Clock in Clock out System

Kennedy Janto[#1], Alex Gomez[#2], Duc Nguyen[#3], Taylor Vandenberg[#4], Janista Gitbumrungsin[#5]

[#]*Department of Computer Science, California State Polytechnic University, Pomona*
3801 W Temple Avenue, Pomona, CA 91768, United States

[1]kjanto@cpp.edu

[2]alexandrog@cpp.edu

[3]ducquocanhn@cpp.edu

[4]tdvandenberg@cpp.edu

[5]janistag@cpp.edu

***Abstract*— We are going to update the clock in-clock out system, and transition to an online system. The system should include log in, log out, clock in, clock out, edit shift, and edit employee functionalities. We are going to build the system using JavaFX, Java Swing, and SQL. This paper will discuss the methodology behind the analysis and design of the clock in-clock out system. It will explore topics for future development and general findings.**

***Index Terms*— employee, online, analysis, Java, SQL**

I. Introduction

A clock in clock out system allows employees to clock in and out from work as well as take lunch breaks. The goal of this system is to add functionalities to add, delete, edit shift, and edit employee functions.

A. Problem Description

The problem with the old system of clock in, clock out is the inefficiency of using physical objects. The old system required the use of paper or cards which consumed time, space, and resources to store and use. If a card is lost, then it must be replaced with a new one which is inefficient as well as a waste of resources. Additionally, editing current time shifts must be done on a seperate system. Old system is inefficient due to employees often needing to wait in line to use it.

B. Proposed Solution

We will implement a system that utilizes a database to store employee clock in and clock out data. Our system will also store and manage employee data in the online database. All data and changes will also be logged to an archive database. In this system, there is no need for a physical object to clock in or clock out. Employees simply enter their employee ID to login and click on the buttons to clock in and clock out. There will also be buttons for employees to take lunch breaks. Managers can also clock in and clock out like employees, but also have access to additional functionalities such as add shift, remove shift, edit shift, add employee, remove employee, and edit employee. Time recorded in the system is precise to the second. The benefits of this system is a faster clock in and clock out process for users. It also provides a more efficient storage for employee data and creates reliable data to calculate paychecks due to the system being based on the time an employee used the time clock function rather than handwritten data. The efficiency of the new system is superior to the original system because it utilizes simple passwords and buttons to make use of its functions. The new system eliminates the possibility of damaging or losing a physical employee card, as well as errors that might be caused as a card is worn down. Managers will more efficiently have the ability to edit employee data as well as their work hours.

II. Analysis

A. System Requirements

During the analysis of the clock in clock out system, there were several key use cases identified as well as the need to reserve functions for particular types of users. The main requirement of the system is to allow employees to quickly clock in or out for work. The system must implement security measures so that employees cannot clock in for someone else. The login window should use a unique employee ID to identify different employees. This login function is also where the system differentiates between a regular employee and an employee with manager status. Once logged in, employees should be sent to the main window where they can clock in or out. This window should present only a few buttons to make it simple for users to locate the button related to their desired use case. There will be a clock in, clock out, start break, and end break button. These will be large and easy for employees to interact with. Employees typically keep track of when they began their shift so they can estimate when to take a break or what time their shift ends. This is why there should also be a time clock that displays the current time on the bottom of the window to show users when they began their shift. This information will provide employees the ease of knowing when they clocked in instead of checking on their watch or phone. What makes this system more efficient than the old system is the added ability of controlling data within the system. At times, employees may forget to clock in or clock out. In this case, a manager needs to fix the shift data in the system which our system can achieve by utilizing the edit shift function. Once a manager logs into the system, they will see the option to add, edit or delete shifts. This will be another window within the system that will display all shifts in the database. Managers will be able to select a certain shift by entering the employee ID and shift date. Once selected, they can delete the shift by clicking the delete button. If the manager wants to edit the shift, then

they click the edit button which will prompt a few text boxes to fill in and update the shift info. Adding a shift is similar to the edit function where the manager fills in the info in the text box and then clicks add. These functions will also be available to the employee data in the database. When an employee quits or a new employee is hired, managers must also make relevant changes in the database. All these requirements will be presented in a UI that is neatly laid out with easy to find functions for users.

