CSE 360 Project Report Number 6

Team W20

Team Member Names:

1. Italia Alanis-Martin
2. Buthaina Alassafi
3. Vincent Dang
4. Karen Garcia
5. Aditi Kelwalkar
6. Junayd Lateef

Table of Contents

1. Executive Summary

The expenditure of upgrading EffortLogger has completed its first iteration on the engineering process cycle. The team has taken the fundamental concepts of the original EffortLogger and expanded upon them, brainstorming ways to meet the customer's requirements alongside optimizing efficiency. Now the team describes the finished product in its completion, with screencasts to exemplify all the components of this finished product.

The customer, who is the owner of the previous EffortLogger, came to the team with a set of three major requirements to fulfill: to increase company security, to incorporate employee privacy, and to accustom agile engineering processes. For context, the existing EffortLogger product timed employees (in minutes) on the duration of their tasks in order to send an efficiency log to their managers. This posed several challenges in the corporate, societal, and engineering arenas.

First, company data is at risk at its current address in an Excel sheet. Every piece of information is recorded and outputted in one central place which can easily be stolen if not careful. However, hackers will have difficulty trying to break the new EffortLogger, which has auxiliary strict data entry checks and also help serve the purpose of blocking off entities who aim to steal corporate information. The team has implemented this by uprooting the current database structure from an Excel spreadsheet to a coded data structure. In the updated version, we used a HashMap data structure, which has encryption built into it; no outsider can ever know the keys that describe the location where data is queried too. Also implemented are encryption keys themselves, which encrypt and decrypt the address of an employee object. Lastly implemented are integer overflow and underflow checks, which serve to save the program from crashing or the buffer to overflow.

Next, we address the problem of employees who are concerned for their privacy. Stressed employees lead to disrupted productivity and lower quality of work life. The primary concern of the old EffortLogger was that employee identities were displayed when managers were viewing the logs, leading to feelings of intense pressure and unhealthy competition. To eliminate this issue altogether, the new EffortLogger will now record an employee's position when displaying their logs, not their individual names. Employee objects are also simplified so that they hold very little outside information about the employees; an object only contains its username, password, position, and lists of efforts or defects. Also, employees are now able to edit their logs as well, which can assist in reducing both technical mishaps and human error.

Lastly, the team addressed the necessity for a revolutionized product that better assists agile-ran engineering processes. The team did so by incorporating a major concept common in agile practices called story points. Story points are the measure of effort and work in agile task sprints, and are now the new measure of effort in EffortLogger: Story Points. Story points delineate the difficulty, or the amount of work or effort it takes to complete a specific task. In our product, story points for a project range from 1 to 10, 1 being the easiest and 10 being the most strenuous. To build on the idea of story points, the application has also supported a *Planning Poker* feature which allows for an honest decision-making process in which the agile team comes together to name point values for certain tasks. This feature allows each employee

to view their filtered historical data as guidance to assign a point value on a brand-new task. This allows for a fair discussion of story point value delineation, and makes for an extremely productive setting point for the tasks ahead. The program is also equipped with a definitions page, which defines all possible stages of an agile project. With these three features, an engineering team who functions in an agile manner will be better equipped to handle business.

The process of changing EffortLogger to meet the new requirements has included several planning stages. The group has written a Concept of Operations, and a Requirements Definition document, detailed the system architecture, executed the design, and completed the testing process. The final prototype is presented through screencasts.

2. Concept of Operations

2.1 Project Description

2.1.1 Background

The Effort Logger system needs to be updated because it is not efficient enough, the security it provides is not adequate, and the searching of logs of different tasks needs to be more accurate. The system also needs to be updated because as more companies and larger companies use it, it needs to be modified and fixed if necessary. The goals of the system operation are for the user to be able to create, search, and edit effort logs easily. Project managers should also be able to view and search through their team's Effort Logs easily and without knowing the identity of each team member. Also, the system must have a better method for measuring the effort it took to complete a task, other than time. Our team's goal is to provide the most efficient and user-friendly system that protects the identities and data of its users.

2.1.2 Assumptions and Constraints

There were not too many constraints and assumptions to be made about the updated solution. The main thing to keep in mind is that we are limited to using the programming languages JavaFX, and Java for creating the main software. We can also assume that we are not limited to how our data is stored and the only thing to focus on is the efficiency of searching through the database, but also how to make the runtime of our overall program efficient to improve the quality of our solution. The system should also include good security measures that serve to protect user data and user's identities. The users must also be able to easily determine how much effort it took to complete the logged tasks.

2.2 Overview of the Envisioned System

2.2.1 Overview

First, the Effort Logger will open to a login page where the user will enter their username and password, if the username and password match to that of a known user, they will be granted access to the system. Once granted access, the main menu will open, this is where the user will be able to choose where to navigate next. They can choose to create a new log, view their old logs, and edit existing logs. If they are a project manager, they can view the logs of their entire team. The user can also choose to create and view defects in the defect console. If a user selects the option to view team logs, they must first confirm that they are a project manager. They do this by being prompted to enter in a verification key that only managers would know. If they enter the key correctly, they will be granted access to the team log information. If the user gets the verification wrong too many times, they will be automatically kicked back to the main menu page. When creating a new Effort Log, the user will select the project, life cycle step, and effort category of the activity. In order to measure effort, the user will give the task story points on a scale of 1-10, they will also input a few short story points to go into further detail about what the task entails. The user can then choose to view or edit existing logs, or they can add defects. In the Effort Log editor, the user can modify all aspects of the log they are trying to edit including the name, project, etc. They can also clear, delete, or update the log The Defect Console is where the user can enter new defects or edit existing

ones. Here, they will specify the project in which the defect takes place and all the attributes that go with the defect. Then they can proceed to the Defect Log to view the new defects. In the Defect and Effort Logs, there will be an option to filter and search through the logs. This would help users find specific logs during planning poker, which will help them give the backlog items discussed in the meeting a more accurate rating. There is also a Team Effort Log and Team Defect Log that only managers have access to where they can see the logs of every person using the Effort Logger.

2.2.2 System Scope

The system will include about 11 different pages: the Login page, the Main Menu, the Manager Menu, the Signup page the Effort Log Console, the Effort Log, the Defect Log, the Defect Log Console, the Manager Verification page, the Team Defect Log, and the Team Effort Log. Each page will have buttons that allow the user to navigate to other pages easily. There are also security measures in place such as how in the manager verification page, if you get the password wrong too many times, you will be kicked back to the main menu page.

2.3 Description of Envisioned System

2.3.1 Needs, Goals, and Objectives of Envisioned System

The system needs to effectively create, edit, and delete effort and defect logs. It also needs to display each log properly and search through the logs to find specific information that the user may need. The system should protect the information of team members by concealing the identity of who made each log and only allowing managers to view the entire team's log history. The system should accurately record all of the log information and organize it in the Effort/Defect log pages. Our goal for the system is to make it user friendly and efficient as well as to make searching for specific log information faster and easier. The Effort Log program that was provided by the professor serves as an example for what the team will try to emulate in regards to the GUI and overall functionality, however security measures will be added as well as a new method for searching through the logs. Another thing that is being added to the provided Effort Log program is the user giving each task a rating of difficulty and inputting short story points in the Effort Log Console. This is being added so that effort can be more accurately measured and observed by users.

2.3.2 Overview of System and Key Elements

The buttons on the main menu of the Effort Logger will each navigate the user to a specific page in the system. Inside of the effort log console, defect log console, and effort log editor pages there will be various buttons and drop-down menus that prompt the user to provide information about the log they are creating or editing. Before the user can navigate to the team log page, there will be a page where the user will need to verify that they are a manager. On this page there will be a text box where the user will be prompted to enter the verification key and upon hitting the "Enter" button the program will verify if the key entered by the user matches the verification key. In the effort log and defect log pages, the information of each log will be laid out similar to the images below. There will also be a search bar where the user can search keywords and a filtering option so that the user can find specific logs.

2.3.3 Interfaces

The effort logger system may have various interfaces with external systems or major envisioned elements of those systems. The system will have an interface with the users and operators who interact with it. This interface will include a GUI that will allow the user to input their effort/defect logs and make necessary edits, if possible. In terms of the data interface, the system will need to interface with other data systems. This will occur when it is time to extract relevant data in regards to user effort. There will also be an authentication interface that the system will interact with. The purpose of this interaction is to ensure that users are authorized with the correct login credentials. The team manager of the employees will be receiving a weekly email containing an access key to team logs. This requires the effort logging system to interface with an email or notification interface. Finally, the effort logger will most likely also interface with a reporting system. This interaction will occur when something goes wrong with the main system, or when a team member or manager is locked out, and needs administrator verification.

