SOFTWARE ARCHITECTURE & MODELING

FOR

CLOCKRBOT

TEAM BLUE
PHUONG HO
CHANCE JOHNSON
JOSEPH KAMROWSKI
NIC LANTIER

MCNEESE STATE UNIVERSITY

WEDNESDAY, NOVEMBER 30TH, 2016

TABLE OF CONTENTS

UML CLASS DIAGRAMS

LoginPage
- Username : String - Password : int
<<constructor>> +LoginPage(Username : String, Password : int) + verifyUsername(); + verifyPassword();

SupervisorPage
+ disable(); + edit(); + clockGAout(); + undo(); + infoReport(); + printTime(); + viewReal-time(); + logout();

GAPage
+ clockin(); + clockout(); + logout(); + break();

FOUR VIEWS

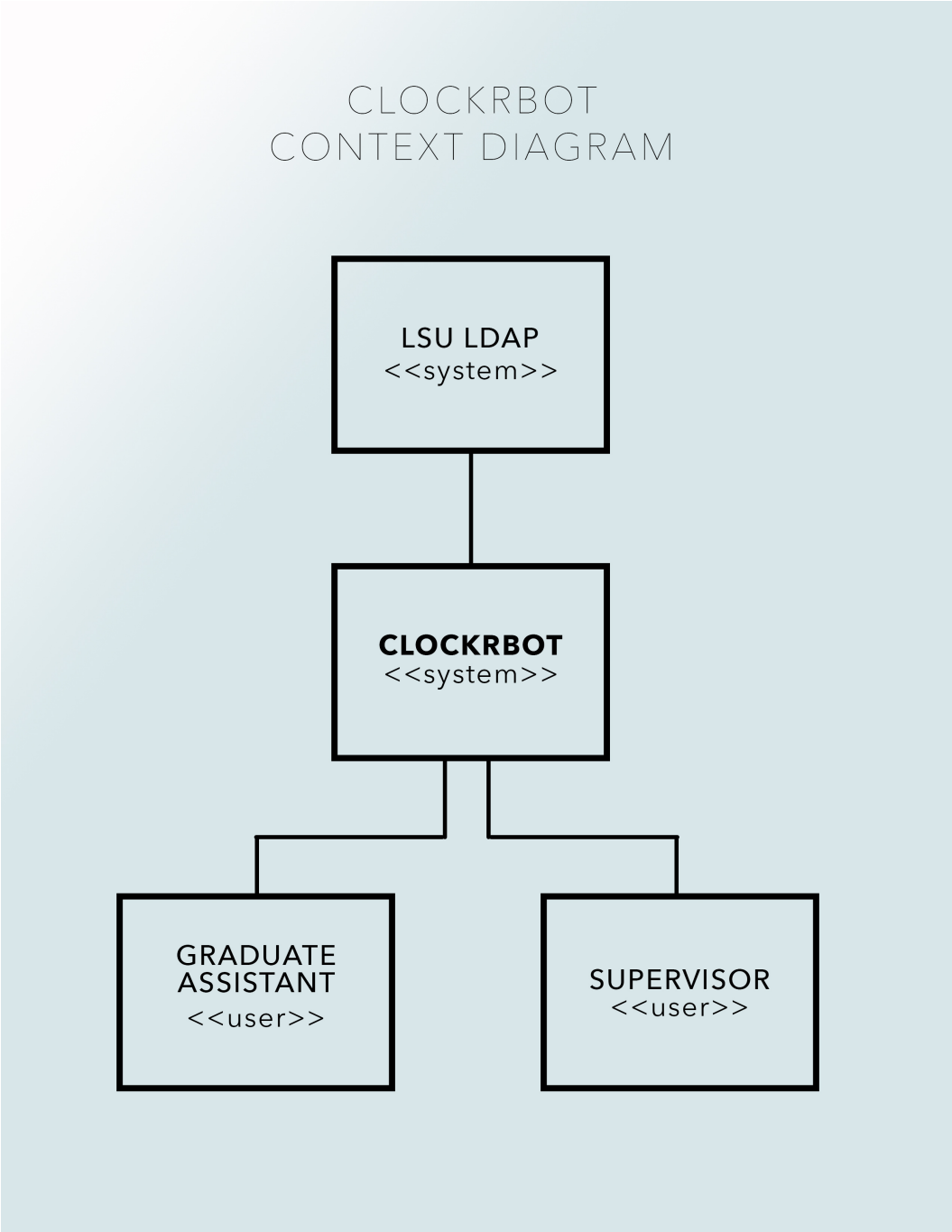
The logical view within the program shows what each class are going to be named as well as what the classes are going to be required to do within the program. The logical views the different functionalities within each classes within the program. Within our program, Team Blue would construct classes to classify each of the different web pages; the GA page, the Supervisor page, and the login page. Also, we would have to construct objects/classes within each main page class in order to decide on which page does what function. Due to knowing what objects/classes do for each main page class will clarify not only what each page should be capable of, but also on how the page could look. Logical view is used to plan on what can be implemented within the program as a whole by looking at the functions of each individual class.

The development view of the program shows on how the software can be used within the system. The development view is an example of programmers needing to be able to adapt their program with any software system from a company. For our program, Team Blue would have to see on what software the customer has and would implement the software to adept the program with the specific software.
This particular view is very useful when needing to know what type of software is necessary within the program to make it function properly. If a programmer does not look at the development view of a program/project, the program/project could give an error message and would not work properly and the programmer would not know what software it would work with.

The process view of the program shows how the processes within the program are functioning at run time; while the program is activated. While looking at the process view, the programmer is able to see which processes activate at each step of the program and see if the program is functioning properly. The view is very helpful for our project by seeing on which processes would work for each class during the starting time and running time of the program. With Team Blue's program having many processes interpreted within the program, we see on which process are working properly for each class of the program. If we see a process not functioning properly within the program, we see on which process is incorrect and immediately change the design of the process.

The physical view of the program is explaining how the system is put together in the system engineer's point of view. When asking how the system was made, you would ask the system engineer who made it and he/she should be able to explain every step and changes were made from his/her point of view. This also includes what system hardware was used within the program and implementation, however, the program Team Blue has designed does not include any hardware to implement. This view is very helpful because the programmers should be able to explain exactly what he did to the program and be able to show on how it correlates with the system. For the programmers to know exactly what software/hardware was implemented, they should be able to display accurate results of the program functioning properly.