B. Actors and Stakeholders

After further analysis, we identified a few stakeholders that fall under Internal, External, Executive, and Operational. The internal executive stakeholders are the Board of Directors, Senior Management, and Finance Department. They will access the data from the system for other usage but will not directly use the clock in and out function. The internal operational stakeholders are Operational Management and Employees. These are the ones who will use the system directly to clock in and out of work. Lasty, the external executive stakeholders are Regulators, Finance Department, and Partner Organization. They will help keep the system in use but will have no direct relationship or usage of the system.

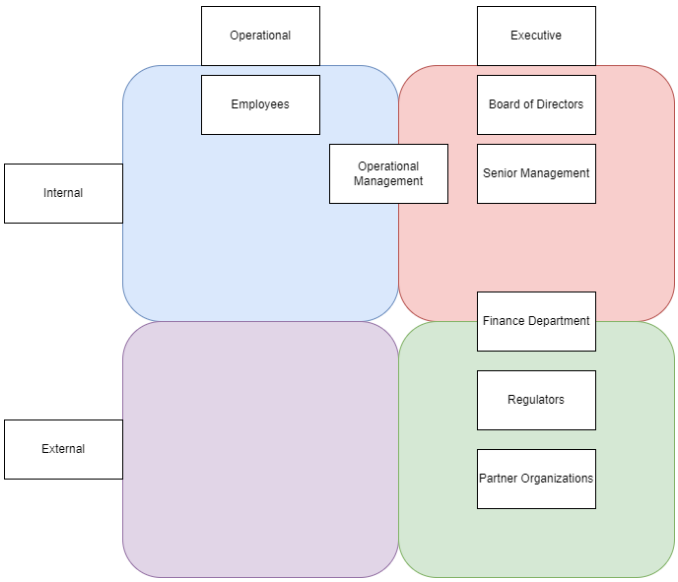

Fig. 1. Stakeholder diagram

C. Analysis Model

After finishing the analysis of the actors in our Clock in clock out system, we need to draw out an analysis model for our system. The analysis model contains all the necessary components to implement into our system. It assists in visualizing the system in addition to viewing it through raw code. Our system has two different actors which are employees and managers. The analysis model will help us distinguish which functions to implement for which actor and what the relationship between them is. Referring back to this while writing the program helps the team stay on track working towards the original vision rather than steering off into other directions.

D. Use Cases

Use Case	**Brief Use Case Description**
Clock In	Employee comes into work and enters his or her pin into the system. System will confirm the clock in and store the exact time at the moment into the database.
Clock Out	Employee leaves work and enters his or her pin into the system. System will confirm the clock out and store the exact time at the moment into the database as well as total time since last clock in.
Break Start	Employee enters the pin into the system and activates the break option which pauses their shift timer in the system.
Break End	Employee enters pin into system and deactivates their break which resumes their shift timer in the system.
Edit Shift Data	Manager enters the pin and selects the edit shift data option which shows them a list of employees that can be selected and have their shift data edited such as hours.
Add new employee	Manager enters the pin and selects add employee option which creates a new employee in the system and assigns a pin.
Remove employee	Manager enters the pin and selects the employee where they can select an employee to remove from the system.

Fig. 2. Use Case Diagram

The use cases are derived from the system requirements. All the functions required in the system are listed as a use case and at least one of the actors is responsible for each use case. The primary functions, clock in and clock out, are seen as two separate use cases. Break start and break end are also two separate use cases. These four use cases are controlled by the four buttons on the time clock window. They all grab the current system time and record it into the database. Edit Shift, add employee, and remove employee are use cases reserved for employees with manager status. These use cases will have control over the data in the database.

E.

Use case name:	Clock in	
Scenario:	Clock in while logged in.	
Triggering event:	Employee wants to start their shift.	
Brief description	Employee logs in with their pin, and clicks the clock in button.	
Actors:	Employee, Manager	
Related use cases:		
Stakeholders:	Employees, Operational Management, Senior Management, Finance Department	
Preconditions:	Employee PIN is valid.	
Postconditions:	Employee is logged in. Employee's clock in time is saved in database.	
Flow of activities:	Actor	System
	1. Employee enters their pin.	1.1 System checks for employee in database. 1.2 System displays manager home screen if employee is a manager. 1.3 System displays employee home screen in employee is not a manager.
	2. Employee clicks the "Clock In" button.	2.1 System verifies employee is not already clocked in. 2.2 System stores clock in time in database. 2.3 System displays confirmation notification.
Exception conditions:	1.1 Employee does not exist 2.1 Employee is already clocked in	