2.3.4 Modes of Operation

The effort logging system will need various modes to accomplish its intended purpose. One of those modes is a training mode that facilitates the training and onboarding of new employees and managers. This mode will include video tutorials that utilize sample data to help users understand the functionality of the system. A reporting mode is needed to generate reports about user productivity to stakeholders. This mode will also be used when there is a bug in the system that causes technical errors for users. At a certain time, the system may need a disposal mode to securely process data deletion and system component destruction. Finally, there should be an operational mode where the system will run in its primary mode. This mode includes all the functionalities of the system that allow it to fulfill its basic requirements.

2.3.5 Proposed Capabilities

The first capability that will be needed in the program is the username and password verification during login. This capability will ensure that the username and password are the correct length, have the correct special characters, and that they match an existing user's username and password. The next capability will be for manager verification. Here, the program will check to make sure that the verification key entered by the user matches the key that only managers know. This ensures that only managers will have access to the team logs. During the creation, deletion, and editing of effort and defect log entries the program will have to insert the given information into the appropriate log in an organized way. Inside the effort and defect logs, the searching feature must use keywords entered by the user to parse through the log and find entries that have similar keywords. The filter feature will serve to further narrow down the search.

2.4 Operational Scenarios, Use Cases and/or Design Reference Missions

2.4.1 Nominal Conditions

When the system runs under normal circumstances with no anomalies, the user will log in to the Effort Logger on the first try. Then, if they are looking to create a new entry, they will navigate to the Effort Log Console by clicking on the button. Here, they will enter in all of the appropriate information for the task they are trying to complete. Then they will work on

completing their task and once the task is completed they will give the task a rating and enter in relevant story points and the new entry will be created. After the new entry has been entered, it will be appropriately sorted into the Effort Log. Then, if the user wants to edit an existing entry they will navigate back to the main menu and select the Effort Log Editor button. Here they will specify which entry they are trying to edit and make the appropriate changes. After the changes are made, the entries information will be updated inside of the Effort Log and reorganizing will take place if needed. In the Effort Log, they can search for a specific entry by inputting keywords into the search bar. Doing this will result in them finding the entries they were looking for. They can also simply narrow their search by project or lifecycle step using the filtering feature. After that, if the user wanted to create a new defect they would navigate back to the main menu and select the Defect Console button, then they would specify whether they wanted to edit an existing defect or add a new defect. They will then enter the appropriate information using the buttons and drop down menus. After the defect has been created, it will be appropriately sorted into the Defect Log. If the user is a project manager and they want to view their team's log, they will navigate back to the main menu and select the Manager Menu button. They will then be taken to the Manager Verification page where they will correctly enter the verification key. They will then be able to select between the Team Defect Log, the Team Effort log, and the Employee signup pages. In the Team logs, managers will be able to view entries from every employee. In the signup page, they will be able to create the login information for new employees. Finally, they will go back to the main menu and log out.

2.4.2 Off-Nominal Conditions

The first scenario covers what would happen if a user tries entering a username and password that doesn't match with the username and password of a user in the system. In this situation, the user will simply not be granted access to the system.

The next scenario covers what would happen if someone who is not a manager attempted to access the team’s log. The user would select the Team Log button and the system would take them to the manager verification page. Here they will be prompted to enter the verification key and they will enter it incorrectly. After five incorrect attempts, the program will take them back to the main menu, kicking them out of the manager verification page.

The last scenario is what would happen if the entry that the user is searching for does not exist in the Effort Log. The user will navigate to the Effort Log and attempt searching for keywords only to find that they are getting no results. They will then try to enter different keywords in hopes of finding a similar entry, but they won't find what they're looking for because they have never logged something similar.

2.5 Risks and Potential Issues

Risks and potential issues with the development include creating and monitoring the database where each user's individual effort and defect entries will be stored. as well as creating and monitoring the database where the entire team's effort and defect entries will be stored. This is a risk because the databases must organize the entries while also keeping each individual user's identity hidden. This will be a tricky thing to implement because the Effort Log would have to not only organize each individual user’s information but also every team member’s information combined.

The searching feature of the Effort Logger could become a potential risk because the user can enter any word that they want into the search bar. This means that the system has a greater variety of things it could search for. This could lead to issues that would slow down the team if not implemented correctly.

Another potential issue is the schedule for the project because our team has little over a month to finish this project and that could lead to potential scheduling conflicts. Final exams are approaching fast, and at this point in the semester student schedules start getting very busy, so we as a team must have clear communication about meetings in order to complete this project on time. The team must also utilize its meeting time to make everyone 100% clear on everything that will be done for this project especially as the semester comes to a close.

3. Requirements Definition

3.1. Overview

The stakeholders have all agreed to align requirements with the primary concerns stated by the client of the Effort Logger system. These requirements include a focus on user privacy, system security, and Agile-scaled development. In terms of user privacy, the main focus is ensuring that the user remains anonymous and that their information is not shared with unauthorized personnel. The implication of this requirement requires extensive testing and mitigation steps to protect the system from being breached. The implementation of Agile-scaled features guides the design process in a way that leads to a final product reliant on iterative and team-focused development.

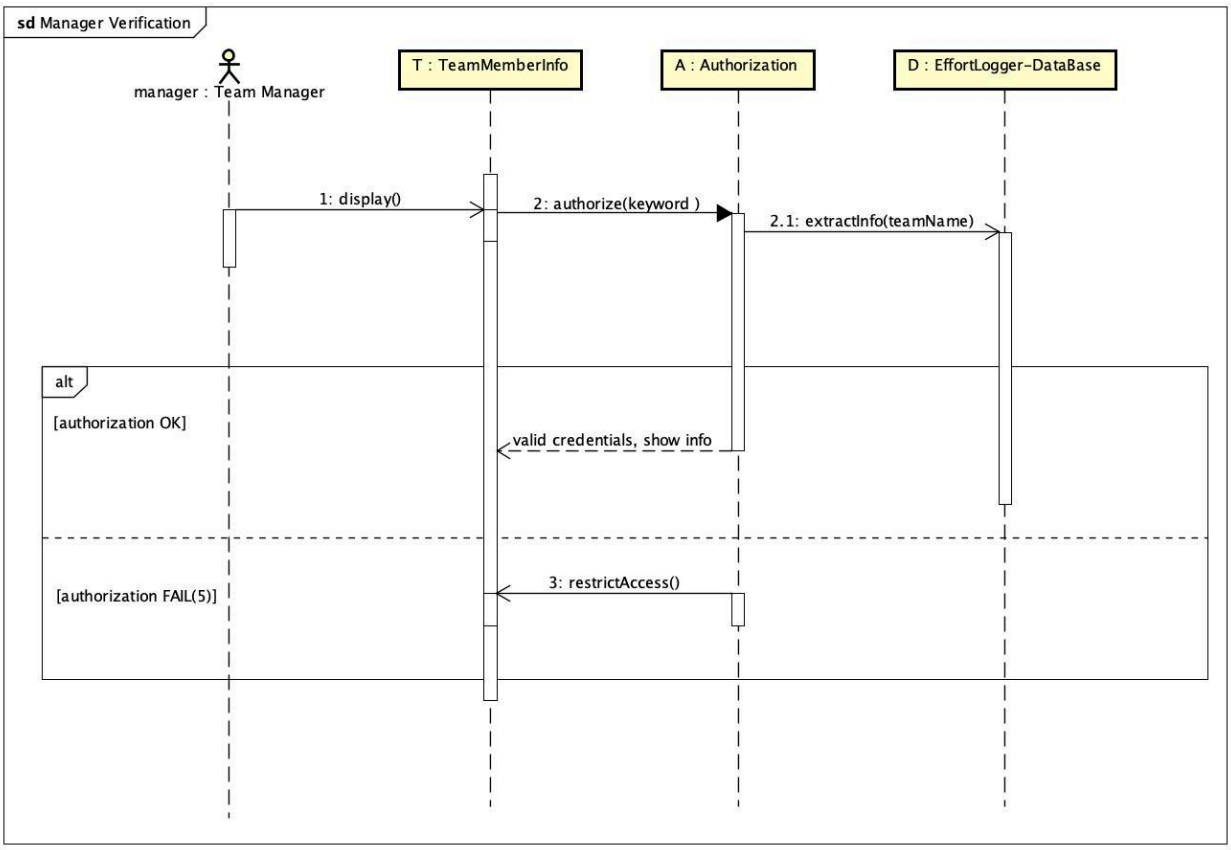
3.2. Context

3.2.1. Sequence Diagrams

The most significant sequence diagrams are those that outline basic user functionality. The main purpose of the effort logging system is to measure effort through effort and defect logs. Therefore, there should be a diagram for the viewing and editing of these logs. Making these logs involves specifying project definitions, such as life cycle step, effort category, plan and deliverable category, etc. Since this step was assigned its own console in the previous effort logger system implementation, it would stand to be significant enough to have its own sequence diagram. To meet the user privacy requirement, login credentials will be required. There should be a sequence diagram that details how the login process will look, and also how to deal with issues concerning invalid user credentials. The manager will have special access to team logs. This will involve a separate authentication process that requires a key. That process should also be documented in a sequence diagram.

