Virtual Biology Lab Simulation: Mitosis Group 11

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

Project Website[1]

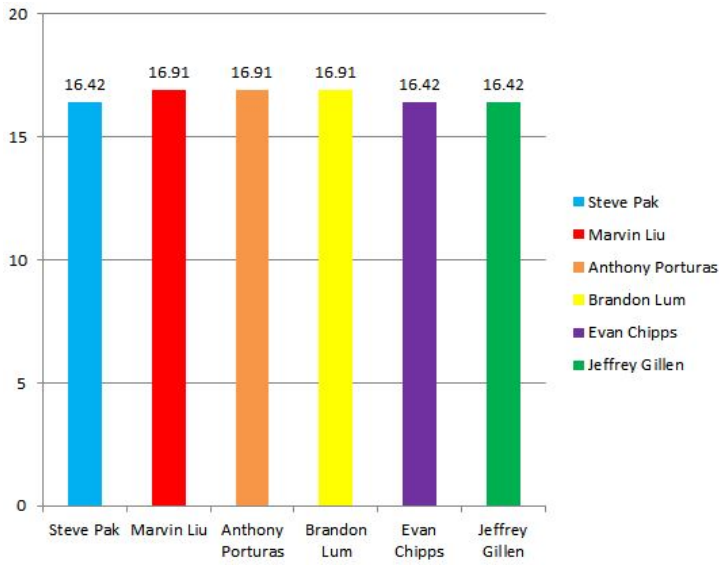
[1] https://sites.google.com/site/virtualbiologylab/

Individual Contributions

Figure 1: Responsibility Matrix

Steve Pak	Marvin Liu	Anthony Porturas	Brandon Lum	Evan Chipps	Jeffrey Gillen
16.42	16.91	16.91	16.91	16.42	16.42

Figure 2: Responsibility Allocation



Steve Pak 16.42%—16.42 points

Marvin Liu 16.91%—16.91 points

Anthony Porturas 16.91%—16.91 points

Brandon Lum 16.91%—16.91 points

Evan Chipps 16.42%—16.42 points

Jeff Gillen 16.42%—16.42 points

Contents

1 Introduction

This report is to serve as a continuation of Report 1. Much of this report will refer to Report 1, such as the Use Cases, Requirements and other such information. Please be familiar with Report 1, before reading this report, as this report builds off what was asserted from Report 1.

This report goes in detail on how the Use cases translate into objects and classes, and the mapping can be seen through a traceability matrix. The report also goes into detail the specifications of the different classes, and how they interact. Some hardware analysis is done based on the specification of the classes and software. And lastly, unit test cases are fleshed out in this report.

2 Interaction Diagrams

2.1 UC8: Simulation

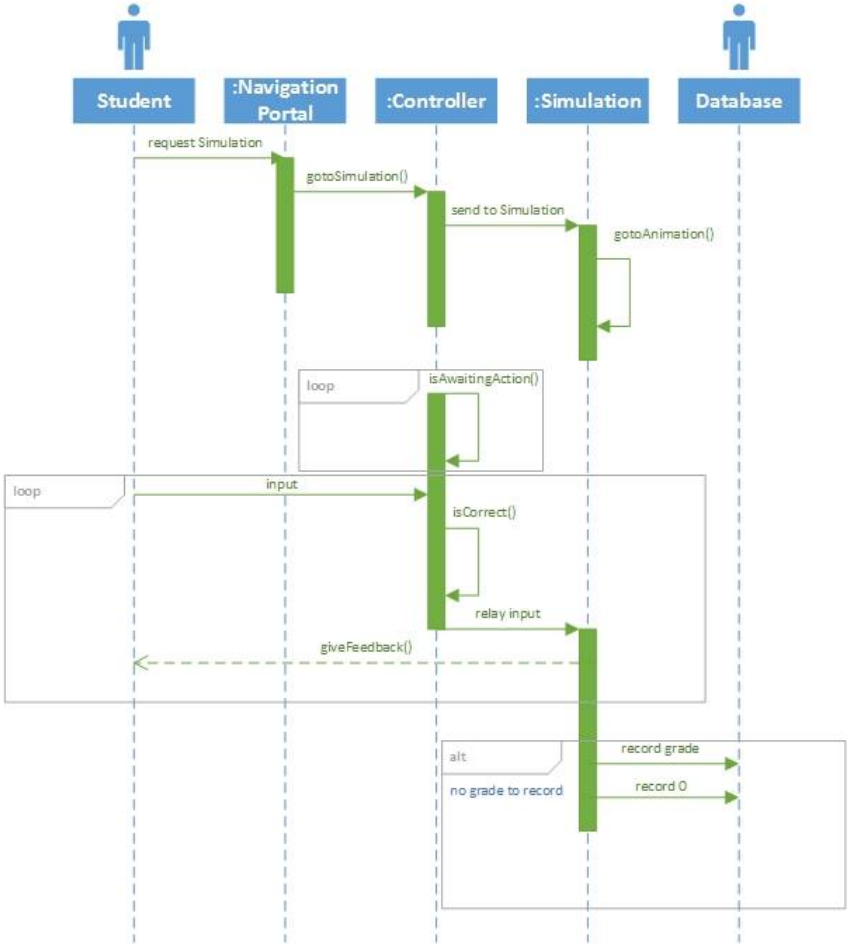


Figure 3: Sequence Diagram of UC7

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.

