Secure Development Life Cycle: Assignment 5

Sophia Kim, Jeffrey Suen and Keanu Williams

ICS 427

A. Introduction

As simple as it sounds, being a college student can get quite overwhelming; taking multiple classes at once, managing all of the homework assignments, and keeping track of upcoming assessments, aside from having other responsibilities such as work can take a toll on someone's well-being. And it's quite common to have students yell out in confusion, "We had a homework due today?"

CalendarG, developed by team The Running Frogs, which consists of Sophia Kim, Jeffrey Suen and Keanu Williams as the project developers, is a web-based application seeking to assist students with managing their homework assignments as well as other important matters, such as projects, tests and quizzes. Using this application, Students will be able to utilize the functions provided such as adding, viewing, editing and removing lists of things that they need to do. Students will also get reminders every so often to stay on track and not miss a single deadline. Prior to the actual development, we are considering using Javascript and Node.js in IntelliJ IDEA, but will modify accordingly as needed along the development journey.

B. Requirements

1. Security and Privacy Requirements

The app will require valid login information such as username and password prior to accessing the user's relevant information stored in the system's database. To strengthen the security perspective of the system, it will be necessary to implement a secure authentication mechanism that checks whether the login information that's entered is valid. In addition, we will set requirements for the user's password, such as setting the minimum number of characters to at least 8, and combining special characters, numbers and alphabets to satisfy the requirement. The passwords will then get hashed using the Message Digest Algorithm and get stored into the system's database. Another functionality that

will be implemented is locking the account after 5 failed attempts at logging in. If a user gets locked out of their account, they may seek to recover their account (if they are the original user and not a hacker with malicious intent), which could be through answering security questions or entering unique codes sent to their phones or emails (which could be stored in the database after hashing). The system also will have a session timeout, which means that it will automatically log the user out after 15 minutes of being idle. To keep track of security flaws and any issues that may arise during development, our team will use the GitHub issue tracker and effectively use the labels (e.g. bug, duplicate, enhancement, etc.) to indicate the reason for raising the issue, and correctly and quickly address the issues in order to provide better services and experiences to our users.

2. Quality Gates

   The levels of security and privacy within our system can be divided into four parts: critical, important, moderate and low. For the critical-level scenario, an elevation of privilege could occur, where a user or someone with malicious intent may view or modify files that they are not authorized to manipulate, which will cause unwanted and possibly harmful changes in the system. For the important-level scenario we can consider lack of data protection. If the user's information or any data that is important in the system is not encrypted, then it's at a greater risk of being vulnerable and exposed to unwanted parties compared to encrypted data. For moderate-level scenario, the risk could result from the storage of information on a local machine, which results in a persistent location and thus persistent information that resides within. This could provide an easier access to the information as the location and the information in that location are persistent, once the location and what it contains are identified it will make things easier for a person with bad intent to find the information and possibly steal or manipulate the data to their liking. This ties back to the important-level scenario, where data encryption is emphasized so that even if the location of information is identified, there's still encryption to take care of before the person can obtain any information. A low-level scenario would be the lack of notice and consent. The user's information is collected and stored locally without necessarily letting the

user know. Although the information is not shared or published, the users may disapprove of such performances from the system's end.

3. Risk Assessment Plan for Security and Privacy

   In our application, the aspects that will need threat modeling and security reviews would be the storage of valid login information from the user, and the user's confidential information that they may put such as their contact information, as well as ensuring that the login information matches the record found from our database. To assess security and privacy risks that could come up during development, we will first find all valuable assets, such as the user's personal information. Then we will identify the potential consequences that could result from losing the assets. If the consequences include data loss or any harm to the user or to the team, we will identify the threats and their levels. The threats can include malicious human actions, such as stealing of the user data and using it for anything that's harmful. Next, we will identify vulnerability and assess the likelihood of their exploitation and work to get rid of the vulnerabilities or improve upon them. Then we can assess the risk, labeling its threat value as high, moderate or low and come up with a risk management plan to find a solution that will address the threat following the steps listed above. To prevent any further damage or loss, we can come up with a strategy to investigate the cause of the previous threat and try our best to not repeat it.

