.01 | Ensure that the computer can access the simulation. Run the simulation, trying to make mistakes occasionally, checking to see if there are many options available to the user. If the user has many options, and the simulation is not linear, but branches out, the test is sucessful. |
| ATC2.02 | Ensure that the computer can access the simulation and is at the part of the simulation of controlling the microscope. If the zoom function works as pleases to the user, the test is successful. |
| ATC3.01 | Ensure that the computer can access the simulation. Commit an known error in the simulation. The system should respond with some sort of feedback message. If so, the test is successful. |
| ATC4.01 | Ensure that the computer can access the simulation and the grading policy is that the grade is reduced when the error count passes the threshold. Commit an amount of errors equal to the threshold of errors allowed. If the system responds with a message saying that the user commited too many errors, and reflect that on the grade at the end of the simulation, the test is successful. |

| | |
|---|---|
| ATC4.02 | Ensure that the computer can access the simulation and the grading policy is that the simulation restarts when the error count passes the threshold. Commit an amount of errors equal to the threshold of error allowed. If the system responds with a message saying that the user commited too many errors, and restart the simulation. |

Table 3: Functional Requirements Acceptance Test Cases

These acceptance tests obviously don't go into the extreme details, but should give a good idea on how the system fulfills the requirements.

One last note: Complete analysis has not been done in the totality of the requirements, due to time constraints. Only a select important ones were chosen to demonstrate and explain how the project will function.

## 3.2 Non-Functional Requirements

The team defined the Non-Functional Requirements as the features that improved the quality-of-life of the system as a whole. Non-Functional Requirements usually fell under one of the category in FURPS+[3], but sometimes the requirements fell into more than one category. Although Non-Functional Requirements are harder to quantify and seem trivial, they are just as important as the Functional Requirements. The list is given below:

| Identifier | Priority Weight | Requirement |
|---|---|---|
| REQ-9 | 5 | The lab simulation shall be accurate in showing clear differences of each stage of mitosis: interphase, prophase, metaphase, anaphase, telophase. |
| REQ-10 | 4 | The feedback system shall immediately display comments on what went wrong if the students made a mistake and if they answered the questions correctly. |
| REQ-11 | 3 | The system shall periodically save the students progress after every action, allowing the student to exit and resume at a later point. |
| REQ-12 | 2 | The simulation shall be maneuverable, allowing for clicking next to move on and building a table of contents after each completed part of the lab experiment. |
| REQ-13 | 2 | The lab simulation shall be accurate in showing clear differences between plant and animal cells. |
| REQ-14 | 2 | The system should include a navigation portal which serves as a central node to access other menus, following after the log in. |

Table 4: Non-Functional Requirements

As state above before the table, Non-functional requirements are more about the quality of the content as opposed to the content itself. REQ-9 is a necessary requirement. This project is to be used as a learning tool, and if students do not learn the proper differences of each stage of this lab, this

[3]http://en.wikipedia.org/wiki/FURPS

project would be a failure. Each stage should be clearly defined and should not be confusing in any way. For this reason, REQ-8 is labeled with priority weight of 5. A similar requirement, REQ-13, is also essential to teach the students, although, it is a secondary goal of the lab, not a main goal

In that same vein, REQ-10 allows the student to learn from their mistake. This simulation should not be seen as a cruel test of their knowledge, but a tool to learn new things, and to reinforce what students already know. It is not meant to find each and every way to make a student fail. That is why the feedback system should be helpful and encouraging and allow the student to know exactly where they commited an error. This project as a whole should make biology more enjoyable and easier to learn. Another benefit is that students rely less on a human TA who can only do so much. For example, a class of 25 students are performing an experiment. During the experiment, students may have to ask the lab instructor questions. In cases where multiple student have a question, there is a lot of wasted time, as the instructor must go to each student with a question. This problem would be solved if this requirement is fulfilled, which is why it has a weight of 4.

Sometimes, students cannot finish the lab in the alloted time. In the case of physical lab experiments, the student would have to clean up his station, and then reset his station when he can work on it again. In the virtual lab, this problem doesn't exist. There is no need for a station set up, or clean up. If the student doesn't finish on time, the system will save his progress, and allow him to come back whenever he's available. The way this works is similar to a save point. These save points would be set up every one or two steps in the simulation, and if the student passes the step, the system will save that state. That way, if the student leaves in the next step, when he comes back, the system will load up that state. In that same sense, the student will also backtrack his progress if he wants to reinforce the concepts. These features can be seen in REQ-11 and REQ-12.

The system needs to be able to navigate through the different menus. This can be done through the Navigation Portal. This is set at a low priority as it is not necessary for the system to be successful, set at a weight of 2.

### 3.2.1 Acceptance Test

Following the conventions in the functional requirement acceptance tests, here are the nonfunctional requirement acceptance tests.

| | |
|---|---|
| ATC9.01 | Ensure that the computer has access to the simulation. Navigate through the simulation, making sure that the different stages are well defined and correct, following the definitions in the glossary. If so, the test is successful. |
| ATC10.01 | Ensure that the computer has access to the simulation. Commit an known error in the simulation. If the feedback message is unhelpful, or absent this test fails. If the message is very clear and explains the mistake well, the test is successful. |
| ATC11.01 | Ensure that the computer has access to the simulation. Start the simulation, and stop somewhere in the middle, and exit the simulation. Access the simulation again, and if the simulation starts at a near point previous to the point where the user left off, with the progress saved, the test is successful. |

Table 5: Non-Functional Requirements Acceptance Test Cases

Once again, this is not an exhaustive list of the acceptance test cases for all of the requirements. It does, however give the customer a very good feel on the fulfilment of the higher priority requirements. A final note: All of the test have only 2 final state, success or failure. If the test says that a condition makes the test successful, and that condition isn't met, the test is considered failed.

## 3.3 On-Screen Appearance Requirements

In order to have a simulation, graphic images of each object in the lab must be found or created. Since it is hard to determine whether a slide or a knife has more priority, the team decided that most trivial, internet searchable image had the same priority, but other graphics that are specific to this system, and must be created from scratch have higher priority. For example, the Navigation Portal background is specific to the system where as a knife can be very simply found by doing a quick google search.

| Identifier | Priority Weight | Requirement |
|---|---|---|
| REQ-14 | 5 | The system shall have an aesthetically pleasing user interface. This interface will be capable of displaying the following things:<br>1. The log in screen for both student and professor.<br>2. The Navigation Portal, with menu buttons to access the simulation or grades for any user, or the grading scheme menu and the manage users menu for professors.<br>3. The simulation itself which includes the graphics for each physical element of the lab and the different pictures of mitosis and the cell as well as the tools.<br>4. Feedback messages to the screen sent by the feedback system.<br>5. A registration screen for non-users. |

Table 6: On-Screen Appearance Requirements

The On-Screen Appearance requirements are essential to this project and system. Without any of these items, the simulation cannot run. If the students cannot see the different stages of mitosis, or if a graphic of a microscope was missing, the system would not seem to work, even if it was coded properly. For this reason, this requirement was given a weight of 5.

### 3.3.1 Graphics

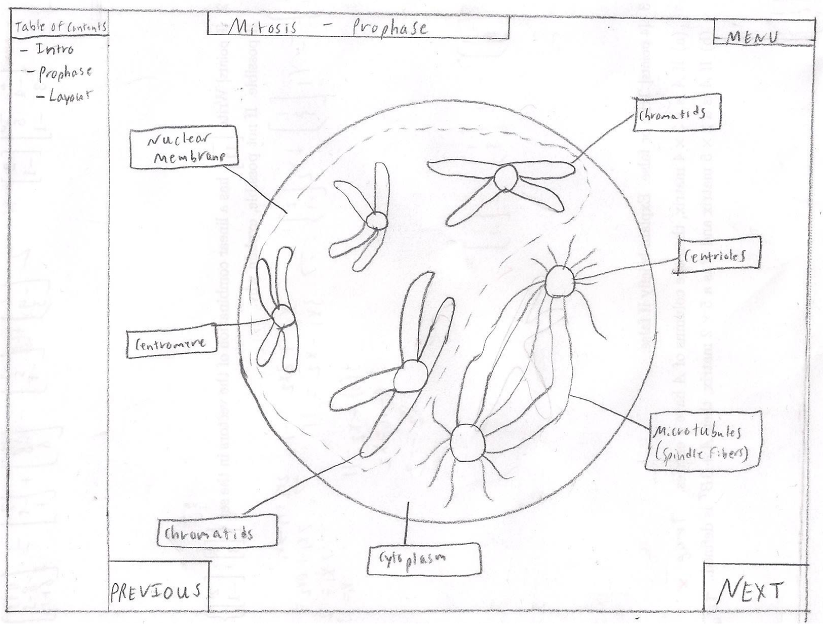


Figure 6: User Interface

# 4 Functional Requirements Specifications

This section contains information about the several use cases, describing different scenarios that a user may go through, stakeholders of this project will be described, a discription of the actors and goals, and some system sequence diagrams on the higher priority use cases.

Although the problem statement specifically stated that this would be a web based project, there was no mention of a database. However, a database is necessary for some of the features of the program, including: Professor has access to student performance(grades), Students are allowed to continue the experiment where they left off. The purpose and the responsibility of the database is to store data, that is accessible to the user, based on these use cases, which determines permissions of each user. First, we discuss the stake holders.

## 4.1 Stakeholders

A stakeholder is described as anyone or any company/organization that may have interest in the project. A list of potential stakeholders, and their reason of interest is listed below:

**High Schools/Universities**
: The reason of interest is at first glance very obvious. It costs a lost of money to build and maintain a lab. With all the costs that a University goes through, it is very obvious that they would choose to opt out of buying expensive equipment handled by inexperienced students, and choose to go with a cheaper Virtual Biology Lab that takes minial maintenance.

**Professors/Teachers**
: Professors and Teachers are extremely busy and have their own agendas. Sometimes they don't want, or even have time, to deal with issues with certain lab experiments. There is so much room for error and it is impossible to fix all the flaws of a physical lab.

**Students**
: Students would have a great interest in this product. The virtual solution will be more intuitive and learning based, so that students spend more time learning instead of sitting and waiting for a professor. Because of this, students may finish labs faster and spend time studying for other classes.

**Lab Technicians**
: Dealing with broken machinery and replacing tools that mischievous students decided to ruin creates a lot of unnecessary work for the lab technicians. Instead, with this new virtual solution, they would be able to simply maintain a system, and install software.

Not all potential users are included in this stakeholders list, as this project may find use outside of its intended purpose. However, these four entities comprise of the main audience that this project is targeting.

## 4.2 Actors and Goals

An actor is defined as any person or device that interacts with the system. An initiating actor can initiate a use case of the system while a participating actor cannot initiate a use case, but will still interact with the system. A list of actors and their goals are listed.

### Initiating Actors

- Professor
  - to manage student users(add/remove) as well as view/edit grades
  - to create laboratory grading policies
  - to view student progress
  - to log in
- Student
  - to perform lab simulation experiment and receive feedback
  - to view grades
  - to track and save his/her progress, and backtrack
  - to log in
- Unregistered User
  - to register for lab course

### Participating Actors

- Feedback system
- Database

The 2 main human users that will actively use this system are the professor and the student. Since the system is reactionary to the users actions, and should not initiate any situations, the Professor and the User will be considered as initiating actors. One more initiating actor is the Unregistered User. These are the users that the system does not recognize, because they never registered into the system. The unregistered user will eventually fall into the other 2 initiating actors, if they intend to access the system.

This system is built around the goals of the initiating actors. For this reason, a lot, if not all of the goals of the students are included as features of the system. An example of this is as follows: The system is built around lessening the load on the Professor. By allowing them to create grading policies, professors no longer have a need to observe each and every lab group to judge performance. The listed goals are a result of the problem statement and analysis.

Participating actors are entities that cannot initiate any use cases, but will still respond and interact to use cases. The database and the feedback system, the 2 participating actors in this system, directly/indirectly interact with the users and make a reaction/response to a certain action. Because they have individual responsibilities that the system rely on, but cannot make an action until a initiating actor makes an action, they are considered participating actors.

## 4.3 Use Cases

Use cases are essential in mapping out the capabilities of any system. The Virtual Biology Lab consists of 9 use cases, and derive from the requirements(see requirements section). One thing to note, use cases are any situation the user can use the system. One thing to note, it can only be considered a use case if an actor initiates a use case through an action. Waiting for an action does not qualify as a use case in itself. The use cases below show the various ways the user can utilize the system. A use case diagram, which really captures the interactions, can also be seen below.

### 4.3.1 Casual Description

**UC1: Navigation Portal**

A user will be able to access grades, the simulation, or pick the grading scheme. Depending on which user is logged in will determine which options will be chosen. For example, if the professor is logged in, the navigation portal will allow him to pick the grading scheme and see the grades of the students that are registered for the lab. This will fulfill the requirements below.
Derived from REQ-1, REQ-5, REQ-8, and REQ-14.

**UC2: Log In**

A user can log in from the main page. The user will input their username and password to which the system will recognize the portal to which it should send the user based on the information provided at registration (mandatory sub use case, <<include>> from UC-1: NavigationPortal).
Derived from REQ-6.

**UC3: Register**

A non-registered user will register with the system, providing user name (usually NetID), password, and type of individual (student or professor) with valid verification.
Derived from REQ-7.

**UC4: ViewGrades**

The students will be able to view their own grades. The professor will be able to view the grades of each student and change them accordingly (optional sub use case, <<extend>>UC-1: NavigationPortal).
Derived from REQ-5 and REQ-8

**UC5: SelectGradingScheme**

The professor will be able to choose how to grade the lab. He will be able to choose how many strikes are allowed, what to do when a student makes an error, etc. (optional sub use case, <<extend>>UC-1: NavigationPortal).
Derived from REQ-4.

**UC6: ManageUsers**

Students that sign up for the course will become alphabetized. The administer of the course (in this case, the professor) will be able to add and remove students from the database so the virtual lab can be reused. This will ensure the flexibility of the lab (optional sub use case, <<extend>>UC-4: ViewGrades).
Derived from REQ-5.

**UC7: GiveFeedback**

The system will respond with messages depending on the action of the user. It will give specific feedback that corresponds to a correct or incorrect answer such as acknowledgement of the correct answer, an explanation, or tips to get to the correct answer (mandatory sub use case, <<include>>from UC-8: Simulation).
Derived from REQ-2 and REQ-10.

**UC8: Simulation**

A user can watch and participate in the mitosis lab simulation. They will watch the animation on the screen, read relative text and interact with the virtual lab when prompted to do so. This interaction between the user and what is happening on-screen is what makes the virtual bio lab functional. (optional sub use case, <<extend>>UC-1: NavigationPortal).
Derived from REQ-1, REQ-2, REQ-9, REQ-12, and REQ-13.

**UC9: ManageHistory**

The system will record the progress and each step that the students make. The students will be able to exit at any time and enter where they left off (mandatory sub use case, <<include>>from UC-8: Simulation).
Derived from REQ-3 and REQ-11.

One thing to keep in mind, is that any user is not directly interacting with another user. However, for the system to use all the use cases, the bare minimum of users is 1 professor and 1 student. The casual description shows what each user is capable of, and what actions can be made while also showing how the system reacts and responds to those actions. Every use case derives from at least 1 requirement.

### 4.3.2 Use Case Diagram

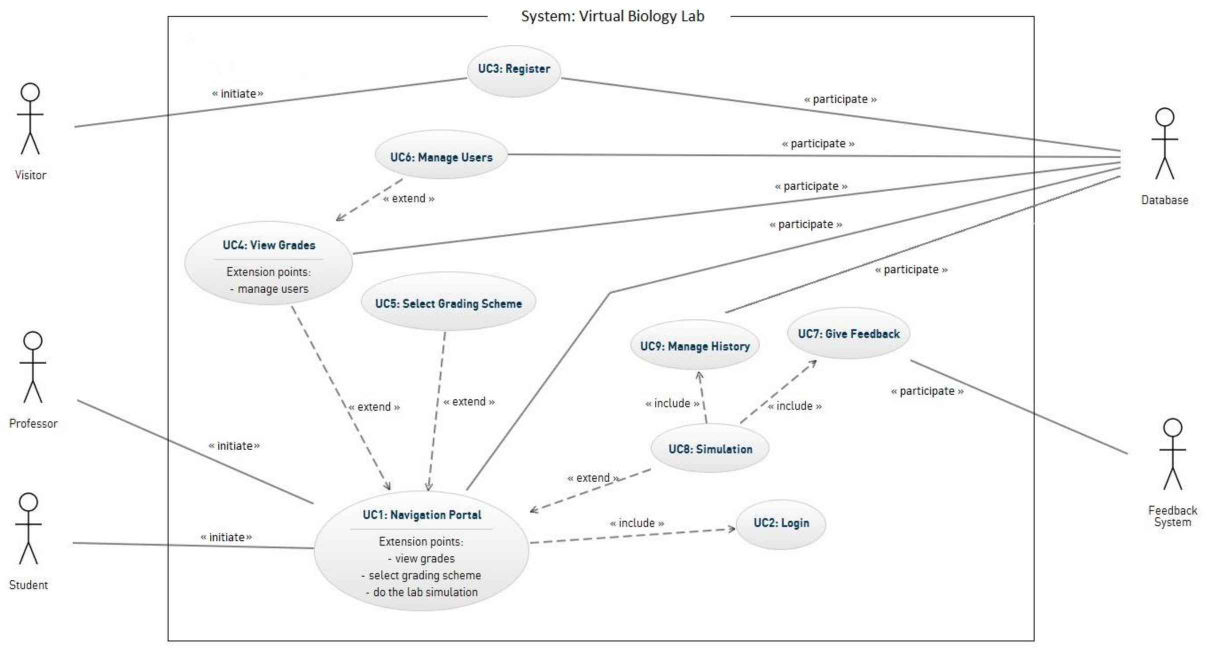


Figure 7: Use Case Diagram of Full System

### 4.3.3 Traceability Matrix

| REQ-# | PW | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 | UC7 | UC8 | UC9 |
|---|---|---|---|---|---|---|---|---|---|---|
| REQ-1 | 5 | X | | | | | | | X | |
| REQ-2 | 4 | | | | | | | X | X | |
| REQ-3 | 4 | | | | | | | | | X |
| REQ-4 | 3 | | | | | X | | | | |
| REQ-5 | 2 | X | | | X | | X | | | |
| REQ-6 | 2 | | X | | | | | | | |
| REQ-7 | 2 | | | X | | | | | | |
| REQ-8 | 1 | X | | | X | | | | | |
| REQ-9 | 5 | | | | | | | | X | |
| REQ-10 | 4 | | | | | | | X | | |
| REQ-11 | 3 | | | | | | | | | X |
| REQ-12 | 2 | | | | | | | | X | |
| REQ-13 | 2 | | | | | | | | X | |
| REQ-14 | 2 | X | | | | | | | | |

Table 7: Traceability Matrix

The traceability matrix shows how the system requirements that we came up map to our use cases. We calculated the priority weights of the use cases, and therefore, we can order our use cases by priority:

UC8 > UC1, UC7 > UC9 > UC4, UC5 > UC2, UC3, UC6

It makes absolute sense that the realistic laboratory simulation and the feedback system are highest priority, as they are our most distinguishable and unique features that the system is based around. The navigation portal is also a high priority use case, because a lot of the use cases rely on the navigation portal, including the login as well as accessing the settings.

### 4.3.4 Fully Dressed Use Case Description

We selected UC8, UC7, UC1 to detail in the fully dressed use case description. The fully-dressed description explain the purpose of the use case, the role of each of the actors involved, what qualifies as a successful as well as a failed end condition, and the flow of events. It also explains how the system reaches each end condition through the flow of events.

| Use Case UC8: | Simulation |
|---|---|
| Related Requirements: | REQ-1, REQ-2, REQ-5, REQ-12, REQ-13 |
| Initiating Actor: | Student |
| Actor's Goal: | To view and interact with a simulation of a mitosis lab |
| Participating Actors: | Professor, Database, Feedback System |
| Preconditions: | Student is a registered user |
| Success End Condition: | The simulation is completed and the grade is recorded and saved in the Database. |
| Failed End Condition: | The simulation is not finished and/or progress is not saved in the Database. |
| Flow of Events for Main Success Scenario: | |
| include::GiveFeedback (UC-7), include::ManageHistory (UC-9) | |
| $->$ | 1. Student accesses the simulation through the navigation portal |
| $<-$ | 2. System displays an introduction to the mitosis lab. |
| $->$ | 3. Student interacts with the simulation by dragging and dropping, typing, or clicking. |
| $<-$ | 4. System gives feedback to the student. |
| $<-$ | 5. System will create a table of contents for each part that is correctly completed by the student. |
| | Step 3, 4, and 5 are repeated until the simulation is over. |
| $<-$ | 6. System gives a grade to the student and the corresponding grade is recorded in the Database. |
| Flow of Events for Extensions (Alternate Scenarios): | |
| | 6a. System does not have a grade to input into the Database. |
| $<-$ | 1. System automatically gives a grade of a 0 and is recorded in the Database. |

Table 8: Fully-Dressed Description, UC8: Simulation

The determined success condition is that the simulation is completed by the student. This is considered to be a success because the student's goal is to interact with the simulation of the lab, and the student is trying to learn more about mitosis and complete the lab for a grade. To complete the lab is to fulfill the goal of the student. If the student were to not finish, the success end condition has not been met, so it is considered a failure, until he completes the simulation. Even worse, if, for some reason the progress didn't save and get stored in the database, it would be as if the student did not perform anything. No grade would be recorded. This would be considered a true failure.