2.2 UC7: Feedback

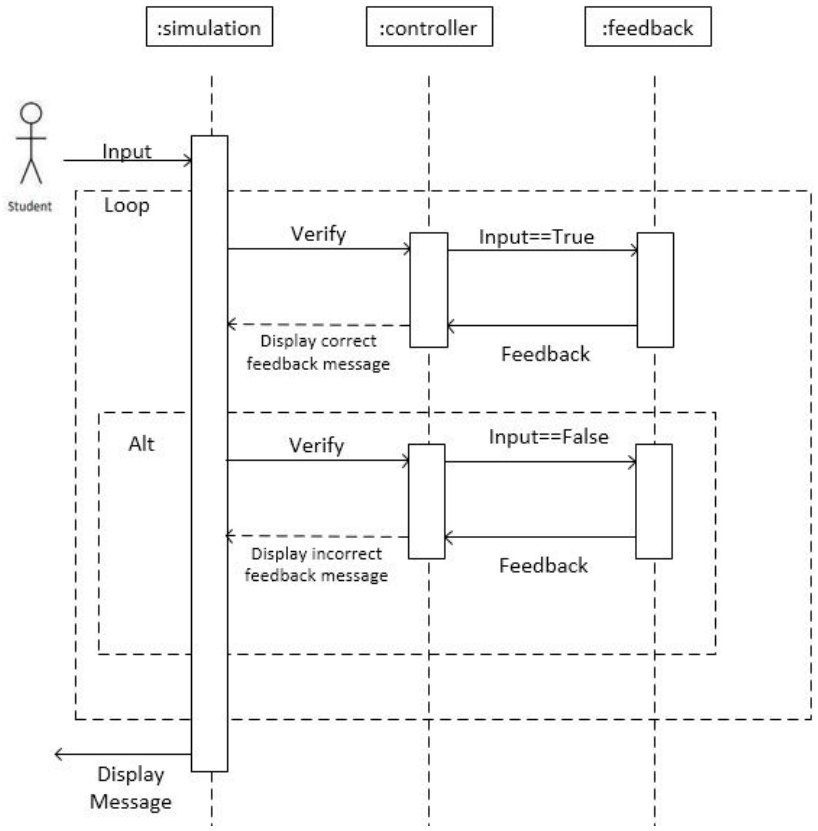


Figure 4: Sequence Diagram of UC7

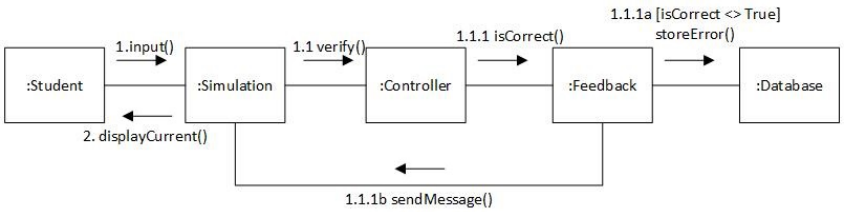


Figure 5: Communication Diagram of UC7

2.2.1 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
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.

2.3 UC1: Navigation Portal

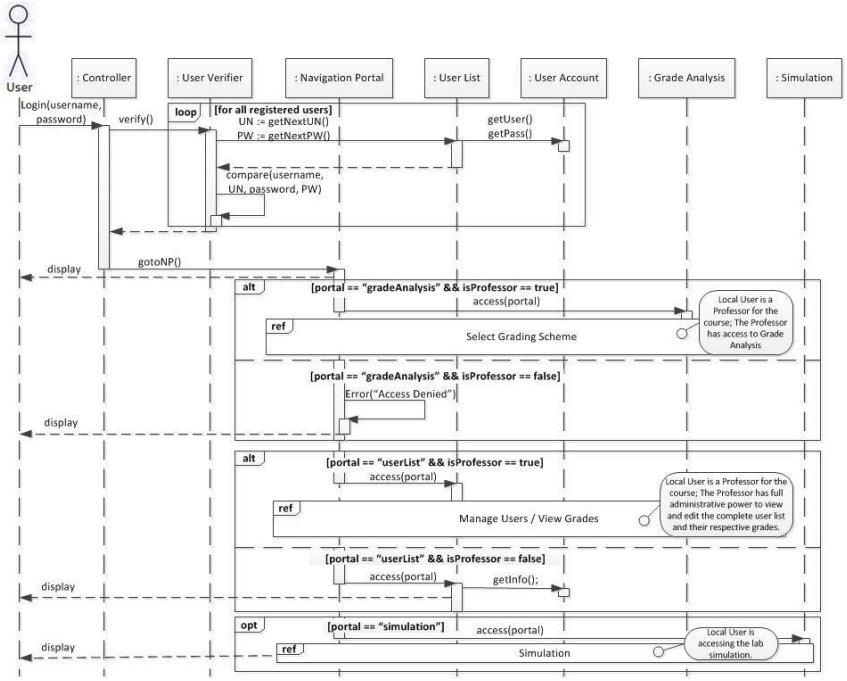


Figure 6: Sequence Diagram of UC1

2.3.1 Responsibility Description

Send message to Controller to validate whether the user is a student or a professor.
Send validation answer to User Verifier.