C. Design

1. Design Requirements

   Since the functionalities that we wish to implement at the end of the development process require a user to be logged in using a valid login information, this ties into our privacy and security requirements discussed earlier, which include setting requirements for a strong user password, session timeout and limited login attempts. Other requirements such as hashing of passwords and personal information before storing them in the database will be handled in backend and don't overlap too much with the functionality of user experience so we will leave that out from this section. To handle the required combination for a strong

password, we will simply prompt the user to create a password that satisfies the requirement shown on the screen. If the user fails to satisfy the requirement, a red box will appear at the top of the page indicating that the user's password has been denied and prompt the user to re-enter a valid password. In handling of session timeout, we will implement a timer function in the backend that checks for inactivity. From the user's end, the user will be able to see a box stating the user has been logged out due to 15 minutes of inactivity and provide them the option to log back in. For failed login attempts, the user will see a red box indicating that the login attempt has failed due to either an invalid username or password up to 5 attempts. After that, the user will see a box that indicates the account has been locked and gives them the option to recover their account by their choice of recovery methods. The design features implemented in the app will be clear and precise to optimize user experience.

2. Attack Surface Analysis and Reduction

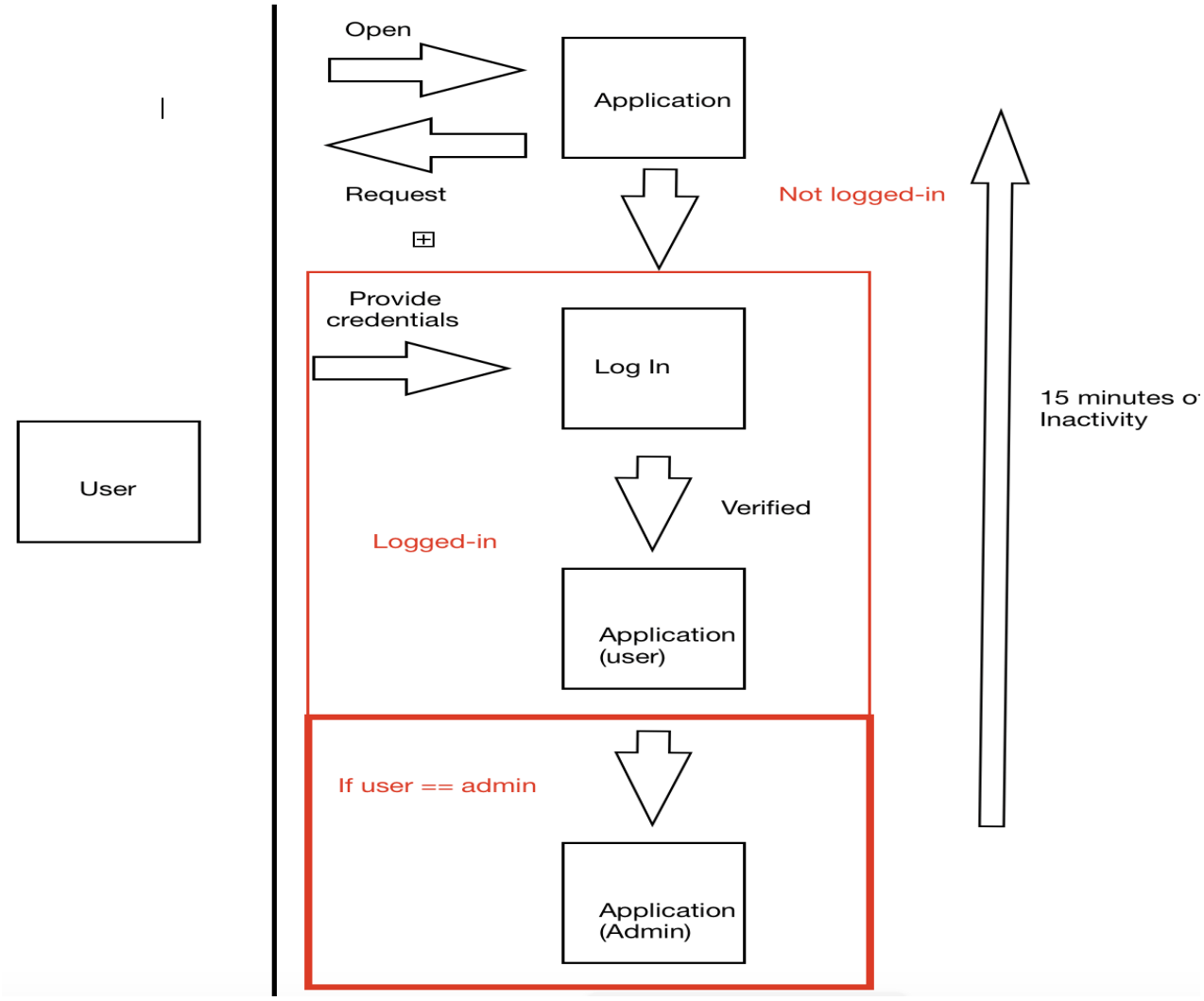
   The privilege levels that the system will have include admin, logged-in and not logged-in. Since this is a web application, the admin will be able to manage the database where all the users' personal information and login information will be saved in. The admin is in charge of the handling of security and privacy aspects of the system, such as unlocking locked accounts after verifying the user's authorization and validity. The logged-in users will have the privilege to view, edit, add, and remove anything that is associated with their own account. A user that is not logged-in have the option to either sign up or log in depending on whether they have an account with our system. The user may also explore certain pages such as the Contact Us or About Us pages. Some vulnerabilities of a similar programs, such as Google Calendar, include allowing interactions between users, such as inviting another user to add an event (e.g. project, club meeting, etc.). This is a vulnerability as a link is usually sent out for the invitation but the invitation could get interrupted and modified to include malicious content. Another vulnerability arises from the fact that the application is web-based. This opens up possible risks involving spoofing of user's information which may not occur to the user or the developers directly. We intend to limit user-to-user