In order to get into the simulation, the student would have to log in, in order for the system to recognize what kind of user is accessing the system. This use case contains this as a precondition. The simulation option is available through the navigation portal, so the student should click the option for the simulation from the portal. The student will then interact with the simulation that the system will provide. As the student progresses through the simulation, the database will store the state of the student and a table of contents of his history will appear(see UI specification section). This continues, while the system is giving feedback and tracking errors. When the simulation ends, the student will be sent to a conclusion page which shows his grade based on his errors.

| Use Case UC7: | Give Feedback |
|---|---|
| Related Requirements: | REQ-2, REQ-10 |
| Initiating Actor: | Student |
| Actor's Goal: | To receive feedback based on given answer |
| Participating Actors: | Feedback System, Database |
| Preconditions: | The student is logged in and starts and interacts with the simulation. |
| Success End Condition: | Feedback is received by the student and will know what to do. |
| Failed End Condition: | No Feedback is given |
| Flow of Events for Main Success Scenario: | |
| include::GiveFeedback (UC-7), include::ManageHistory (UC-9) | |
| $->$ | 1. Student interacts with the simulation |
| $<-$ | 2. System gives feedback messages based on the answer that the student gives. |
| $->$ | 3. Student moves onto the next step. |
| | Step 1, 2, and 3 are repeated until the simulation is over. |
| Flow of Events for Extensions (Alternate Scenarios): | |
| $<-$ | 2a. Student makes an incorrect answer. 1. System will give tips on how to improve the students answer and add 1 to the number of strikes that are allowed. The Database will keep track of how many strikes there are. |
| $<-$ | 2. System will return back to the screen that the student made an error in so that he/she can fix the mistake. |

Table 9: Fully-Dressed Description, UC1: Simulation

The feedback system is arguably the second most important feature of this system, after the simulation. Students want to know how they are performing, while the professor does not want to constantly watch over every group. UC-7 starts off with a student interaction of the feedback system as a precondition. Based on that interaction, the feedback system will then send a message to the user, and send the student to the next step. This process will then repeat until the simulation is over. If no feedback is given, based on the student action, the student cannot tell if they commited an error. This is considered a failure, because the goal was not met.

The alternative scenario of the student committing an error flow of events is as follows: the student makes an error. The system will then send a message giving tips and telling the student that his action was not the correct action to take. The system will then increment the error count(strikes). The system will then return the user to the previous stage, allowing him to try again, this time with helpful tips to allow him to continue.

| | | |
|---|---|---|
| Use Case UC1: | | Navigation Portal |
| Related Requirements: | | REQ-1, REQ-5, REQ-8, REQ-14 |
| Initiating Actor: | | Student, Professor |
| Actor's Goal: | | To choose what they want to view. |
| Participating Actors: | | Database |
| Preconditions: | | Student and Professor are registered users. |
| Success End Condition: | | The user successfully ends up where they wants to go. |
| Failed End Condition: | | The user ends up at the incorrect location. |
| Flow of Events for Main Success Scenario: | | |
| include::Login (UC-2) | | |
| $->$ | 1. The system authenticates whether or not the user is a student or a professor and decide which portal to go to. | |
| $<-$ | 2. The system displays all available options for the user to navigate through. | |
| $->$ | 3. User selects which window to open and view. | |
| $<-$ | 4. The system responds by displaying the selected option. | |
| $<-$ | 5. System will create a table of contents for each part that is correctly completed by the student. | |
| Flow of Events for Extensions (Alternate Scenarios): | | |
| | 3a. The system transfers to the incorrect webpage. | |
| $->$ | 1. The user selects the menu button and selects the navigation portal. | |
| $<-$ | 2. The system displays all available options for the user to navigate through. | |
| $->$ | 3. The user selects which window to open and view. | |
| $<-$ | 4. The system responds by displaying the selected option. | |

Table 10: Fully-Dressed Description, UC1: Navigation Portal

Users will be able to navigate to their desired web page using the navigation portal. A users login credentials will be validated by the system before proceeding to the appropriate navigation portal. The navigation portal will display all of the possible navigation choices to the user after they have successfully logged in. Once a choice is selected the navigation portal will bring the user to the selected location. If the user wishes to change their selection, they will be able to navigate back to the navigation portal and select another location.

### 4.3.5 System Sequence Diagram

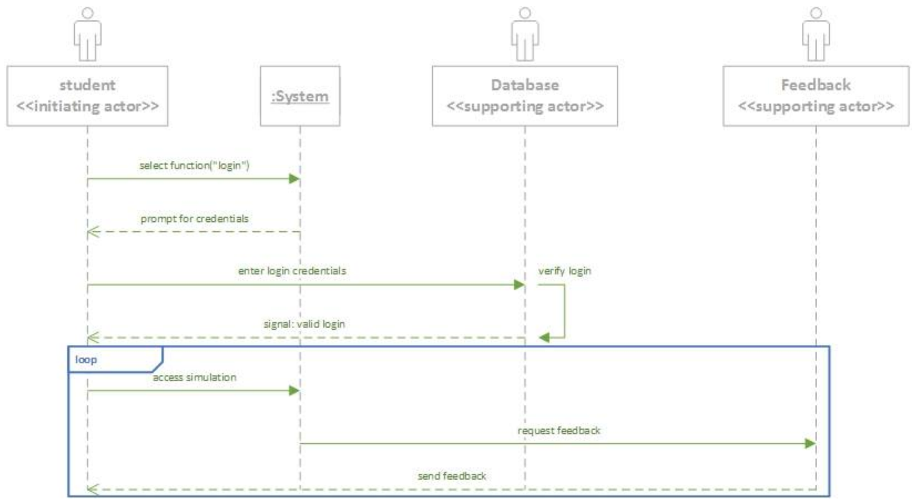


Figure 8: System Sequence Diagram of UC8: Simulation

The professor and the student are two initiating actors who can both communicate with the system in different ways. These two use cases were chosen with the intention of illustrating the capabilities of the entire system in interacting with both different actors. In the first use case, the student is the initiating actor of the sequence diagram. Initially the user calls the login function. This function causes the system to prompt the user for their credentials. Upon entering their login credentials, the input information will be checked against the stored information in the database. The database then confirms the user is a student, and it signals to the user that the credentials are verified. The student will be allowed to access the lab simulation. While performing the lab simulations, the system will periodically communicate to feedback, which is a supporting actor. When the system communicates with feedback, feedback will send information to the student concerning performance on the simulation.

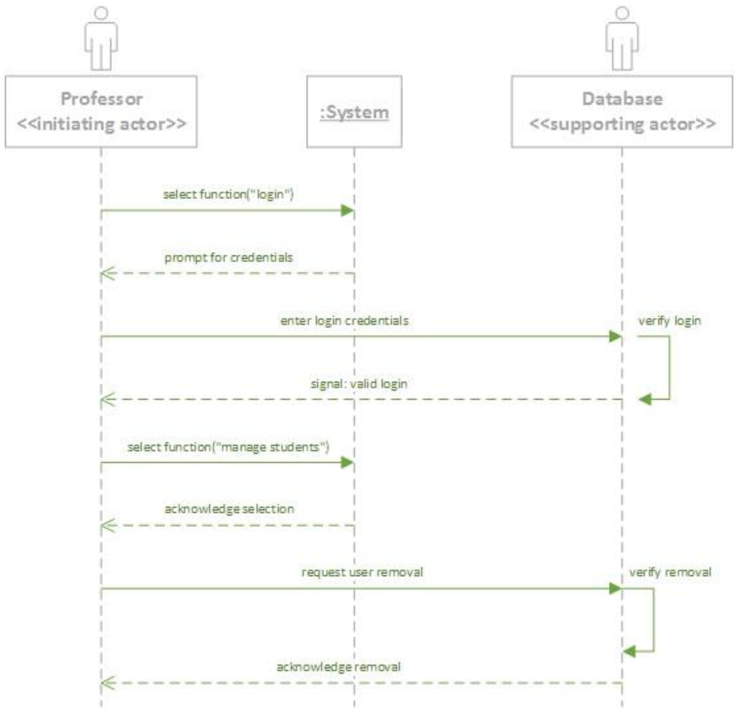
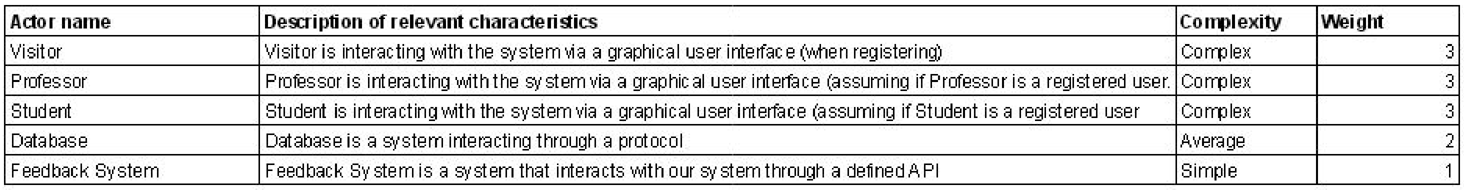


Figure 9: System Sequence Diagram of UC6: Manage Users

The next use case involves the professor communicating with the system. The professor has a similar foundation to its sequence diagram. The login process is identical, where the system prompts the user for their credentials. Upon the professor entering the proper credentials, the information is cross referenced against information stored in the database. Once it is confirmed that the user is a professor, the database will signal to the user that the login was valid. The professor then selects the manage students function from the system. Once the system acknowledges this call, the professor requests the database to remove a user. The database will then recognize the fact that a verifiable user has made this request, and will signal to the user that the removal has been made.

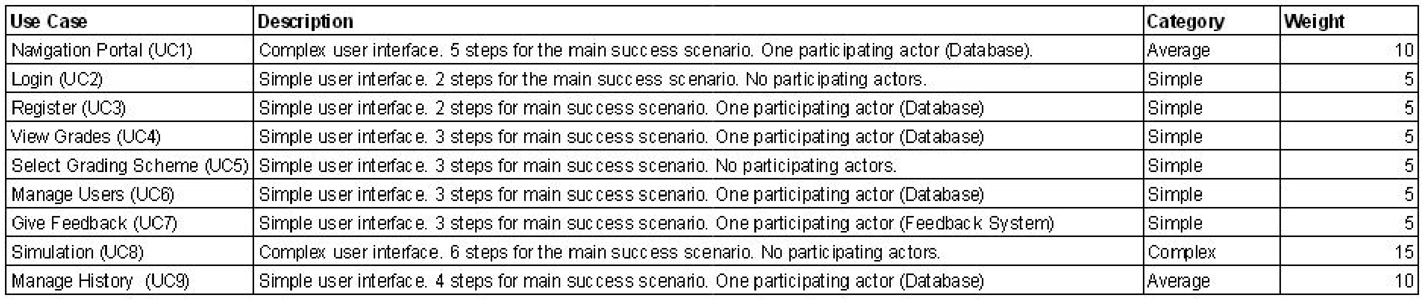
## 5 Effort Estimation Using Use Case Points



| Actor name | Description of relevant characteristics | Complexity | Weight |
|---|---|---|---|
| Visitor | Visitor is interacting with the system via a graphical user interface (when registering) | Complex | 3 |
| Professor | Professor is interacting with the system via a graphical user interface (assuming if Professor is a registered user. | Complex | 3 |
| Student | Student is interacting with the system via a graphical user interface (assuming if Student is a registered user | Complex | 3 |
| Database | Database is a system interacting through a protocol | Average | 2 |
| Feedback System | Feedback System is a system that interacts with our system through a defined API | Simple | 1 |

Figure 10: Actor classification for the Virtual Biology Lab

The Unadjusted Actor Weight was calculated to be:
UAW = 1 x Simple + 1 x Average + 3 x Complex = 1 x 1 + 1 x 2 + 3 x 3 = 12

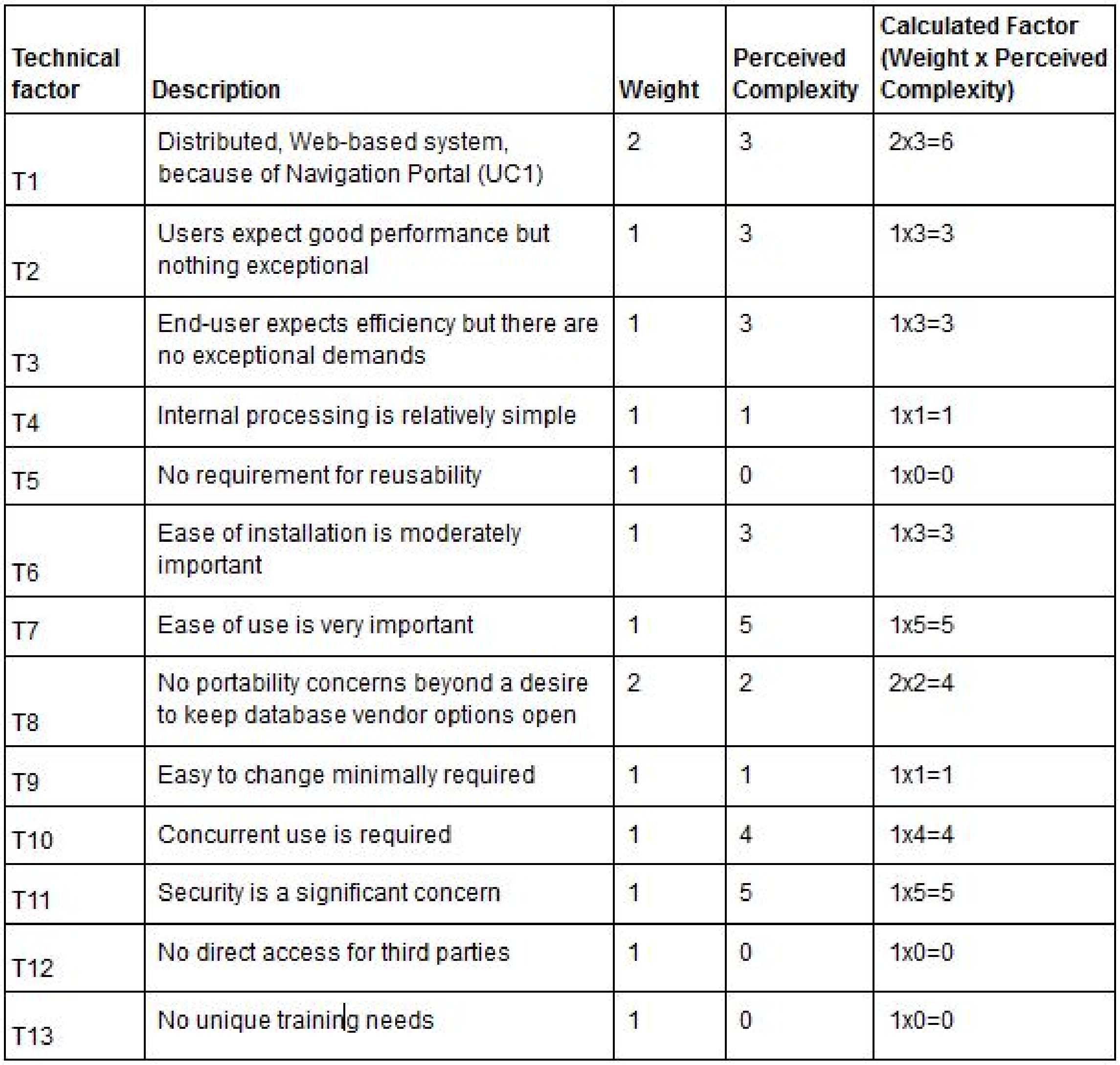


| Use Case | Description | Category | Weight |
|---|---|---|---|
| Navigation Portal (UC1) | Complex user interface. 5 steps for the main success scenario. One participating actor (Database). | Average | 10 |
| Login (UC2) | Simple user interface. 2 steps for the main success scenario. No participating actors. | Simple | 5 |
| Register (UC3) | Simple user interface. 2 steps for main success scenario. One participating actor (Database) | Simple | 5 |
| View Grades (UC4) | Simple user interface. 3 steps for main success scenario. One participating actor (Database) | Simple | 5 |
| Select Grading Scheme (UC5) | Simple user interface. 3 steps for main success scenario. No participating actors. | Simple | 5 |
| Manage Users (UC6) | Simple user interface. 3 steps for main success scenario. One participating actor (Database) | Simple | 5 |
| Give Feedback (UC7) | Simple user interface. 3 steps for main success scenario. One participating actor (Feedback System) | Simple | 5 |
| Simulation (UC8) | Complex user interface. 6 steps for the main success scenario. No participating actors. | Complex | 15 |
| Manage History (UC9) | Simple user interface. 4 steps for main success scenario. One participating actor (Database) | Average | 10 |

Figure 11: Use case classification for the Virtual Biology Lab

The Unadjusted User Case Weight was calculated to be:
UUCW = 6 x Simple + 2 x Average + 1 x Complex = 6 x 5 + 2 x 10 + 1 x 15 = 65

With these two values, the Unadjusted Use Case Points were able to be calculated using the formula below:
UUCP = UAW + UUCW = 12 + 65 = 77

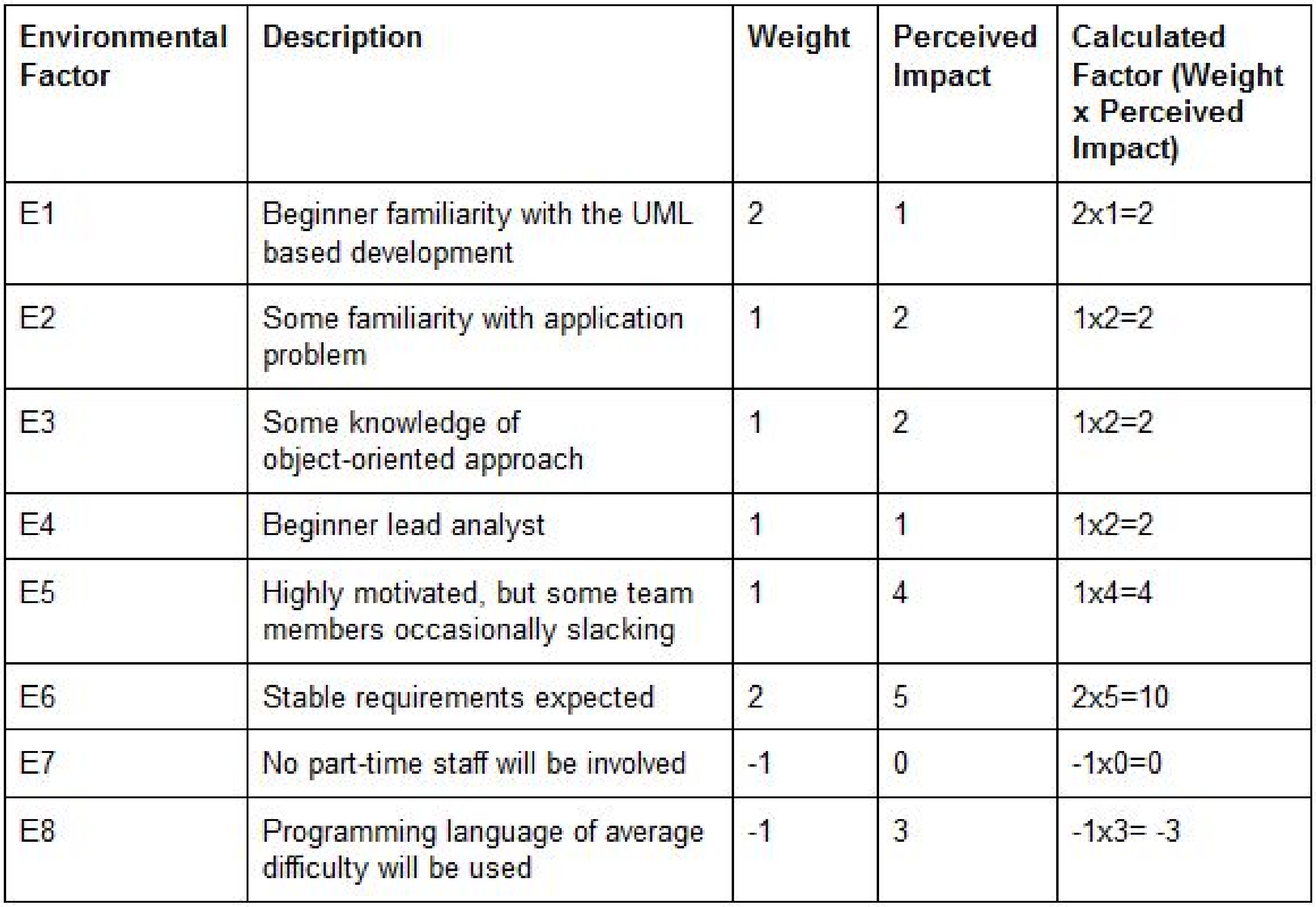


| Technical factor | Description | Weight | Perceived Complexity | Calculated Factor (Weight x Perceived Complexity) |
|---|---|---|---|---|
| T1 | Distributed, Web-based system, because of Navigation Portal (UC1) | 2 | 3 | 2x3=6 |
| T2 | Users expect good performance but nothing exceptional | 1 | 3 | 1x3=3 |
| T3 | End-user expects efficiency but there are no exceptional demands | 1 | 3 | 1x3=3 |
| T4 | Internal processing is relatively simple | 1 | 1 | 1x1=1 |
| T5 | No requirement for reusability | 1 | 0 | 1x0=0 |
| T6 | Ease of installation is moderately important | 1 | 3 | 1x3=3 |
| T7 | Ease of use is very important | 1 | 5 | 1x5=5 |
| T8 | No portability concerns beyond a desire to keep database vendor options open | 2 | 2 | 2x2=4 |
| T9 | Easy to change minimally required | 1 | 1 | 1x1=1 |
| T10 | Concurrent use is required | 1 | 4 | 1x4=4 |
| T11 | Security is a significant concern | 1 | 5 | 1x5=5 |
| T12 | No direct access for third parties | 1 | 0 | 1x0=0 |
| T13 | No unique training needs | 1 | 0 | 1x0=0 |