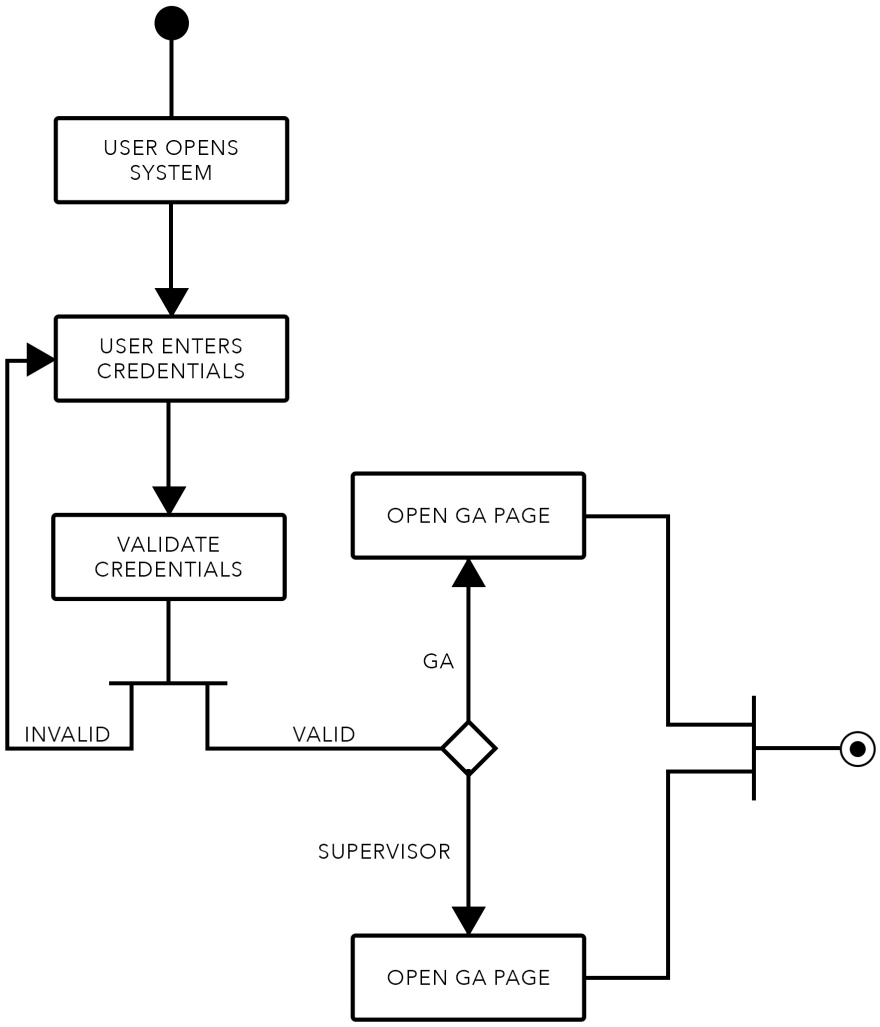
CONTEXT DIAGRAM



PROCESS MODELS

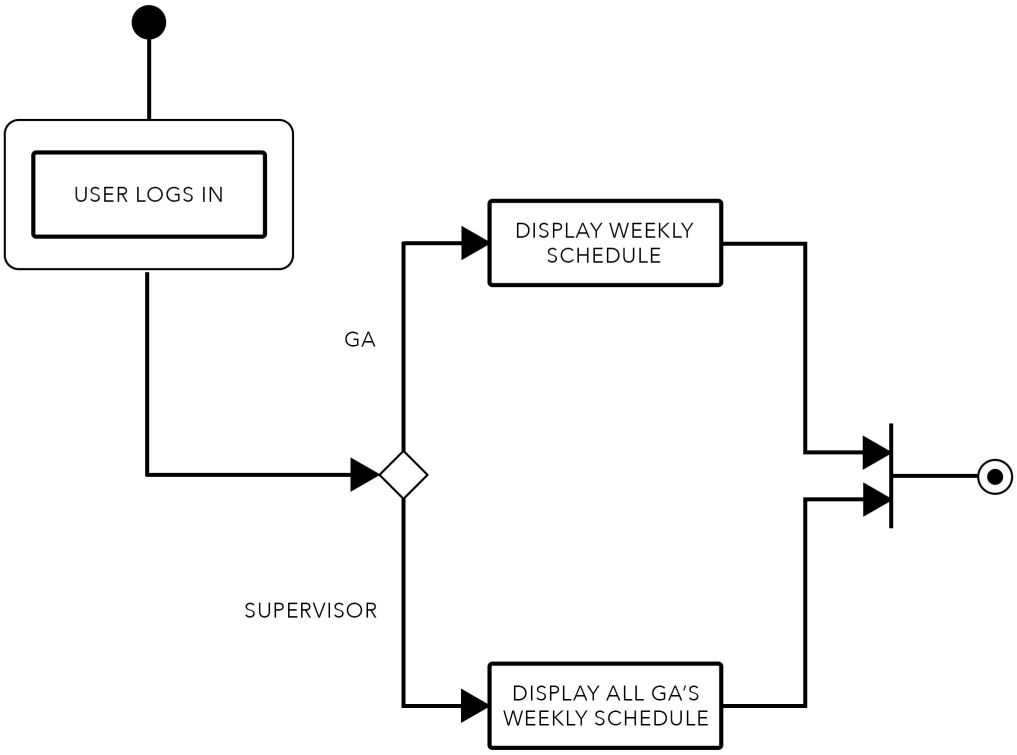
USER LOGS IN

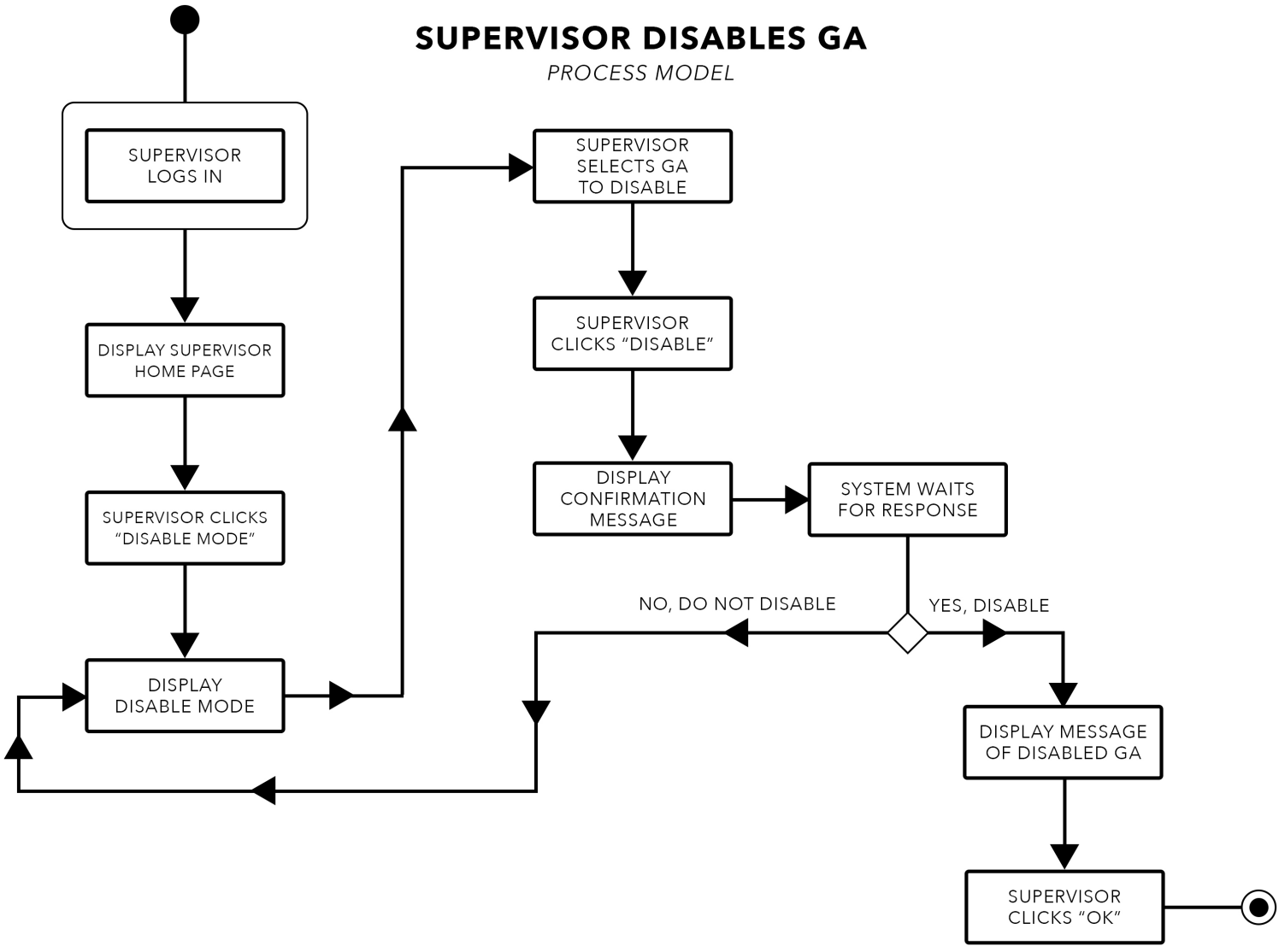
PROCESS MODEL



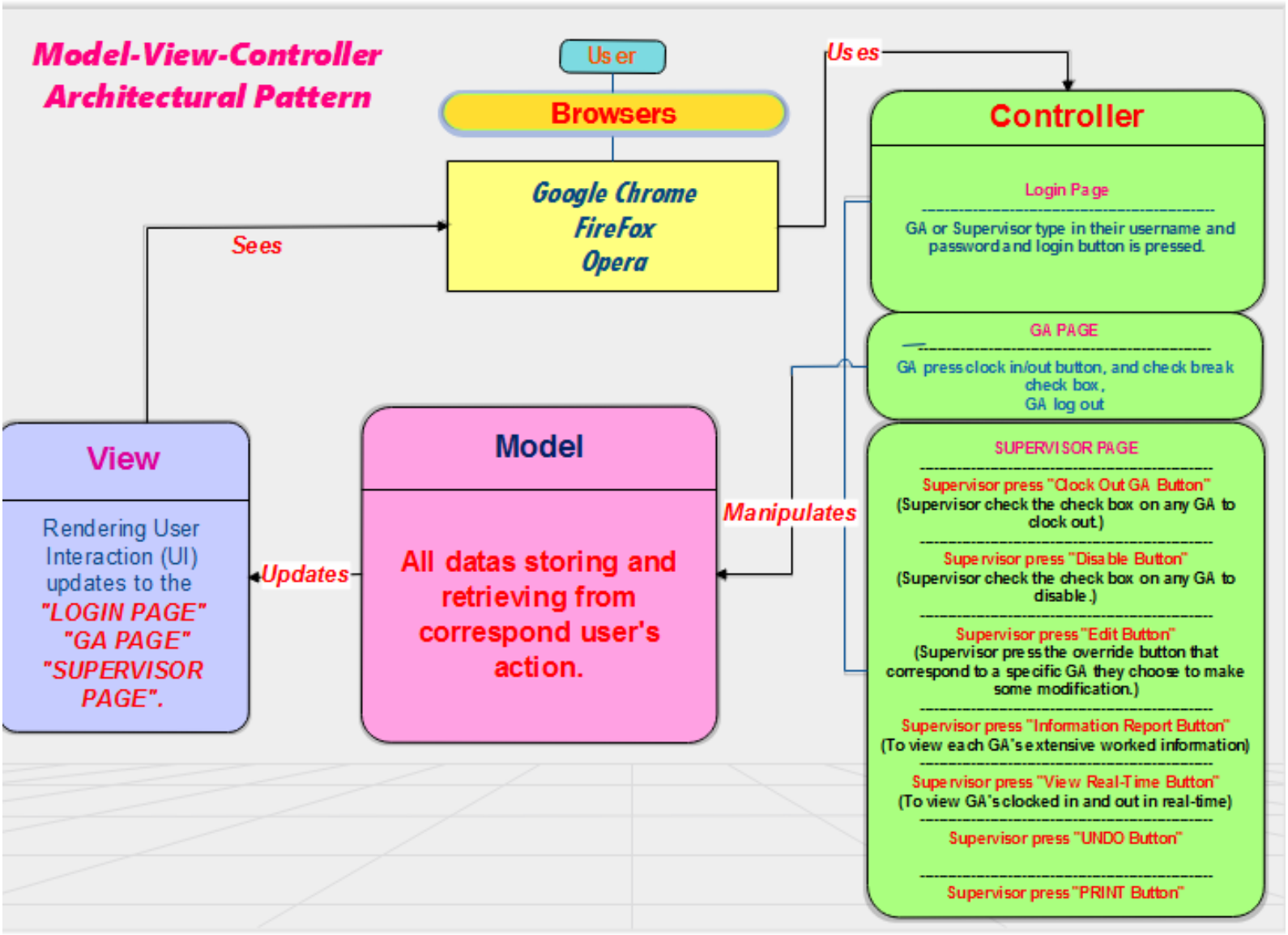
USER VIEWS SCHEDULE

PROCESS MODEL

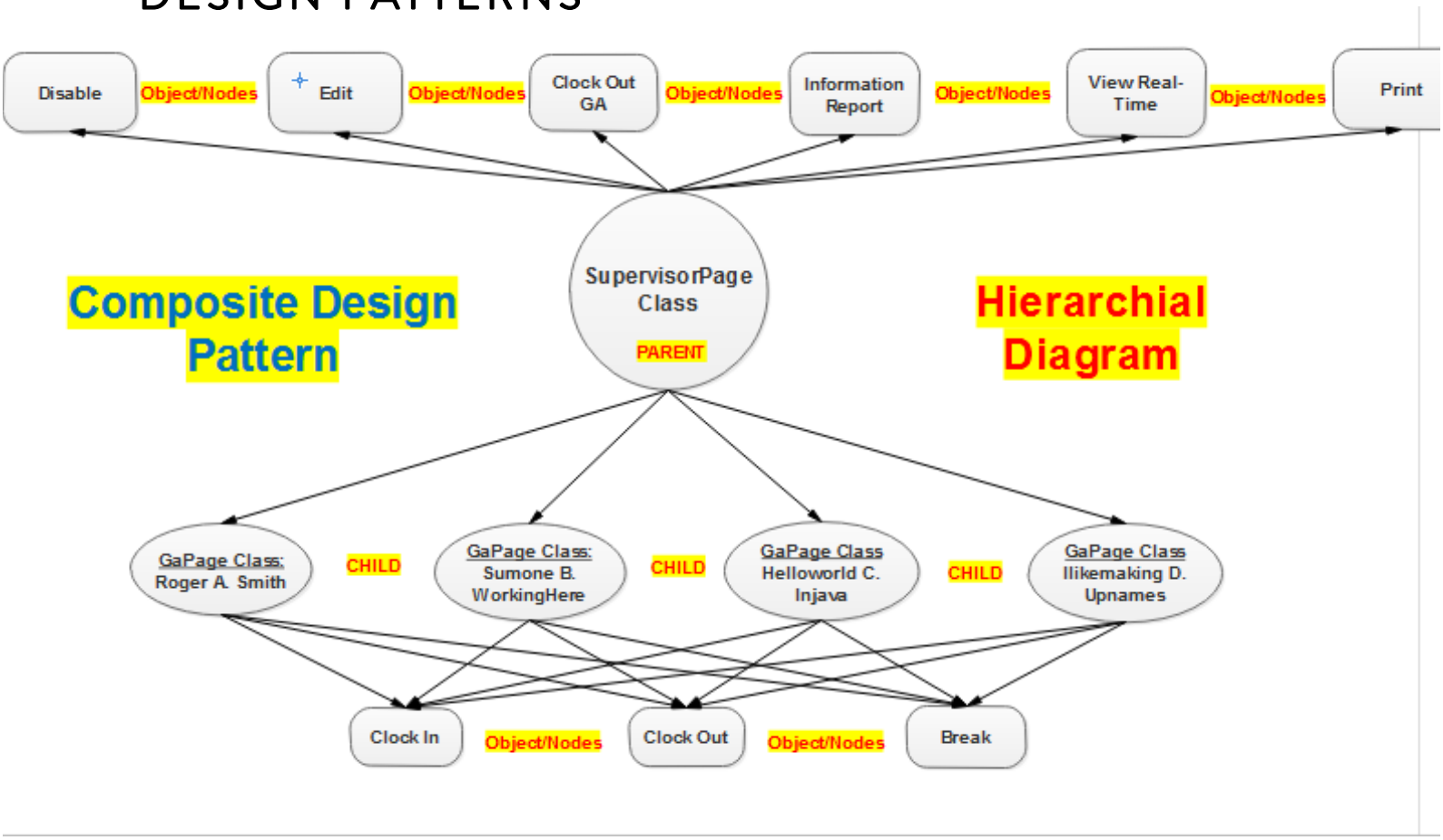



SUPERVISOR DISABLES GA
PROCESS MODEL
SUPERVISOR LOGS IN
DISPLAY SUPERVISOR HOME PAGE
SUPERVISOR CLICKS "DISABLE MODE"
DISPLAY DISABLE MODE
SUPERVISOR SELECTS GA TO DISABLE
SUPERVISOR CLICKS "DISABLE"
DISPLAY CONFIRMATION MESSAGE
SYSTEM WAITS FOR RESPONSE
NO, DO NOT DISABLE
YES, DISABLE
DISPLAY MESSAGE OF DISABLED GA
SUPERVISOR CLICKS "OK"

ARCHITECTURE PATTERN



DESIGN PATTERNS



Façade Design Pattern

One of the design patterns that will be very useful to use for the ClockrBot system is the Façade design pattern. The Façade pattern provides a single interface with the main purpose is to simplify a complex system. The way Façade pattern simplify a complex system is to break the whole system into each small subsystems that have its own responsible interaction sequence functions. We can refer each subsystems as classes, so in regard to that, the ClockrBot system will have three classes name LoginPage, GaPage, and the SupervisorPage. For the LoginPage class, the backend of it the username is declared as a string and the password is declared as an integer. Once the username and password is retrieved from the user, the verifyUsername() and verifyPassword() function perform its role by verifying the inputted data to determine if that input data is for the GaPage or SupervisorPage to direct to its appropriate class. If the username and password data is verified as Ga then the GaPage class will be activated. In the GaPage class, some of its particular function are clockIn(), clockOut(), break(), and logout().