interactions as little to none, and stay away from collecting highly sensitive information such as the user's credit card numbers or physical address.

3. Threat Modeling

   Possible threats that could arise within the usage of app include unsecure network where it opens up the risk of hackers obtaining the users' sensitive information and thus allowing the chances for violating authenticity, integrity and confidentiality in the STRIDE threat model. Since it will be difficult to prevent this issue without implementing a secure connection feature, what we could do is to limit the information we collect from the users to something that is safe enough to be exposed to the public or to a third party. And hashing the sensitive information such as the user's passwords will also make it harder for the hackers to decrypt the information they collect. Another threat could result from the user being in an unsafe environment and forgetting to log out of their account. This could violate confidentiality, authorization and authenticity in the STRIDE threat model. We can prevent this threat by implementing the session timeout feature, so that the account gets automatically logged out if the session has been inactive for more than 15 minutes. The threats that are listed above could be considered a mild threat, which would result in a score of 0.5 on the given threat analysis by Microsoft. (Diagram is shown on the next page)

Application Diagram:



D. Implementation

1. Approved Tools

Compiler/Tool	Minimum Required Version and Switches/Options	Optimal/Recommended Version and Switches/Options	Comments
Javascript HTML CSS Interpreter	IntelliJ IDEA Version 2018.1	IntelliJ IDEA Version 2019.3	
Database	MongoDB Web SQL		May be changed as necessary
Source Code Management System	GitHub GitHub Desktop		
Web Hosting Service	UH Unix Webspace MeteorApp		May be changed as necessary
Frameworks	Semantic UI React		
Browser Developer Tools	Chrome 80.0.3987.116/Firefox 4	Chrome 80.0.3987.116/Firefox 73	