Fig. 3 Detailed Use Case Diagram for Clock in

For the detailed use case diagram, we provided an in-depth look into the Clock In use case. This use case functions when the user clicks the clock in button from the main screen after logging in. It only works when a correct employee ID is entered in the login screen or else this button is not available. The triggering event of this use case is when the employee wants to start their shift. The actors for this use case are employees and managers. Since managers are employees, they will be using the same function to clock in to work. The flow of activity is the system will check for the employee in the database when the employee enters their ID. When the system verifies that the login was correct, it will pop up the main screen which will show the four buttons including the clock in button. Once the button is clicked, the current time is recorded into the database as the start work time. The employee should then start work. This use case would not work if the employee ID entered is incorrect or if the user has already clocked in.

F. Domain Model Class Diagram

After doing all the analysis of the different use cases involved with the system, we needed to draw a domain model diagram. A domain model diagram shows all the different classes and attributes but does not include the data types or any other program specific information. This is beneficial because it serves as a blueprint without showing unnecessary information that is related to the code. We had many changes to our diagram from the original to the finish. As we were implementing the program, we cut down on what information we needed to keep track of and use in the system.

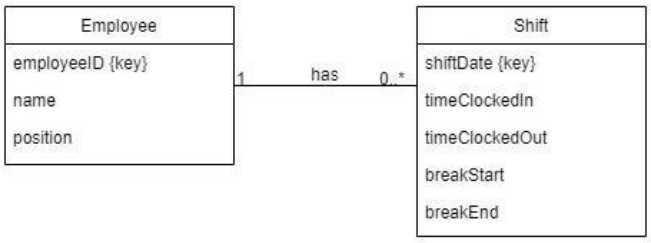

Fig. 4. Domain Model Class Diagram

Our domain model diagram consists of two classes which are employee and shift. Employee contains attributes of name, position, and employeeID which is the key attribute. The shift class contains timeClockedIn, timeClockedOut, breakStart, breakEnd, and shiftDate which is the key attribute. The relationship between the two classes is one and only one employee has zero or many shifts.

G. Activity Diagram

This activity diagram visualizes the typical day use of the system by an employee. This includes the process of clocking in, taking break, finishing break, and clocking out from work. This diagram shows the overall flow of activity and what information is being passed back and forth from actor to system.

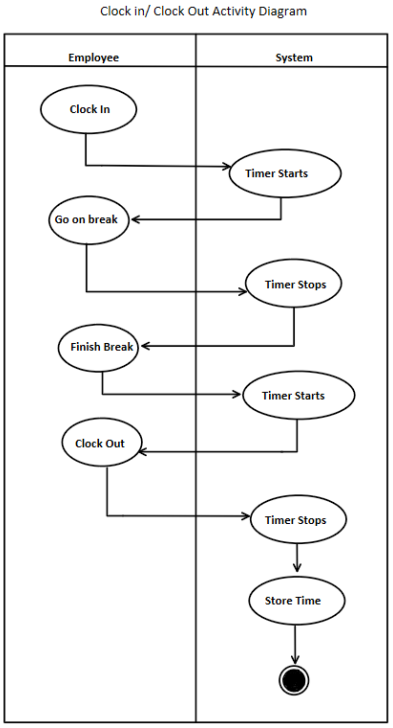

Fig. 5. Activity Diagram for Clock in/ Clock out

When an employee starts off work, they will log in to the system and clock in. This triggers the system to collect the current time and store it into the database. When the employee goes on break, they will go back to this system and press the button to start break. The system will again collect the current time. The system will repeat the same process when the

employee clicks finish break and clock out. Once the employee clocks out, the system will collect all the data and update it to the database as a new entry to the shift data table.

H. System Sequence Diagram

A system sequence diagram goes through the detail in the flow of activity for a use case. It shows what information is passed between the user and the system. This system sequence diagram illustrates the clock in use case.