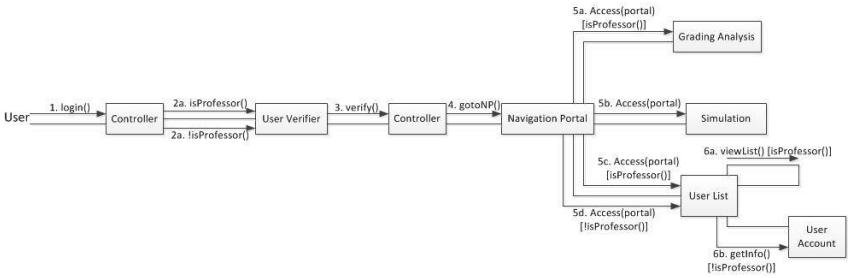


Figure 7: Communication Diagram of UC1

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)

3 Class Diagram and Interface Specification

3.1 Class Diagram

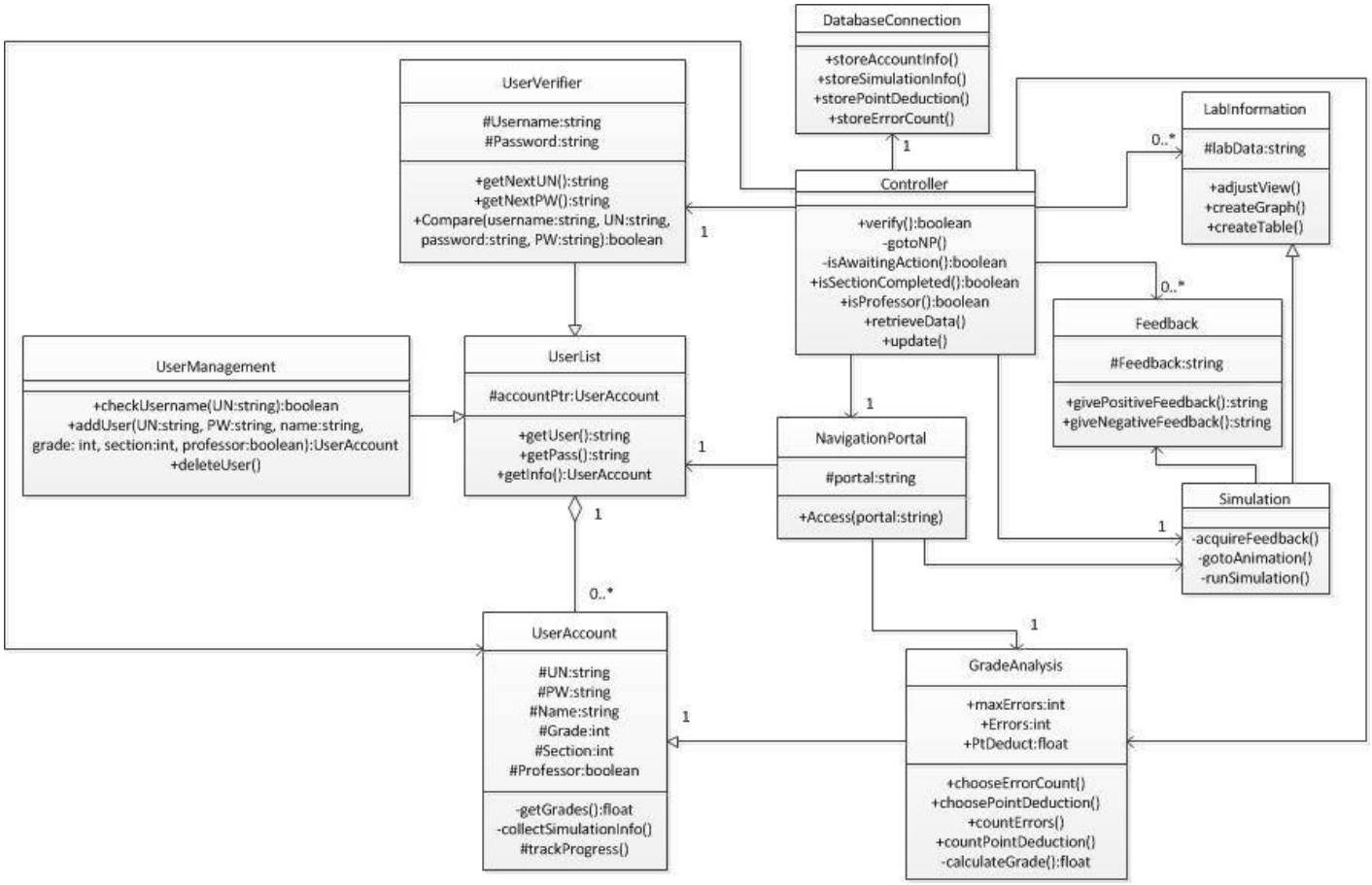


Figure 8: Class Diagram

3.2 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()

3.3 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

3.4 Traceability Matrix

Domain Concepts	Software Classes										
	NavigationPortal	Controller	DatabaseConnection	UserVerifier	UserManagement	UserList	GradeAnalysis	Simulation	Feedback	LabInformation	UserAccount
Navigation Portal	x										
Controller		x									
Database Connection			x								
User Verifier				x							
User Management					x						
User List						x					
Grade Analysis							x				
Simulation								x	x		
Lab Information										x	
User Account											x

Figure 9: 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.