2. Deprecated/Unsafe Functions

Deprecated/Unsafe Functions	Alternative Methods
document.write() – HTML, Javascript	none
document.writeIn() – HTML, Javascript	none
.innerHTML – HTML, Javascript	none
.eval() – Javascript	none – it's evil
.append() – jQuery	none - not safe for unescaped input
postMessage – Javascript	set restrictions to prevent data leak
location.assign() - Javascript	none
.insert() – jQuery	none
.html() – jQuery	none
.after() – jQuery	none
no wrap attribute – HTML	white-space
embed tag – HTML	object tag
menu tag – HTML	ul tag
center tag – HTML	text-align:center
acronym tag – HTML	abbr tag
isindex tag – HTML	form tag
height attribute – HTML	padding attribute
font tag – HTML	font-family, font-size, color

u tag – HTML	Text-decoration

3. Static Analysis Tool

   There was quite a few options to choose from for Javascript's static analysis tool. There's JSLint, which is a web-based analytical tool that aids in verifying the code quality. It works by returning a message with a problem description and location when it detects a problem. A fork of JSLint, JSHint, is a flexible tool to spot errors and issues. It's able to detect syntax errors and implicit data type conversion as well as leaking variables. It cannot, however, determine whether the software is efficient and correct. Another tool our group found was ESLint, which is helpful when trying to discover strange patterns or lines of code that do not comply with the style guidelines. It allows the users to view the errors without having to execute the code, which saves time.

   After spending some time looking at the pros and cons of each and seeing the actual usages of all three, our team decided to use ESLint as it is the most efficient out of all and all the team members have previous experiences using ESLint in a class. Some pros that we found while using ESLint during the initial development process is that it's very flexible. It allows us to tweak any settings and rules to define error levels to determine what should be a warning or an error and what can be ignored. It also allows us to write our own plugins if we wish to, although they already have many pre-made plugins available. Some difficulties we experienced were that at times it was a bit slow to catch any issues or errors that it didn't notify us until after we executed, at which point we felt that it wasn't the most efficient analysis tool available. It also has an extensive configuration that is required, done by installation through npm.

E. Verification

1. Dynamic Analysis Tools

   i. Google Dev Tools Canary

      Google Dev Tools Canary was one of the dynamic analysis tools our group found. It's one of the Chrome's four development release

channels, which allows developers to execute the program and view and test any changes, new features and bugs that were implemented in the latest version of the program. It's installed separately from all of the other development release channels of Chrome, despite having a very similar interface. Google Canary offers features like Audits and dev tools to analyze the website performance. Chrome Canary allows us to browse the program using a websocket, which is used to view the input and output parameters for the functions embedded in the program such as the log-in and creating a new data in the database. Canary also hashed the password of the user that just logged in or signed up for an account, which is particularly useful in this program as we need to have security feature(s) implemented to provide a safe environment for the users. It was also great to see the parameters being passed into any functions or schemas we created in the program and see how the server and the program are interacting with each other. Some difficulties we encountered while using Chrome Canary are that it can be extremely slow when rendering graphics and loading sites or settings. We've also read complaints online that it breaks completely about once a month and takes a whole day or more to fix, which can be a huge minus for a lot of developers who depend entire on it for their debugging. Some good things, aside from a few we mentioned above are that it gets rebuilt everyday so the bugs are fixed faster than other channels of Chrome which have specific release dates for a new version. Overall, it is a very easy-to-follow and dynamic for our use and we have decided to continue to stick with Canary for our dynamic analysis tool.