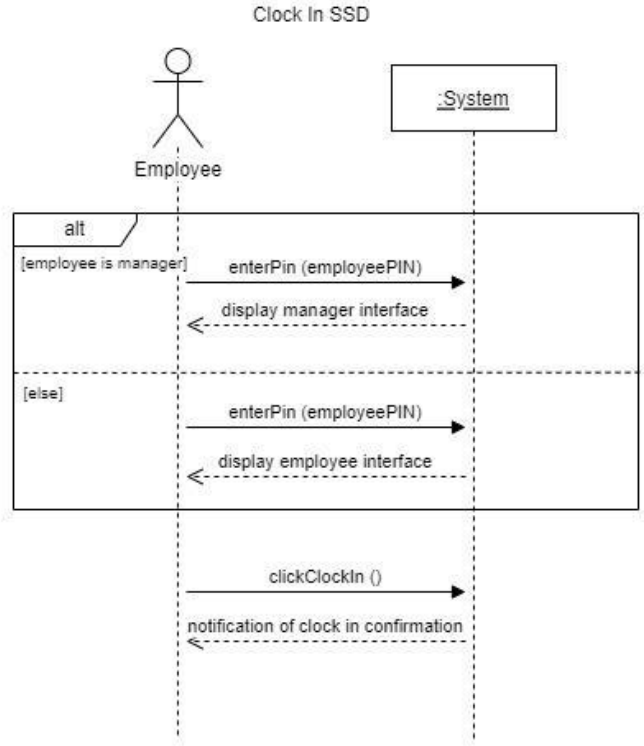

Fig. 6. System Sequence Diagram for Clock in

When an employee is clocking in, it is sending information to the system while the system is sending information back. The first thing shown in the diagram is an alt box which illustrates an if else action. If the user logging in is a manager, then the system will respond by popping up the manager interface. Else if the user logging in is not a manager, then they will receive the employee interface. After that, the user will click the clock in button initiating the clockIn function to the system and the system will return the clock in confirmation which is the current time.

III. DESIGN

The design model is the next step in the project creation process. In design, the purpose is to transition smoothly to implementation. The design model would include data types and return types. We would also need to specify parameters for functions. The benefit of the design model is that it generates an outline before implementing/coding the system. We approach design as a vision for the system. During implementation, a problem may arise and we may deviate our design due to bugs, but design provides us with a blueprint to begin.

A. Class Design

The transition from analysis to design begins with converting our domain model class diagram to a design class diagram, where we include the data type and basic functions regarding the classes.

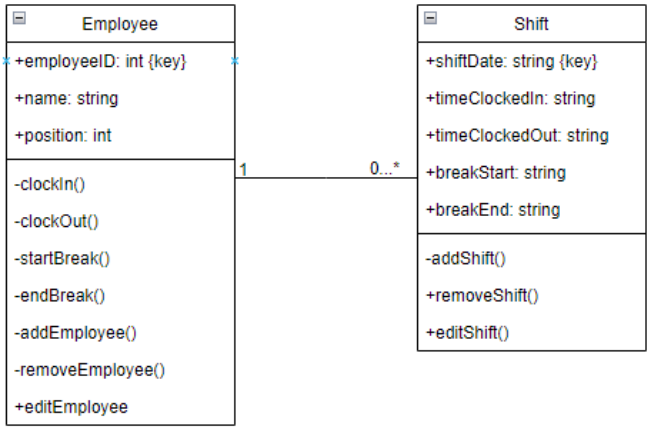

Fig. 7. Design Class Diagram

Our design class diagram is similar to our domain model class diagram but includes the attribute types. We made employeeID and position of type int and all the other attributes of type string. The system is able to take in the time as a string because it will format it into the correct data type. The functions of the employee are all the use cases stated previously. It also includes manager functionalities of add, remove, and edit.

B. Graphical User Interface

A graphical user interface or GUI is the screen that the user sees to interact with the system and the database. Each of the main functions in this system is a button in the GUI that the user can click. Our approach to designing the interface was to create the screen first and program the functions into it.