4 System Architecture and System Design

4.1 Architectural Styles

Wikipedia[2] 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[3]) 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.

4.2 Identifying Subsystems

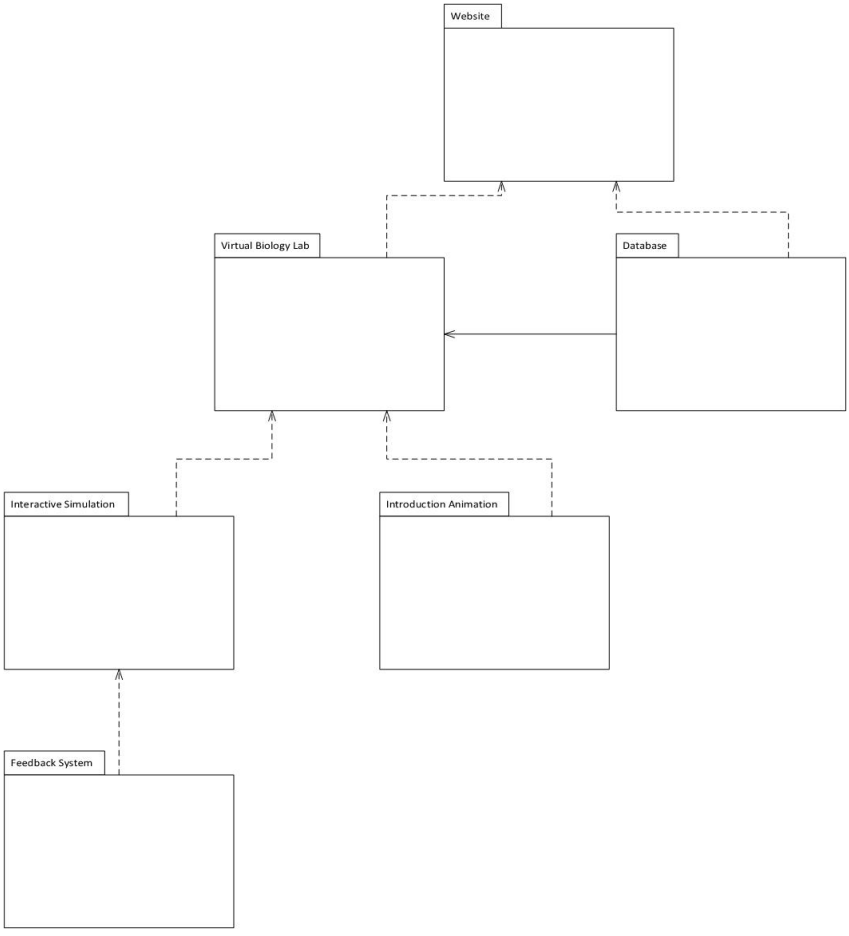


Figure 10: System Architecture: Subsystem Package Diagram

[2]http://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.2F_Patterns
[3]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.

4.3 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.

4.4 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.

4.5 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.

4.6 Global Control Flow

4.6.1 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.

4.6.2 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.

4.6.3 Concurrency

We do not have multiple threads running within our simulation.

4.7 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.

5 Algorithms and Data Structures

5.1 Algorithms

Because there are no heavy mathematical model that this project uses, and there are no complex methods of computation or physics calculations of graphics, this project does not use any complex algorithms, only simple ones like calculation of a grade, which is only simple subtraction.

However, as implementation of the project continues, the need for algorithms may come up, and we will use the appropriate ones if and when the time comes to use them.

5.2 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.

6 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.

7 Design of Tests

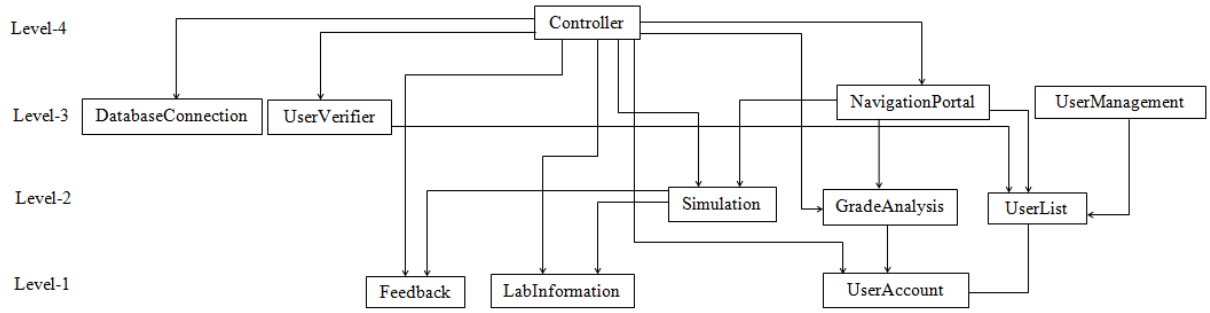


Figure 11: Units Hierarchy

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.