ii. React Developer Tools

React Developer Tools is a Chrome DevTools extension for the React Javascript library, which is very specific to our program as we are using both React and Javascript as our main tools. As an extension of the Chrome DevTools, it allows us to inspect the React component hierarchies in the Developer Tools. Two new tabs are added when this

extension is installed, called Components and Profiler. The Components tab shows the root React components and subcomponents on the rendered page, which allows us to inspect and edit the current props and state of the components and their hierarchies. The Profiler tab is there to allow us to record the performance information, such as how long the rendering and completion took. React Developer Tools was easier compared to Canary to understand as it was very specific to the use of Javascript and React components within the pages. It was especially helpful identifying any errors or bugs that occurred during the styling stage of the development, trying to figure out the hierarchies of the components rendering on the page and making the necessary changes in the DevTools to have the page look exactly like our ideal designs. Despite having some advantages over Google Canary due to its purpose and nature, we decided to not use React Developer Tools as our main dynamic analysis tool as we felt that it was mostly focused on the React components rather than the entirety of the webpage.

iii. Iroh.js

Another dynamic analysis tool that our team considered and tested but did not proceed with using after the first few test cases was Iroh.js. Iroh.js is an open source dynamic code analysis tool for Javascript. It provides many features like recording code flow during execution, intercept runtime information and modify program's behavior and allowing users to collect data that is only available during runtime. As many have said online, it allows the user to collect the data of the program, analyze the behavior and manipulates parameters while the program is running, all of which are fantastic features. However, unlike Google Canary and React Developer Tools which are either a browser or an extension of a browser, it requires us to know what the Iroh.js API is doing and utilize that knowledge to implement the analysis within specific functions of the program rather than the entire program itself. Hence it sometimes took longer to understand what it was actually doing

or how it worked rather than analyzing the running program. Despite its great uses, we decided to not use Iroh.js as our dynamic analysis tool.

2. Attack Surface Review

Compiler/Tool	Minimum Required Version and Switches/Options	Optimal/Recommended Version and Switches/Options	Changes from Assignment 2	Comments
Javascript HTML CSS Interpreter	IntelliJ IDEA Version 2018.1	IntelliJ IDEA Version 2019.3	Version 2019.3.3	No vulnerabilities
Database	MongoDB Web SQL		MongoDB Only	No vulnerabilities
Source Code Management System	GitHub GitHub Desktop		No changes	No vulnerabilities
Web Hosting Service	UH Unix Webspace MeteorApp		No changes	No vulnerabilities
Frameworks	Semantic UI React		No changes	No vulnerabilities
Browser Developer Tools	Chrome 80.0.3987.116/Firefox 4	Chrome 80.0.3987.116/Firefox 73	Chrome 80.0.3987.132/Firefox 74	No vulnerabilities

4. Fuzz Testing
    a. First Attempt – Phishing Email

    We attempted to send out fake phishing emails to the accounts that have been set up, notifying the users that there is an important announcement that they should log into see and providing a link for them to click. Upon clicking the link, a new window will show up that simply asks the users for their account information such as their email and password, but no further action is needed. Although in this case the page link provided in the phishing email is a part of our web application (that will later be discarded), and the information that the users enter get saved into our own database, if the phishing emails were to be sent out from people with malicious intent then users' credentials would be at risk. This resulted in the success rate of 70% as most of the users that received the email clicked the link and provided their login credentials (thinking that they were logging into the site). To prevent any security breach related to this matter, it would be important to have the users be aware that any changes in the TOS would be visible to anybody so there shouldn't be any emails getting sent out asking them to click the link and log in right after. It would also be important to secure our email account so that it doesn't give the hackers the access to the company's email.

    b. Second Attempt – Brute Force Attack

    We used brute force attack tools to break into the application, attempting to guess the user's password correctly given thousands of possible passwords. One of the tools we used was THC-Hydra, which is well-known for cracking online passwords using the wordlist built in the system. After a few minutes, the return statement from Hydra indicated that the password was cracked for our admin account. The success rate is at 100% as it was able to figure out the password in just a few minutes. This would be addressed by requiring the users to build more complicated passwords upon signing up for our application, which has already been implemented at this point of development. It would also be important to lock the users out after

numerous login attempt, as someone could easily go to our application and try to break in using brute force, and if the system doesn't lock the user out and notify them of the numerous changes then it will have the vulnerability open and visible to anyone, which puts the accounts at risk as hackers could try to log in using this method.