The responsibility of the clockIn() function is to perform the action behind the scenes when the user clicks the clock in button, it should be able to retrieve that specific time of clock in data and display it to the GA user interface and also sends that data to the supervisor class so that it can be retrieve for the supervisor to obtain that data. The clockOut() and break() functions performs the same actions just like the clockIn() function that was just described. The logout() function performs the action by deactivated the Ga account, but it does not send that data to any other classes. So all those functions is pertain to the GaPage class. If the username and password data is verified as supervisor then the SupervisorPage class will be activated. In the SupervisorPage class, some of its particular function are disable(), edit(), clockGAOut(), undo(), infoReport(), viewRealTime(), printTime(), and logout(). The responsibility of the disable() function is to perform the action of disabling the GA's account by restricting to a particular GA from having access to the account. Once the Supervisor clicks on the disable button, from the backend the disable function will perform the action from the SupervisorPage class and sends that data to the GaPage class to take the effect on disabling it. The edit() function perform the action by sending the update changing data to the GaPage class to take the effect on the new changes that the supervisor has made. The clockGAOut() function, the action is done from the SupervisorPage class. Once the Supervisor manually clocks out the Ga the clockGAOut() function data gets sent to the GaPage class to take effect on the changes. The undo() function, data gets sent from the SupervisorPage class to the GaPage class back and forth based on whatever action the supervisor has taken. The infoReport(), viewRealTime(), printTime() functions those function pulls data from the GaPage class to retrieve the Ga's information to purposely display the data onto the supervisor interface. The logout() function performs the action by deactivated the supervisor account, but it does not send that data to any other classes. With all the backend information from the performance of each sub-systems that was describe from above, by using Facade design pattern the user will only experience with just a single interface of the system without having to know the backend complexity of each single steps about the system performance, the single interface will take care of the rest.

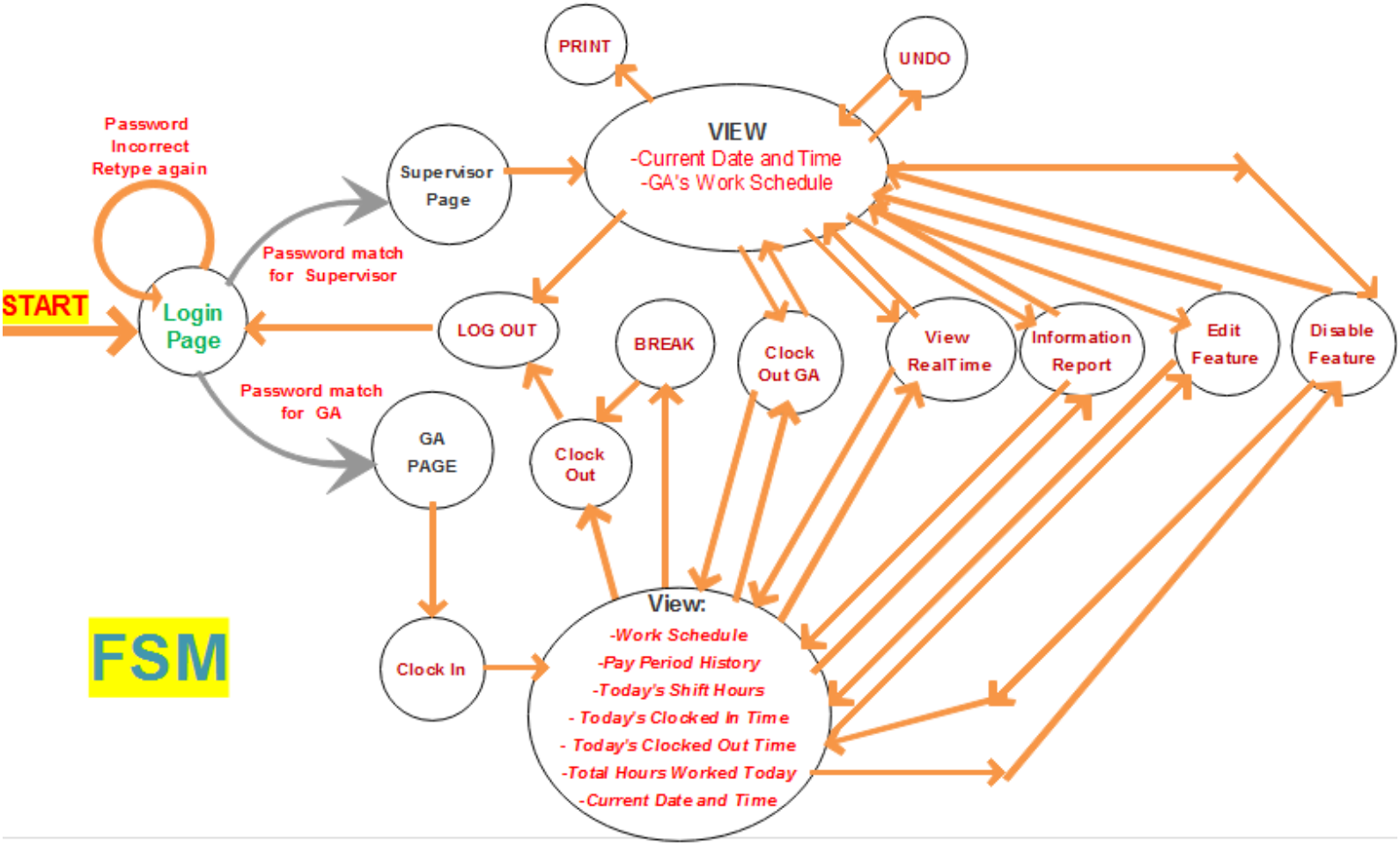
*Refer to the UML Diagram for the Façade Pattern Diagram

Composite Design Pattern

Another design pattern that will be very useful to use for the ClockrBot system is called the Composite design pattern. The Composite design pattern is a structural pattern which modifies the structure of an object. It is suitable for the need to work with objects which forms a hierarchy tree. The ClockrBot system will be a hierarchical structure that need a common functionality across its structure such as when the system deals with multiple GA's accounts that has all the same attributes functions. It can be consider that the supervisorPage class would be the parent managing multiple GA's accounts, which the GAPage class can be consider as many children. An example of hierarchy tree for this ClockrBot system would be the Supervisor page will be the parent, many GA's page will be the children, and many features both the Supervisor and GA's page has will be consider objects as many leaf nodes. The advantage of using the Composite design pattern is that it provide an organize hierarchical structure to the system in an efficient manner. The Component such as the SupervisorPage class has the ability to declare an interface for accessing and managing its child components, which refer to as the GaPage class, and defining the behavior for the primitive objects in the composition would refer to as the features the Supervisor and the GA's has in its own page.

*Refer to the diagram above for the Composite Design Pattern

FINITE STATE MACHINES



GA Description Tables

GA Page
This State is where you can visibly see the daily work schedule, the time(s) the GA clocked the current day, the total amount of hours worked today, and the work schedule for the current week. The buttons that are on the page are: the clock in, clock out, break, and logout buttons.

Clock In
This is one of the features on the GA page that is used for the GA's to confirm that they are at work.

Clock Out
This is one of the features on the GA page that is used for the GA's to confirm that they are not at work and had to leave for one of many reasons.

Break
This is one of the features used to let the supervisor know that the GA did take their break that day.

Log Out
This feature is used by the GA when they are ready to leave the lab and have already clocked out.

Disable Feature State
This state reads the data from the GA's View state and changes the data that corresponds to the name of this state and sends the data back to this state and then it sends the data to the supervisor's view state to view his or her action towards the selected GA's account.

LogOut State
Supervisor logs out of their account.

Login Page State
Supervisor enter password if incorrect he or she will have to reenter the password again. If the password is correct then the password data will get sent and allow supervisor to have access to their account.

Supervisor Page State
Supervisor can view the current date and time and the GA's Work Schedule retrieve from the View state.

View State
This is the main centralize state where all the data gets retrieve from many other states. This will have enough data for the print and undo features to work.

Print State
There is enough data gets retrieve from the View state to print the information out.