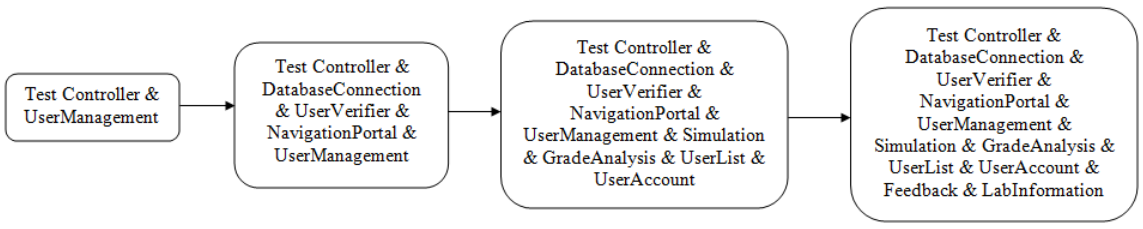


Figure 12: Top-Down Integration(based on Units Hierarchy)

7.1 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.

7.1.1 Controller Class

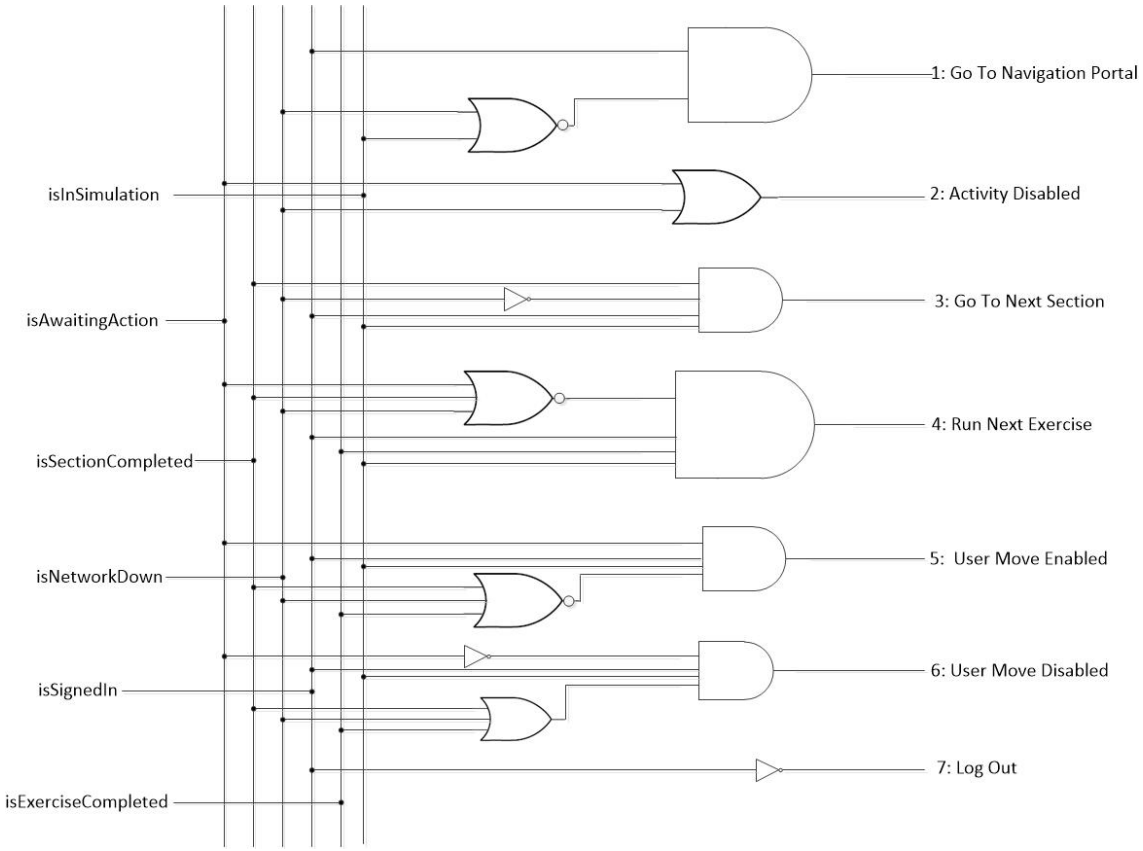


Figure 13: Controller Class

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.