Figure 12: Technical Complexity Factors for the Virtual Biology Lab

The Technical Complexity Factor can be calculated using the equation:
TCF = Constant-1 + Constant-2 x Technical Factor Total
where Constant-1 = 0.6 and Constant-2 = 0.01

The Technical Factor Total is the sum of the Calculated Factors, which is 35.
Therefore, the TCF turns out to be:
TCF = 0.6 + 0.01 x 35 = 0.95
This results in a reduction of the UCP by 5 percent.



| Environmental Factor | Description | Weight | Perceived Impact | Calculated Factor (Weight x Perceived Impact) |
|---|---|---|---|---|
| E1 | Beginner familiarity with the UML based development | 2 | 1 | 2x1=2 |
| E2 | Some familiarity with application problem | 1 | 2 | 1x2=2 |
| E3 | Some knowledge of object-oriented approach | 1 | 2 | 1x2=2 |
| E4 | Beginner lead analyst | 1 | 1 | 1x2=2 |
| E5 | Highly motivated, but some team members occasionally slacking | 1 | 4 | 1x4=4 |
| E6 | Stable requirements expected | 2 | 5 | 2x5=10 |
| E7 | No part-time staff will be involved | -1 | 0 | -1x0=0 |
| E8 | Programming language of average difficulty will be used | -1 | 3 | -1x3= -3 |

Figure 13: Environmental Complexity Factors for the Virtual Biology Lab

The Environmental Complexity Factor can be calculated using the equation:
ECF = Constant-1 + Constant-2 x Environmental Factor Total
where Constant-1 = 1.4 and Constant-2 = -0.03

The Environmental Factor Total is the sum of the Calculated Factors, which is 19.
Therefore, the ECF turns out to be:
ECF = 1.4 - 0.03 * 19 = 0.83
This results in a reduction of ECF by 17 percent.

Now, the UCP can be calculated by using the equation:
UCP = UUCP x TCF x ECPF where UUCP = 77, TCF = 0.95, and ECF = 0.83
The final UCP = 78.78

# 6 Domain Analysis

## 6.1 Domain Model

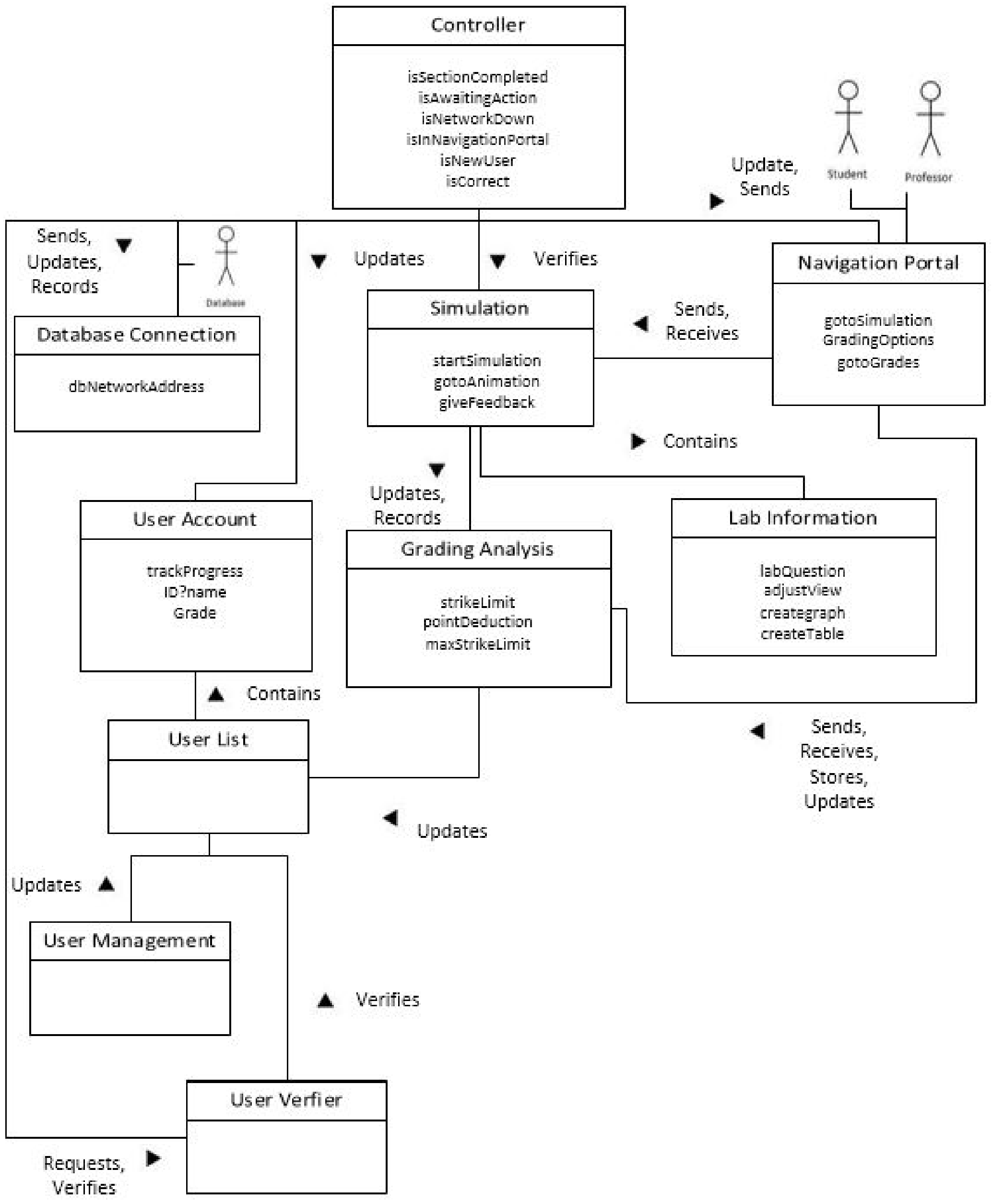
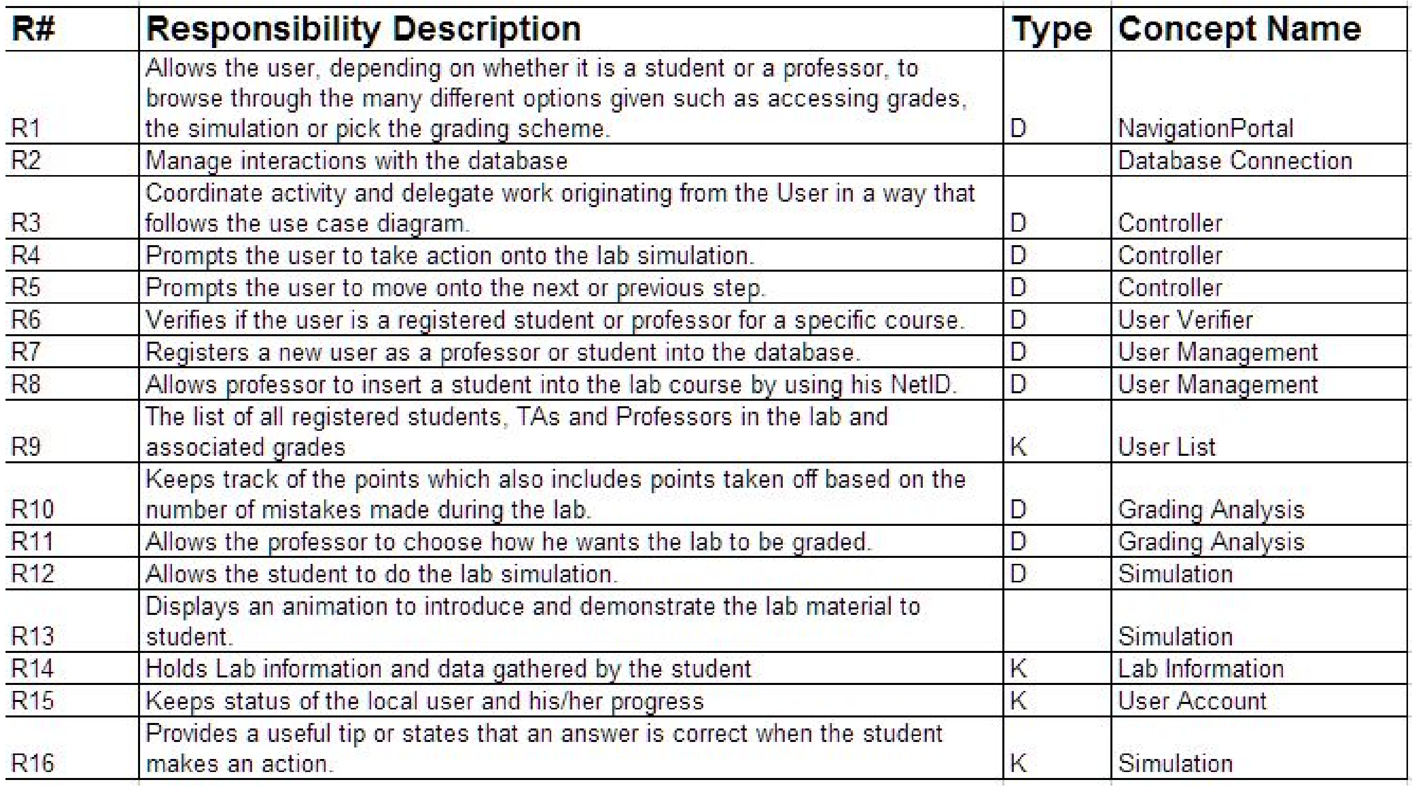


Figure 14: UML Domain Model

### 6.1.1 Concept Definitions



| R# | Responsibility Description | Type | Concept Name |
|---|---|---|---|
| R1 | Allows the user, depending on whether it is a student or a professor, to browse through the many different options given such as accessing grades, the simulation or pick the grading scheme. | D | NavigationPortal |
| R2 | Manage interactions with the database | | Database Connection |
| R3 | Coordinate activity and delegate work originating from the User in a way that follows the use case diagram. | D | Controller |
| R4 | Prompts the user to take action onto the lab simulation. | D | Controller |
| R5 | Prompts the user to move onto the next or previous step. | D | Controller |
| R6 | Verifies if the user is a registered student or professor for a specific course. | D | User Verifier |
| R7 | Registers a new user as a professor or student into the database. | D | User Management |
| R8 | Allows professor to insert a student into the lab course by using his NetID. | D | User Management |
| R9 | The list of all registered students, TAs and Professors in the lab and associated grades | K | User List |
| R10 | Keeps track of the points which also includes points taken off based on the number of mistakes made during the lab. | D | Grading Analysis |
| R11 | Allows the professor to choose how he wants the lab to be graded. | D | Grading Analysis |
| R12 | Allows the student to do the lab simulation. | D | Simulation |
| R13 | Displays an animation to introduce and demonstrate the lab material to student. | | Simulation |
| R14 | Holds Lab information and data gathered by the student | K | Lab Information |
| R15 | Keeps status of the local user and his/her progress | K | User Account |
| R16 | Provides a useful tip or states that an answer is correct when the student makes an action. | K | Simulation |

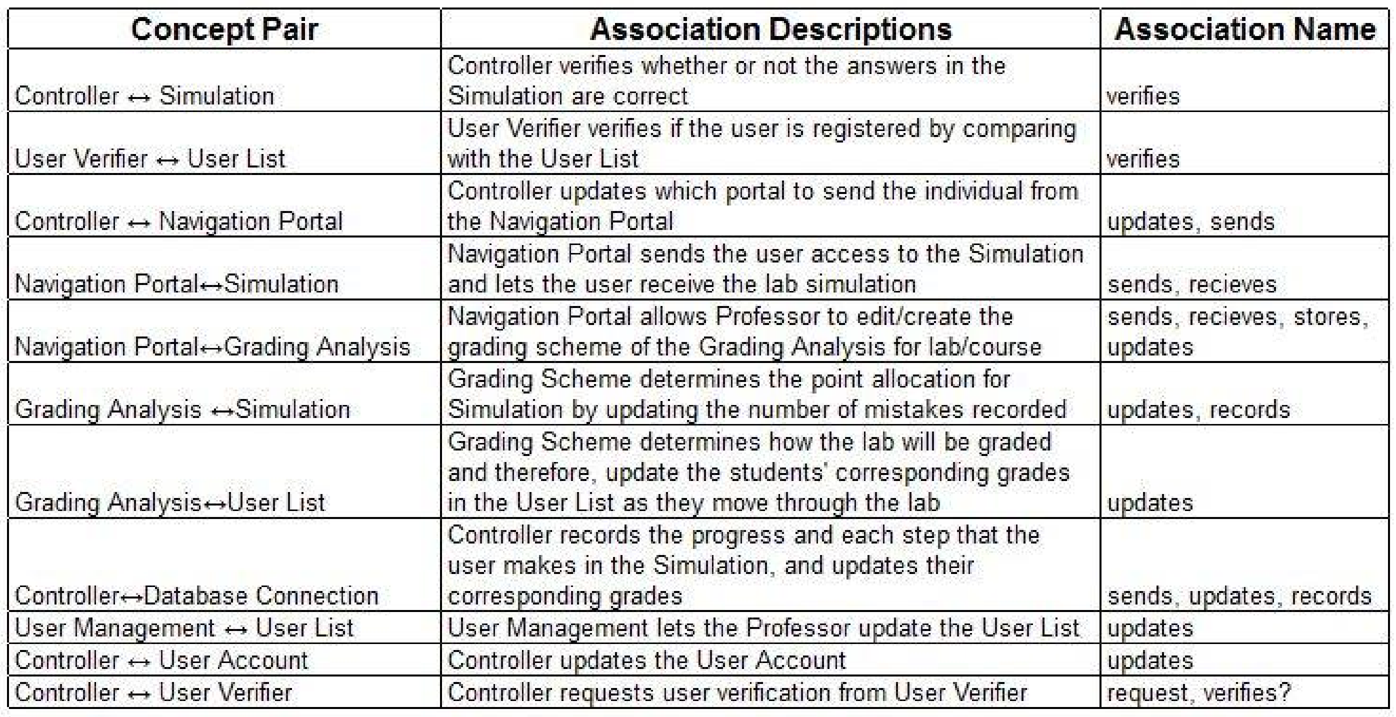
Figure 15: Concept Definitions

The concept definitions can be derived from the different responsibilities, which came from the use cases of the system. Our concepts include: Navigation Portal, Database Connection, Controller, User Verifier, User Management, User List, Grading Analysis, Simulation, Lab Information, and User Account. The responsibilities of each concept is listed above. The Navigation Portal is responsible for allowing the user to access the different options, such as getting to the grades page or to go to the simulation.

The Database connection is responsible for getting the data from other concepts into the database. The responsibility table should reflect that. The controller helps the user interact with the system, acting somewhat like a game pad in the fact that the user inputs something to the controller which will then interact with the inner system, and then the controller will tell the user to make an action prompt. User verifier has the sole task of checking whether the user is a professor or a student.

User management concept allows the system to give users(Professors) to change the User list, to add or remove or edit, and the user list is a list of all the students and their associated grades as well as the professors. The grading analysis is responsible for the grading scheme as well as keeping track of the grades and updating the user list grades. The simulation concept is responsible for the access of the actual simulation as well as the feedback messages. The User account holds each users information(Name, Type of user, grade), and the User list is comprised of all these User Accounts. The lab information holds the information on the experiment and sends data to the simulation.

### 6.1.2 Association Definition

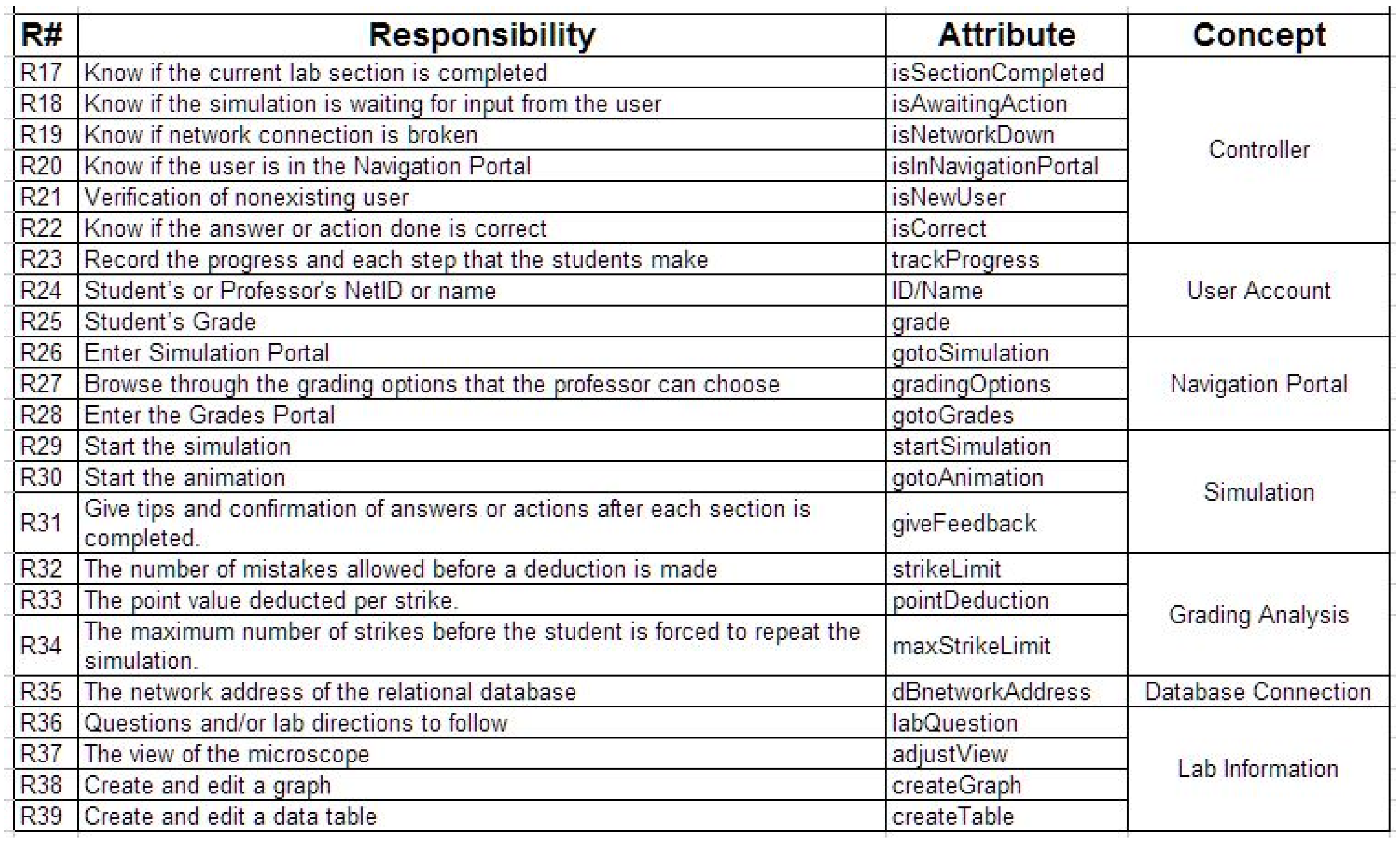


| Concept Pair | Association Descriptions | Association Name |
|---|---|---|
| Controller ↔ Simulation | Controller verifies whether or not the answers in the Simulation are correct | verifies |
| User Verifier ↔ User List | User Verifier verifies if the user is registered by comparing with the User List | verifies |
| Controller ↔ Navigation Portal | Controller updates which portal to send the individual from the Navigation Portal | updates, sends |
| Navigation Portal↔Simulation | Navigation Portal sends the user access to the Simulation and lets the user receive the lab simulation | sends, recieves |
| Navigation Portal↔Grading Analysis | Navigation Portal allows Professor to edit/create the grading scheme of the Grading Analysis for lab/course | sends, recieves, stores, updates |
| Grading Analysis ↔Simulation | Grading Scheme determines the point allocation for Simulation by updating the number of mistakes recorded | updates, records |
| Grading Analysis↔User List | Grading Scheme determines how the lab will be graded and therefore, update the students' corresponding grades in the User List as they move through the lab | updates |
| Controller↔Database Connection | Controller records the progress and each step that the user makes in the Simulation, and updates their corresponding grades | sends, updates, records |
| User Management ↔ User List | User Management lets the Professor update the User List | updates |
| Controller ↔ User Account | Controller updates the User Account | updates |
| Controller ↔ User Verifier | Controller requests user verification from User Verifier | request, verifies? |

Figure 16: Association Description

When a student submits an answer during the simulation the simulation will communicate with the controller to verify the correctness of the students answer. The controller will record a students progress and relay the information to database to store the information.The controller will also communicate with the user verifier to facilitate user verification. User account will communicate with the controller to update the user accounts. The user verifier and the user list will communicate with each other make sure all the users are registered correctly. Users navigate to other pages using the navigation portal. When a user selects a destination, the navigation portal will communicate with the controller to send the user to their destination. The navigation portal will communicate with the simulation to allow students to access the simulation when they request it. The navigation portal will also communicate with the grading analysis to allow the professor edit the grading scheme for the lab. The grading analysis will communicate with the simulation to determine the point allocation for the lab. Grading analysis will need to communicate with the user list in order to update the students grades. User management will communicate with user list to give the professor the ability to update the user list.