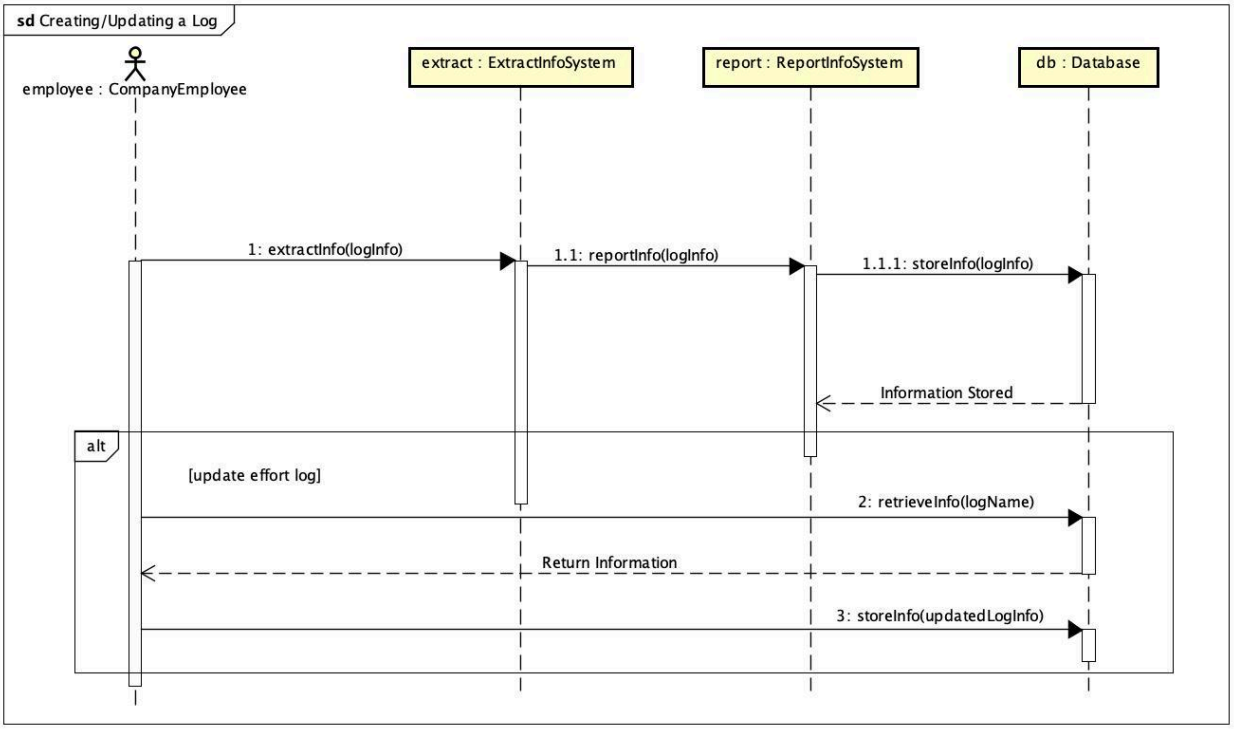
Manager Verification Sequence Diagram:

This sequence diagram outlines the steps taken by the system when a manager tries to view the team logs. There are three instances of three different objects, namely authorization, team member info, and the effort logger database. The flow is such that the authorization instance verifies the identity of the manager based on the keyword they have inputted. After that verification, the team member info instance will collect the specified team name, and pass on that information to the database. The database will then display the correct team log information. In the alternative situation where the team manager is not authorized, they will have no access to the team member’s effort log info after the fifth login attempt.


sd Manager Verification
manager : Team Manager
T : TeamMemberInfo
A : Authorization
D : EffortLogger-DataBase
1: display()
2: authorize(keyword)
2.1: extractInfo(teamName)
alt
[authorization OK]
valid credentials, show info
[authorization FAIL(5)]
3: restrictAccess()

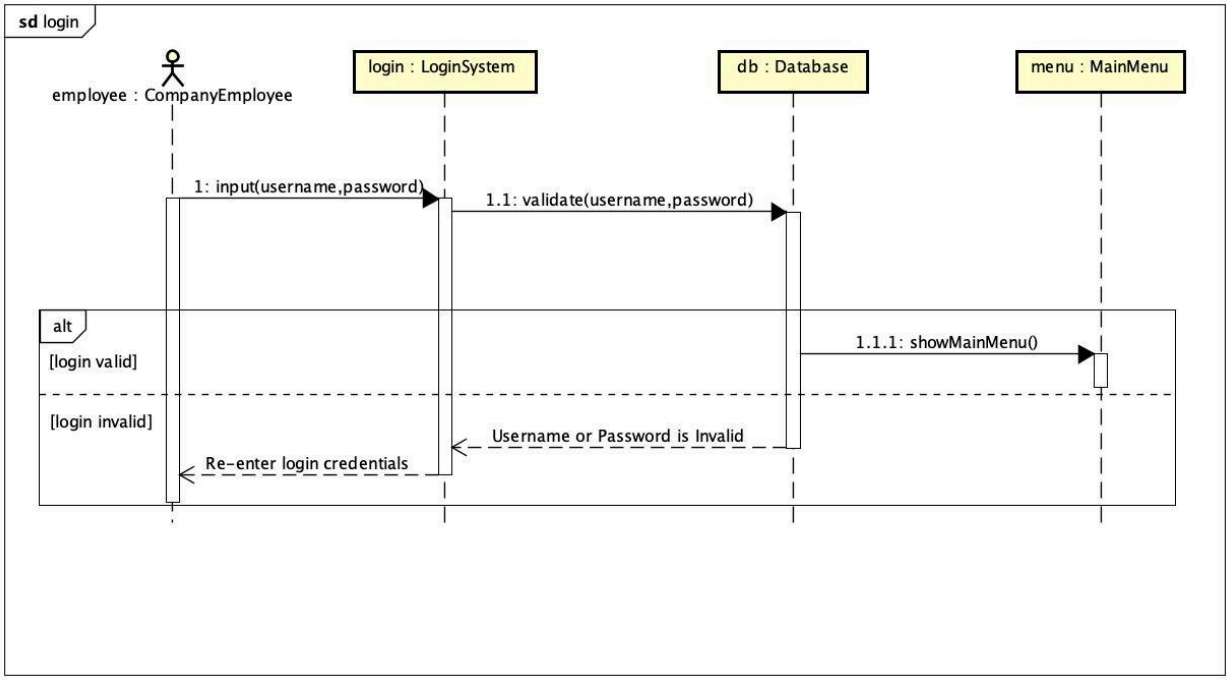
Creating/Updating Logs Sequence Diagram:

This sequence diagram outlines the steps taken to make an effort/defect log. There are four instances of four objects present, namely the company employee, extract info system, report info system, and database system instances. The first step is extracting the information related to the effort log definitions and descriptions. This information is then passed on to a reporting system that transfers the log information to the database. The database confirms the storage of information to the reporting system.