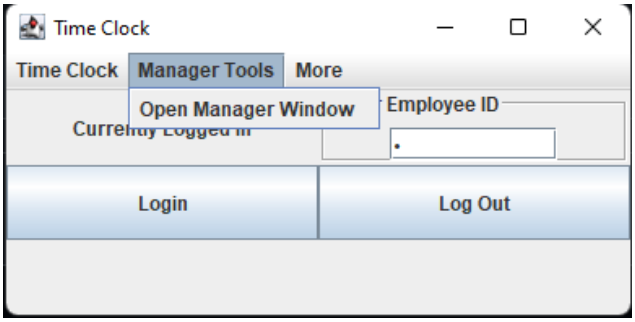

Fig. 7. GUI: Employee Login Screen

The design consists of text fields, labels, drop downs, and buttons. The text fields accept user input and the buttons are to activate a function to navigate through the interfaces. Once a user enters a correct employee ID and clicks the login button, the system takes us to the main function screen.

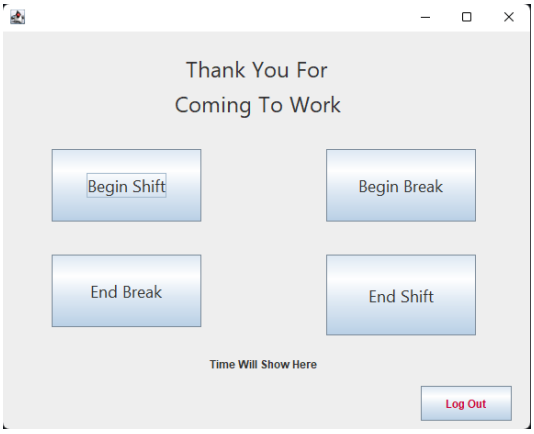

Fig. 8. GUI: Main Function Screen

This screen is where the employee is able to clock in or out of work as well as taking breaks. Once the begin shift button is clicked, the system will record the current time into the database and display it as a text on the bottom.

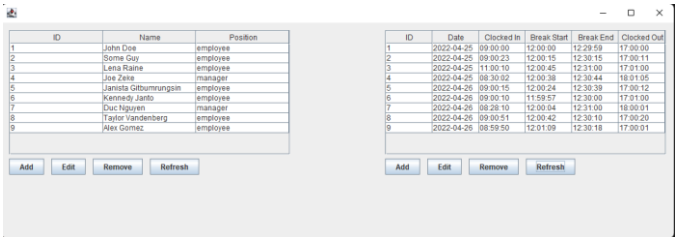

Fig. 9. GUI: Manager Tool Screen

This final screen is only available to a manager. To access this screen, a manager must be logged in and there will be a drop down option to click on a button called manager tools. Once clicked, this screen will pop up. This screen is where managers can manage the data within the database. The left table displays all the employee data while the right table displays all the shift data from the database. Managers can add, edit, or remove employees from the left table by clicking on the corresponding buttons. There will be a pop up asking for which employee ID to select and then it will ask to input new data into text boxes depending on which function you chose. Those same functions for add, edit and remove are also available to the right table of shift data.

C. Database Design

After completing a domain model diagram of our system, we were able to move on to designing a database based on it. This led us to our first database iteration (as seen in Fig. 10) with three tables. The "employee" table represented employee information, the "timelog" table represented daily shift time logs and "shift" represented information about employees' designated shifts. This design was scrapped for our final design (as seen in Fig. 11) which primarily removed the original shift table and renamed the "timelog" table as shift. Tables "employee_archv" and "shift_archv" were also added later as archive tables containing copies of records written to and updated in the regular tables.

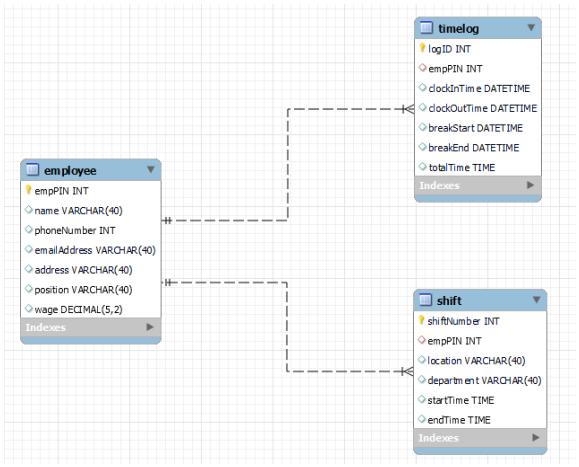

Fig. 10. Database Design First Iteration

The major changes to our database from our first to our last iteration were based on similar changes in the domain model diagram