c. Third Attempt – Locate Config folder

The last attempt at breaking into the application was by attempting to locate the config folder within our system. The config folder holds the schemas and collections which the user account details are stored, which are crucial information such as physical address of the users. Also, if a hacker were to obtain the location of this holder, the application itself is at risk as it is vulnerable and open for any possible attacks including modification of data and code. We used Chrome Canary Developer Tools to locate the file, but the attempt resulted in failure as Meteor detects such folder and files and keeps them hidden. Even after attempting to enter the directory by embedding it within the URL, the location of the folder could not be detected. It is convenient that Meteor has these security features to help the developers from fearing yet another possible security breach. Since the attempt was unsuccessful, the success rate is 0%.

5. Static Analysis Review

As mentioned in the Static Analysis Tool portion of the report in the previous development process, our team has been using ESLint as our main static analysis tool due to the familiarity and effectiveness. Using ESLint and utilizing a plugin that is the best fit for our purposes and coding style, ESLint has allowed us to keep the lines of code more consistent and cleaner, while also guiding us to use certain functions and variable types as necessary. ESLint has helped us to double check and research before proceeding with writing code that can be a bit confusing or uncertain, because we believe that without having this tool we would have easily overlooked a lot of possible warnings and errors within our files.

6. Dynamic Analysis Review

   As mentioned in the Dynamic Analysis Tool portion of the report, our team has been using Chrome Canary Developer Tools as our main dynamic analysis tool due to its convenience. Overall, Canary allows to seek for any possible security breaches that could be occurring within our system by showing us the network data while performing various actions on the application such as logging in and verifying the email credentials. So far we have done all possible implementations to prevent any sensitive information being visible to the public eye, by using bcrypt to hash the passwords when it enters the database. We have been using Canary every time a new feature gets implemented to see if there will be any important information that will be visible on the network data, but so far we have yet to come across any aspect that requires fix immediately. We will continue to use Canary to seek out for vulnerabilities until the end of the development process.

F. Release

1. Incident Response Plan

   Privacy Escalation Team

   In case of any possible threats to calendarG, the Privacy Escalation Team will be divided accordingly to handle each aspect:

   -**Sophia Kim (Legal Representative)**: Sophia will handle any concerns regarding the legal aspects of the threat, which may include lawsuits, seeking for any unclarity or issues within the company's TOS, and ensuring the company's compliance of rules and regulations. When the threat occurs, she will first consult the rest of the members of the root of the security or privacy breach that has occurred and take necessary steps for any legal actions.

   -Jeff Suen (Public Relations Representative): Jeff will handle the issues regarding the public relations. His tasks will include dealing with the responsibilities such as reaching out to the users to receive feedbacks, cooperate with the public to address the issue in the best interests of both the company and the users. He will also oversee notifying the users of the threats that had occurred and drafting the changes in TOS, if any.

-**Keanu Williams (Escalation Manager)**: Keanu will recruit security, privacy and business experts and completing the privacy escalation following the appropriate procedure and ensure that the same threat does not occur again. He will oversee the overall process of privacy escalation and devote his time to delegate tasks to each team member. He will evaluate the severity and, along with the recruited members of the team, will organize the steps needed to resolve the issue.

Point of Contact

In case of emergency, the users may reach us at: therunningfrog.biz@gmail.com to seek possible solutions to the issues they are having or address any security or privacy concerns.

Privacy Escalation Procedure

i. When the issue has been notified to the escalation manager and the team, the team under the manager's lead will evaluate the content and decide whether more information needs to be gathered.

ii. Information about the issue, such as where the escalation came from, how much of an impact and threat it is, the validity of it, and known facts of it, will be put together so that the manager can distribute the information to appropriate branches or personnel to come up with solutions.

iii. Depending on the branch or the personnel, the solution-seeking process may differ, but should fall under one of the categories listed:

    1. Managing the security and privacy sector of the company to prevent future threats.
    2. Reaching out to the public for help articles, notification of the threat, and changes in TOS.
    3. Training and enforcing effective and efficient communication mechanisms for better flow of the overall work experience regarding protecting privacy and guaranteeing security to the users.

iv. The escalation manager should ensure that all aspects of the escalation are addressed and resolved as planned. After the solutions have taken place, the team should evaluate the effectiveness of each approach taken to solve the issues. For the approaches that became effective solutions should be kept as

a record so that a similar threat does not occur again, and also to ensure that there is a solution that works for any possible threats in the future.