Login Sequence Diagram:

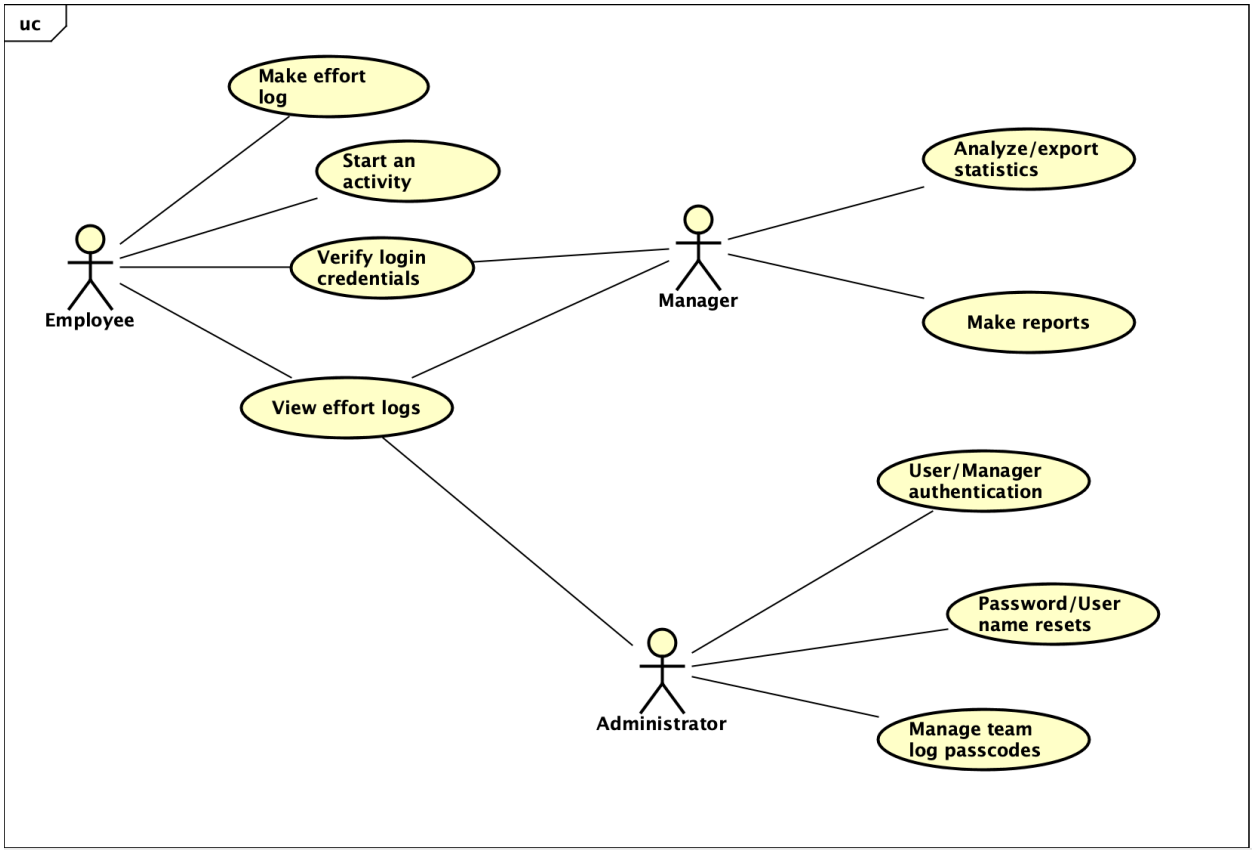
The following sequence diagram outlines the set of interactions that occur when an employee logs into the effort logger system. There are four class instances here, namely the employee, login system, database, and main menu instances. The username and password are passed on to the login system, which validates the information with the database. If the user's credentials exist, they are then taken to the main menu. In the alternative situation that the login credentials are wrong, the user will be asked to re-enter the username and password.



3.3. Use Cases

3.3.1 Overview

The following use case diagram covers all the scenarios in the use cases listed below. These use cases center around the primary functions of the effort logging system, two of which include the user making an effort log, and managers viewing these logs to monitor team productivity. In terms of the employee, their use case entails starting an activity, making/viewing effort logs, and verifying their login credentials. The manager will also have to verify their identity and they will have access to the employee's effort logs. The managers are responsible for analyzing the employee's efforts and making reports to the higher ups about productivity. The administrator will step in to authenticate the employee or manager in the case that they are a valid user who has forgotten their login credentials. They will also be in charge of sending out weekly passcodes to managers. These passcodes give the manager access to the team logs.



3.3.2 Use Case Details

Use case 1

Description:

One of the important outputs of requirements engineering is system security. The detailed design ensures this by setting up a credential-checking system. Each employee is provided with login information by the company. This information can be used by the user to make effort and defect logs, as well as view previous logs. One of the layers of security taken is identity verification, whereby the company verifies the identity of the employee before creating an account. This reduces the risk of impersonation and prevents unauthorized access. By centrally creating employee accounts, the company keeps track of who has access to what resources. In a dire situation, this helps the company quickly detect and respond to any suspicious activity.

Main success scenario:

A user opens up the effort logging system on their personal computer. They are asked to enter their login credentials, consisting of a username and password. Credentials are entered, and the user is authenticated. The user is then led to a screen that has all of the consols (effort, defect, project definition, etc.) listed. The respective option is chosen and the user is led to another screen.

Alternative scenarios:

If the user enters their credentials and their username or password is incorrect, they will have 2 remaining attempts to verify their identity. After the third attempt, they will be locked out and will need to request a special administrator key to reset their password/username.

If the user does not remember their username or password, they will have the option to reset it. This will require the verification of some other identification method, such as driver license number, SSN, etc. Afterwards, the user will be prompted to enter the new username/password.

Use case 2

Description:

This use case is related to the requirements engineering output of agile development. The gap between that output and specified design is the metric of effort used. Previously, time was used as an estimate for how much effort a defect or effort log takes. This was determined to be inefficient, and thus will be replaced by an agile metric. The user will write a one sentence user story to give context to the effort/defect log, and will use that story to determine how many story points should be allocated.

Main success scenario:

User enters the effort logging system to document a new activity. They specify the user story related to the effort log, and give that user story a story point rating. The user selects the project definitions pertaining to the effort, and then posts the effort. The user then exits out of the program.

Alternative scenarios:

If the user makes a mistake in regards to project definitions of the effort log they posted, they will have the chance to fix it. The user will go to the effort log editor, select the effort log they just posted, and make any needed changes or updates.

Use case 3

Description:

Use case three concerns the requirements engineering output of employee anonymity. This is bridged to the detailed design through a team manager verification system. Employees of the client were worried that their effort log information may be used against them. It is a primary objective to maintain the confidentiality of the employee as well as restrict certain user privilege in terms of viewing productivity details. As exemplified by this use case, only the team manager will have access to the effort logs of team members. Even then, the identity of the employee will be concealed, and they will only be known by their professional title (software developer, software engineer, etc.).

Main success scenario:

A team manager tries to access the effort logs of the team members in order to view productivity statistics and make reports about them. The manager will go to their work email, and find the weekly sent out passcode for the team logs. They will open the effort log program, and click on a button that leads them to another page. They will be prompted to enter the passcode they found in their email. The manager will then be directed to the team logs.

Alternative scenarios:

If the manager enters the wrong password more than five times, they will be locked out and will require administrator help to log back in.

If the manager does not find the weekly sent out passcode in their email, they can contact admin to verify their identity and obtain the passcode.

3.4. Software Requirements Documentation

The functions required for the renovation of EffortLogger expands over three major stakeholder requirements. In the past deliverables, known and expected requirements have been identified, studied, and planned for without direct immediate designing; In order to build up to this stage the group has created user stories, then from those use cases, identified goals and key elements, highlighted the functional and non-functional requirements, composed a Concept of Operations, then finally outlined the risk upsides and downsides. All these factors have been thoroughly detailed in previous deliverables as well as this one. Now the next step is to engineer functional prototypes of the proposed design, which will be the focus of Section 5: Design of this deliverable.

Since this application will be used by PC users as a stand-alone program, the coding for the program will rely on data structures and function independently from external databases, such as SQL, which is the current planned execution of the program. This is liable to change if data issues arise, such as a low-response time due to an intensive storage of memory. Therefore the ideal data structure to implement this program will be a hash table, which can hold immense amounts of information with quick storage and retrieval times. The user interface will be built with JavaFX in conjunction with SceneBuilder, a visual layout tool for JavaFX. The interface will be composed of a log-in screen, a main menu screen, and multiple “tool” screens. These tools will be: creating an account, creating a log, viewing individual logs, viewing team logs, and viewing task-relevant historical individual logs. Every one of these tools will use their own screen and subtabs, utilizing a unique UI for each. The validation of this design has been confirmed through group discussion, and will continue to be altered if necessary.

These design choices will be tested and implemented through Java. Each member of the team will be creating individual components and individually testing features for this first stage of coding the project after having intensely studied the requirements. The team is certain they are ready to begin implementation and testing, which will be thoroughly discussed in Section 5: Design. Here, each member of the team will detail a risk-reduction prototype and begin coding a feature of the new EffortLogger. The testing for each prototype will also be documented in this

section.