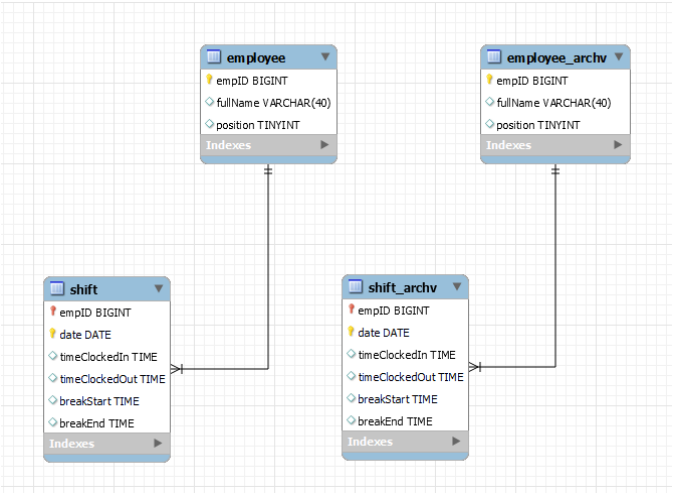

Fig. 11. Database Design Second Iteration

D. AGILE development method

Since our team decided to go with the AGILE development model, we were able to do QA and testing phase every single sprint without fail. This has increased our chances to catch major bugs. faulty syntax and doing quick fixes whenever problems arise. Our own AGILE model development process is as followed:

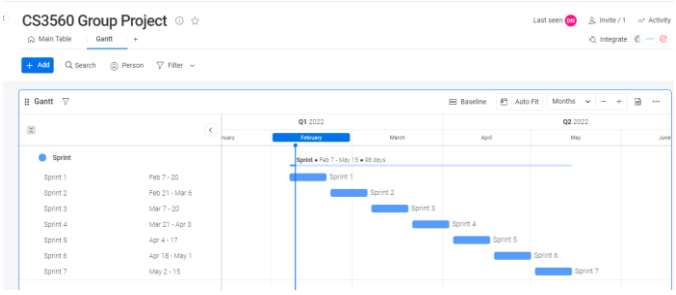

Fig. 12. monday.com Gantt Chart

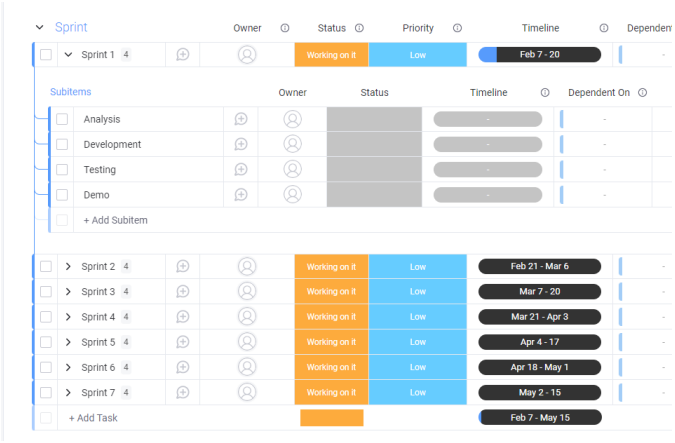

Fig. 13. Sprint Detail On monday.com

First, some ground work before we can go in-depth on how we do our QA and testing procedure. By using the project management website monday.com (as seen from Fig. 12 and Fig 13, we have decided to go with an AGILE development method which includes 7 Sprint periods that last throughout the semester. Each Sprint will be 2-week long that contains its own Analysis, Development, Testing and Demo phase that everyone in the group will actively involve and report to its respective leader. Having a Gantt chart helps us visualize the plan better so we can keep track of what happened and what is to come.