### 6.1.3 Attribute Definitions



| R# | Responsibility | Attribute | Concept |
|---|---|---|---|
| R17 | Know if the current lab section is completed | isSectionCompleted | Controller |
| R18 | Know if the simulation is waiting for input from the user | isAwaitingAction | |
| R19 | Know if network connection is broken | isNetworkDown | |
| R20 | Know if the user is in the Navigation Portal | isInNavigationPortal | |
| R21 | Verification of nonexisting user | isNewUser | |
| R22 | Know if the answer or action done is correct | isCorrect | |
| R23 | Record the progress and each step that the students make | trackProgress | User Account |
| R24 | Student's or Professor's NetID or name | ID/Name | |
| R25 | Student's Grade | grade | |
| R26 | Enter Simulation Portal | gotoSimulation | Navigation Portal |
| R27 | Browse through the grading options that the professor can choose | gradingOptions | |
| R28 | Enter the Grades Portal | gotoGrades | |
| R29 | Start the simulation | startSimulation | Simulation |
| R30 | Start the animation | gotoAnimation | |
| R31 | Give tips and confirmation of answers or actions after each section is completed. | giveFeedback | |
| R32 | The number of mistakes allowed before a deduction is made | strikeLimit | Grading Analysis |
| R33 | The point value deducted per strike. | pointDeduction | |
| R34 | The maximum number of strikes before the student is forced to repeat the simulation. | maxStrikeLimit | |
| R35 | The network address of the relational database | dBnetworkAddress | Database Connection |
| R36 | Questions and/or lab directions to follow | labQuestion | Lab Information |
| R37 | The view of the microscope | adjustView | |
| R38 | Create and edit a graph | createGraph | |
| R39 | Create and edit a data table | createTable | |

Figure 17: Attribute Description

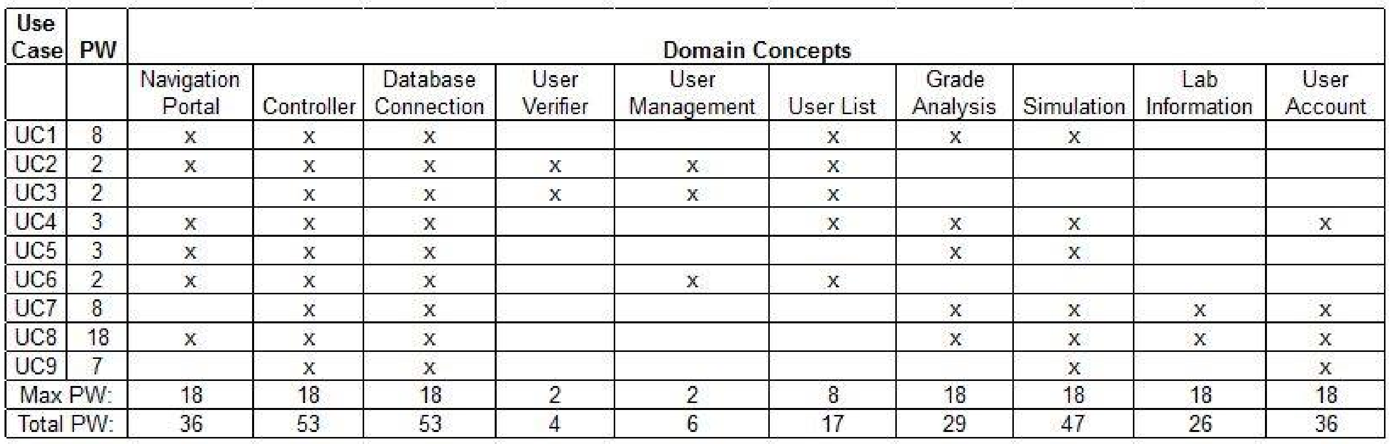
The isSectionCompleted attribute verifies if the section of the lab simulation is completed. The isAwaitingAction attribute lets the system know it is waiting for information from the user. In order to verify that the network connection is available, isNetworkDown checks the connection. IsinNavigation portal checks if the system is in the navigation portal. When a new user is registering, isNewUser authenticates that the user is not already in the user database. When a student enters an answer, isCorrect checks if the inputted answer by the student is correct.

The system allows for the student to begin the lab where they left off because of the trackProgress attribute. The user name for the student or professor is required for the database to verify a registered user, the attribute for the name of the user is ID/name. The grade the student receives in the lab is stored in the grade database, the attribute for the grade is grade. The attribute which allows the student to enter the lab simulation is goToSimulation. The professor can choose criteria on how wrong answers will affect students grades on the simulation in the attribute gradingOptions. The attribute goToGrades allows the user to access their individual grades, while allowing the professor to see the grades of all the registered students. The attribute which prompts the system to start the simulation is startSimulation. An opening animation will perform for the student illustrating important concepts of the upcoming lab in the form of the attribute startAnimation.

The attribute feedback will give conceptual to the student at important junctures of the simulation. This is the amount of strikes a student can receive before having points deducted from their grade for the simulation is adjustable by the professor, and it is in the form of the attribute strikeLimit. The attribute pointDeduction withholds points from the students final grade every time the student does a wrong action on the lab simulation or gets a question wrong. In order to insure that the student is properly learning the required information from the simulation, there is a limit of how many questions they can get wrong before having to restart the whole simulation. This limit can be set by the professor in the form of the attribute maxStrikeLimit.

When information regarding grades or users is needed from the database this address is needed to connect the system to the database properly. The attribute dbNetworkAccess will connect the database to the system. Throughout the simulation the student will be asked questions to verify that they have learned the required information which will be in the form of the attribute labquestion. The attribute adjustView will allow the student to zoom in or out on the microscope to properly view the mitosis process in a sample that cannot be seen by the naked eye. The attributes createGraph and createTable will allow students to create or edit graphs and tables illustrating the lab simulation.

### 6.1.4 Traceability Matrix



| Use Case | PW | Domain Concepts | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Navigation Portal | Controller | Database Connection | User Verifier | User Management | User List | Grade Analysis | Simulation | Lab Information | User Account |
| UC1 | 8 | x | x | x | | | x | x | x | | |
| UC2 | 2 | x | x | x | x | x | x | | | | |
| UC3 | 2 | | x | x | x | x | x | | | | |
| UC4 | 3 | x | x | x | | | x | x | x | | x |
| UC5 | 3 | x | x | x | | | | x | x | | |
| UC6 | 2 | x | x | x | | x | x | | | | |
| UC7 | 8 | | x | x | | | | x | x | x | x |
| UC8 | 18 | x | x | x | | | | x | x | x | x |
| UC9 | 7 | | x | x | | | | | x | | x |
| Max PW: | | 18 | 18 | 18 | 2 | 2 | 8 | 18 | 18 | 18 | 18 |
| Total PW: | | 36 | 53 | 53 | 4 | 6 | 17 | 29 | 47 | 26 | 36 |

Figure 18: Tracebility Matrix(concepts vs use case)

## 6.2 System Operation Contracts

System operation contracts for the Fully-Dressed Use Cases are seen below, as they pertain to the format here:

- Name: appropriate name
- Responsibilities: perform a function
- Post Conditions
- Pre-Conditions

### UC8: Simulation

- Name: simulation
- Responsibilities:
  - Start the simulation as well as the introduction animation
  - Take input from user interacting with simulation
  - Send useful feedback
  - Store history in the form of the table of contents(see UI specifications)
- Preconditions:
  - Logged in and registered user
  - Went through the Navigation Portal to access simulation
- Postconditions:
  - Introduction animation was started
  - Student was able to perform lab experiment

### UC7: Give Feedback

- Name: giveFeedback
- Responsibilities:
  - Analyze student action
  - Send message based off student action
  - If error, increment error count
  - If error count reaches threshold, take appropriate action
  - Return to previous screen
- Preconditions:
  - Logged in and registered user

  - Went through the Navigation Portal to access simulation
  - Student reached point in simulation to prompt giveFeedback

- Postconditions:
  - Helpful feedback was given to the student.

## UC1: Navigation Portal

- Name: navPortal
- Responsibilities:
  - Allow the user to login
  - Show a menu/start page showing the options available to that user
  - Allow the user to access those options
- Preconditions:
  - Registered user
- Postconditions:
  - Student was able to access their previous grades
  - Professor was able to access User List
  - Student was able to access the simulation
  - Professor was able to get to the Grading Analysis

The operation contract shows the capabilities of each of system in each of the different use cases, as well as what is expected of the system. The pre and post-conditions explain the state of the system before and after the use case scenario, and the responsibility shows the functions necessary to reach those conditions.

# 7 Interaction Diagrams

## 7.1 UC8: Simulation

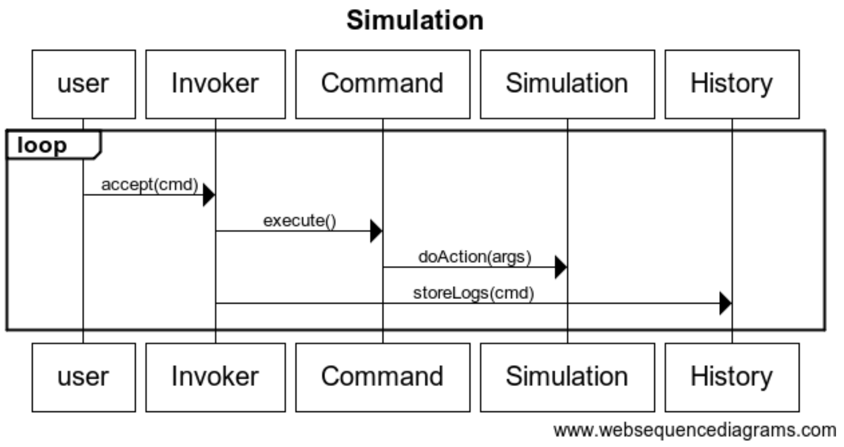


Figure 19: Sequence Diagram of UC7

After the User chooses to perform the selected lab from the Navigation Portal. The lab simulation begins to perform. The first thing that happens is the user watches a preliminary animation that illustrates facets about the mitosis lab. After the animation is complete, the user is able to begin executing the virtual lab. Initially, the animal or plant cell must be placed on the slide. Once this has been completed, the student is supposed to put dye on the cell, so the process of mitosis can be more clearly visible when looking through the microscope. After the dye has been placed, the user places the sample onto the microscope, so it can be viewed. The user can adjust the microscope accordingly, making the sample perfectly viewable. Throughout the process, the system will ask the user questions, some of which may require the user to construct a table or a graph. Feedback is given to the student after they have answered, and if the student answered a question wrong or performed part of the experiment wrong, the strike limit counter will be incremented. If the strike limit counter reaches the strike limit set by the teacher, the user will be forced to restart the whole simulation.

This Use Case was designed using the Command Design Pattern. The reason for using this pattern was that the commands of the User unto the Simulation object were complex, so to ease the calls of the user onto the simulation, an invoker object was created to handle the calls. An additional reason, was that the ability to undo and redo as well as keep log history is inline with the goal of this use case.

## 7.2 UC7: Feedback

### 7.2.1 Responsibility Description

Send input to controller to validate answer.
Send validation answer to feedback system
Send feedback back to controller
Send feedback message to simulation

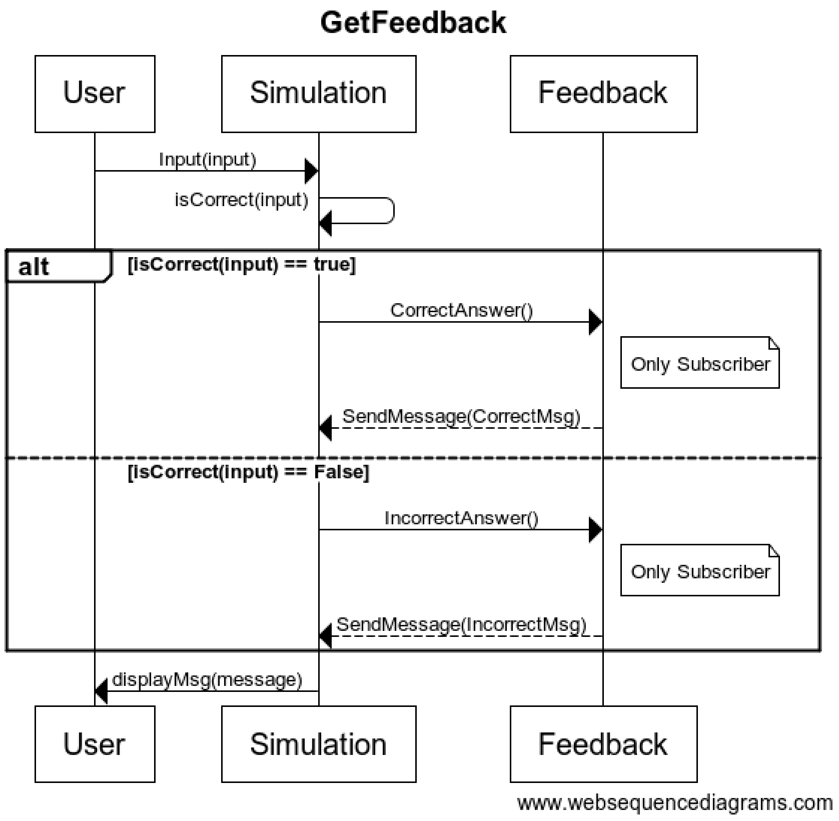


Figure 20: Sequence Diagram of UC7

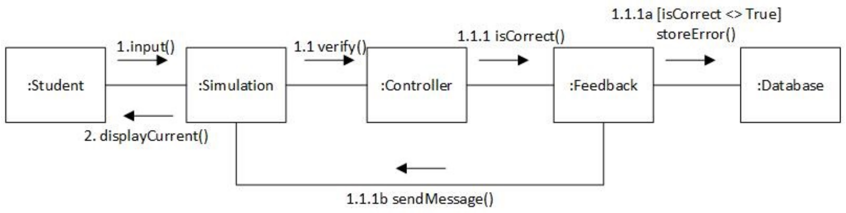


Figure 21: Communication Diagram of UC7

Display feedback message to user from simulation

The Feedback Use case starts with the user(student) already accessing the interactive simulation and currently committing actions. A successful end case begins with the user committing an action and the controller verifying that it is a legal action. The controller will then send the input response to the feedback. The feedback will send the data containing the corresponding correct message back to the simulation object. The alternate scenario, for when the user inputs an incorrect message, the same process goes into effect, verification, sending the "incorrect" message to the simulation. However, the feedback will now save the error in the database, for grade calculation purposes.

This use case uses the Publisher-Subscriber pattern. The pub-sub pattern is extremely useful for indirect communication. The User does not directly speak to the Feedback object, but to the Simulation object. For this purpose the Feedback object will subscribe to the Simulation, with 2 different receive methods, one for a correct answer, and one for an incorrect answer.

## 7.3 UC1: Navigation Portal

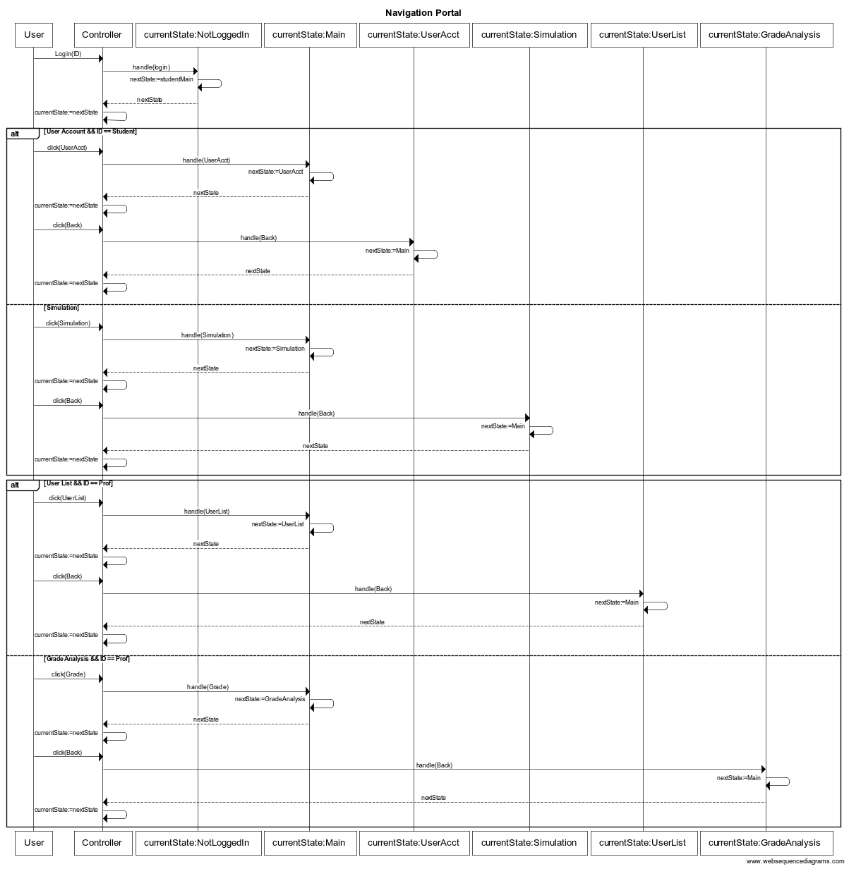


Figure 22: Sequence Diagram of UC1

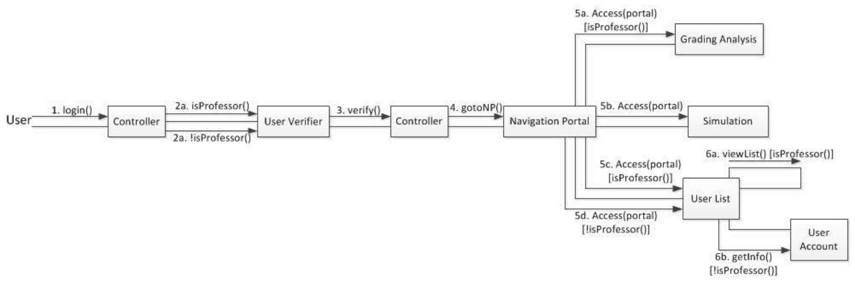


Figure 23: Communication Diagram of UC1

### 7.3.1 Responsibility Description

Send message to Controller to validate whether the user is a student or a professor.

Send validation answer to User Verifier.

Send message to Controller to bring the user to the appropriate Navigation Portal depending on the type of user.

Send message to Simulation to bring the user to the lab simulation if chosen.

Send message to the Grade Analysis to bring the user to choose the grading scheme if chosen and if the user is a registered professor.

Send message to User List to bring the user to the user list if chosen and if the user is a registered professor

Send message to User List to bring the user to his own account if chosen and if the user is a registered student.

The purpose of the Navigation Portal is to serve as a central node to access the different parts of the application. While the diagrams above may seem somewhat convoluted, it is essentially just the mapping of actions and interactions just to navigate to one section to another, while gathering the appropriate data needed. It's responsibilities include bringing the user to the desired state as well as getting the data needed for that particular section of the application(I.E. student grades for Grading Analysis).