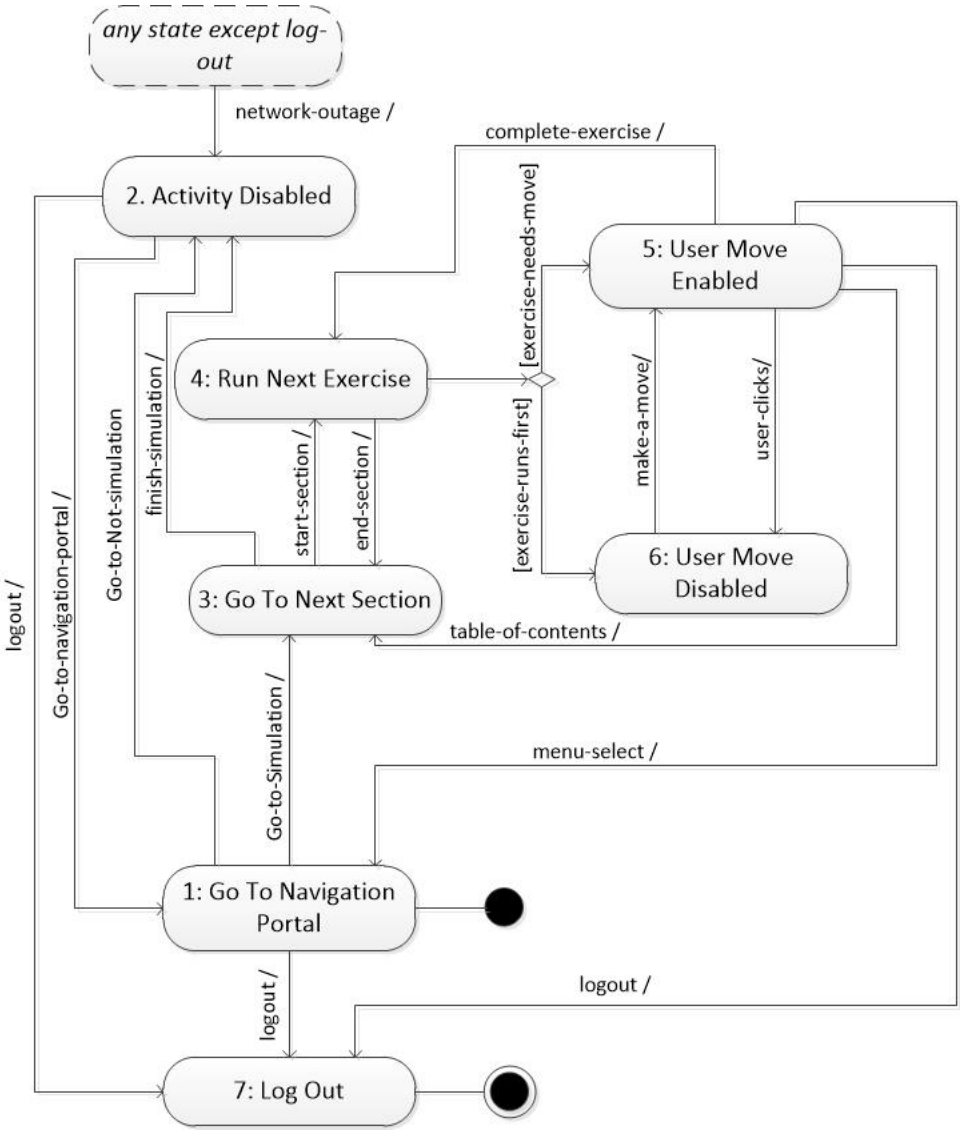


Figure 14: Controller Class State Diagram

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 15: Controller State Diagram Table

7.1.2 Navigation Portal

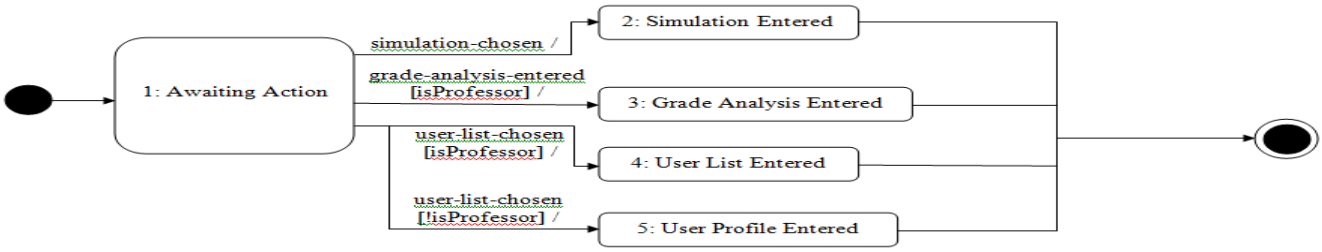


Figure 16: Navigation Portal State Diagram

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 17: Navigation Portal State Diagram Table