Name	Duration	Start	Finish	Tasks	Resource Names
Sprint 1	**14 days**	**2/7/2022**	**2/20/2022**	Plan out and Learn concepts	
analysis	4 days	2/7/2022	2/10/2022	-	
development	7 days	2/11/2022	2/17/2022	-	
testing	2 days	2/18/2022	2/19/2022	-	
demo	1 days	2/20/2022	2/20/2022	-	
Meeting day 1	week 1				
Meeting day 2	week 2			create session plan	
Sprint 2	**14 days**	**2/21/2022**	**3/6/2022**	Learn concepts + GUI design	Focused members: Janista and Taylor
analysis	4 days	2/21/2022	2/24/2022	- Think and draw design for program	
development	7 days	2/25/2022	03/03/2022	- Learn and Implement in C#	
testing	2 days	3/4/2022	3/5/2022	- Show what we have for the implementation so far	
demo	1 days	3/6/2022	3/6/2022	- Show the initial implementation to the professor	
Meeting day 1	week 1			Work on group assignment #1	
Meeting day 2	week 2			Make plans	
Course Item due:					
Group Assignment #1					
Sprint 3	**14 days**	**3/7/2022**	**3/20/2022**	Plan out and Learn concepts / put in progress report	
analysis	4 days	3/7/2022	3/10/2022	-	
development	7 days	3/11/2022	3/17/2022	-	
testing	2 days	3/18/2022	3/19/2022	-	
demo	1 days	3/20/2022	3/20/2022	-	
Meeting day 1	week 1	3/10/2022		Compile for Project Report	
Meeting day 2	week 2			Make plans	

Fig. 14. Sprint Detail In Excel

Second, by utilizing Microsoft Excel, we developed a more detailed plan for each Sprint as what we need to do? What do we need to accomplish by the end of each Sprint? etc. This allows us a detailed and in-depth pre-analysis of what is to come so everyone can start preparing accordingly.

Lastly, since the model of our project development is AGILE, the testing phase happens at every single Sprint, so everyone has to get involved with the QA and testing procedure. We divided the main responsibilities and tasks as followed for every respective phase we have for the project:

+ Project Manager: Kennedy Janto - Overseer and communicate with the professor/ stakeholder about the project overall.
+ Lead Database Coder: Alex Gomez - Learn how to utilize and implement database systems.
+ Lead Backend Programmer: Duc Nguyen - Learn how to utilize and implement the backend of a programming language to connect everything together.
+ Lead Design/ UI Coder: Taylor Vandenberg - Learn how to utilize and implement a functional GUI
+ Lead Analyst: Janista Gitbumrungsin - Research on the overall requirement for the system such as what classes do we need for the program, what information do we need, etc.

After this is established, we then follow all the testing procedures each week with their unique respective test cases and leaders to reduce the amount of disagreement created from individual perspectives. Some example are:

Sprint 1: QA/Testing Case	
Respective Role: Project Manager	
1. Is the project follow the Analysis phase at the beginning of this Sprint?	
2. What is the project missing?	
3. etc,...	

Sprint 1: QA/Testing Case	
Respective Role: Lead Main Coding Language	
1. Can the current programming language support the current Sprint's idea/algorithm?	
2. What is the project demand as of this Testing phase?	
3. etc,...	

IV. Findings and Suggestions

A. Findings

During the analysis portion of the project, we made changes based on our realization of the system's view. We first had in mind that employees and managers are two separate classes to be implemented. Through discussion, we found out that it would be more efficient to identify the manager as an

employee instead. This reflects real life too because managers are just employees to a company with higher roles.

B. Suggestions for the Future

For the future, we suggest having a more organized and planned out schedule for implementation. A lot more time was needed for testing and debugging so programmers should consider that when implementing the system. It is also suggested to change the system so that it is stored which employees work under which managers, so that when managers edit any employee or shift data, they only have access to the employees under them rather than anyone else. Furthermore, the main menu screen in which employees login from should display the name of the employee once logged in, so that users will know they did not log in to the system as a different person.

V. Conclusion

A. Challenges

We obviously had a lot of bugs in our program that were not noticed or cannot be identified right away. No one knew how to connect a database in mySQL to VSCode. Most of us did not have enough prior knowledge on GUI. Only two people that know the basics are Duc and Taylor. Clash of ideas or methods our team decided to go with the AGILE development model, we were able to do QA and testing phase every single sprint without fail. This has increased our chances to catch major bugs of faulty syntax and doing quick fixes whenever problems arise. There were also time constraints that led to our team having to reassess what should be a part of the program and what should not be.

B. What We Learned

When we started this project, most of our team had little to no prior knowledge of databases and GUI. That is why a majority of our learning is on those topics. We learned about MySql database and how to create and connect a local database to our java program. We also learned design planning before implementation. We created many UML diagrams and made changes to them over time as we learned more about them. We also learned GUI sketching which was later implemented into the system. All of us learned how to do GUI implementation through NetBeans and JavaFX which surprisingly connected easily. The last thing we realized was that time moved fast in the development process. We were relaxed with the amount of time we had left to implement the program. All of a sudden, we had two weeks left and realized that we could have worked more efficiently. However, we still completed the program in a timely manner with good quality.