The Navigation Portal will be designed using the State design pattern. The thought behind the State design patterns is that the system will behave differently depending on what the state of the system is. Since there are multiple states that the Navigation Portal can have, and the actions on those states are different, the state design pattern was used.

# 8 Class Diagram and Interface Specification

## 8.1 Class Diagram

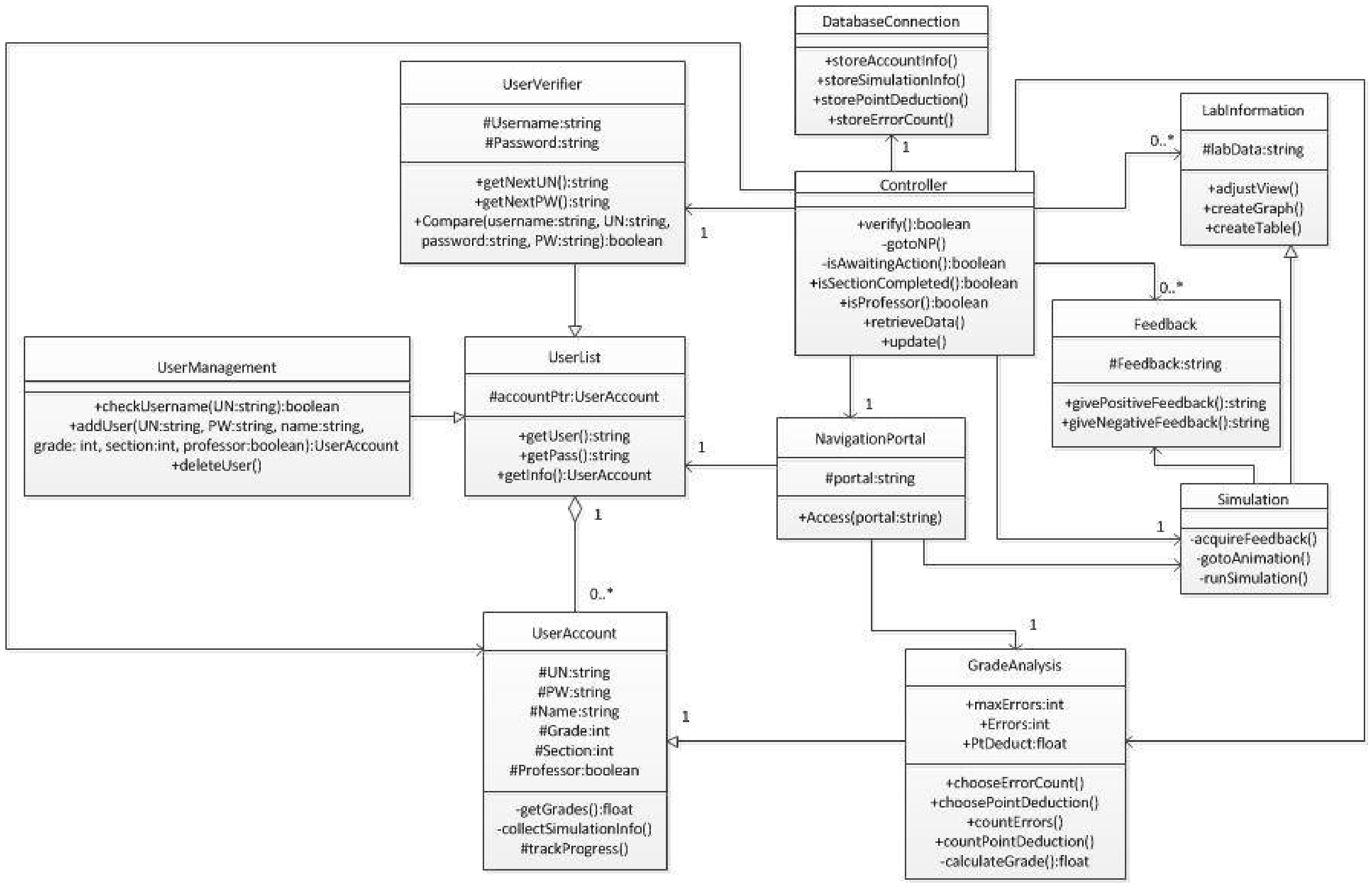


Figure 24: Class Diagram

## 8.2 Interface Specification

This next section may seem overloaded with a lot of definitions, but these are just the specification of the operation contracts of all of the different classes of the system. This list consists of the Data Members as well as the Member Functions of each class. As this is just the list, the explanation will be in the following section.

- Controller
  - verify()
  - gotoNP()
  - isAwaitingAction()
  - isProfessor()
  - retrieveData()
  - update()
- NavigationPotal
  - access(portal)
    - portal = simulation, gradeAnalysis, or userList
- UserVerifier
  - Data Members
    - username
    - password
  - Member Functions
    - getNextUN()
    - getNextPW()
    - compare(username, UN, password, PW)
- UserList
  - Data Members
    - accountPtr
  - Functions
    - getUser()
    - getPass()
    - getInfo()
- UserAccount
  - Data Members
    - UN
    - PW

    - name
    - grade
    - section
    - professor
  - Member Functions
    - getGrades()
    - collectSimulationInfo()
    - trackProgress()
- UserManagement
  - Data Members
    - UserList
  - Member Functions
    - checkUserName()
      - if(used) − > disp(Error Message)
      - if(unused) − > disp(Usuable Username Message)
- Grade Analysis
  - Data Members
    - maxErrors
    - Errors
    - PtDeduct
  - Member Functions
    - chooseErrorCount()
    - choosePointDeduction()
    - chooseRepeat()
    - countErrors()
    - countPointDeduction()
- Simulation
  - Member Functions
    - gotoAnimation()
    - runSimulation()
- DatabaseConnection
  - Member Functions
    - storeAccountInfo()
    - storeSimulationInfo()
    - storePointDeduction()

    - storeErrorCount()

- Feedback
  - Data Members
    - Feedback
  - Member Functions
    - givePositiveFeedback()
    - giveNegativeFeedback()
- LabInformation
  - Data Members
    - labData
  - Member Functions
    - adjustView()
    - createGraph()
    - createTable()

## 8.3 Data Types and Operation Signatures

This section serves the purpose of fully explaining the system's classes, going indepth of the class description as well as a brief meaning of each operation and attribute in plain language.

- Controller - The controller class will contain all of the function the controller will utilize.
  - verify() - Verify if input is valid
  - gotoNP() - Will bring the user to the Navigation Portal
  - isAwaitingAction() - Keeps controller ready to receive user input
  - isSectionCompleted() - Checks to see if the user has completed the current section of the simulation
  - isProfessor() - Checks to see if current user is a professor.
  - retrieveData() - Returns data from ***
  - update() - ****
- NavigationPortal -
  - access(portal) - Brings the user to the selected portal. Where the portal can be the Simulation, Grade Analysis, or User List
- User Verifier
  - username - Stores entered username
  - password - Stores entered password
  - getNextUN() - Returns username
  - getNextPW() - Returns password
  - compare(username, UN, password, PW) - Compares login credentials entered by user to the credentials stored in the database.
- User List
  - accountPtr - Points to the users account
  - getUser() - Returns users username
  - getPass() - Returns users password
  - getInfo() - Returns users account information
- User Account
  - username - Stores users username
  - password - Stores users password
  - name - Stores users name
  - grade - Stores users grade
  - section - Stores users section number

  - professor - Stores the name of the users professor
  - getGrades() - Returns the users grades
  - collectSimulationInfo() - *
  - trackProgress() - Tracks the progress the user has made

- User Management
  - userList - Contains a list of all of the users
  - checkUsername() - Check if requested username is available or not

- GradeAnalysis
  - chooseErrorCount() allows the professor to set a limit on how many errors the student is allowed to make before the student is forced to restart the laboratory assignment
  - choosePointDeduction() allows the professor to choose how many points are deducted upon each mistake made by the student; mistakes are caused by wrong answer choices for the lab questions or by performing the lab simulation wrong
  - chooseRepeat() allows the professor to choose how many mistakes are allowed to be made before the student is forced to restart the lab simulation
  - countErrors() counts how many errors are made by the student while performing the lab simulation
  - countPointDeduction() counts the amount of points taken off the final grade for the simulation done by the student; this grade is derived by how many mistakes are made by the student on the lab

- Simulation
  - gotoAnimation() prompts the simulation to illustrate an animation to the students which teaches them important concepts for performing the mitosis lab

- DatabaseConnection
  - storeAccountInfo() stores the account information for both students and professors; this includes login information and previous grades for students
  - storeSimulationInfo() stores the mitosis lab simulation information which is performed by students
  - storePointDeduction() stores the amount of points taken off the final grade for the simulation done by the student; this grade is derived by how many mistakes are made by the student on the lab
  - storeErrorCount() stores how many errors are made by the student while performing the lab simulation

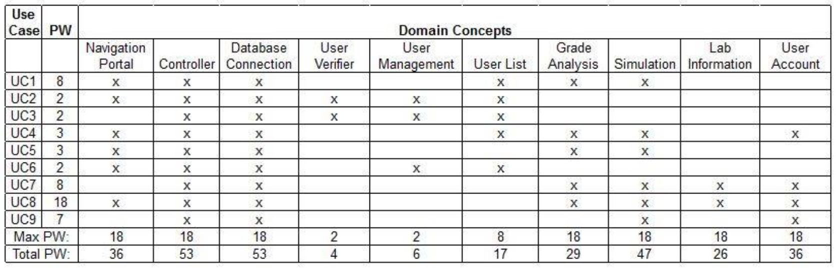
- Feedback
  - givePositiveFeedback() throughout the lab simulation, the system will give the student positive feedback after the correct answer of a question or at key junctures in the simulation

  - giveNegativeFeedback()  throughout the lab simulation, the system will give the student negatuve feedback after the wrong answer of a question or after part of the lab has been performed incorrectly by the student

- LabInformation

  - adjustView()  when viewing the cell sample, the student needs to use a microscope to be able to properly view the process of mitosis; this allows the student to adjust the microscope until the sample is in proper view
  - createGraph()  allows the student to create a graph containing data garnered from performance of the lab simulation
  - createTable() - allows the student to create a graph containing data garnered from performance of the lab simulation

## 8.4 Traceability Matrix



| Use Case | PW | Domain Concepts | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Navigation Portal | Controller | Database Connection | User Verifier | User Management | User List | Grade Analysis | Simulation | Lab Information | User Account |
| UC1 | 8 | x | x | x | | | x | x | x | | |
| UC2 | 2 | x | x | x | x | x | x | | | | |
| UC3 | 2 | | x | x | x | x | x | | | | |
| UC4 | 3 | x | x | x | | | x | x | x | | x |
| UC5 | 3 | x | x | x | | | | x | x | | |
| UC6 | 2 | x | x | x | | x | x | | | | |
| UC7 | 8 | | x | x | | | | x | x | x | x |
| UC8 | 18 | x | x | x | | | | x | x | x | x |
| UC9 | 7 | | x | x | | | | | x | | x |
| Max PW: | | 18 | 18 | 18 | 2 | 2 | 8 | 18 | 18 | 18 | 18 |
| Total PW: | | 36 | 53 | 53 | 4 | 6 | 17 | 29 | 47 | 26 | 36 |

Figure 25: Traceability Matrix of Domain Concepts to Classes

The NavigationPortal class was derived from the Navigation Portal concept. This class satisfies the corresponding domain concept because it will be used to bring the user to wherever he needs to go, whether it is the simulation, the user list, or the grade analysis portal. The Controller class was derived from the Controller concept. This class satisfies the corresponding domain concept because it will be used to update, retrieve, verify, access, etc. details that are necessary for each class and actor to interact. The class is essentially the heart of the program as it is the middle-man.

The DatabaseConnection class was derived from the Database Connection concept. This class satisfies the corresponding domain concept because it will be used as the connecting point between the program and the database itself. In order to access the database, which holds all the information, a connection is necessary.

The UserVerifier class was derived from the User Verifier concept. This class satisfies the corresponding domain concept because it will be used to verify whether or not the user that is logged in is a registered student or professor for a specific course. Without this class, security would not be implemented and anyone can access any aspect of the program, which is not a desirable aspect.

The UserManagement class was derived from the User Management concept. This class satisfies the corresponding domain concept because it will be used to register a new user, whether it is a professor or a student, into the database, as well as allowing the professor to insert a student into the lab course itself. It also gives the option of deleting a user from the database. This lets the professor drop students from the lab or clean the slate for a new class.

The UserList class was derived from the User List concept. This class satisfies the corresponding domain concept because it will hold a list of every student, professor, and Teaching Assistant registered for the course. This is the list that holds every bit of information that is related to each user. If the user is a professor and is accessing this list, he is able to see every student's grades. If the user is a student, he is only able to see his own grade.

The GradeAnalysis class was derived from the Grade Analysis concept. This class satisfies the corresponding domain concept because it allows the professor to select how he wants the lab to be graded. He can determine how many mistakes that a student can make before deducting points, he can determine how many points to deduct, and he can choose whether or not a student would need to be forced to restart the simulation for a reduced grade. This class will also keep track of how many mistakes a student has made, and therefore how many point deductions there are. This will update the grade of the student as well.

The Simulation class was derived from the Simulation concept. This class satisfies the corresponding domain concept because it is used for a user to access the lab simulation. It is only specific to the simulation itself, not the feedback system. It will access an introduction animation that will demonstrate the lab material and allow the user to interact with the lab simulation itself.

The Feedback class was derived from the Simulation concept. This class satisfies the corresponding domain concept because it is used to provide feedback to the user. It will give positive feedback or tips if the user has answered a question correctly or a correct action is performed and it will give a negative feedback if the user has done the opposite.

The LabInformation class was derived from the Lab Information concept. This class satisfies the corresponding domain concept because it allows students to record data and information for the lab. It will also provide students with different viewing angles to facilitate the characterization of each step of mitosis. This class will also provide the list of questions that will need to be answered by the students as they progress through the lab.

The UserAccount class was derived from the User Account and Manage History concept. This class satisfies the corresponding domain concepts because it will contain a students name, grades, account information, and a record of his progress. We have combined the two concepts because it will be easier to implement in the code, rather than have them as separate classes. The name falls under UserAccount since each account will have its own progress tracker. This class will be contained in the UserList class as there can be multiple accounts. This is where the students will be able to access their own grades when accessing the User List portal from the Navigation Portal.

## 8.5 Design Patterns

## 8.6 Object Constraint Language (OCL) Contracts

These contracts are constraints on a class that allows the users of the class, implementers, and extenders to share the same assumptions about the data. They are between the class implementer about the promises of what can be expected and the class user about the obligations that must be met before the class is used.

### 8.6.1 Invariants

**context** UserAccount **inv:**
$self.getGrades() > 0$

**context** GradeAnalysis **inv:**
$self.chooseErrorCount() >= 0$
$self.choosePointDeduction() >= 0$
$self.countErrors() >= 0$
$self.countPointDeduction() >= 0$
$self.calculateGrades() >= 0$

**context** Simulation **inv:**
$self.acquireFeedback() == True$

**context** DatabaseConnection **inv:**
$self.storeAccountInfo() == True$
$self.storeSimulationInfo() == True$
$self.storePointDeduction() == True$
$self.storeErrorCount() == True$

### 8.6.2 Preconditions

**context** Controller::gotoNP() **pre:**
$self.checkUsername() == True$

**context** NavigationPortal::Access() **pre:**
$(self.isProfessor() == True || self.isProfessor() == False)$
$\&\& self.checkUsername() == True$

**context** GradeAnalysis::chooseErrorCount() **pre:**
$portal ==$ ”$GradeAnalysis$” $\&\& self.isProfessor() == True$

**context** GradeAnalysis::choosePointDeduction() **pre:**
$portal ==$ ”$GradeAnalysis$” $\&\& self.isProfessor() == True$

**context** GradeAnalysis::chooseErrorCount() **pre:**
$portal ==$ ”$GradeAnalysis$” $\&\& self.isProfessor() == True$

**context** Simulation::gotoAnimation() **pre:**
$portal == "Simulation" \&\& self.checkUsername() == True$

**context** Simulation::runSimulation() **pre:**

$portal == "Simulation" \&\& self.checkUsername() == True$

**context** Feedback::givePositiveFeedback() **pre:**
$portal == "Simulation" \&\& self.isProfessor() == True \&\& self.runSimulation()$

**context** Feedback::giveNegativeFeedback() **pre:**
$portal == "Simulation" \&\& self.isProfessor() == True \&\& self.runSimulation()$

**context** LabInformation::adjustView() **pre:**
$portal == "Simulation" \&\& self.isProfessor() == True \&\& self.runSimulation()$

**context** LabInformation::createGraph() **pre:**
$portal == "Simulation" \&\& self.isProfessor() == True \&\& self.runSimulation()$

**context** LabInformation::createTable() **pre:**
$portal == "Simulation" \&\& self.isProfessor() == True \&\& self.runSimulation()$

### 8.6.3 Postconditions

**context** Controller::gotoNP() **post:**
$self.Access()$

**context** Simulation::runSimulation() **post:**
$self.storeErrorCount() \&\& self.storeSimulationInfo() \&\&$
$self.storePointDeduction() \&\& self.calculateGrade()$

# 9 System Architecture and System Design

## 9.1 Architectural Styles

Wikipedia[4] defines software architecture as a denotation of the high level structures of a software system. It is also described as the set of structures needed to reason about the software system, which comprise the software elements, the relations between them, and the properties of both elements and relations.

The Virtual Biology Lab System will be broken down into 2 major structures: the website component and the lab simulation component. The website component will be written in python with the Django framework. Django is considered a Model-View-Controller(wikipedia[5]) framework, so this project will follow that architecture pattern. It will also be built using MySQL as a backend database.

The Lab Simulation component will be implemented using Adobe Flash Player, and will allow hands-on interactive learning.

## 9.2 Identifying Subsystems

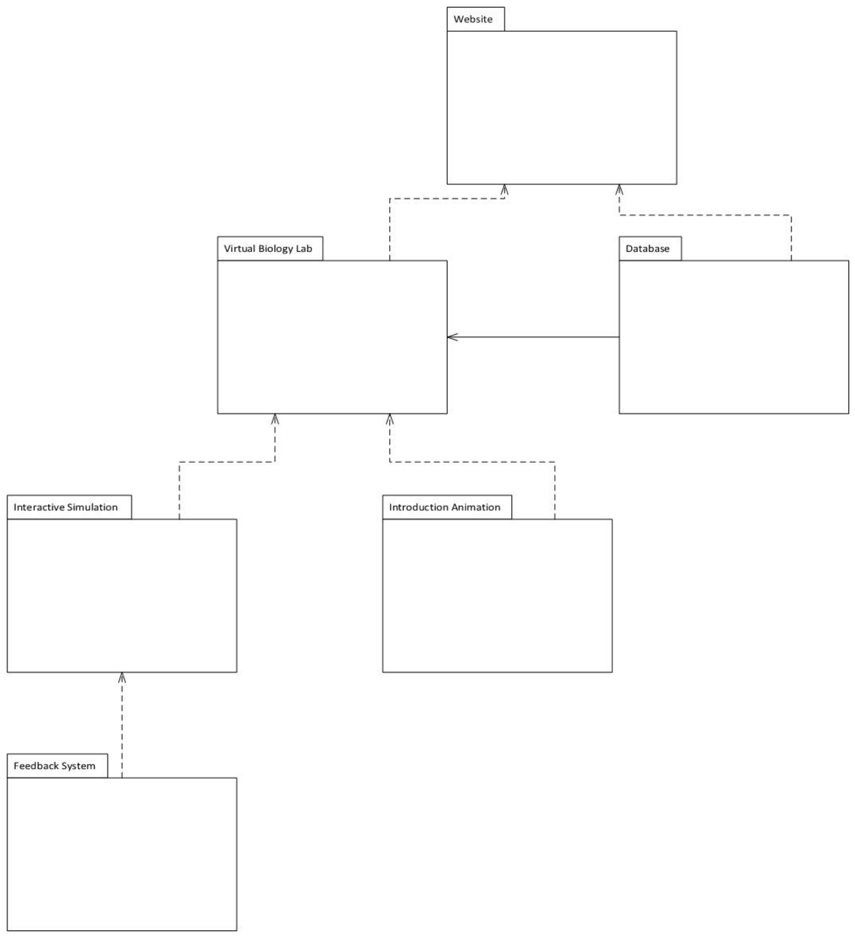


Figure 26: System Architecture: Subsystem Package Diagram

[4]http://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.2F_Patterns
[5]http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

The Virtual Biology Lab system can be broken down into smaller subsystems. The subsystem as shown in the diagram has 4 major tiers. The overlying system is labeled as the Website. This is mainly due to the fact that this entire project will be implemented as a web application. This will hold essentially the entire system, but individual responsibilities include the Navigation Portal/login and system settings. This system will be broken down into the Virtual Biology Lab and the Database subsystems.

The Virtual Biology Lab subsystem is responsible for everything related to the biology mitosis experiment, including the introduction animation, simulation and the feedback system. These can be seen as subsystems of the Virtual Biology Lab: Interactive Simulation, Introduction Animation and the Feedback System. Because the feedback is a result of the simulation, it is considered a subsystem of the simulation.