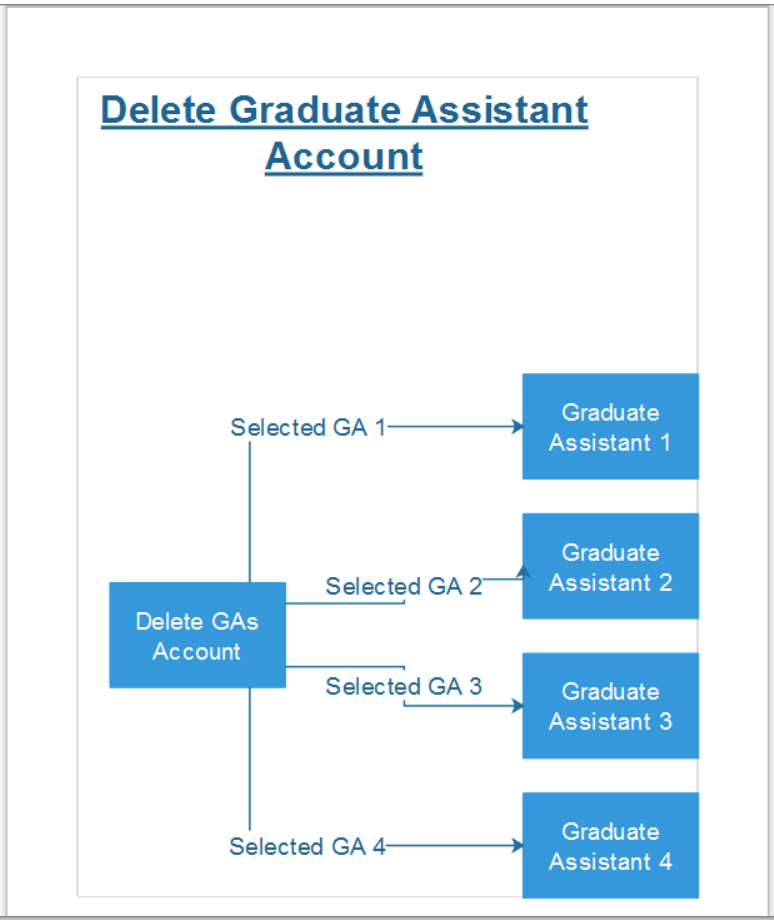
UNDO State
To undo any unwanted action that manipulate with the data.

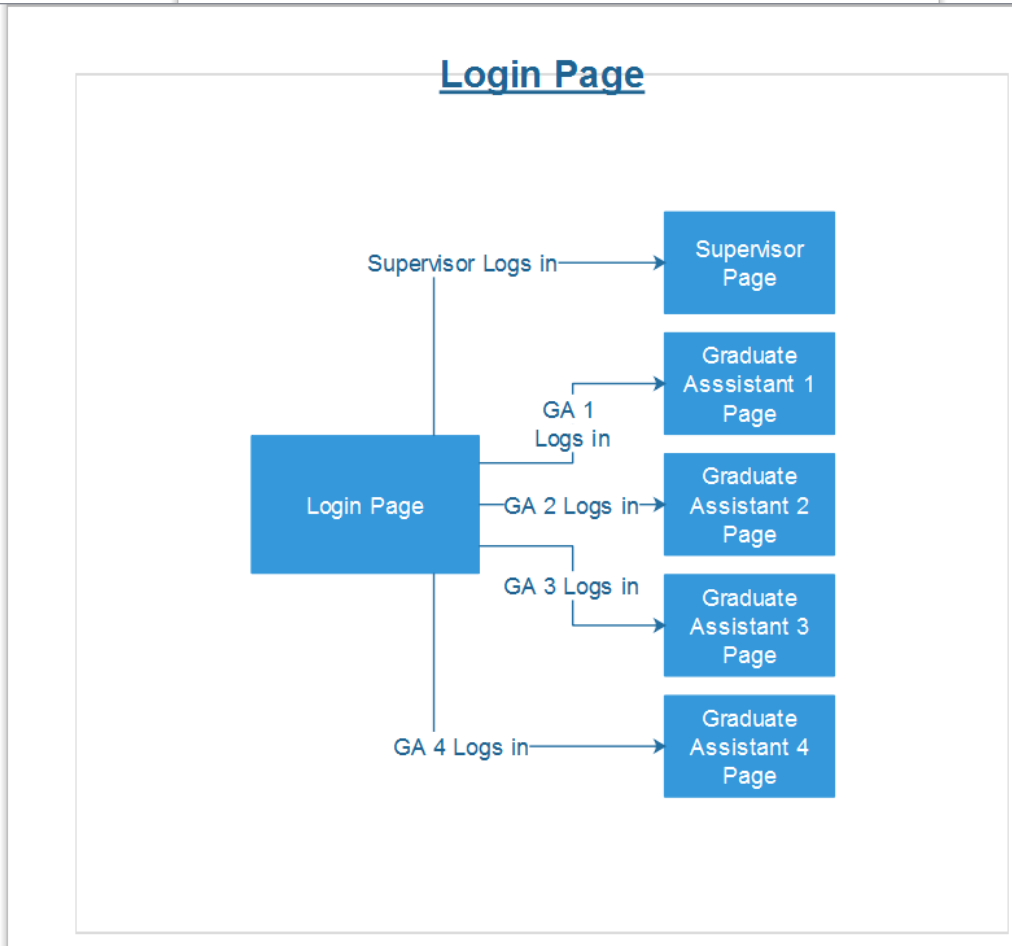
Clock Out GA State
This state reads the data from the GA's View state and changes the data that corresponds to the name of this state and sends the data back to this state and then it sends the data to the supervisor's view state.

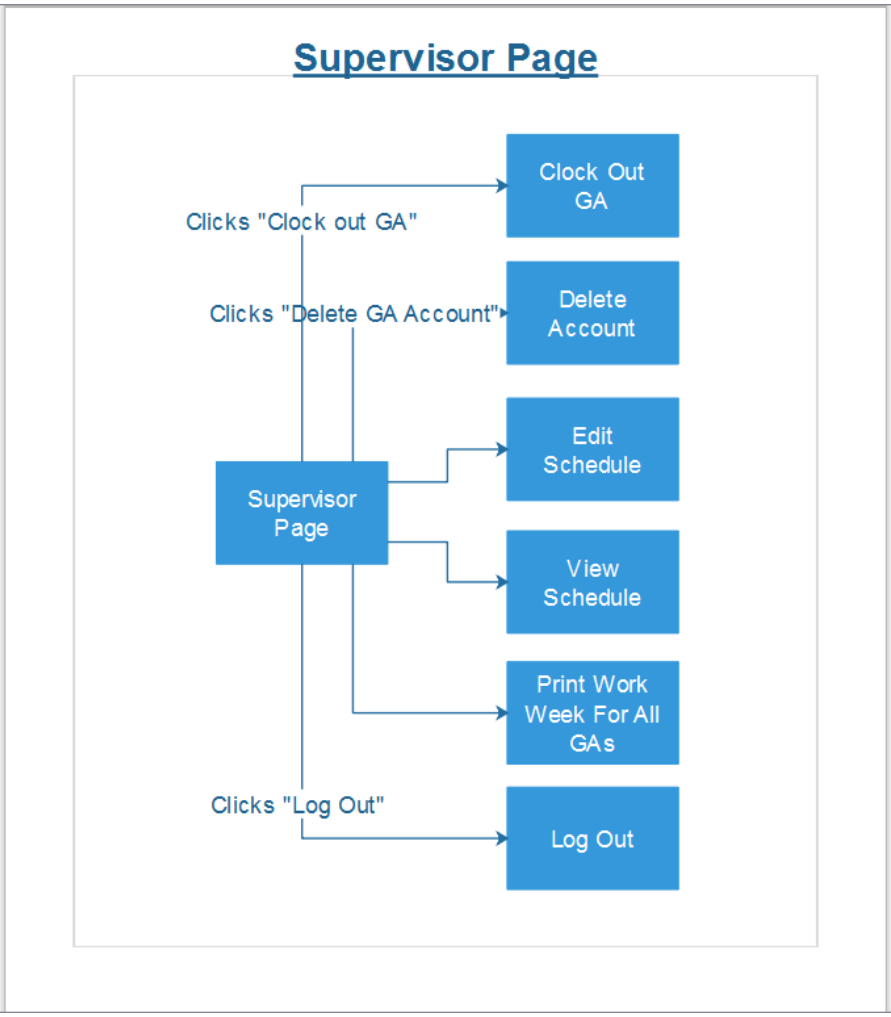
View RealTime State
This state reads the data from the GA's View state and sends the data back to this state and then it sends the data to the supervisor's view state to view real time data.