C. Recap

In conclusion, our project goal was to create an online clocking system for employees at a workplace. It needed to have the main functions of clocking in and clocking out but also additional functions of editing data for managers. We went through the whole AGILE development process and also did thorough analysis prior to. We made changes to our class diagrams and database schemas throughout the analysis process. During the implementation process, we went through a lot of testing and debugging. In the end, we created a functioning system that meets all the project goals from the beginning.

References

1. BAELDUNG, *"Connect Java to a MySQL Database"*.BAELDUNG.COM. www.baeldung.com/java-connect-mysql. Accessed February 27, 2022.

2. freeCodeCamp.org. *"SQL Tutorial - Full Database Course for Beginners"*. YouTube. Published Jul 2, 2018. www.youtube.com/watch?v=HXV3zeQKqGY. Accessed February 27, 2022.

3. CodeCraks. *"How to Create Complete Login and Registration System (Multiple Accounts) in C# 2020 | C# Tutorial"*. YouTube. Published Dec 23, 2020. www.youtube.com/watch?v=IzpnHUo6iAI. Accessed March 15, 2022.

4. BoostMyTool. *"Create Login Form Using C++ and Visual Studio 2022 with SQL Server Database (with Source Code)"*. YouTube. Published Jan 14, 2022. https://www.youtube.com/watch?v=N1DarSDMVWo. Accessed March 15, 2022.

5. IAmTimCorey. *"Using SQLite in C# - Building Simple, Powerful, Portable Databases for Your Application"*. YouTube. Published Jul 17, 2018. www.youtube.com/watch?v=ayp3tHEkRc0. Accessed March 17, 2022.

6. VoidRealms. *"C# 22 - Textbox and more Forms"*. YouTube. Published Aug 24, 2011. www.youtube.com/watch?v=Y9L85OBo1c4. Accessed March 17, 2022.

7. Microsoft. *"Visual Studio 2019 version 16.11 Release Notes"*. docs.microsoft. docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes. Accessed March 19, 2022.

8. *"Connect Java to a MySQL database"*. stackoverflow. Published May 15, 2010. stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database. Accessed April 1, 2022.

9. ProgrammingKnowledge. *"Java prog#13.Display Time and Date In java Netbeans"*. YouTube.

Published May 5, 2012. youtu.be/lMbfAyFYkDM. ACCESSED APRIL 1, 2022.

10. Knowledge to Share. *"How to Open one JFrame from another in java Swing : JFrame Tutorial"*. Published Oct 11, 2018. www.youtube.com/watch?v=3dlvseTkRHg. ACCESSED APRIL 1, 2022.

11. BoostMyTool. *"Connect to MySQL Database from Visual Studio Code and Run SQL Queries using SQLTools Extension". YouTube. Published Oct 5, 2021.* www.youtube.com/watch?v=wzdCpJY6Y4c. ACCESSED APRIL 15, 2022.

12. Programming Guru. *"How to Connect to MySQL Server in Java | Java Connect to MySQL Step by Step | JDBC in Java"*. YouTube. Published Dec 11, 2021.www.youtube.com/watch?v=-0ppy9NjQ0c. ACCESSED APRIL 15, 2022.

13. Ha Minh, Nam . *"Java connect to MySQL database with JDBC"*. CodeJava. Last updated March 07, 2020 www.codejava.net/java-se/jdbc/connect-to-mysql-database-via-jdbc. ACCESSED APRIL 15, 2022.

14. BalusC. "Connect Java to a MySQL database". stackoverflow. Published May 15, 2010. stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database/2840358#2840358. ACCESSED APRIL 20, 2022.

15. BootsMyTool. "Add JAR files to Java Project using Visual Studio Code". YouTube. Published Jun 1, 2021. www.youtube.com/watch?v=3Qm54znQX2E. ACCESSED APRIL 20, 2022.

16. MySQLTUTORIAL. *"How To Copy a MySQL Database"*. www.mysqltutorial.org/mysql-copy-database/. ACCESSED APRIL 29, 2022