7.1.3 Simulation

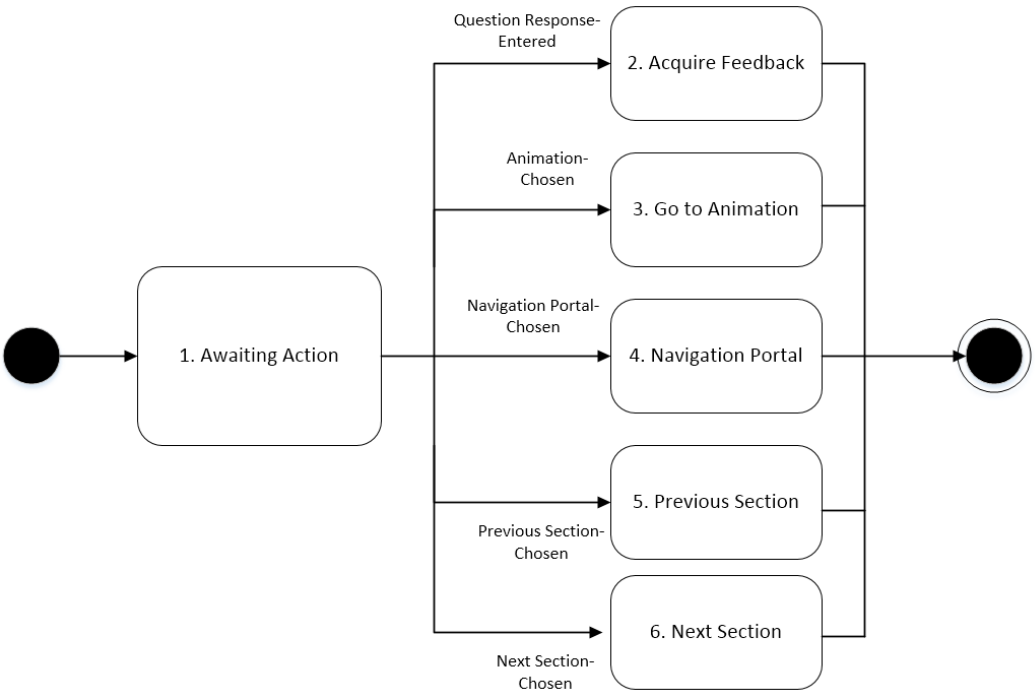


Figure 18: Simulation State Diagram

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 19: Simulation State Diagram Table