The database subsystem directly associates with the Virtual Biology Lab subsystem, storing the grades and progress of each user. For this reason, the association arrow is given.

## 9.3 Mapping Subsystems to Hardware

This system will require a minimum of a single client side machine and a single server machine. The client machine maps to the Virtual Biology Lab subsystem. It will also contain aspects of the website system, such as the Navigation Portal UI/login and the system settings. The server machine maps to Database subsystem. It will also handle some of the Feedback system responsibilities such as storing and calculating the grades.

Although the system should not be overly demanding in resources, some minimum requirements are needed. They can be mostly seen in Hardware Requirements section. However, the client side must have a web browser capable of HTML, CSS, Javascript and Flash, for both the website capabilities and the simulation.

Outside of these machines, no further hardware is required, due to the lack of need for any sensors or measuring devices. This system is mostly self contained.

## 9.4 Persistent Data Storage

Our system updates based on the events taking place. All system and simulation data are stored on the database thus meaning the controller, UI, and database are in constant conversation allowing for the system to do multiple checks and displaying user messages; ie checking login information, pulling simulation history, pulling grades, displaying updated profile and simulation information, etc. Our database will also house errors created by the user/system.

Our system also plans on having a backup system, whether it be local or cloud, or both. Because we are planning to back up data and information, we must also set up times in the day when the system can backup all changes made throughout the day. The best choice that I have seen so far that makes the most sense is to back up towards the end of the day when the least amount of traffic is on. Maybe setup a warning message to all users when backups are taking place and provide them with the time so they may know of possible slowdowns in the system.

It is crucial to backup and update the system information as often as possible in case of any issues so not to lose progress made by the student.

## 9.5 Network Protocol

Our network protocol is expected to be HTTP/HTTPS for user to system, SSL for system to database. Because we are planning on using HTTP/HTTPS, we need a stable and reliable communication channel thus we plan on using TCP to provide point-to point channels to communicate without any hitches.

## 9.6 Global Control Flow

### 9.6.1 Execution Orderness

Navigating the system as a whole is very much event-driven. The user will have the option to go to other portals within the website from virtually anywhere. Example, a user can go to the simulation portal from the navigation portal, but also has the option to go to the simulation from the View Grades portal.

On the other hand, the simulation is usually procedure-driven, meaning the user must follow steps in the order they are given to complete the lab. Most labs follow a set procedure to create the results that are desired. Following lab procedures play a key role in understanding material progressively.

### 9.6.2 Time Dependency

To coincide with being event-driven, navigating the system is also event-responsive. Depending on what events take place/chosen, the system reacts.

On the other hand, the simulation can be time-based. Most lab courses require the student to finish their procedures during the lab period which constrains the student to execute the procedures of the lab correctly within said timeframe. Each period varies depending on the school in most cases. For our system, the professor can constrain the students to finish the lab within a certain time frame, 3hrs on a specific day, or just open up the labs for the students to do as they please.

### 9.6.3 Concurrency

We do not have multiple threads running within our simulation.

## 9.7 Hardware Requirements

1. **Server:** login, connection between host environment and user environment
   (a) Network connection, preferably T1
2. **Database:** Simulation/Lab Data; User Login Account information; Feedback messages; Error messages; Strike System;
   (a) Hard drives
      i. Preferably a few terabytes, probably best to put into a RAID. Size of the RAID depends on the user size.
3. **Backup:** All database information
   (a) Hard drives

            i. Preferably at least double the size of the Database just to be safe.

4. Cloud

    (a) Can store backup in cloud to be accessible anywhere in case of issues to server or if hard drives have an error and lose data.

As development continues, the hardware requirements will become more accurate, and these requirements are only a draft, and not finalized.

# 10 Algorithms and Data Structures

## 10.1 Algorithms

The mathematical model of a grading scheme was implemented. This would mainly use a weighted system. There are different types of mistakes that can be made while students are doing the simulation. There are simple errors, such as placing the dye on the microscope slide before the onion is placed, and there are more important errors that should not be made as often, such as defining the stages of mitosis. Obviously, the more simple errors would be weighted less and the more important errors that should not be constantly made would be weighted more. Also, if the same error is made, a set amount of points would be taken off. A default grading scheme would be already set, but the professor would be able to change the criteria of how many points would be taken off, how much to weight each mistake, etc. The mathematical model would be as follows:

## 10.2 Data Structures

We are implementing a linked list for the user accounts. The linked list itself will be the user list, with each node an instance of the individual user account for every registered user. They have the name of the student as the value and a pointer to the next student in alphabetical order. Since the user list will be alphabetized as students are added in, a linked list will be better than an array. The reason for this is because linked lists are very flexible. Items (or in this case students) can be added or removed from anywhere in the list, rather than adding items to the end of an array and then sorting multiple times. Another reason is that there isnt a static number of students to add to the database. Unlike arrays, linked lists are dynamic data structures, which means that there is no need to define an initial size of the list.

We are also going to be using a hash table for our feedback system so that we can have an associative array that can map keys to values. With this, whenever a student makes a mistake or answers, the system will use the key to pull the specific message that needs to be displayed. A hash table will allow the program to be more efficient than using a bunch of if else statements. It can simply look up some key based on the event that occurs, and quickly pull the information or message needed.

# 11 User Interface Design and Implementation

There is one minor change. The bar of tabs above the simulation portal mock-up is to be removed. Some of the tabs that were listed, such as strikes and demo, are no longer portals and have been combined with other ones, such as Grade Analysis and Simulation. It will simplify the visual appearance for the user, reducing distractions. It also helps simplify the coding because we wouldnt have to add the option of a mini navigation portal inside another portal. Instead, the user will have the option of a button to enter the Navigation Portal. There, they will have the choices of different portals. This way each screen has a specified task, reducing the need to understand and comprehend for the user.

The major change required for the GUI design made in the screen mock-up. Almost everything will be removed. Instead, there are three buttons that will be used. They are the three portals: Grade Analysis, User List, and Simulation. Each button will take the user to the individual page. This implements visual simplicity and requires less complex code to make. It is also more aesthetically pleasing without having to look around for what is desired.

The design is simplistic because it only contains what is necessary. The login screen has two options: logging in and registering into the lab. The main user interface that will be visible throughout most of the simulation is a bit more complicated. However, it is still a combination of aesthetically pleasing and usefulness. The table of contents serves as a way to easily traverse the lab sections. The top right of the screen will contain your username, to notify who is logged in. It will also have a logout feature to exit the simulation and let someone else on. Lastly, there will be a menu button to redirect the user to the Navigation Portal. The largest box in the middle contains the main lab experiment, with interactivity such as drag and drop. All in all, we kept what was necessary for the user to have the tools needed to operate smoothly.

## 12 Design of Tests

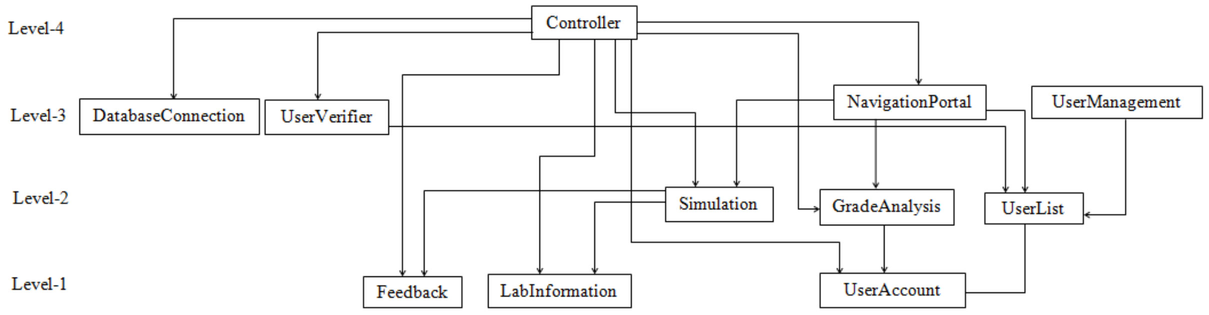


Figure 27: Units Hierarchy

Our tests will only cover the higher priority use cases, rather than the lower priority requirements. The reason for this is because it is not practical to test every single test case, so we will only be testing the test cases that refer to the most important use cases. They are also the more common ones that are required to run successfully for completion of the lab.

As seen in the units hierarchy, there are four levels. We will be implementing top-down integration because our program does not have many lower-level components. Since the top most components, such as the Navigation Portal, are very important to the virtual biology lab, they cannot be tested last because if a fault or bug is found, we may need to redesign the entire system. With top-down integration, we will be starting with the testing of the highest level of hierarchy that no other units depend on. This way, if the top level units pass testing, we know that there is a stable ground for the other units to be based upon. Our test cases are essentially derived directly from the requirements and use cases that we have discussed earlier. The only disadvantage is that we would need to develop test stubs. These test stubs can be full of errors and can be time consuming as well. Other than that single drawback, top-down integration is the best integration testing strategy to implement for our program because it is focused hugely on the user interface itself.

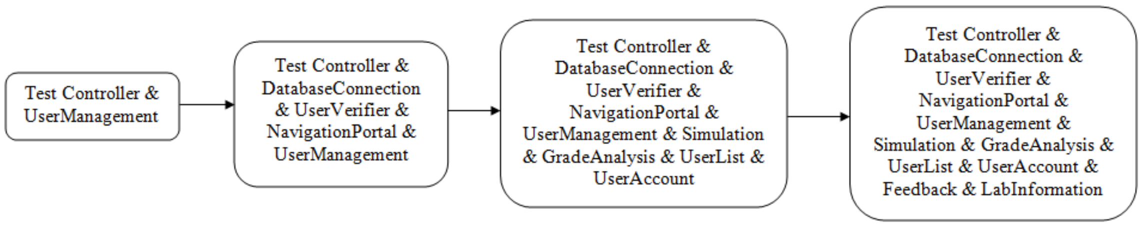

Test Controller & UserManagement
Test Controller & DatabaseConnection & UserVerifier & NavigationPortal & UserManagement
Test Controller & DatabaseConnection & UserVerifier & NavigationPortal & UserManagement & Simulation & GradeAnalysis & UserList & UserAccount
Test Controller & DatabaseConnection & UserVerifier & NavigationPortal & UserManagement & Simulation & GradeAnalysis & UserList & UserAccount & Feedback & LabInformation

Figure 28: Top-Down Integration(based on Units Hierarchy)

## 12.1 Class State Diagrams

State Diagrams are used to show the state of a class at a given point during its use. The reason we need to design state diagrams is to help with the implementation of testing. If, after a certain action or method call, the class did not end up in the correct state, there exists a bug. State Diagrams, along with Unit tests are used to get rid of the more prominent bugs within the project. Only the 4 major classes were given state diagrams: Controller, Navigation Portal, Simulation, and Feedback classes.

### 12.1.1 Controller Class

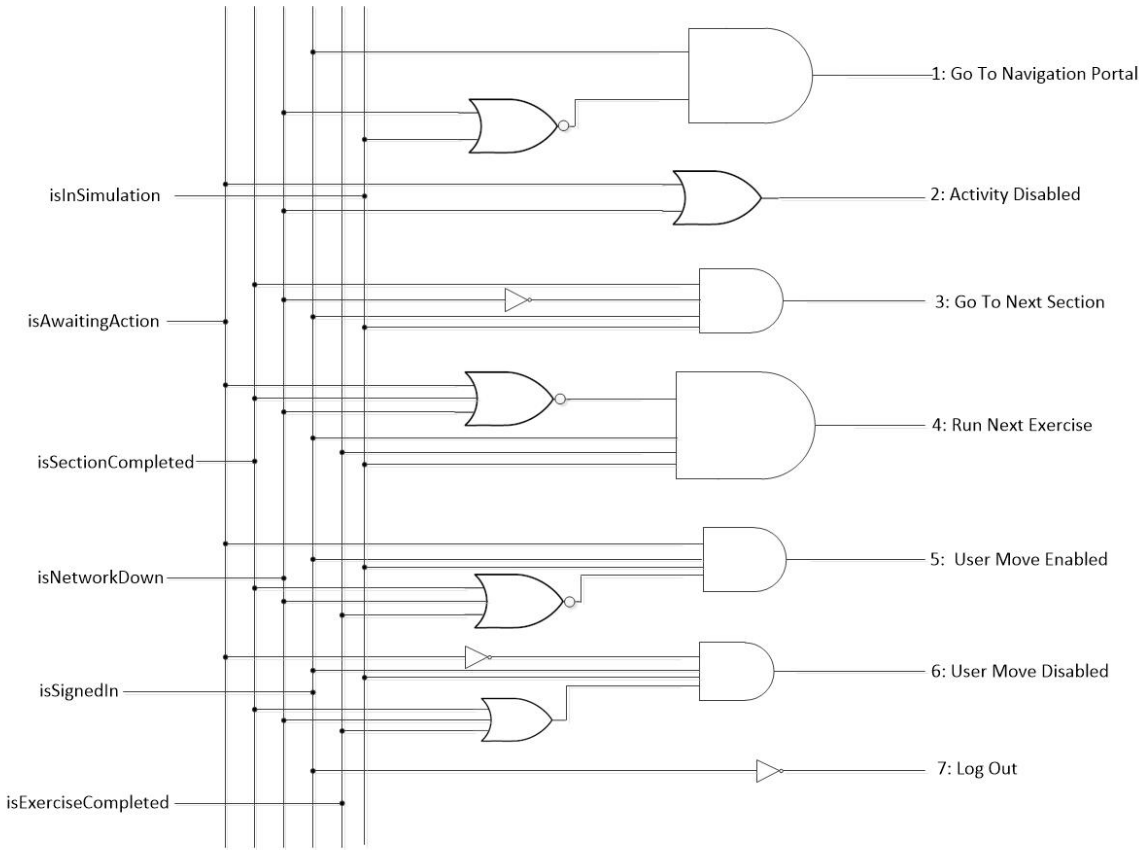


Figure 29: Controller Class

isAwaitingAction is used to indicate that the system is waiting for an input response from the User. isSectionCompleted is for notifying the system when the current lab section is completed. isExerciseCompleted is for notifying the system when the current exercise within the lab section is completed. isNetworkDown is used for indicating whether or not the network is running. isSignedIn is used for checking if the current user is logged into the system. isInSimulation is used to identify whether or not the user is in the simulation portal.

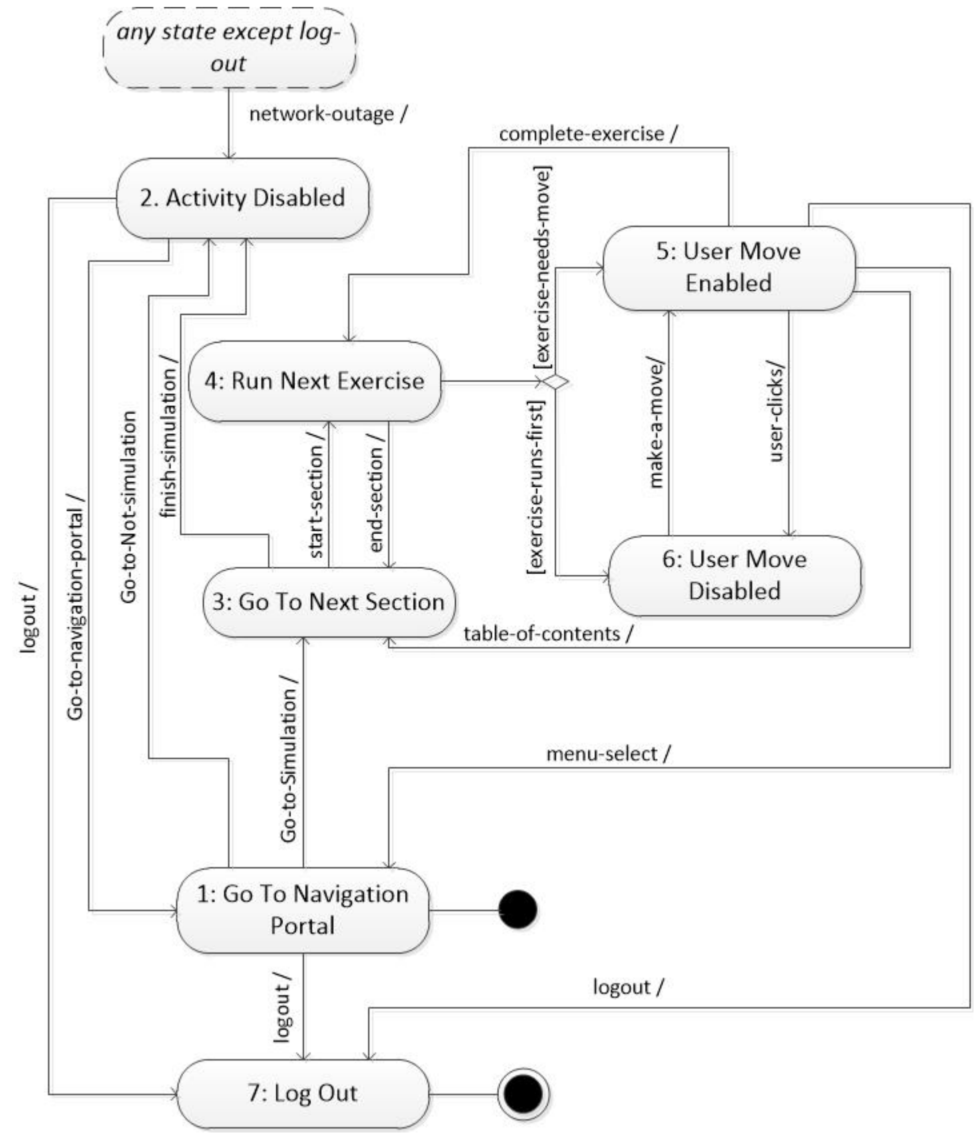
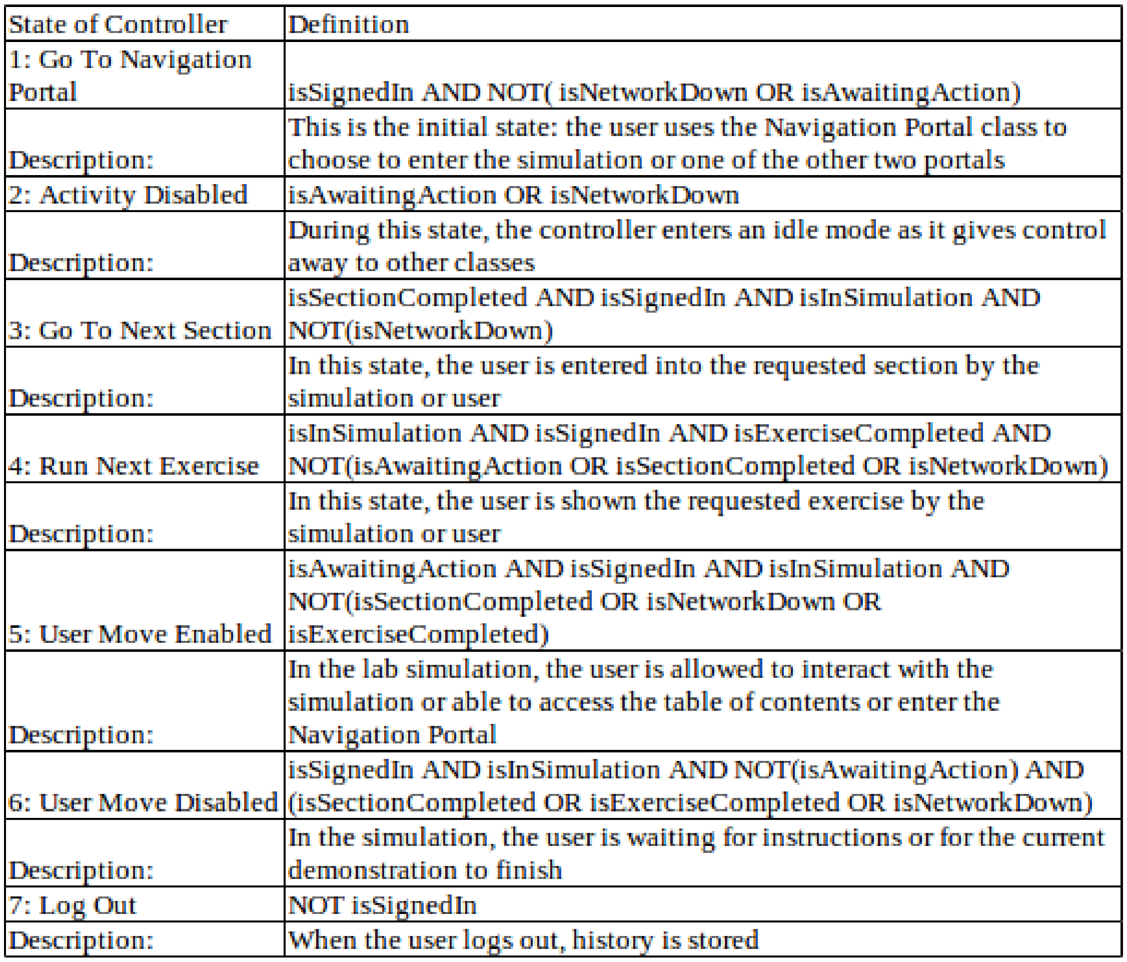