2. Final Security Review

   Some threats that were identified in the beginning and during the development cycle include the user being in an unsafe environment where an intruder can use the user's account and modify the data, user's personally identifiable information getting exposed and the permanent storage of the data within our local machines. After reviewing the threat models as well as the static analysis, dynamic analysis and quality gates, we have concluded that our team should obtain the "Passed FSR with exceptions" grade. Since the beginning of the development cycle, our team has focused and spent tremendous amount of time on protecting the users' privacy and guaranteeing security when using our application. Thus, we have implemented functionalities such as tracking the user's password strength, verifying their email address, encrypting their password and other personal information, and giving them the ability to change their email address and password under a secure environment. For the most part, we have utilized ESLint and Google Canary to ensure that the attack surface is minimized and ensure the security functionalities we have implemented are working properly. However, due to the given circumstances, we were unable to implement all the functions that we had in mind originally, one of them which includes the session timeout feature. Despite following and completing all steps in the Secure Development Lifecycle, being unable to seek solutions for all possible threats would put us under the Passed FSR with exceptions. If the time allows for future development, we would like to fully implement all security features to pass FSR and therefore certify that our app is secure to be released.

3. Certified Release & Archive Report
   i. Link to the release version of the program: https://github.com/The-Running-Frogs/calendarG/releases/tag/v1.0
   ii. Features
      1. Create and edit events related to personal or academic life.
      2. Receive notifications for reminders for the events via the application or the email.

3. Change the account settings for better protection.
4. Email verification process to ensure authenticity of the account.
5. Password strength tracker to ensure that the users use passwords that are not easy to crack.

iii. Version Number

For the current release, the version number is 1.0.

iv. Future Development Plans

1. We would like to finish implementing the session timeout and locking out accounts after some failed attempts at logging in.
2. We would like to take the calendar functionalities further and implement a scheduler feature to possibly compete with calendar apps designed by startup companies.
3. We would like to incorporate a system like Laulima, where it can be used by both students and professors to send out reminders for certain assignments while the students keep track of their personal life so everything is all kept in one place.

v. Technical Notes

1. Installing the application
    a. Install Meteor (https://www.meteor.com/install).
    b. Download a copy of calendarG (https://github.com/The-Running-Frogs/calendarG) or clone it using git.
    c. cd into the app/ direcotry and install libraries with: *meteor npm install*
    d. Run the system with *meteor npm run start*
    e. If all necessary libraries were installed, the application will appear at http://localhost:3000.
    f. Sign up for an account with calendarG if you don't already have one.
2. Specifications for use
    a. calendarG is mostly targeting students as the audience, as being a student can get overwhelming with the amount of

work and events they get. calendarG strives to assist students in having an organized and balance life, by allowing students to keep all their life events, assignment and assessment due dates all in one page. However, anyone who seeks to have a balanced life with a simple, friendly calendar app can use calendarG as well.

i. When you first go to the site, user will be greeted by the landing page that explains what calendarG does.
ii. The user can create an account, or log in if they don't have one.
iii. After creating the account, an email verification will get sent to the user's email.
iv. The user must click the link provided in the email to verify their email address.
v. After the account has been verified, the user's home page will show a blank calendar.
vi. The user will be able to add and edit events.
vii. To access account settings, users can click on their name on the top right corner and click "Settings"
viii. In settings, the user can change their username and password, and change the notification options.
ix. When they're done using the app, they can sign out, and they will be redirected to the landing page.