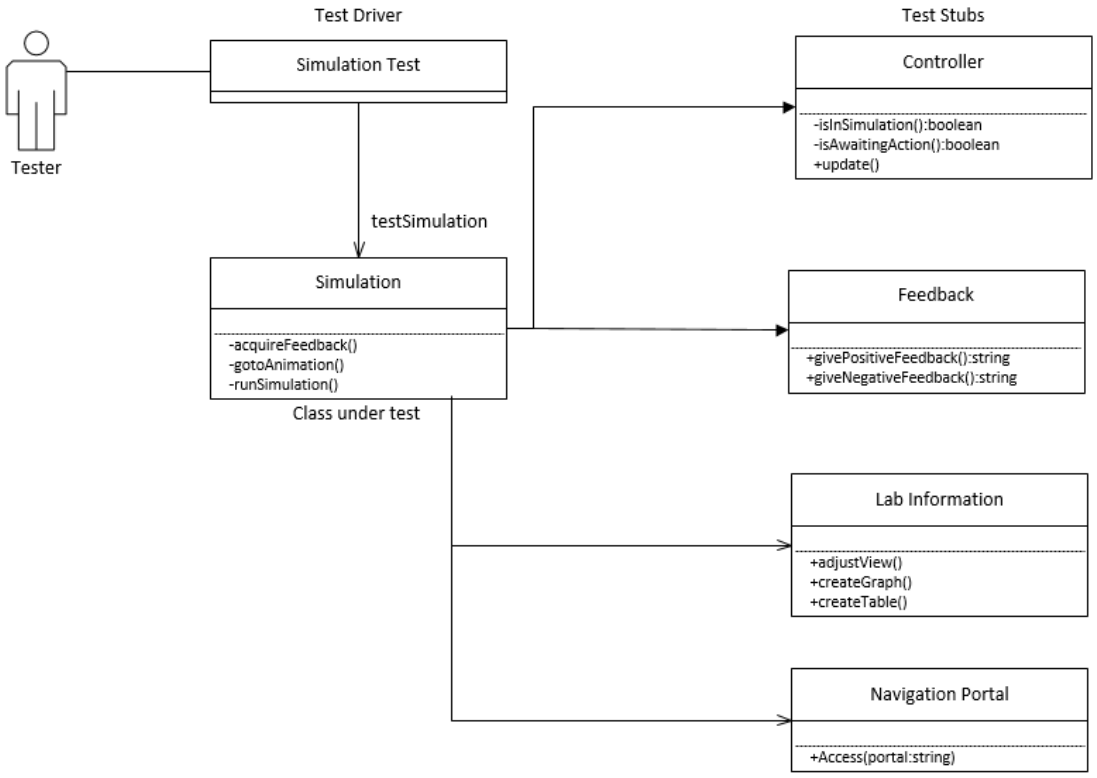


Figure 20: Simulation Class Test

7.1.4 Feedback

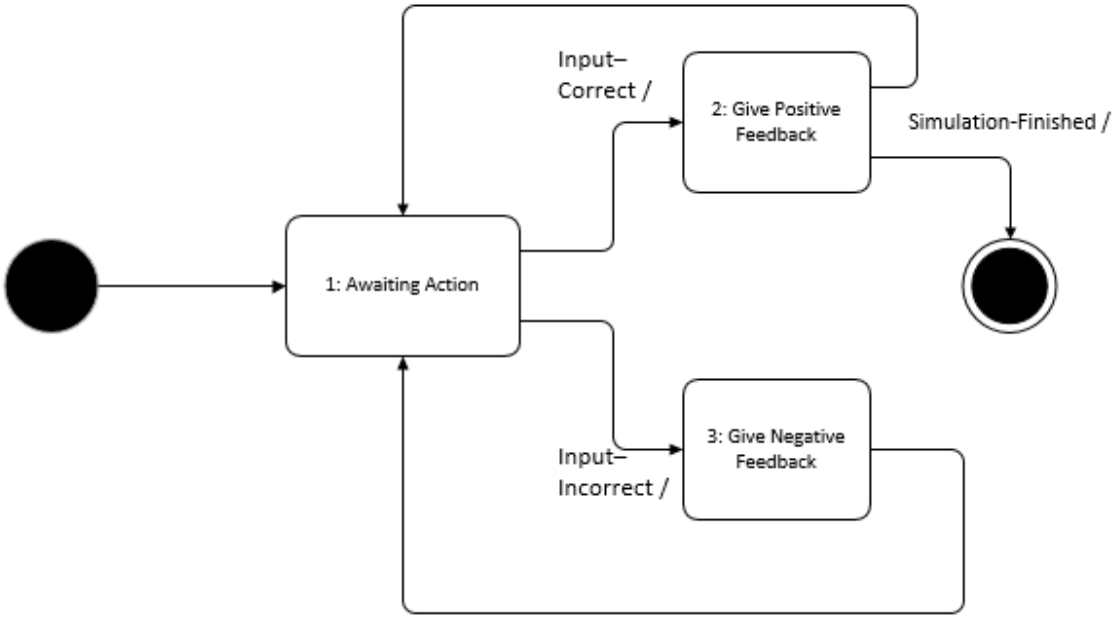


Figure 21: Feedback State Diagram

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 22: Feedback State Diagram Table

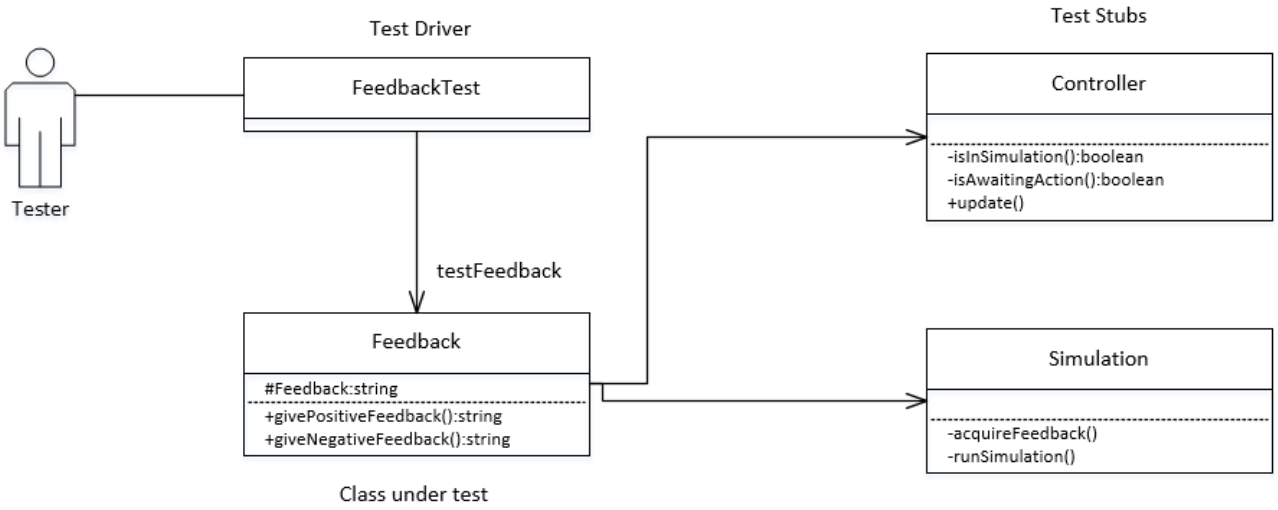


Figure 23: Feedback Class Test

7.2 Test Cases

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

Test Case Identifier: TC1 Use Case Tested: UC1 Main Success Scenario: System allows validated user to access the desired page 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 24: Test Case 1: TC1

Test Case Identifier: TC2 Use Case Tested: UC7 Main Success Scenario: System gives feedback to the user during the simulation 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 25: Test Case 2: TC2

Test Case Identifier: TC3 Use Case Tested: UC8 Main Success Scenario:User is a student and successfully runs the simulation 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 26: 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.

8 Plan of Work

Merging work: Complete
Compiling work: Complete

8.1 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.

8.2 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.

8.2.1 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.

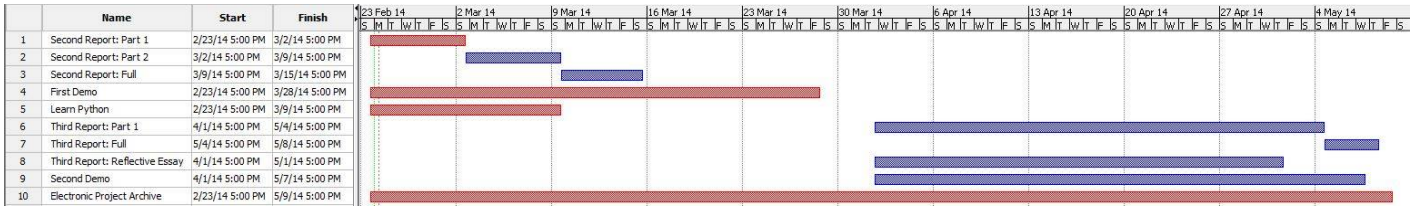
8.2.2 Already Functional Use Cases:

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.

8.2.3 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.

8.3 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 27: Gannt Timeline

8.4 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

9 References

9.1 Biology-Related

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

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

9.2 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