Figure 30: Controller Class State Diagram



| State of Controller | Definition |
|---|---|
| 1: Go To Navigation Portal | isSignedIn AND NOT( isNetworkDown OR isAwaitingAction) |
| Description: | This is the initial state: the user uses the Navigation Portal class to choose to enter the simulation or one of the other two portals |
| 2: Activity Disabled | isAwaitingAction OR isNetworkDown |
| Description: | During this state, the controller enters an idle mode as it gives control away to other classes |
| 3: Go To Next Section | isSectionCompleted AND isSignedIn AND isInSimulation AND NOT(isNetworkDown) |
| Description: | In this state, the user is entered into the requested section by the simulation or user |
| 4: Run Next Exercise | isInSimulation AND isSignedIn AND isExerciseCompleted AND NOT(isAwaitingAction OR isSectionCompleted OR isNetworkDown) |
| Description: | In this state, the user is shown the requested exercise by the simulation or user |
| 5: User Move Enabled | isAwaitingAction AND isSignedIn AND isInSimulation AND NOT(isSectionCompleted OR isNetworkDown OR isExerciseCompleted) |
| Description: | In the lab simulation, the user is allowed to interact with the simulation or able to access the table of contents or enter the Navigation Portal |
| 6: User Move Disabled | isSignedIn AND isInSimulation AND NOT(isAwaitingAction) AND (isSectionCompleted OR isExerciseCompleted OR isNetworkDown) |
| Description: | In the simulation, the user is waiting for instructions or for the current demonstration to finish |
| 7: Log Out | NOT isSignedIn |
| Description: | When the user logs out, history is stored |

Figure 31: Controller State Diagram Table

### 12.1.2 Navigation Portal

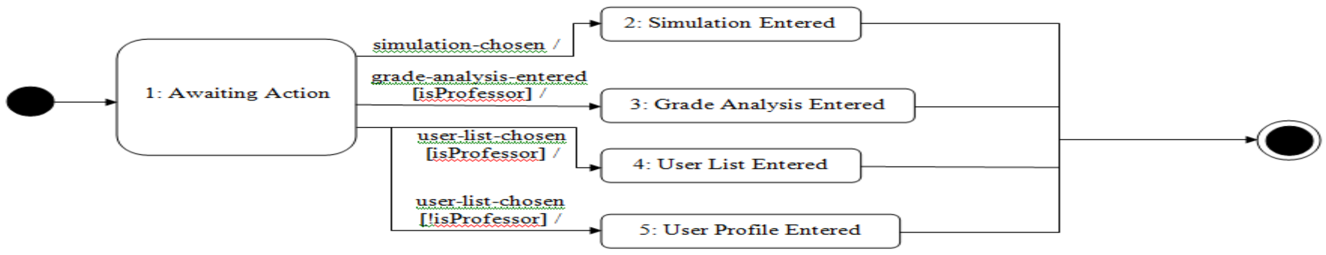
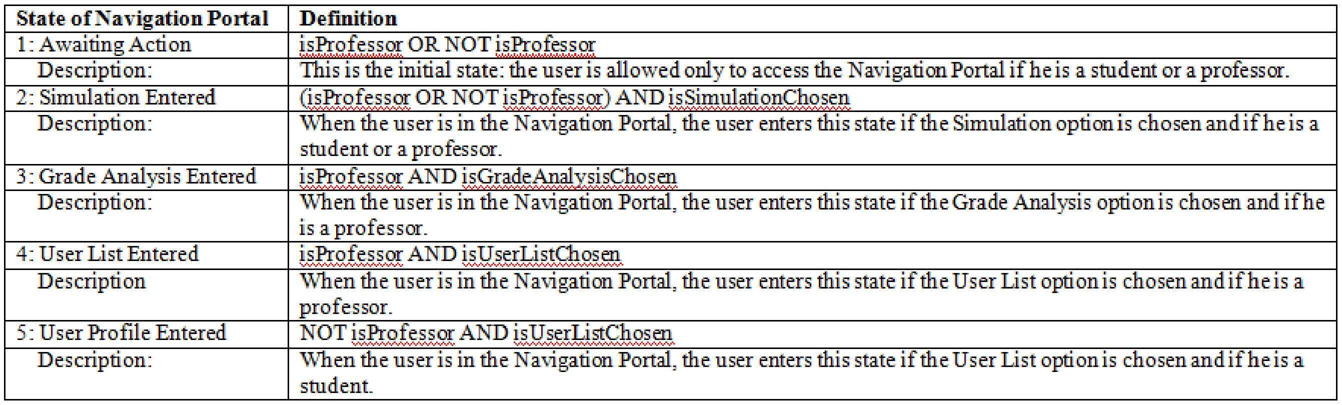

1: Awaiting Action
simulation-chosen /
grade-analysis-entered [isProfessor] /
user-list-chosen [isProfessor] /
user-list-chosen [!isProfessor] /
2: Simulation Entered
3: Grade Analysis Entered
4: User List Entered
5: User Profile Entered

Figure 32: Navigation Portal State Diagram



| State of Navigation Portal | Definition |
|---|---|
| 1: Awaiting Action | isProfessor OR NOT isProfessor |
| Description: | This is the initial state: the user is allowed only to access the Navigation Portal if he is a student or a professor. |
| 2: Simulation Entered | (isProfessor OR NOT isProfessor) AND isSimulationChosen |
| Description: | When the user is in the Navigation Portal, the user enters this state if the Simulation option is chosen and if he is a student or a professor. |
| 3: Grade Analysis Entered | isProfessor AND isGradeAnalysisChosen |
| Description: | When the user is in the Navigation Portal, the user enters this state if the Grade Analysis option is chosen and if he is a professor. |
| 4: User List Entered | isProfessor AND isUserListChosen |
| Description | When the user is in the Navigation Portal, the user enters this state if the User List option is chosen and if he is a professor. |
| 5: User Profile Entered | NOT isProfessor AND isUserListChosen |
| Description: | When the user is in the Navigation Portal, the user enters this state if the User List option is chosen and if he is a student. |

Figure 33: Navigation Portal State Diagram Table

### 12.1.3 Simulation

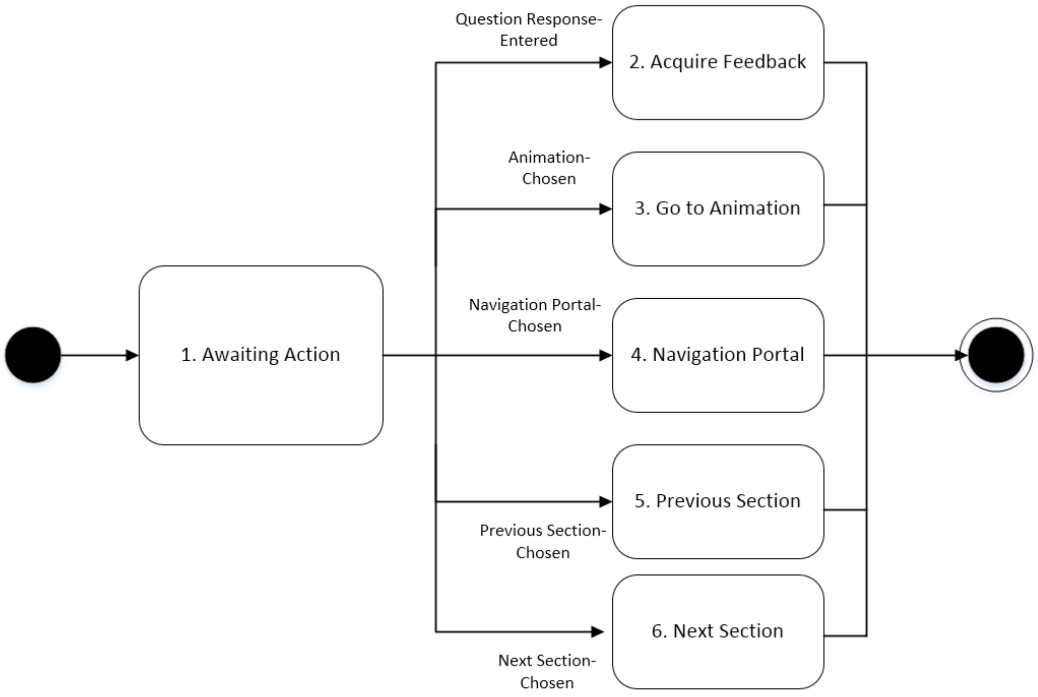
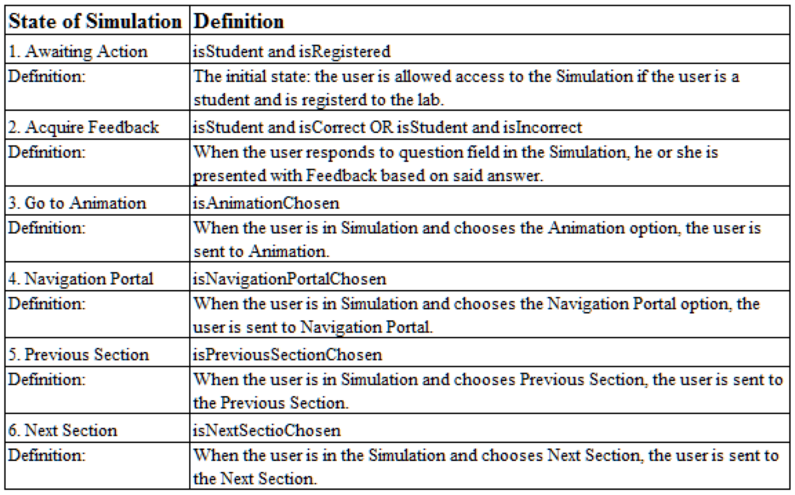


Figure 34: Simulation State Diagram



| **State of Simulation** | **Definition** |
|---|---|
| 1. Awaiting Action | isStudent and isRegistered |
| Definition: | The initial state: the user is allowed access to the Simulation if the user is a student and is registerd to the lab. |
| 2. Acquire Feedback | isStudent and isCorrect OR isStudent and isIncorrect |
| Definition: | When the user responds to question field in the Simulation, he or she is presented with Feedback based on said answer. |
| 3. Go to Animation | isAnimationChosen |
| Definition: | When the user is in Simulation and chooses the Animation option, the user is sent to Animation. |
| 4. Navigation Portal | isNavigationPortalChosen |
| Definition: | When the user is in Simulation and chooses the Navigation Portal option, the user is sent to Navigation Portal. |
| 5. Previous Section | isPreviousSectionChosen |
| Definition: | When the user is in Simulation and chooses Previous Section, the user is sent to the Previous Section. |
| 6. Next Section | isNextSectioChosen |
| Definition: | When the user is in the Simulation and chooses Next Section, the user is sent to the Next Section. |

Figure 35: Simulation State Diagram Table

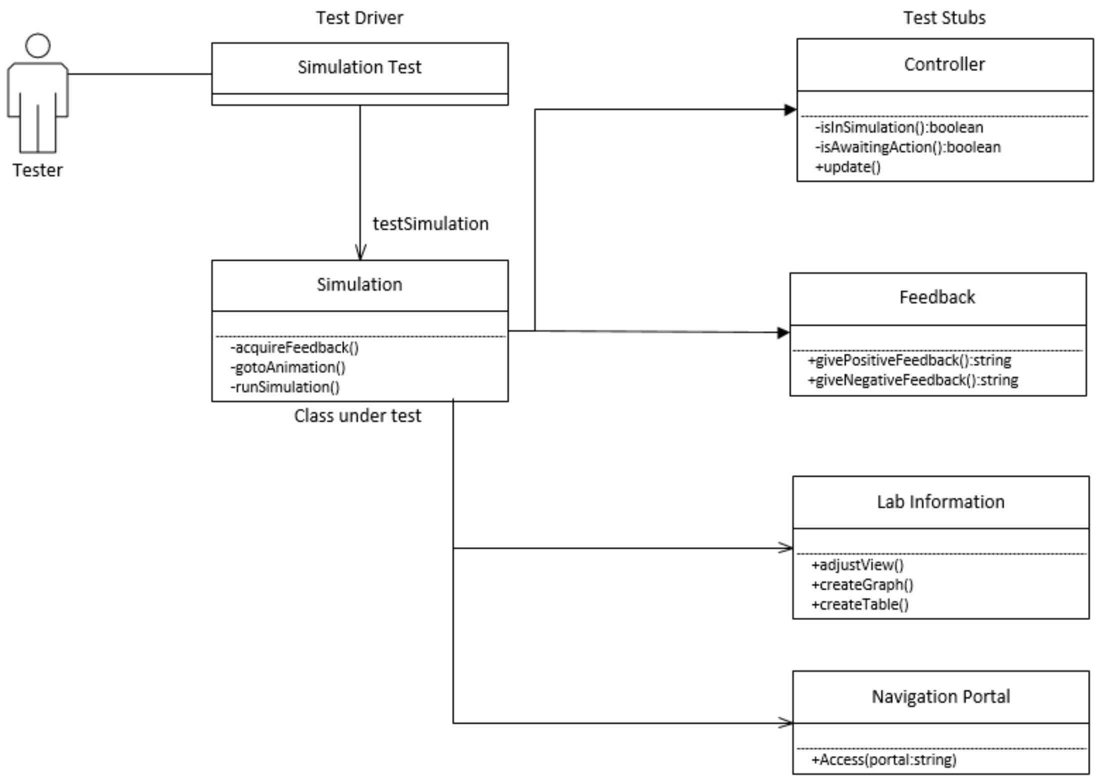


Figure 36: Simulation Class Test

### 12.1.4 Feedback

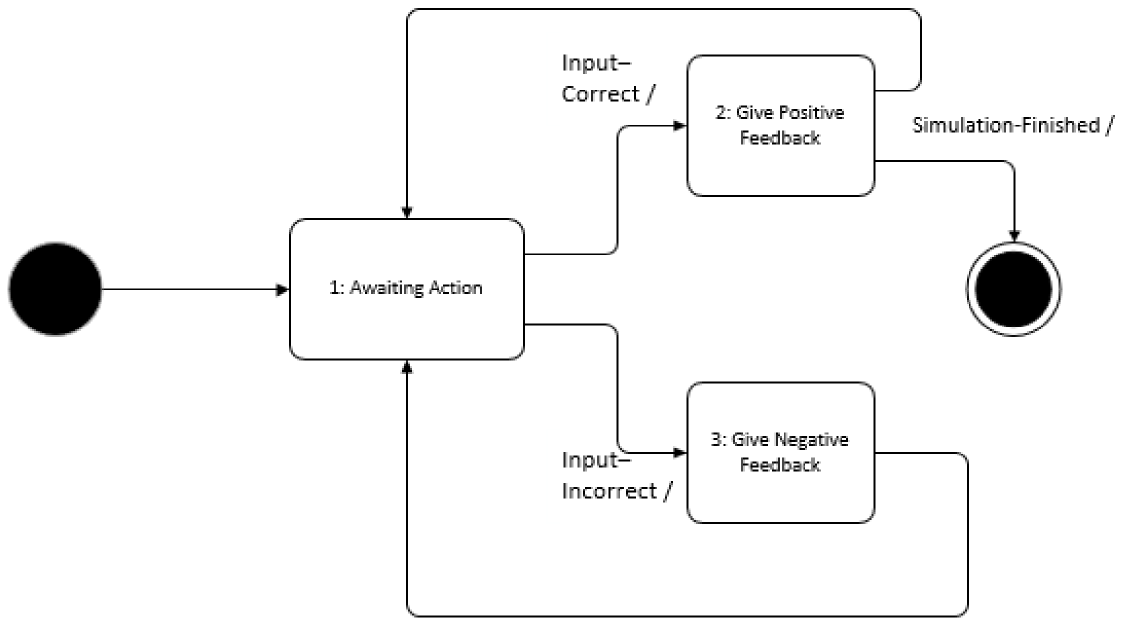
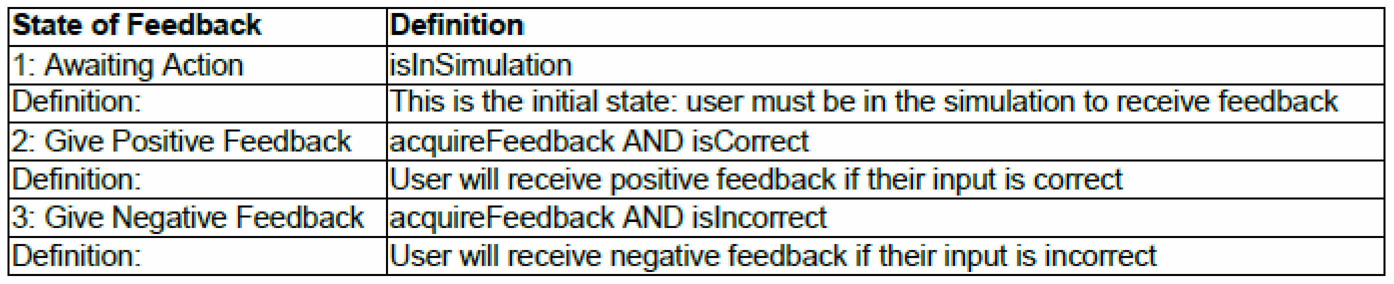


Figure 37: Feedback State Diagram



| **State of Feedback** | **Definition** |
|---|---|
| 1: Awaiting Action | isInSimulation |
| Definition: | This is the initial state: user must be in the simulation to receive feedback |
| 2: Give Positive Feedback | acquireFeedback AND isCorrect |
| Definition: | User will receive positive feedback if their input is correct |
| 3: Give Negative Feedback | acquireFeedback AND isIncorrect |
| Definition: | User will receive negative feedback if their input is incorrect |

Figure 38: Feedback State Diagram Table

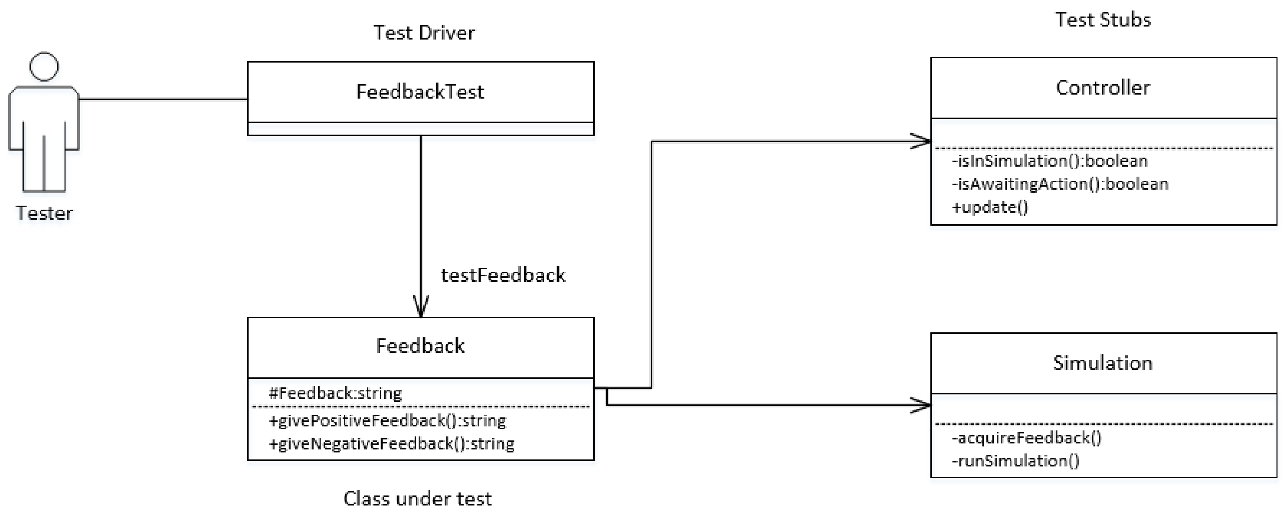
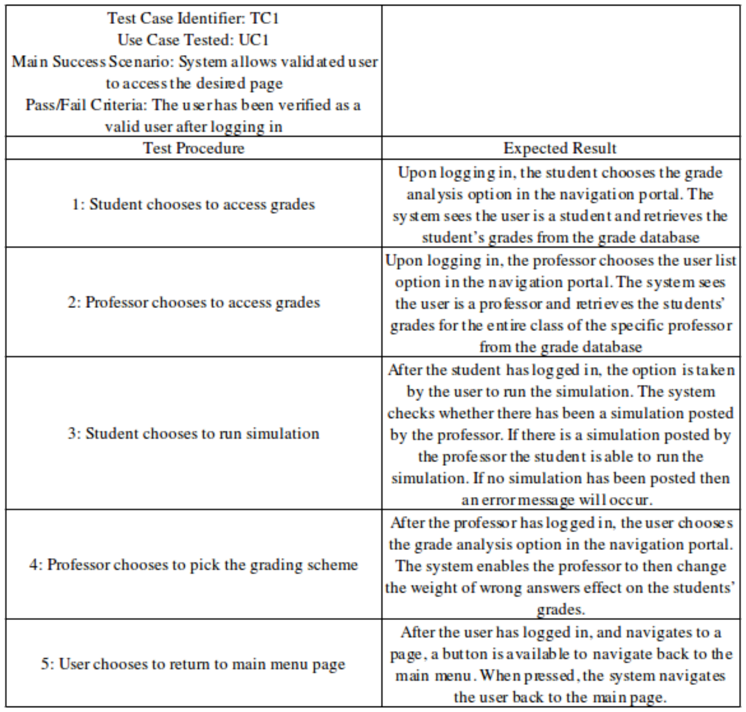