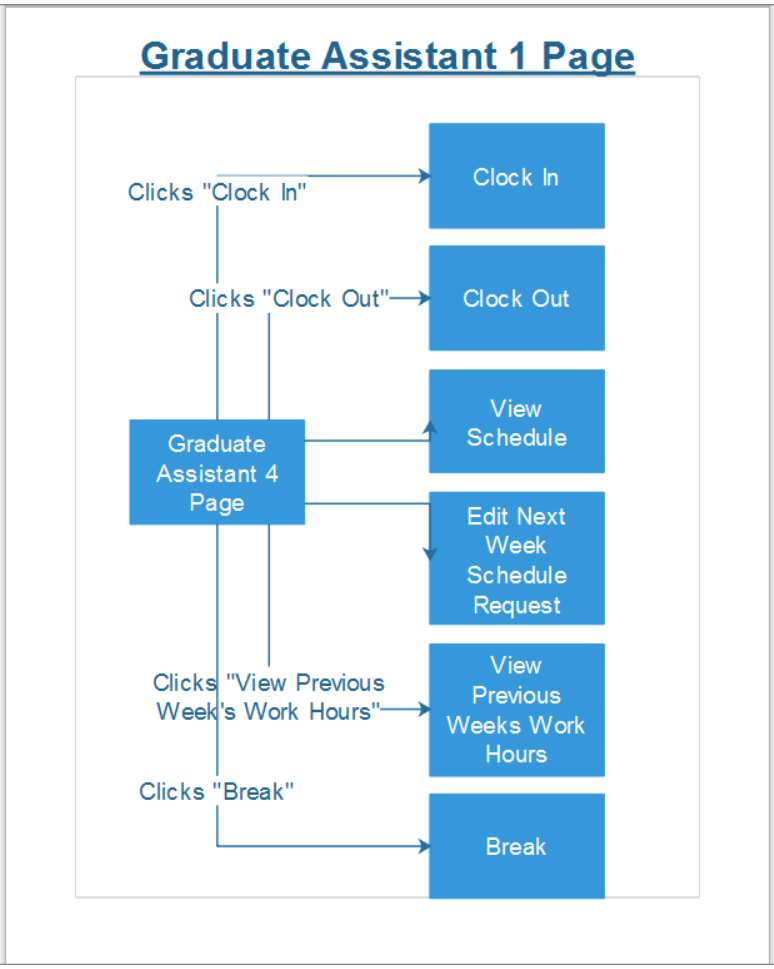
Information Report State
This state reads the data from the GA's View state and sends the data back to this state and then it sends the data to the supervisor's view state.

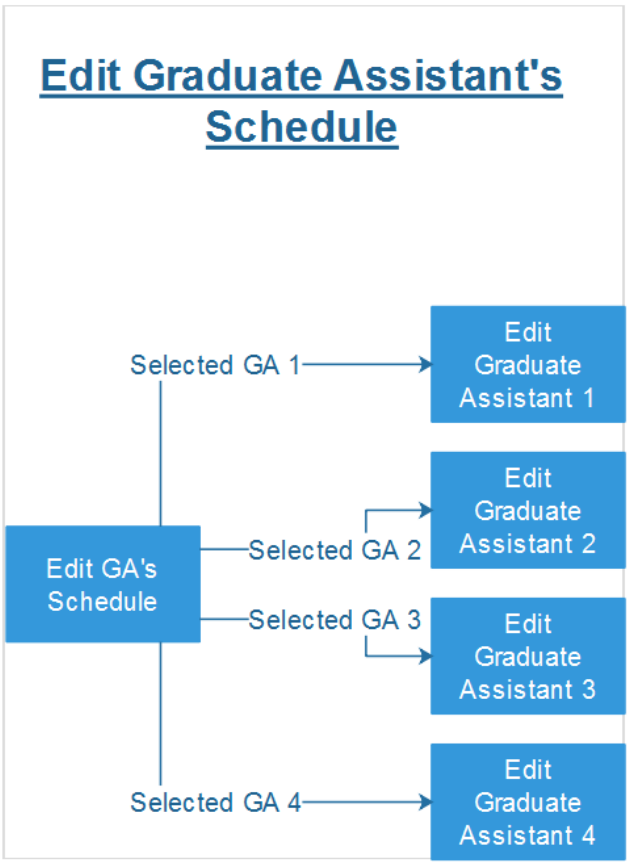
Edit Feature State
This state reads the data from the GA's View state and changes the data that corresponds to the name of this state and sends the data back to this state and then it sends the data to the supervisor's view state to let the supervisor notice the changes he or she made towards the selected GA's account.

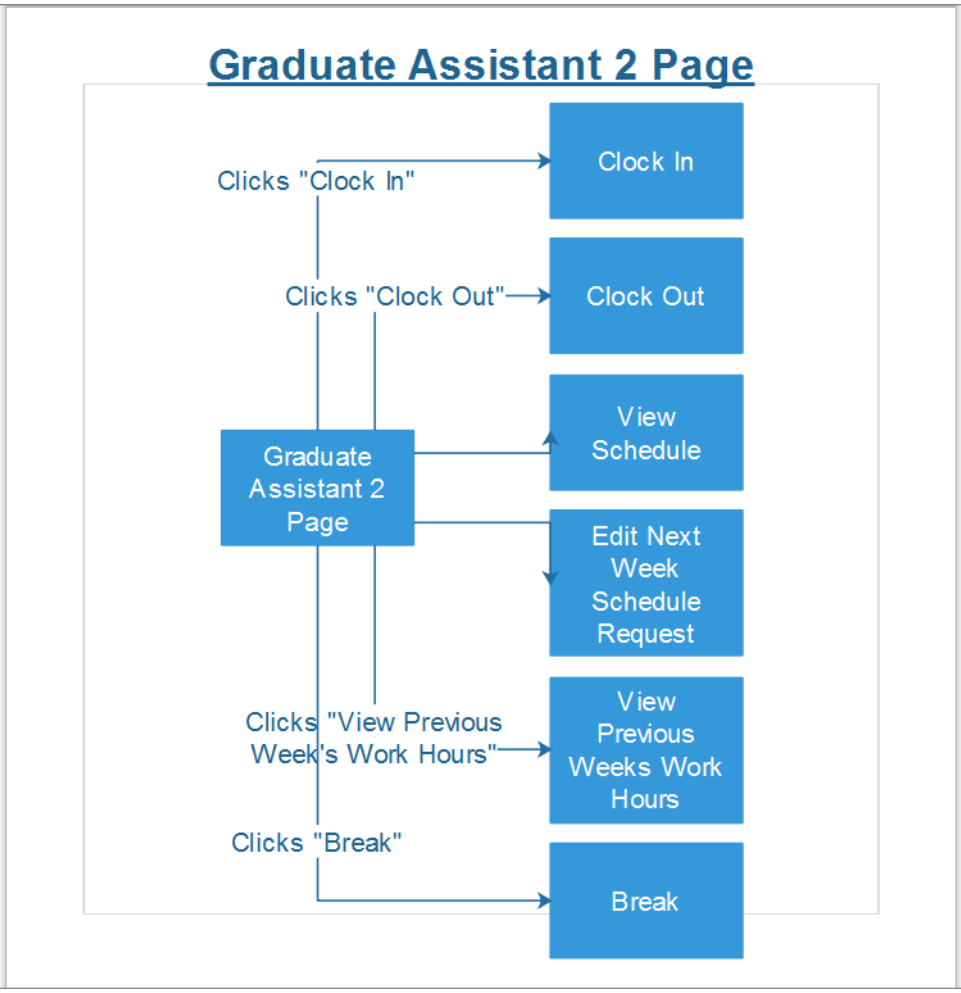

Delete Graduate Assistant Account
Delete GAs Account
Selected GA 1
Selected GA 2
Selected GA 3
Selected GA 4
Graduate Assistant 1
Graduate Assistant 2
Graduate Assistant 3
Graduate Assistant 4

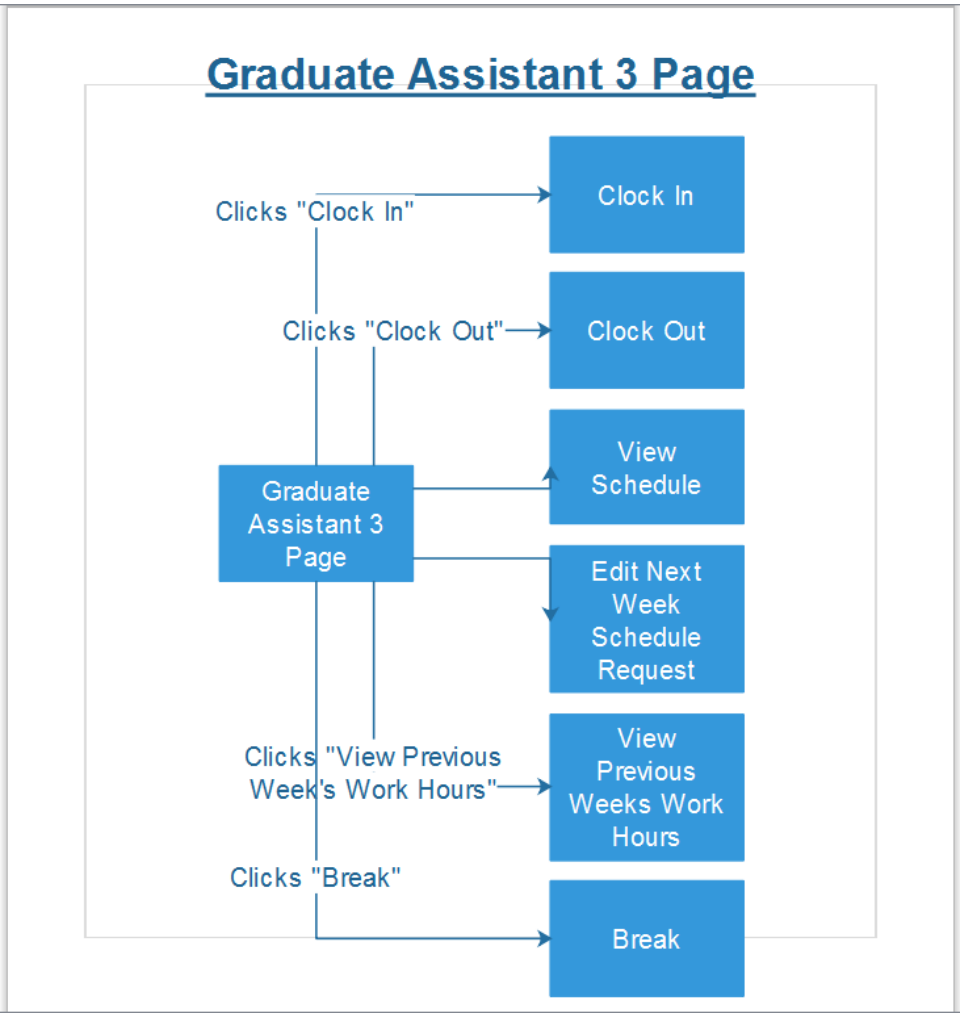

Login Page
Login Page
Supervisor Logs in
GA 1 Logs in
GA 2 Logs in
GA 3 Logs in
GA 4 Logs in
Supervisor Page
Graduate Asssistant 1 Page
Graduate Assistant 2 Page
Graduate Assistant 3 Page
Graduate Assistant 4 Page

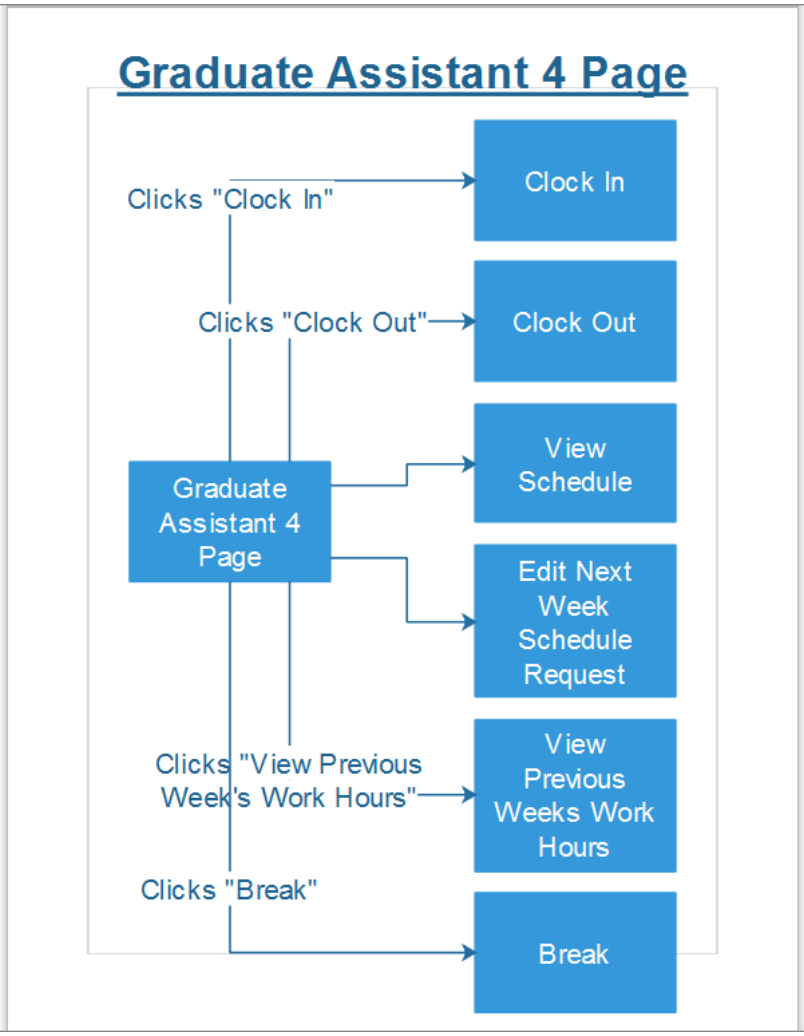

Supervisor Page
Clicks "Clock out GA"
Clicks "Delete GA Account"
Clicks "Log Out"
Supervisor Page
Clock Out GA
Delete Account
Edit Schedule
View Schedule
Print Work Week For All GAs
Log Out

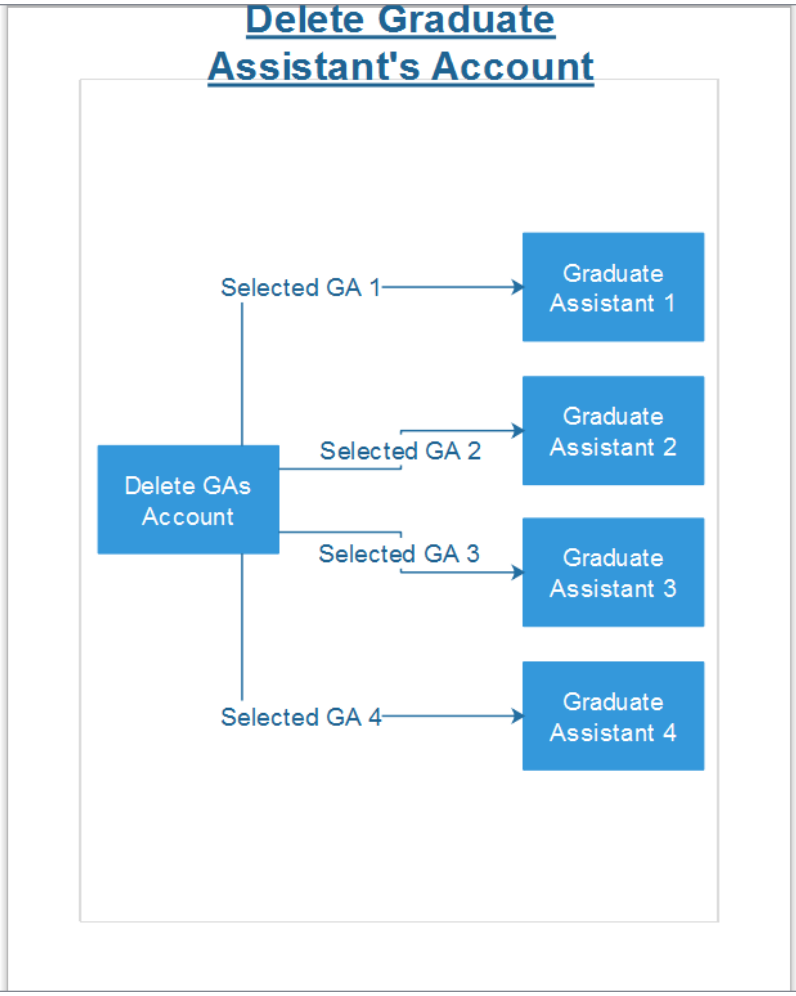

Graduate Assistant 1 Page
Clicks "Clock In"
Clock In
Clicks "Clock Out"
Clock Out
Graduate Assistant 4 Page
View Schedule
Edit Next Week Schedule Request
Clicks "View Previous Week's Work Hours"
View Previous Weeks Work Hours
Clicks "Break"
Break

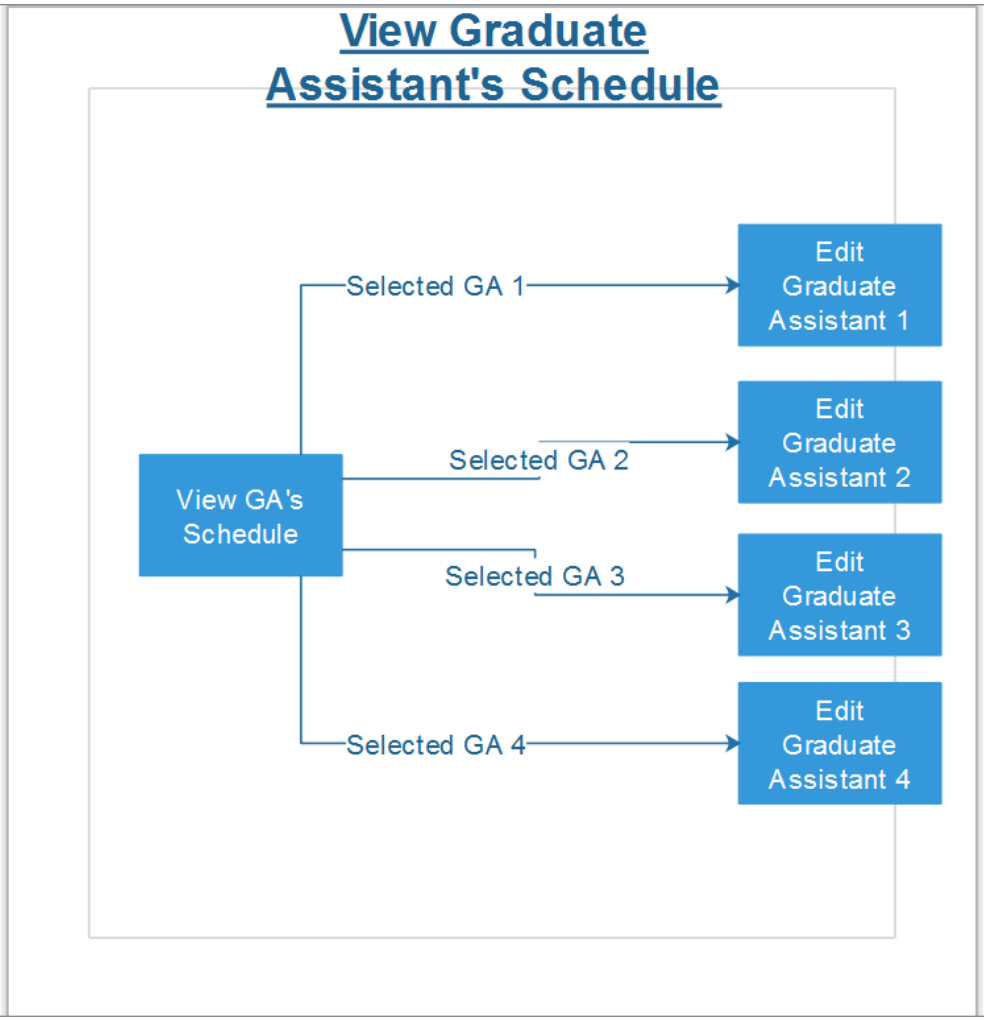

Edit Graduate Assistant's Schedule
Edit GA's Schedule
Selected GA 1
Selected GA 2
Selected GA 3
Selected GA 4
Edit Graduate Assistant 1
Edit Graduate Assistant 2
Edit Graduate Assistant 3
Edit Graduate Assistant 4


Graduate Assistant 2 Page
Clicks "Clock In"
Clock In
Clicks "Clock Out"
Clock Out
Graduate Assistant 2 Page
View Schedule
Edit Next Week Schedule Request
Clicks "View Previous Week's Work Hours"
View Previous Weeks Work Hours
Clicks "Break"
Break

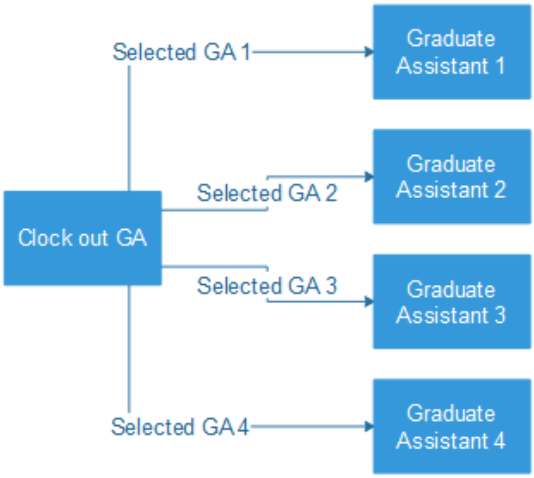

Graduate Assistant 3 Page
Clicks "Clock In"
Clock In
Clicks "Clock Out"
Clock Out
View Schedule
Graduate Assistant 3 Page
Edit Next Week Schedule Request
Clicks "View Previous Week's Work Hours"
View Previous Weeks Work Hours
Clicks "Break"
Break


Graduate Assistant 4 Page
Clicks "Clock In"
Clock In
Clicks "Clock Out"
Clock Out
Graduate Assistant 4 Page
View Schedule
Edit Next Week Schedule Request
Clicks "View Previous Week's Work Hours"
View Previous Weeks Work Hours
Clicks "Break"
Break


Delete Graduate Assistant's Account
Delete GAs Account
Selected GA 1
Selected GA 2
Selected GA 3
Selected GA 4
Graduate Assistant 1
Graduate Assistant 2
Graduate Assistant 3
Graduate Assistant 4


View Graduate Assistant's Schedule
View GA's Schedule
Selected GA 1
Selected GA 2
Selected GA 3
Selected GA 4
Edit Graduate Assistant 1
Edit Graduate Assistant 2
Edit Graduate Assistant 3
Edit Graduate Assistant 4

Clock Out Graduate Assistant



REFLECTION AND ANALYSIS

What we learned from this process

Throughout the process we learned new things about collaboration as well as how system modeling and architectural design plays in to software engineering. We had to learn how to effectively communicate and relay feedback amongst team members. We also learned how to adapt to unexpected changes. These unexpected changes required us to develop new versions of our models and architectures. We also learned how to manage task distribution in order to ensure no team member is bogged down. Finally, we have continued to learn about the extensiveness of software engineering and how much thought, planning, and designing actually goes in to developing a piece of software.

The software we used

We used a variety of software to accomplish our individual tasks and diagrams: Microsoft Word, EDraw Max, Adobe Photoshop

Was the software difficult to learn?

None of the software was difficult to learn. We were already familiar with Microsoft Word, and EDraw Max is very similar to Word. We also used Photoshop because we thought it would be quicker to use a software we were already comfortable with rather than learn a new one.

How did we like the software?

EDraw Max is a very user-friendly, very stable, easy to learn software. Anyone could learn how to use this simple tool. Microsoft Word is much like EDraw Max. Photoshop is a very powerful software that can do many things, but definitely takes awhile to master.

The most difficult models to create for this project

Drawing the model to represent the Composite design pattern was the more difficult model to create.

The easiest models to create for this project

The context diagram and the UML Class diagrams were the easiest models to create for this project.

Do these models help with implementing a system like this?

Yes. These models provide an adequate starting point for the implementation of this system. The models do a good job of showing the theory behind the system as well as the organization in order to develop a successfully working system.

The most helpful models during implementation

The Model View Controller pattern really helps give a visual blueprint of how the system functions. The context model is useful for realizing what this system interacts with.

The least helpful models during implementation

The context model would be the least helpful model for implementation. It doesn't really help with the actual development of the system.

How hard was it to use Git?

Git was not very hard to use. With more experience, it will become even easier to use.

Would we use Git again?

We would definitely use Git again. It is powerful and comes in handy when managing a large project with a team.

Did version control help or did it get in the way?

The benefit of using version control was seen with the ease of tracking each person's progress and seeing the contributions and updates to each task. It was very beneficial to be able to see who committed what at what time and allow everyone in the project to be able to view it.