Since the team is fully aware of the requirements and understands future functionality to be implemented, the code will be expandable and reusable for the purposes of future integration. This way, as new functionality is introduced, deployment will simply build off the previous version delivered.

4. Architecture

4.1 Architecturally Significant Tradeoffs

The EffortLogger application's priorities include security and privacy. These non-functional requirements have a greater priority than the functional requirements of collecting data such as defects and story points in order to estimate the effort put into a project lifecycle step. Our improved EffortLogger will no longer have a clock that keeps track of the time required to finish the task. The tradeoff of this decision is that the application will measure effort through story points, which are more compatible with agile development. Most of the functionality of the original application will be kept in the new EffortLogger application, instead of expanding the functionality. The tradeoff of this decision is that the application will focus on recording an individual's effort data instead of facilitating project planning or scheduling based on historical effort data.

Security is the tradeoff with the highest priority in the improved application because EffortLogger collects the personal information of employees and confidential information about a company's project. Competitors could take advantage of it if they get access to this information. If there are no security measures in place, hackers could get access to EffortLogger's data, creating a data leak and compromising the application and its users. In order for employees and companies to be willing to use this application, a log-in system is added to the application. Even though EffortLogger is hosted on each individual's personal computer and each application will only have one user, the security system of a personal computer is not sufficient to protect EffortLogger's data. If a user leaves their laptop unlocked and unattended, someone else could open the EffortLogger application and view their collected data, compromising security and privacy. Therefore, the architecture of EffortLogger has a foremost authentication layer in the form of a log-in feature. The log-in feature requires a user to sign-up by creating a username and password, and then requires the user to log-in with their newly created credentials The tradeoff of adding a log-in feature is increasing the complexity of the application and decreasing the accessibility of the data for users.

Additionally, privacy for users is the next-highest priority tradeoff that affects the architecture of the EffortLogger application. Privacy is necessary for users to ensure that the data that EffortLogger collects is not used for alternative purposes. Because collected data should only be visible to the individual users, an encryption layer is necessary on the collected data. An encryption algorithm can take input such as the effort category, completion time, or types of defects that need to be corrected and convert them into a string of random characters using a key value. This string of characters can only be understood if it is put through a decryption algorithm using the same key value it used to encrypt the message. Encryption helps ensure privacy by restricting the visibility of the data to only the user who has the key for the data. The tradeoff for improved privacy is that the application is slower since it needs to encrypt and decrypt data whenever the user needs access to it, decreasing performance and increasing complexity.

Another architecturally critical design trade off is adding input validation to the EffortLogger application. Input validation ensures the user inputs information that is

syntactically, contextually, and semantically correct when prompted for it. Without input validation checks, users can input information that is incorrect into the EffortLogs and hackers can attempt SQL injections in the username or password fields of the log-in feature. SQL injections can allow hackers to access the data stored inside the application, compromising security and privacy. Checking the user input for length and meaning can prevent SQL injections from having an effect, increasing the reliability of the application. The tradeoff for adding input validation is decreased performance and increased complexity because it will take more time and resources to ensure user input is correct.

4.2 Architectural Analysis

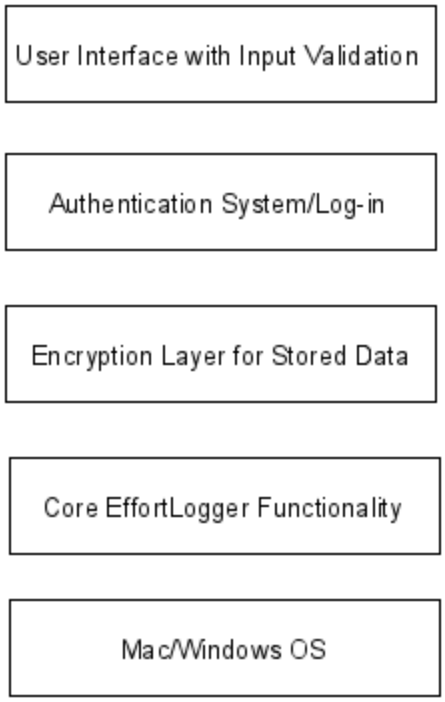
The results of the architectural analysis are that the quality attribute requirements include that the application should be user-friendly, have security measures, ensure privacy, and be reliable. The application should be user-friendly so that users of the previous EffortLogger application can get accustomed to the new application easily. This means that the new application should carry over most of the old functionality, which includes features to log defects, story points, and other productivity and effort data. The improved version of EffortLogger should also implement further security measures. Even though the application is on a personal computer, the computer's log-in screen is not sufficient security for the application. An additional authentication system is necessary to secure the application and create a multi-layered security system. In order to implement measures for privacy, the data stored in the EffortLogger application should be encrypted. Encrypted data cannot be read unless an individual has the key that was used to encode the data. Encryption restricts the people who can view and understand the collected data to the people who have the correct key, which should only be the application's user. The application should be reliable and should handle incorrect user input safely so that the application does not crash and so that the application is not vulnerable to SQL injections. The architecture of the application will be layered to include all these structural components.

4.3 Logical and Structural Decomposition

The decomposition of the improved EffortLogger resembles the diagram depicted below. The new EffortLogger will have a user interface that includes checks for valid user input, an authentication system, an encryption layer to encode stored data, and the original core functionality of EffortLogger. The user interface satisfies the requirement that the application should be easy to use and reliable, since it includes input validation. The UI will be implemented with JavaFX. The UI should be its own class since it plays a role in the entire application. The authentication system covers the security requirement and needs to be a separate layer in the architecture so that hackers cannot bypass it and compromise the confidential data stored in the application. The requirement of privacy is covered by the encryption layer, which serves as a barrier for users who do not have the correct key to access the stored logs. The encryption layer will utilize AES encryption/decryption and the javax.crypto library. The data will be stored in a hash map. The original core functionality of the effort logger will be the last layer before the computer's operating system. This layer addresses retaining the old functionality of the application.

Diagram of the Layered Architecture:

Architecture



5. Design

5.1. Design Goals

The database prototype is a foundational aspect of the new EffortLogger, and it assists many roles throughout its lifecycle. It will assist implementers by offering a predefined data structure that does much of the heavy lifting. Because it is also coded in Java, the management of data is eliminated, saving effort and time. Testers and integrators will enjoy debugging because of the high readability of the code. The methods are straightforward and comments are abundant. Operators and other business professionals will be supported with an easy-to-manage and easy-to-use infrastructure.

The credentials system for both logging in and signing up follows a similar flow to prevent any possibility of injections. EffortLogger starts off by checking if the username and password inputted follow the credential rules. Along with that, the credentials system will match the inputted credentials, if they are valid, with any of the user data in the system and will log the user rather than trying to use strings that may crash the software or leak crucial details. We wanted to make this version of EffortLogger more secure from hackers and only allow specific people from editing logs and other aspects of the data.

The encryption prototype class uses the Advanced Encryption Standard (AES) method to encrypt and decrypt Strings of data using a pseudo-randomly generated key. This class can take a String of data and encrypt it into a String of random characters. The encoded message can only be decrypted with the same key used to encrypt the message, restricting the visibility of the encoded data. This class ensures the confidentiality of the data by only allowing the user, who has the correct key, to decrypt and comprehend the data stored in memory. An encryption object has public methods to encrypt and decrypt, allowing implementers and integrators easy access to the encryption and decryption algorithms from anywhere in the final program. They do not need to worry about ensuring the privacy of the stored data because an Encryption object interfaces encryption for them.

The implementation of the manager verification feature is to ensure that only managers have access to the Effort Logs of the entire team. The manager verification feature will verify that the key entered by the user matches with the key that managers would be given in order to access the team logs. If the user enters the key incorrectly, the system will kick them out of the page and back to the main menu.

The implementation of an integer overflow checker helps developers to hone in on creating new features and improving the overall quality of the software, without needing to worry about the potential vulnerabilities associated with integer overflow. Testers can use this design to identify bugs that were otherwise overlooked. These issues can then be reported to development teams for further investigation. Integrators will benefit from the design prototype by ensuring that all functional elements of the software product are seamlessly layered. This helps them avoid costly integration issues that can lead to deliverables not being completed on time. Through this design, operators can ensure that the software is secure and stable. This will help them mitigate the chances of security breaches that can result in data loss and infringement.

5.2. Design Elements

The database holding all the company data takes place in a HashMap data structure. The company data in context includes employee information, defect and effort logs, ranks of positions, and confidential comments related to the company's projects. Each slot in the map has a key and a value where the employee's ID is the key and a list of their respective logs is the value. The database is responsible for keeping the information secure and quickly accessible, which assists in fulfilling the requirement of a heightened security. Between the inherent encryption in the hash map and coded encryption prototype, this security functionality of the new EffortLogger will be verifiable if the retrieval and storage of data functions properly every time, placing and returning the requested data. It will also be verifiable if the data is never visibly compromised at any point of the program session, or if the program does not crash.

credentials:

The main purpose of the credentials system is to improve the security of the data inside EffortLogger. For this feature, there are two main stages of the architecture which are the validation stage, and the search stage. The validation stage has the system grab the credentials from the textboxes and see if they meet the criteria for usernames and passwords. This is essential for preventing any injections or inputs that may cause errors that stop the entire process. The other stage is the search stage which is where the credentials are compared with other credentials according to their possible index in the database, which then searches through that slot in the hash table for any matching credentials. This version of the credential system not only meets the security requirements by how the credentials are thoroughly inspected, but it also is efficient at searching for matching credentials since it only compares the inputted credentials with a few of the credentials in the database.

The encryption class satisfies the requirement of privacy because it prevents confidential data from being stored in memory fully visible to external parties or hackers. Users of EffortLogger are storing confidential information about their company and projects within the application's memory. Users need assurance that their data will not be viewed by external parties and used against them, making privacy an essential component of EffortLogger. The encryption class encodes the stored data by transforming the data into a series of random characters using a specific key. The encoded data is incomprehensible and can only be decrypted using the same key used to perform encryption. While the data is stored in memory, it will be encrypted to ensure its privacy since hackers will not be able to understand the contents of the application's memory. Once a valid user needs to view their stored data, only then will the data be decrypted and displayed to the user. The encryption class ensures privacy by making the data incomprehensible to anyone who is not a valid user with the correct key.

The manager verification system's purpose is to ensure the security of the entire team's logs as well as provide privacy for employees. One of the main requirements of the Effort Logger program is to provide users with privacy by keeping them anonymous and keeping their data secure. The manager verification system takes a user's input and compares it to a preset verification key that only managers would know. If the user correctly enters the key, they have verified that they are a manager and will be granted access to the Team Logs. If the user enters the key incorrectly five times, they will be kicked out of the manager verification page and back to the main menu. Users won't be able to access the Team Logs without going through the

manager verification process first. This ensures that hackers and employees that aren't managers can't access the Team Logs.

In terms of improved security, the integer overflow design prototype supports input validation, whereby input values for a specific range of integers are verified. This prevents memory-based attacks, such as buffer overflow. In addition, methods of code injection via integer overflow are mitigated by preventing the manipulation of numerical values to execute malicious code. For Agile-supported development, the integer overflow check design can easily be integrated into iterative-based processes. These processes allow linear improvement that is based on the feedback of stakeholders. The design also poses a chance for collaborative development, as it can be used by testers, developers, and other professional team members. The design heightens privacy by protecting data. Attackers will try to exploit integer overflow vulnerabilities to gain access to sensitive data. Implementing the integer overflow prototype design can easily prevent that.

5.3. Upside and Downside Risks

5.3.1. Risk Reduction Prototype 1: Credential Check System

Rename the title for this element, explain the upside or downside, and the final resolution for the risk. For this submission, you will be required to provide a link to a screencast that will describe the resolution of the risk, demonstrate the final resolution, and explain the implementation showing the code. Place the screencast in an ASU Google Drive, set its permissions so anyone with an ASU account and the URL can view it, and place the URL here.

One of the elements that were prototyped was a log-in and sign-up system. The upside to adding a Log-In function to the EffortLogger program is allowing only a select few users to access the recorded data. Companies can have new employees sign-up without having to go through a lengthy process to add them. Lastly, there are a few measures set in place to prevent any possible SQL injections or any other hacking techniques to prevent. The downside of risks is that there are only a few measures in place. This means that if we do not stay updated on any new hacking techniques, the EffortLogger database will be more vulnerable to malware attacks and other hacking techniques. Our final solution for this application is to ban as many special characters that may be used for injections to limit any injections and errors, while also keeping credentials difficult to guess. This serves as a defense against any existing injection techniques and possible new injection techniques. The link to the screencast is listed here: https://drive.google.com/file/d/1Mvuqn45aaGN_WU1XpRyB5MDx8nZ_k17R/view?usp=sharing

5.3.2 Risk Reduction Prototype 2: Integer Overflow Check

The upside risk of preventing memory-based attacks is improving the security of software systems. The downside is that this process can be very time-consuming and expensive. For linear improvement, the upside risk is that software will continuously be in a state of evolution with the end goal of increased efficiency and effectiveness. The downside risk is that this evolution may become hard to manage by introducing instability. The upside risk of protecting sensitive information is ensuring the privacy of customers and employees. The downside is trying to balance increased security and user accessibility. The final resolution involves using an integer overflow/underflow check every time an arithmetic operation is

performed. The link to the screencast is listed here:
https://drive.google.com/file/d/1DHJw2tzdCtHcAq9WWl0P2E1rpppgN8o7/view?usp=sharing

5.3.3 Risk Reduction Prototype 3: GUI Revamped

The upside risk of having a streamlined GUI allows for easier accessibility and visibility of key interaction between the users and the program. In addition, it improves security and privacy by having multiple pages so that all the data isn't presented all at once; it includes a login page as well for further security. It'll highlight where a user will be able to input their information in a quick and efficient manner. It also allows for faster performance and better privacy by always locking parts of the system from access. Finally, information recorded in the database can be retrieved and viewed quickly and neatly for users to look through and manage. For this prototype, we scrapped the old excel based GUI and replaced it with a more clean and easy to read look. Along with that, we implemented multiple pages that can be accessed with buttons and a main page with buttons to different pages of the console. We also implemented a login page that would move to the main page once the login function is integrated and added user-friendly icons and highlighted key features and functions. The link to the screencast is listed here:
https://drive.google.com/file/d/19Ec6eAzo20AjJ6rRZgZv7EGmlEPm8NZL/view?usp=share_link

5.3.4 Risk Reduction Prototype 4: Encryption Class

The encryption feature is an upside risk. Encryption allows for the restriction of user privacy in terms of who has access to the application's collected data. Several encryption algorithms have already been researched, and the improved EffortLogger application will use a Java implementation of the proven AES encryption/decryption algorithm, making encryption an upside risk. With the AES encryption algorithm, the risk of unrelated individuals viewing the data is reduced, improving privacy. It also adds another layer of security to prevent hackers from accessing and understanding EffortLogger's recorded data. A downside risk of implementing encryption is that accessing the information provided in the logs will be slower because it would take more time to encrypt and decrypt the data. In this prototype, we have created an Encryption object that was instantiated with a chosen key size. If an incorrect key size was entered, the program threw an Invalid Parameter Exception. When a string is inputted it is encrypted using a randomly generated key and was correctly decrypted. If the wrong key was used to decrypt the message, the program threw a Bad Tag Exception. This prevents any hacker from being able to create a copy of the key. The link to the screencast is listed here:

https://drive.google.com/drive/folders/1xG7BMKuXvG-y7tRZfw9Wr87KT8qmgC3G?usp=share_link

5.3.5 Risk Reduction Prototype 5: Database

The upside risk of a properly functioning database is a seamless product for the purposes of EffortLogger usage. It will allow software engineering teams to collaborate efficiently, without worrying about privacy or security concerns. The main downside risk was ensuring proper retrieval and placement of the information being stored in the database, but after that was implemented and tested, the risk dissolved. Now the only downside risk is that growing too large of an employee base that the hash table can no longer support it, in which case a new method of storage will have to be implemented. This prototype has coded and

tested deep database storage and retrieval and implemented the code that is linked with the other prototypes. The link to the screencast is listed here: https://drive.google.com/file/d/1IR7lFsVaSSzaa2EGciJKdJAOaVnSJ1rz/view?usp=sharing

5.3.6 Risk Reduction Prototype 6: Manager Verification

The upside risk of the manager verification system is that only trusted individuals in upper management will have access to the data of an entire team. Hackers and lower-level employees will not be able to view the data of other employees. The downside risk is that there are other, more advanced hacking techniques that this security measure cannot combat. In the final version of the prototype, we added a manager verification system with different flows. When the user enters the verification key, the program will determine whether the entered key matches the predetermined verification key. If the user has entered the correct key, the program will navigate to another page where the user can view and search through the team logs. If the user enters the key incorrectly, the message: “Incorrect Key!” will be displayed on the screen and the user will be given another chance to enter the correct key.If the user enters the key wrong five times, the program will kick them back to the main menu. The link to the screencast is listed here:

https://drive.google.com/drive/folders/1TSUyY1Cd0Bv9bCi076BssKiEF_q5JcWq?usp=share_link

5.4. Design Reuse

5.4.1. Design Reuse 1: Credential Check System

For this submission, you will be required to provide a link to a screencast that will describe the design reuse, demonstrate the usage, and explain the implementation showing the code. Place the screencast in an ASU Google Drive, set its permissions so anyone with an ASU account and the URL can view it, and place the URL here.

The external solution that was used for the credential check system explains how hackers will hide SQL commands into the inputs so that way the software outputs critical information. The best method to combat these vulnerabilities is to create restrictions on the characters being used and limit the size of the inputs. This makes it more and more difficult for hackers to create commands that will leak important information or cause the system to crash. Inputs for both sign-up and sign-in features are done with the Scanner class so that way the input can be recorded from the terminal used by the software. There are also string functions used to check the characters of each credential input and are used uniquely when creating functions that will check a different aspect of the input to see if it meets the requirements. Lastly, Try and catch statements are used to prevent the code from crashing and stopping all processes. The screencast for the reuse aspects of the credentials checks system. Screencast: https://drive.google.com/file/d/1DHJw2tzdCtHcAq9WWl0P2E1rpppgN8o7/view?usp=sharing .

5.4.2. Design Reuse Prototype 2: Integer Overflow

The main elements of reuse come from instances of classes that are already found within Java libraries. These classes are predefined and allow for wide reuse across innumerable projects. Pre-defined exception methods are used to indicate integer overflow. Finally, the system exit method of the java system is used to terminate the virtual machine when a user has

inputted a certain operand. For this prototype, the scanner class is used to collect input from the user regarding the arithmetic operator and operands. Any arithmetic exceptions are thrown when overflow or underflow are detected and the java.lang.System.exit() method is used to indicate the successful termination of the running java virtual machine. This occurs when a user has inputted a 0 as the second operand of a division operation. The link to the screencast is listed here:

https://drive.google.com/drive/folders/11Gzoy2nVPvar23NC-SHIkrUamDTEMoDu?usp=sharing

5.4.3. Design Reuse Prototype 3: Encryption Class

The Encryption class that we created reuses the java.util and javax.crypto libraries, which are built-in libraries in Java. These classes are used in the Encryption class to perform encryption and decryption and ensure privacy. From the javax.crypto library, we used the KeyGenerator class to create a SecretKey key, the Cipher class to create encryption and decryption ciphers, and the GCMParameterSpec class to set up the Galois-Counter Mode (GCM) of decryption. From the java.util library, we used the Base64 class to convert an array of bytes into a String and vice versa. These external classes all play a hand in encrypting and decrypting a String message. The javax.crypto import is used to create the KeyGenerator class creates a key using the instance "AES," the SecretKey class, which stores the generated key as an object, the Cipher class, which creates encryption/decryption cipher objects that take a key and perform encryption/decryption, and the GCMParameterSpec class, which creates a parameter for the decryption cipher object that sets up Galois-Counter Mode (provides authenticated encryption). The java.util import is also used to use the Base64 class to convert byte arrays into Strings and vice versa. The link to the screencast is listed here:

https://drive.google.com/drive/folders/1f6lebeDMT0C9LV3z2lg1e2vWsUIYWob6?usp=sharing

5.4.4. Design Reuse Prototype 4: Database

The primary element of code reuse in this prototype was Java's utility library which describes a data structure serving as the database backbone for the new EffortLogger. Prototyping with this predefined structure reduced many risks that would have arisen through the use of a data structure coded by scratch. It ensures strong security with the addition of encrypting and decrypting objects, meeting the security requirements. In this prototype, polymorphism is used to generalize effort and defect log objects and we have supplemented the data structure with other attributes and methods to fit the project. Lastly, we linked the database functionality to the GUI code so that way all edits to the database can be made. The link to the screencast is listed here:

https://drive.google.com/file/d/1pxEZ6ZMLc5irtes9G1tnhBfuP2HiNgBZ/view?usp=sharing

5.4.5. Design Reuse Prototype 5: GUI

The code reuse in this prototype was the JavaFX's SceneBuilder library, which assisted in developing and creating the foundation of the GUI within a testing environment. Prototyping with the library makes it easier to design the pages of the console and reduces the risk of having to spend extra time adjusting the GUI of Effort Console through code. It also allows for easier

implementation with other prototypes. In this prototype, we used JavaFX to create a control function that allows a user to navigate between the menu. We also made a login page, an effort console page, and more where users can interact with the console or input text and check if the inputs are correct. Lastly, we made a control function that swaps the data being shown in the console depending on the information being selected. The link to the screencast is listed here:

https://drive.google.com/file/d/11JlM2CoTaaJ7rmb0uQ8eVPi2Gr2W4riZ/view?usp=share_link

5.4.6. Design Reuse Prototype 6: Manager Verification

The code reuse in this prototype was the JavaFx library. From here, we used the existing Button, VBox, Label, and TextBox classes. This helps in the design of the pages and reduces the time spent designing the pages so that the team can focus on the functionality of the program. In this prototype, JavaFx takes the input given by a user from a TextBox and checks that input against an existing verification key. If the user gets the verification key wrong, a label will appear letting them know that the input is incorrect. However, if they get the key correct the scene will switch to the manager menu. If they get the key incorrect, the scene will switch to the main menu.

https://drive.google.com/drive/folders/1TSUyY1Cd0Bv9bCi076BssKiEF_q5JcWq?usp=share_link

6. Final Integrated Prototype

6.1. Project Prototype Goals

The goals of this prototype are the following: create an application that can swiftly move from console to console without causing any errors or bugs, create a system that will be able to save both the data recorded and credentials of each company, implement the features that we created during the previous phase into the EffortLogger application. We also want to test out this prototype's features to determine which features best fit the customer's requirements. This way we can provide the customers with a product that exceeds their expectations.

6.2. Key Features

6.2.1. Key Feature 1: Credentials System

As soon as EffortLogger is opened up by the user, they will first be greeted by the log-in page. This feature is used to remove the customer's concerns with security by preventing any unauthorized users from having access to the company's data. If the user is logging in, which will take whatever is inputted as credentials by the user and check to see if the credentials match with any employee in the employee database. After checking the credentials to see if they are valid and match an employee's log-in info, EffortLogger will either log the user in, if they have the correct credentials, or they will stay at the log-in screen and have a red text show up stating that the inputted values are invalid or do not match any of the employee's credentials in the system. When a manager is signing up a new user a similar process will occur for checking the username and password. In this process, the user will input their credentials accordingly and then hit the "sign-up" button. Depending on if the inputted credentials are valid the user will either be given a message that the new user has been created or kept at the signup screen. If the credentials are invalid then the user will be given a red warning message about their credentials being invalid. If the credentials are valid, then the text fields that help the inputted employee information will be cleared and the new employee will be inputted. The link to the screencast of this feature in EffortLogger is here.

https://drive.google.com/file/d/1HyONnkGH2-BFz4DUrFbJ_V9acC36G_NK/view?usp=share_link

6.2.2. Key Feature 2: Hash Table Data Structure

EffortLogger is responsible for storing a lot of information for different companies. Using an efficient data structure, like hash tables, is paramount to retrieve data and maintain the software's performance. Due to EffortLogger's responsibility for information about employee credentials and effort, encountered defects, and other company data, a hash table will be utilized for its hashing capabilities. When hashing these objects into a slot it will take one aspect of the object being hashed it will convert the string into an integer and use the hashing technique to sort the object. Hashing is also used to ensure an even spread of objects through each of the slots. The hash table's properties are two components: a key and a corresponding value. The key in this project will be the employee, and each key will have a list of logs for its corresponding value. The list of logs contains both defect and effort logs, but they each can be accessed individually. Java's hash table will be utilized in the Database object class for storage

and retrieval. The Database also keeps track of the number of employees and a local variable used to help identify the employee or user that is currently logged into the application. The screencast link displaying the functions of this key feature of EffortLogger is here:

https://drive.google.com/file/d/1oMIIsI5ccN1kdJ6CT3ivQmbhZR53isht/view?usp=sharing

6.2.3. Key Feature 3: GUI with implementation

The GUI of the entire EffortLogger application is being revamped to make it easier to navigate and give it a pleasing look. We added different input controls, like textboxes, dropdown menus, and buttons, that are labeled in a way that any user can understand. There are four types of consoles that are accessible to all users who have EffortLogger. The first is the credentials screen which is used to log in or sign-up for a company. This screen will contain the title of the software, two text fields that will be used to input the credentials that the user is logging in with or signing up with, and one button below the credentials text fields with the one to the left running the log-in function. If there are any errors with the values inputted, then a red error message will appear, below the text fields that hold the credentials, that tells the user why the credentials were not accepted. If the credentials are acceptable and the log-in button is pressed then the user is taken to the main menu. For the sign-up page, there will be three text fields. One for the username, one for the password, and another for the job title with a submit button at the bottom of the page and an error message label above all of the text fields. The next console is the main menu which holds all the features that EffortLogger provides. The main menu will only consist of several buttons that lead to different consoles and features. The buttons for the different consoles and features will be at the center of the screen and the log-out button that will return the user to the credentials screen is at the bottom of the stack of buttons. Each of the buttons will be labeled according to what feature, console, or function they lead to so that we can make this version of EffortLogger easy to use. The only change that is made is that there are more project options and agile-based steps that can be recorded and story points are the main method for calculating effort. There are no “Start Activity” or “End Activity” buttons to calculate the time for the effort activity. The next type is the data consoles that are used to input or modify a specific type of data. The types of consoles are similar to the previous version of EffortLogger, which are the Effort Console, Effort Modification Console, and the Defect Console. Each of these console layouts will be similar to their previous version counterparts. The only major change that will be seen is that the consoles will be given a more modern and stylish look to them. The last type of console is the logging screen which displays all the different logs and data that are recorded. The log screen can switch between the effort logs and the defect logs. This is controlled by a Dropbox that will hold both of the logging options. When either one of these is selected, the data respective to the type of logs being displayed will appear in the order in which they are inputted. The Employee names who conducted the activity will not be displayed to ensure employee privacy unless the user is viewing the team logs from the manager menu. The logging GUI also has a search bar above the section where the logs are displayed that users may enter text into. The link to the screencast of this feature in EffortLogger is here:

https://drive.google.com/drive/folders/1kpPYEuXyvZ7GB-dHQbK-x-sbUMF06aAM?usp=share_link

6.2.4. Key Feature 4: Manager Verification

EffortLogger is a tool that is used to help business groups manage the work that is being done at the time. Giving the manager specific roles and having a feature that confirms that the current user is the manager improves security and allows the manager to keep track of their employees. This feature is used when the user is trying to access any of the manager-specific features that cannot be done by other group members. Once the user clicks on the manager-specific feature, they will be brought to a screen where they will input a verification code to ensure that they are a manager. Once they have inputted a code, it will be checked similarly to the credentials check feature and if the code is correct they will be brought to the GUI for the manager-specific feature. If the code is incorrect then they will be given a red message informing them that the code is incorrect. The link to the screencast of this feature in EffortLogger is here:

https://drive.google.com/file/d/1niSPckHNemyzBdBHdkKjFy9Egt36xAWi/view?usp=share_link

6.2.5. Key Feature 5: Logging System

When displaying all of the data inputted, EffortLogger uses a separate screen titled, “Logs”, that holds all of the effort and defect information that is inputted into the system. The Log console will display all data that has been stored based on what kind of data it is (Defect or Effort). The user who has access to the logs will be able to switch from the effort logs to the defect logs. The logging system impacts both the recording of data and displaying of the data. For recording the data, the Logging system takes whatever information was inputted. For displaying the data it will get the information from the type of data that is being displayed, decrypt it, and display it by the date that it was inputted. There will also be a drop-down menu at the bottom of the log console that allows users to switch the type of data that they want to see. Employee information is not displayed as a way of employee privacy. This way other employees cannot see what work other employees have done and only gives that information to managers and users with higher roles. Along with all of these features, a search feature is also implemented that will display all of the logs with the keywords that are inputted into the search bar. When displaying the logs, each aspect of the log will be put into a JavaFX table. The link to the screencast of this feature in EffortLogger is here: https://drive.google.com/file/d/1RIGvQMXANq35h_7GV6QnjpiAsrXNcR-E/view?usp=share_link

6.2.6. Key Feature 6: Effort Logging with Story Points

The main purpose of EffortLogger is to allow companies and groups to keep track of the amount of work that their employees have done. Since we want to use a value that is easily understood by anyone, we have the user input the number of story points for the task they did. When inputting a new entry for an employee’s effort, one of the things that the user will be asked to input is the number of story points that they would give the activity. Rather than having to state the amount of time it took the person to do, which does not accurately tell how difficult an activity is, we decided to have story points that would be more appropriate since the value of story points takes into account more aspects of the task that related to its difficulty. The link to the screencast of this feature in EffortLogger is here:

https://drive.google.com/drive/folders/1kpPYEuXyvZ7GB-dHQbK-x-sbUMF06aAM?usp=share_li

nk

6.2.7 Key Feature 7: Searching and Filtering

In the EffortLog tab, the user can search through previously entered logs based on lifecycle or user story. They can see the average amount of story points required for a certain type of category, which is helpful in planning poker and facilitate the agile process by measuring effort in terms of story points. We wanted an easy way to implement a feature that will allow users to find tasks they have done so that way they can use the story point values and other information with project planning or story point value estimating.

https://drive.google.com/drive/u/2/folders/1ZZ0lnlRzNzkLTgm7S6DGvGEiBfTXRhqR

6.3. Project Prototype Submission

Here is the link to our final version of EffortLogger. Please be sure to read the README.md file to understand how to log in with the default account.

Log-In info to start:

username: effortlogger

password: 1234effort

Manager Verification Key: rainbow

https://drive.google.com/drive/folders/1Z3MPGS3y740bQY_chti2bivz2Fc3Gajm?usp=sharing

7. Conclusion

In this documentation, we have integrated all of the features and prototypes that held more value according to the customer. In this updated version of EffortLogger, we included multiple features that accommodate the customer's most recent needs and requirements, these features being security, employee privacy, and more agile methods and practices. After analyzing all of the possible upside and downside risks along with all the possible features that were not implemented, we found that the credentials system, manager verification, a logging system, logging with story points, using a hashtable for the database, implementing an easy-to-use GUI, and having data encrypted were the main features that we wanted to add to this updated version.

For our fully integrated version of EffortLogger, we increased security so that the probability of hackers breaking into the system is minimal. For this, we added a credentials check feature, a data encryption feature, and a sign-up feature that is only accessed by managers. Both log-in and sign-up features are made to prevent any injections from occurring or any input being accepted that may break the software. Along with that, we have implemented data encryption to prevent data from being read even if the hacker was able to successfully obtain data from the system. This way they are unable to log into the system and can't access anything the user has.

When considering how to increase employee privacy, we decided to implement the manager verification feature as well as the logging feature that displays only the necessary information. The logging system only displays the aspects of the logs that have been inserted into the software. While each of the employees has their own logs for defects and effort, only the log information will prevent anyone else from seeing the work of others. Only the manager and other authorized users have access to this with their team log feature. Users who know the manager code will be able to view these logs.

Lastly, we wanted to add agile planning steps and the use of story points to implement more agile planning features for the customer to use. Knowing that time is not a good and complete piece of information to determine the difficulty of an activity, we implemented the story point feature so that way we can allow users to rate how difficult their activity is. This way the number of points can be debated before it is inputted into the system. Along with that, the implementation of agile planning steps prevents users from being limited to one type of planning.

Overall, the key features added were made to meet the customer's requirements as well as adding features that improve the performance of EffortLogger from the previous version. In the future, we may begin to consider other features that may be implemented to make EffortLogger a product that can be used by multiple companies.

8. Appendix A: Credit Sheet

Team Member Name	Contributions
Italia Alanis-Martin	I worked on the Concept of Operations. I contributed to the design section, and my risk prototype was the manager verification system.
Buthaina Alassafi	I worked on section 3, requirements definition. I also contributed to section 5, design, by describing my risk reduction prototype.
Vincent Dang	I worked on the conclusion and my prototype design in section 5 GUI and section 6
Karen Garcia	I worked on the Executive Summary and my respective prototype design portions in section 5. This was the database of the EffortLogger.
Aditi Kelwalkar	I worked on the Architecture section and the design sections related to encryption. I also
Junayd Lateef	I again worked on the Credential Check prototype in the design sections and I also worked on the Final Integrated Prototype section where I explained each of the key features implemented in the new version of EffortLogger.