Figure 39: Feedback Class Test

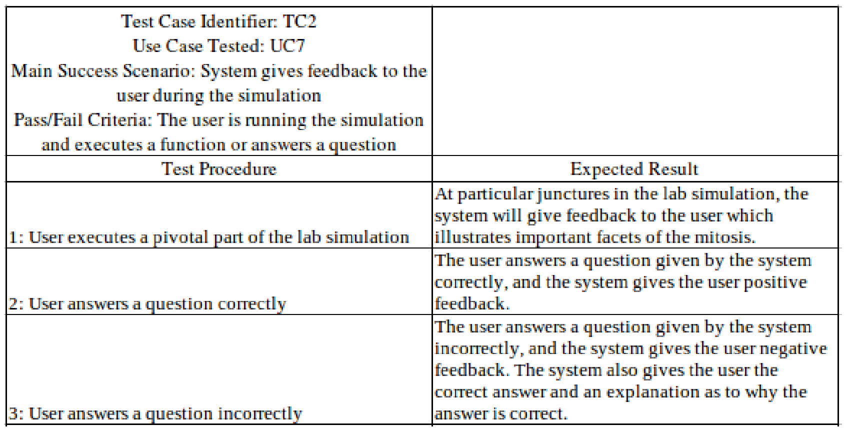
## 12.2 Test Cases

This section delves into the different tests for our most important Use Cases. See the tables below:



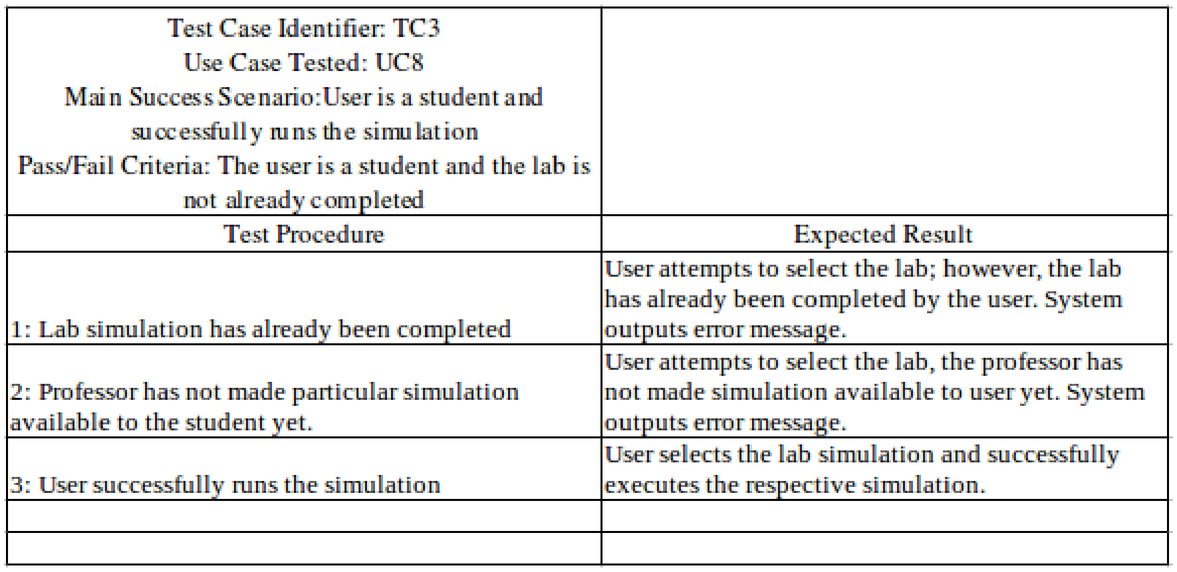
| | |
|---|---|
| Test Case Identifier: TC1<br>Use Case Tested: UC1<br>Main Success Scenario: System allows validated user to access the desired page<br>Pass/Fail Criteria: The user has been verified as a valid user after logging in | |
| Test Procedure | Expected Result |
| 1: Student chooses to access grades | Upon logging in, the student chooses the grade analysis option in the navigation portal. The system sees the user is a student and retrieves the student's grades from the grade database |
| 2: Professor chooses to access grades | Upon logging in, the professor chooses the user list option in the navigation portal. The system sees the user is a professor and retrieves the students' grades for the entire class of the specific professor from the grade database |
| 3: Student chooses to run simulation | After the student has logged in, the option is taken by the user to run the simulation. The system checks whether there has been a simulation posted by the professor. If there is a simulation posted by the professor the student is able to run the simulation. If no simulation has been posted then an error message will occur. |
| 4: Professor chooses to pick the grading scheme | After the professor has logged in, the user chooses the grade analysis option in the navigation portal. The system enables the professor to then change the weight of wrong answers effect on the students' grades. |
| 5: User chooses to return to main menu page | After the user has logged in, and navigates to a page, a button is available to navigate back to the main menu. When pressed, the system navigates the user back to the main page. |

Figure 40: Test Case 1: TC1



| | |
|---|---|
| Test Case Identifier: TC2<br>Use Case Tested: UC7<br>Main Success Scenario: System gives feedback to the user during the simulation<br>Pass/Fail Criteria: The user is running the simulation and executes a function or answers a question | |
| Test Procedure | Expected Result |
| 1: User executes a pivotal part of the lab simulation | At particular junctures in the lab simulation, the system will give feedback to the user which illustrates important facets of the mitosis. |
| 2: User answers a question correctly | The user answers a question given by the system correctly, and the system gives the user positive feedback. |
| 3: User answers a question incorrectly | The user answers a question given by the system incorrectly, and the system gives the user negative feedback. The system also gives the user the correct answer and an explanation as to why the answer is correct. |

Figure 41: Test Case 2: TC2



| | |
|---|---|
| Test Case Identifier: TC3<br>Use Case Tested: UC8<br>Main Success Scenario:User is a student and successfully runs the simulation<br>Pass/Fail Criteria: The user is a student and the lab is not already completed | |
| Test Procedure | Expected Result |
| 1: Lab simulation has already been completed | User attempts to select the lab; however, the lab has already been completed by the user. System outputs error message. |
| 2: Professor has not made particular simulation available to the student yet. | User attempts to select the lab, the professor has not made simulation available to user yet. System outputs error message. |
| 3: User successfully runs the simulation | User selects the lab simulation and successfully executes the respective simulation. |
| | |
| | |

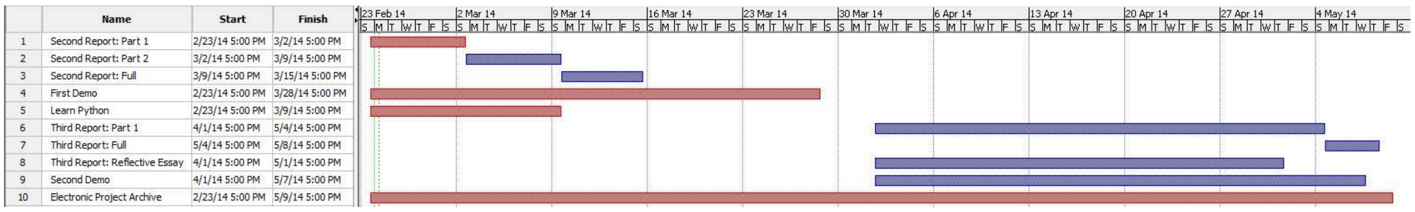
Figure 42: Test Case 3: TC3

These test, as stated above, test the system for our highest priority Use Cases. They are important to implement the test cases for, and since they are ones that are going to be used the most, taking care of mishaps is also higher priority.

To talk about test coverage, these tests explicitly cover a majority of the higher priority use cases, but they do not cover some of the lower priority use cases, such as the ManageHistory(UC9) or the LogIn(UC2). If time allows, these test cases will be implemented, but due to time constraints, these should suffice for a basic system.

# 13 History of Work, Current Status, and Future Work

## 13.1 Plan of Work - Report 1



| | Name | Start | Finish |
|---|---|---|---|
| 1 | Second Report: Part 1 | 2/23/14 5:00 PM | 3/2/14 5:00 PM |
| 2 | Second Report: Part 2 | 3/2/14 5:00 PM | 3/9/14 5:00 PM |
| 3 | Second Report: Full | 3/9/14 5:00 PM | 3/15/14 5:00 PM |
| 4 | First Demo | 2/23/14 5:00 PM | 3/28/14 5:00 PM |
| 5 | Learn Python | 2/23/14 5:00 PM | 3/9/14 5:00 PM |
| 6 | Third Report: Part 1 | 4/1/14 5:00 PM | 5/4/14 5:00 PM |
| 7 | Third Report: Full | 5/4/14 5:00 PM | 5/8/14 5:00 PM |
| 8 | Third Report: Reflective Essay | 4/1/14 5:00 PM | 5/1/14 5:00 PM |
| 9 | Second Demo | 4/1/14 5:00 PM | 5/7/14 5:00 PM |
| 10 | Electronic Project Archive | 2/23/14 5:00 PM | 5/9/14 5:00 PM |

Figure 43: Timeline of the Plan of Work

With the submission of Report #1, the group will move onto writing Report #2 and developing the first demo. The demo and Report #2 have to be worked on simultaneously because the due dates for Report #2 are every 6-7 days following the Report #1 due date. Writing good quality code takes a while, so we will have to be on top of our time management. The real leg work begins when we start to write the software. As described in the product ownership, each pair of group members will work on the specified part of the code. However, since we are mostly inexperienced with python, we will begin by heavily focusing on learning the language. Our designated methods will be www.Codecademy.com, Learn Python the Hard Way, and other reading material. Because we all have some experience programming, at the very least C++, the process of learning python should not take more than 2 weeks. After we passed the tutorials, we will begin finding useful mods (ones that let us represent our code graphically).

After the first demo, which we will present on April 1st, we will start to work on Report #3-revising and improving from our past two reports. Along with the revisions for our report, we will also be revising our first demo to present as a second demo. This time, there are 3 parts that will be worked on at the same time. The first is the Part 1 of Report #3 is the main writing that the group has to work on, thus, it should be started right away. The second is the Reflective Essay which includes an individual write up and should be done on each group members own convenience. The third and final portion is the second demo. Since there is ample time between the first demo and the next due date, working on all three things simultaneously shouldnt pose too much of a problem, but we will not push off the time as there is still a lot to be done. The full report for Report #3 will be written immediately after Part 1. Because it is the last few days of class, we need the time to finish coding. So the report being finished sooner than later is crucial. Finally, the Electronic Project Archive can begin after the second demo. It is then that the coding should be accomplished. It should be finished last, since it compiles everything together.

### 13.1.1 Project Management

The entire project will be done through division of labor. The team will be divided into three groups of two. The lab experiment is broken up into 3 separate parts: The introduction learning simulation and navigation portal, the interactive simulation of mitosis, and the data collection tutorials. Each group will handle the User Interface, the interactive options, finding the graphics, creating the simulation, and providing a feedback system.

### 13.1.2 Introduction Simulation and Navigation Portal

The introduction simulation and the navigation portal will be created by Jeff Gillen and Evan Chipps. Their responsibilities include, but are not limited to, creating the simulation that is knowledgeable and easy to understand, allowing the students to go backwards if something does not make sense, and making it visually appealing. The navigation portal should allow the student to log-in, and give simple ways to navigate to the lab experiment and grades. The navigation should also allow the professor to access the lab, change the different settings of the simulation(such as grading system), and entire class grades.

### 13.1.3 Interactive Mitosis Experiment

Marvin Liu and Anthony Porturas will be responsible for the creation of the interactive simulation itself. This includes the realistic simulation of the process of cutting an onion root and placing it into a microscope to observe the plant cell, and to repeat the process with the animal cell. The system should have many options and include paths leading to common student mistakes. Depending on the action made, the system should respond and give real-time feedback to the student, so that the student can learn from the mistake. The system will keep track of the student's errors, and respond according the professor's policy.

### 13.1.4 Data Collection

The last objective listed in the lab manual states that at the end of the experiments, students should have a working knowledge of how to collect data through the use of graphs and tables. The system will include a graphing tutorial, created by Steve Pak and Brandon Lum. This data collection subsystem should introduce the concepts of graphs and tables, specifically the pie chart and bar graphs. The system should have a quick interactive tutorial, asking them to create both a pie chart and a bar graph based on a set of data. Incorrect graphs should give feedback to the student, saying that the student was wrong, and ask them to try again. Then the system should allow the students to create their own graphs and tables, based on their data from the experiment.

The above requirements for each pair ensures that each member of the group is playing a clearly defined and important role. They also ensure that each member is contributing fairly equally and no one member is taking on a dominant role. To make sure that the subsystems are not vastly different, each pair is responsible for updating the rest of the team. Although some collaboration is necessary for this project to be successful, no subsystem is dependent on the progress of another subsystem.

## 13.2 Plan of Work - Report 2

**Merging work:** Complete
**Compiling work:** Complete

### 13.2.1 Describing Issues:

One of our major issues was the formatting for some of the pieces required for Report 2. Design of tests was a bit unclear on how the section was to be formatted, what exactly went into the report, and also how thorough the professor would like. Upon a quick email to the professor, we were all given the format and what pieces we should focus on for our submission.

### 13.2.2 Project Coordination and Progress Report:

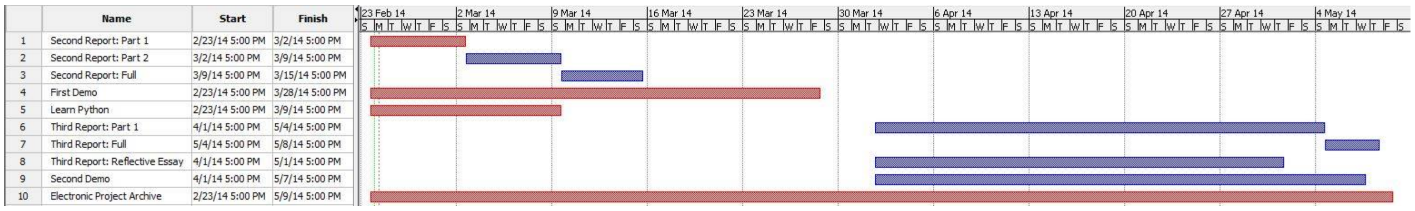
Throughout our process so far, we have had trouble meeting in person so a majority of our group meetings were done virtually. Through our weekly chats, we discussed what we had to do, how we wanted to execute our work, and addressed any questions anyone had in regards to our system. All our work was submitted on time and was looked upon by other group members to check for consistency and completeness. Upon finishing Report 1, we immediately started on Report 2 to get a head start before we start working on our demo. Upon finishing and submitting Report 2, we plan to work on our demo as soon as possible. The sooner we start, the more planning and features we can add to our system.

**Implemented Use Cases:** Our major Use Cases are: 1, 7, & 8 These three use cases are the focal points of our project and are the main focus of our work and testing.

**Already Functional Use Cases:** None at the moment. Creation for the demo has not been started aside from major planning. Construction will begin shortly after we feel comfortable with the majority of our planning.

**Need to be tackled Use Cases:** All use cases still need to be tackled. Again, focusing on our major use cases (U.C. 1, 7, 8) first as they are the most important to our project and what our solution focuses around and then we will work on the other use cases to make sure all requirements are fulfilled.

### 13.2.3 Projected Milestones:



| | Name | Start | Finish |
|---|---|---|---|
| 1 | Second Report: Part 1 | 2/23/14 5:00 PM | 3/2/14 5:00 PM |
| 2 | Second Report: Part 2 | 3/2/14 5:00 PM | 3/9/14 5:00 PM |
| 3 | Second Report: Full | 3/9/14 5:00 PM | 3/15/14 5:00 PM |
| 4 | First Demo | 2/23/14 5:00 PM | 3/28/14 5:00 PM |
| 5 | Learn Python | 2/23/14 5:00 PM | 3/9/14 5:00 PM |
| 6 | Third Report: Part 1 | 4/1/14 5:00 PM | 5/4/14 5:00 PM |
| 7 | Third Report: Full | 5/4/14 5:00 PM | 5/8/14 5:00 PM |
| 8 | Third Report: Reflective Essay | 4/1/14 5:00 PM | 5/1/14 5:00 PM |
| 9 | Second Demo | 4/1/14 5:00 PM | 5/7/14 5:00 PM |
| 10 | Electronic Project Archive | 2/23/14 5:00 PM | 5/9/14 5:00 PM |

Figure 44: Gannt Timeline

### 13.2.4 Breakdown Responsibilities:

**Evan & Jeff:**
Use Case 1: Navigation Portal
Evan Chipps and Jeffrey Gillen are responsible for, but not limited to, creating the introductory simulation material as well as the navigation portal. The Navigation Portal is an important asset for our solution and thus must be the focus of responsibilities for Evan and Jeff. Use Case 1 revolves around the Navigation Portal. The Navigation Portal is crucial for our clients to able to get to the places they need to go with the least amount of effort. The Navigation portal is the link to all major portals on the website. Because it links to so many major pieces of the site, Evan and Jeff will be responsible for testing the navigation portal and all associated classes which include the following:

- Navigation Portal
- User List
- User Management
- User Account
- Grade Analysis
- User Verifier

**Marvin & Anthony:**
Use Case 8: Simulation
Marvin Liu and Anthony Porturas are responsible for creating the interactive simulation. The simulation includes all material and procedures associated with the lab. The lab simulation should have many options including interactive adjustments and needs associated with making the lab procedure as realistic as possible. Marvin and Anthony will be in charge of Use Case 8, the simulation, and all associated classes which include the following:

- Simulation
- Lab Information
- Controller

**Steve & Brandon:**
Use Case 7: Feedback
Steve Pak and Brandon Lum are responsible for creating a dynamic feedback system. This feedback system will be in charge of sending and receiving updates from the simulation. Based on the answers provided by the professor and the responses provided by the students throughout the lab, the feedback system will send out an appropriate message stating any relevant information. Steve and Brandon will be in charge of the Use Case 7, feedback system, and all associated classes which include the following:

- Feedback
- GradeAnalysis
- Database

## 13.3 History of Work

The milestones and deadlines from the previous reports have evolved by a large amount. We have gone milestone to milestone and deadline to deadline. The sense of our project has gotten better and now know what the customer wants. In the beginning, we were confused about what we were supposed to do as this was a new idea to us. Originally, we were going to create the Virtual Biology Lab using python, but we realized that this was not a reasonable option. In the end, we decided to use HTML, CSS, and Javascript to create a web-based simulation. As we got closer and closer to each deadline, we got a better idea as to what was supposed to be done.

The milestones shown in the Gannt Timelines in both the first and second report show the progression of what was to be done. For the final report, there was not much new material to add. We needed to basically put together what we had in Report 1 and 2 and revise. This time, there are basically only two parts that will be worked on. The first is Part 1 of Report 3. First, the new content that must be added is the design patterns and the interaction diagrams that show the new design of the project. The second is the full report. We must include the OCL contracts and include design patterns once again. With this, our project has become more fluid, clearer, and more professional.

## 13.4 Key Accomplishments

- We now excel at the design of UML diagrams.
- Our documentation is very clear and easy to understand.
- We now excel at software engineering development concepts.
- We have created an aesthetically pleasing and professional product.
- We have come up with excellent enhancements to the work of the past group.

## 13.5 Future Work

In the near future, we would like to see our project to be implemented throughout Rutgers University. This would be the ideal direction for our project to go. Otherwise, we would like to see this project be enhanced and improved by other people. This way, if it is not ready to ship as a product, then we would like to get as close as possible to a finished product. If possible, when others are working on this project, we would all like to be present because we know the ins and outs. Since we were the ones that most recently worked on it, we are the most familiar with it. The previous groups before us worked on the project almost a year ago and would not be sufficient.

# 14 References

## 14.1 Biology-Related

**Lab Manual** http://www.ece.rutgers.edu/~marsic/books/SE/projects/ViBE/biolab-1.pdf

**Biology Online** http://www.biology-online.org/

## 14.2 Software-Related

**FURPS+** http://en.wikipedia.org/wiki/FURPS

**Virtual Bio Lab-2012** www.ece.rutgers.edu/~marsic/books/SE/projects/ViBE/2012-g5-report3.pdf

**Architecture Styles** en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.2F_Patterns

**Model-View-Controller** http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller