Assignment 1 - Requirements and Design

A. INTRODUCTION

1. **Team Name:**

   Team Goguma

2. **Team Member List:**

   Gian Calica

   Mercedez Castro

   Tysen Imai-Toyama

   Hyosun Song

3. **Application Title, Description, and Functional Requirements Specification**
   - Application Title: Barbell Buddy
   - Description: This app allows you to focus on your workout while it handles all the math to effortlessly calculate which weight plates to add to the barbell based on your available equipment, desired workout, and lifting goals.
   - Functional Requirements:
     - Account and membership creation, access, and retrieval capabilities.
     - Allow data input by user, such as name, age, weight, etc.
     - Workout and rep plan generator based on user input and desired goals.

- Calculate and display the value and amount of plates to load on the workout bar based on user input.
- Ability to customize units of measurements, such as pounds or kilograms.
- Ability to customize the equipment inventory based on plates available to the user.

4. **Type of program e.g. Browser Extension, Web Application, ..etc**

   Web application

5. **Development Tools**

   The selected language that we chose to create our web application is Javascript. We will also be utilizing Jetbrain's Intellij/WebStorm for our IDE as well as GitHub for our version control collaborative tool.

B. REQUIREMENTS

1. **Security and Privacy Requirements:**
   - Establish and define your security and privacy requirements for your program.
     - User ID must be unique.
     - Require a SHA256 hash on all passwords ensuring that no plain-text passwords are stored.
     - Sensitive data must be transmitted with the POST protocol and must not be stored in persistent cookies.

        - There must exist a process to validate a user's login information and enforcement shall be executed to manage the access to restricted information.
        - Comprehensive account management methods shall be established to identify account types and user levels with the ability to remove, disable, and secure accounts.
        - The system shall only collect and use Personally Identifiable Information (PII) relevant to its purposes.
        - Privileged and super-user accounts (Administrator, root, etc.) must not be used for non-administrator activities.
        - The system shall keep historical records and logs of events and processes executed by the application.
        - Integrity of the system must be protected from corruption, tampering, destruction, overwriting, and malicious insertions or deletions.
    - Also, come up with a system of keeping track of security flaws that may arise throughout the development process. Describe this plan/system in your report.
        - We will keep track of security flaws through Github issues. When a security flaw is found, an issue is to be made for correcting the security flaw that is found. In a sense, using Github's issues as a security flaw todo list.
        - Each security flaw will be documented with a label corresponding to the level of security and privacy as it relates to our program as defined in part B, question 2 below.

2. **Quality Gates (or Bug Bars):**
    - Define levels of security and privacy for your program.

- Critical level
    1. Security: Server permissions and data is compromised by a 3rd-party. Execution of code that is not from the program itself
    2. Privacy: Lack of notice/consent for PII to be transferred. Insecure transfer of PII, via no data protection or even improper use of cookies. Also the ability of opting in/out. Lack of ability for the user to stop collection and transfer of their data.
- Important level:
    1. Security: Denial of service attacks against a high value asset. Elevation of privileges to manipulate sensitive information. The ability to gain access to information anywhere on the system. Tampering of other users' information.
    2. Privacy: Same as critical level with the addition of data minimization, where the user's PII will not be disclosed to 3rd-parties if not necessary for the disclosed business purpose
- Moderate level:
    1. Security: Denial of service attack in general. Spoofing other user accounts. Security assurance, where users or programs can't have higher privileges or control unless it is given to them/it on purpose.
    2. Privacy: PII can be accessed by other people. Cookies persist past it purpose. No retention policy for data stored.
- Low level:

1. Security: Memory leaks, such as leak of random heap memory. Temporary modifications within a situation that does not persist after restarting the application.
2. Privacy: PII is not accessible by others if a folder is shared. PII is hidden within metadata locally without discoverable notice.

3. **Risk Assessment Plan for Security and Privacy:**
    - Security will be assessed by attacking the application for common security flaws that happen within the programming language and application types. Such flaws are to be ranked and identified. Security reviews and threat modeling will have to be for user input and network security of sensitive information being transferred.
    - Privacy will be assessed as to how information will be handled. A checklist as to the privacy impact rating of our application. Such a checklist will determine how invasive our application is to the user, an example being “Continuously monitoring user data” or “transferring of anonymous data”. Items on the checklist are to be analyzed for probability of happening within our application. The checklist goal is to have none of the invasive practices done to our users, thus keeping a good relationship with our users. These privacy assessments are to help lower future costs of the project due to conflict with the user base. Possibility of working with a privacy advisor due to our inexperience of dealing with privacy issues. Thus our draft of a privacy disclosure can be made and reviewed by said privacy advisor.

C. DESIGN

1. **Design Requirements:**

        - The user interface and design should exhibit uniformity and consistency to produce a visually pleasing user experience.
        - The design should be structured and straightforward to mitigate user confusion.
        - The design architecture should be modular to accomodate for changes and maintainability.
        - The design must exhibit extensibility to allow for added features.
        - The implementation of the design should ensure software capability for common media access.
        - The software must be designed to operate under invalid or unpredictable user behavior.
        - The design should allow the application to execute tasks within an acceptable time frame
        - The application must be able to perform the required functions without requiring a large amount of memory.
        - The software must be designed to detect, log, and combat against malicious actions and behaviors.

2. **Attack Surface Analysis and Reduction:**
    - Determine privilege levels for your users and perform an attack surface analysis.
        - There are two levels of users in this application: the regular user level and an administrator level. The administrator level will have read and write privileges to the data of all regular users. This data will include stored information about the user, including their saved app configurations and username. However, administrator levels will not have permission to read or write a user's password. The

regular user level will only have read and write privileges to their own data and not any other user's data.

- Attack Surface Analysis (Scale of 0.1 to 1.0 with 1.0 being high threat):
    1. Account Creation - 0.5
        a. We will most likely use a library to help with account creation, therefore the security of the process when creating accounts would depend on this library for the most part.
    2. Input Format - 0.8
        a. Because we will be taking a lot of user input, failure to sanitize and check the user inputs might allow an attack to do injection attacks.
    3. Profile Information Exposure - 0.2
        a. Users can customize their profile to input their PII. Attackers might access by grabbing the modification request to the database, escalated privilege (somehow have read privilege of other users), or having access to the victim's account.
    4. Escalated Privileges - 1
        a. Attackers might somehow get access to the administrator level privilege that allows them to read and write other user data.

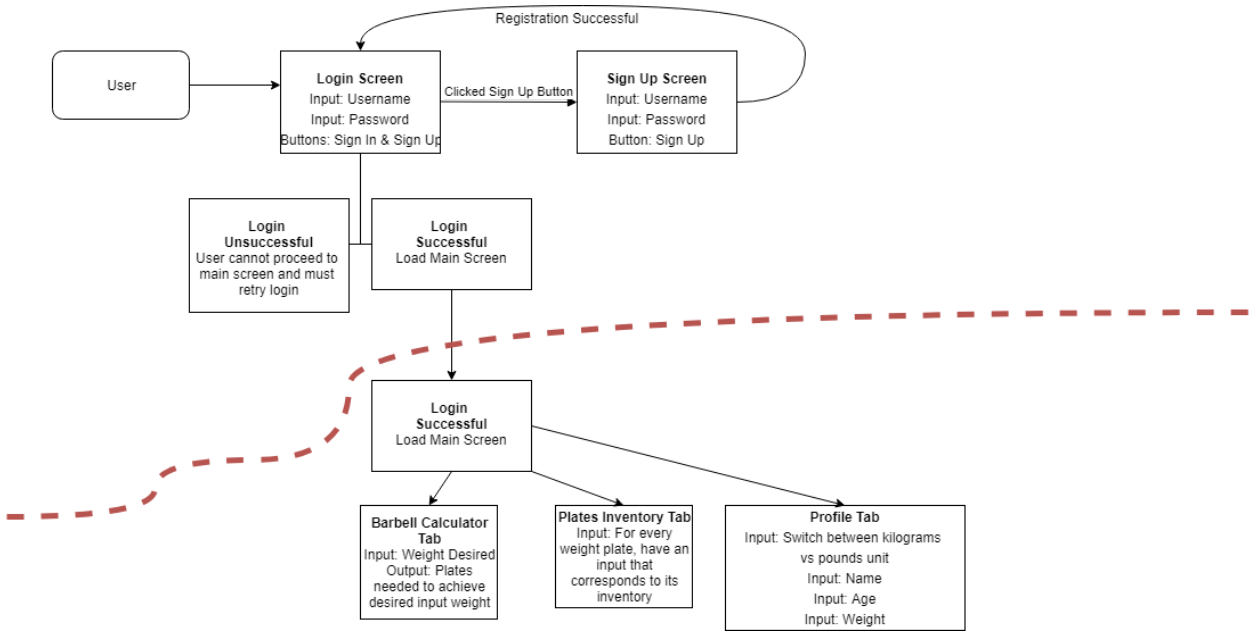
- Since this is the first version of your program, compile a list of vulnerabilities from a similar program, that could apply to yours. Using a similar program as a vulnerability reference will give you an idea of how your program may be vulnerable. Consider the ways a malicious user may attempt to exploit the code or other aspects of the program.
    - Package Dependencies

1. We will be utilizing various libraries and resources to efficiently implement assorted functionality and features. Although a majority of the libraries we choose to employ should be reliable in terms of security, there always exists a possibility of a package posing some sort of vulnerability to our program.

- Client-Side Injection Attacks
    1. Since our application relies on user input, we must ensure that our program regulates and sanitizes all user input. Not doing so may leave our program exposed to malicious insertion, deletion, and modification of data in our program.

3. Threat Modeling:

- Create a threat model for your program.



Threat Categories (based on the ISO 7498-2 model):

- Destruction of information and/or other resources

    - An entity could obtain write privileges to other user data and destroy their data. Destruction of information could also occur through other vulnerabilities such as injection attack. And in an extreme case, an attacker could potentially destroy our entire database. Since our database is stored in the cloud provided by some service, they would have to attack the service provider.
- Corruption or modification of information
    - An entity could obtain write privileges to other user data and modify their data. A corruption or modification of information could also occur through other vulnerabilities such as injection attack. An attacker could also use some weird characters for the user inputs that our application cannot read and process, potentially corrupting information in our database.
- Theft, removal or loss of information and/or other resources
    - A user could attempt to send data to our database but the request fails to send through. A loss of information can occur if an error is not shown to the user and the user is allowed to have thought that their request has been processed. A loss of information could also occur if the database fails to send requested information to the user.
- Disclosure of information
    - Users should only be able to see their own user data. Non-logged in entities should not be able to see any user data. A possible threat, whether it's from a stray console log or incorrect access privileges to our database, could expose this data to an entity that should not have read access to it.
    - An incorrect access privilege to our database might lie in our code logic where we do not exclude non-logged in users from accessing our database. Or our code logic does not check the user that it sent the data to actually belongs to that user.
- Interruption of Services
    - Since this is a web application and user data is stored on the database, a threat is attacking the cloud server our database is stored on. An attacker could also attack the server hosting our web app, therefore preventing a user from using our application.

Assignment 2 - SDLC: Implementation

1. Approved Tools

Compiler/Tool	Minimum Required Version and switches/options	Recommended Versions and switches/options	Comments
Intellij Ultimate	Ver. 2019.3.0	Ver. 2019.3.3	IDE used for development
Node JS	Ver. 12.16.1	Ver. 12.16.1	Javascript runtime environment
Express JS	Ver. 4.17.1	Ver. 4.17.1	NodeJS Web Application Framework
Npm	Ver. 6.13.4	Ver. 6.13.4	Javascript package manager
ESLint	Ver. 6.8.0 npm install eslint --save-dev	Ver 7.0.0 npm install eslint --save-dev	Static analysis tool. Can be used within Intellij instead.
React	npx create-react-app “appname”	npx create-react-app “appname”	UI Library for Javascript
Semantic UI	npm semantic-ui --save	npm semantic-ui --save	Frontend Framework
Passport JS	npm i passport	npm i passport	Authentication Middleware

2. Deprecated/Unsafe Functions

a. Dependencies

i. As our web app is Javascript based, we will be using Javascript packages to do some functionalities that we don't need to code from scratch. These packages are also built using other Javascript packages, making the packages that we choose to use having dependencies. An issue that could arise is compatibility issues. A package that uses another package as a dependency must make sure that it has the updated dependency so that it does not break the package itself. Another issue that could also arise is that a security vulnerability in one dependency would then also be present in the package that we choose to use.

ii. Our general alternative to these problems is to use another package that accomplishes the same functionality that does not have the same problems. Another alternative is to simply code the functionality ourselves that the unsafe dependency would have provided us.

b. Express JS

i. Express JS is the web application framework that we will be using to create and balance our server on. It is also the framework that we will be leveraging to connect and interact with our web app's API. Looking at Express JS past vulnerabilities, most of its vulnerabilities are Denial of Service attacks. Most recently, there were Regular Expression Denial of Service Attacks. This is concerning as the majority of our functionality uses user inputs to function properly.

ii. As Javascript is a popular language, there are many alternatives to Express JS. Our other Javascript-based alternative is to use Koa. If we end up with an underlying problem that is just related to Javascript, we have also considered using other web application frameworks from other programming languages. The other

alternatives are Python's Flask and the Golang programming language. The fantastic thing about the Golang programming language is that it has a lot of already built in library functions to make web applications that can accomplish the same tasks as ExpressJS.

c. NPM
    i. Packages can be infected with malware that steals information from computers. This is done from backdooring through popular packages or uploading a package that seems safe but isn't. Not all packages are to be trusted fully, in which knowing the amount of downloads and if it is still being maintained is a good first sign of a safe package.
    ii. The solution to this problem would be having to code the functionality that the package originally provided ourselves to prevent such vulnerabilities or doing research on the packages we intend to use before installing them. Using yarn to download our packages could be another alternative due to its checksum mechanism to verify the integrity of every installed package before code is executed. Although it would be most recommended to use yarn over coding everything ourselves from scratch to save time and resources for development.
d. IntelliJ Ultimate Updates
    i. IntelliJ is one of the biggest and most popular IDEs used worldwide. It is also a very active development project, often having major updates monthly and weekly minor updates. These updates, minor or major, might introduce a bug or two that affects our development process. It impacts our development process in two ways: 1) introduced security vulnerabilities within the IDE and 2) breaks IDE functionality that aids in code analysis. Although it is

just an IDE, IntelliJ is thoroughly integrated with other online third party software. As with any software, it has had its history of security vulnerabilities introduced in patches. It impacts us in a second way such that minor updates can break IDE functionality that aided us in code analysis. IntelliJ Ultimate is a very smart IDE and is integrated with many languages, including Javascript. It has a lot of smart tools and functionality that aids us in developing Javascript code, many of which help prevent us from writing vulnerable code. Therefore, unknowingly updating to newer versions of IntelliJ Ultimate might affect our development as we develop throughout the semester.

ii. The solution to this is simply not updating beyond what is currently listed in the recommended version for IntelliJ Ultimate. If they do update, they should rollback to the recommended version. Otherwise, if a vulnerability is found in the current recommended version listed, then our alternative is to also rollback our recommended version to a older version. Another alternative is to simply use another IDE, either VSCode or Atom.

3. Static Analysis

a. The static analysis tool we chose to use for our web app is ESLint. We didn't really have any difficulties with this tool as we have all used it previously in other classes for analyzing Javascript code. We've worked with this tool, both in class and outside of class in our own work experience, therefore we are proficient enough to configure it on our own to suit and fulfill our needs. It is a syntax checker, format validator, and also enforces good coding patterns. It can also detect any coding patterns that are considered unsafe or outdated for Javascript.

Assignment 3 - SDLC: Implementation & Verification

1. **Dynamic Analysis**
    a. The dynamic analysis tool we decided to use is called “Iroh.js”. This tool has an API to implement into our code. The analysis is done by using the IrohJS API. The API calls for formatting your code as a string variable and constructing a stage object by using the code string as a constructor parameter. From there, you can then add listeners via the stage object. These listeners range from variables, loops, if-statements and so on. The listeners can be activated from certain parts of the code, such as the return, after, enter, leave. As you can already tell, leave would mean leaving the loop, function or if-statement and return would mean the return value of a function. Then the stage can be evaluated by putting it within an eval() function. Thus, when running the eval() function, text will be printed from the information taken from the stack during runtime of the code. You can also format nested loops or if-statements using the “ “.repeat(e.indent), where e is the variable from the callback function in your event on function where it declares the part of the code to be listening to. When executed, the runtime values will be shown in the console.
    b. Prior to this tool, we mainly just used console logs (console.log()) to monitor the values of our program during runtime. This method was fine for the majority of our development and we were confident that through the console logs that we had the right values that we expected and the components of our web app behaved correctly. However, there were certain functionality in our web app that got complicated to monitor and in those cases is when this tool was really helpful. Such cases were complex *for* loops and deeply nested values in these *for* loops. In these cases, especially in our cases when the *n* for these *for* loops were big, the

outputs of using the regular console.log() became very chaotic and unreadable. This tool allowed us to be able to actually understand readable traces of how our loops behaved and the resulting values of each iteration. Especially in the case of being able to know the specific value of a certain variable at a particular iteration of a loop was very helpful in debugging at times when edge cases appeared.

2. **Attack Surface Review**
    a. One of the packages that we use is called ‘create-react-app’ which was used to bootstrap the client side of our web app. This package uses a package called ‘react-scripts’ that manages the scripts and configuration used by ‘create-react-app.’ On 3/07/2020, we noticed that after NPM, our package manager, audited our packages for any vulnerabilities, we had 17 moderate-level vulnerabilities and they were all through a package called ‘acorn’ which is a dependency used in ‘react-scripts.’ All these vulnerabilities were under “Regular Expression Denial of Service.” There was also a GitHub issue made about this on the same day we discovered it. It looks like this is a NPM issue because according to the discussion in that GitHub issue, the warnings can be resolved using Yarn (another package manager that is an alternative to NPM). But we aren’t sure if it actually solved the problem of the vulnerability or if it merely silenced the warnings. After a few days, there was some more discussion regarding this and a new patch of ‘acorn’ came out that fixed this vulnerability. We were then able to update our packages that resolved the vulnerabilities. At that time when there was no patch to the vulnerability yet, we had decided on solutions if the vulnerability were to persist. This solution included running a command called ‘react-scripts eject.’ Because ‘create-react-app’ was only used to bootstrap our project, it encapsulated all of the modules

it uses internally, hides them under the hood, and uses ‘react-scripts’ to manage the configurations instead of us having to manage it. By using this command, it opens up this hood and we would no longer need ‘react-scripts’, which has ‘acorn’ as its dependency, to manage our configurations and scripts.

Assignment 4 - Verification

1. **Fuzz Testing**
    a. **Postman**
        i. Postman is a popular API client that makes it very easy to test APIs. It allows developers to easily create simple and complex HTTP requests to a server, including the ability to modify headers, and see the response back in a quick manner.
        ii. This tool was very successful in detecting and monitoring to see if our API and its CRUD (Create, Read, Update, and Delete) operations worked as we expected. We really only had two resources of our API that we needed to test: Users and WeightInventory. Users is the collection of users registered in our app and WeightInventory is simply the inventory associated with a user. Since WeightInventory is associated with a user, it means that pretty much all of the requests to that resource should naturally be authorized properly. Using Postman allowed us to easily test a bunch of requests, that contain with or without an authorization token, and isolate this testing simply on the server side as Postman directly calls to the server. Because Postman allows us to easily send requests and modify the headers of our requests to the server, we didn't have to worry about writing client-side code before we fully tested our API on the server-side. Therefore, we could easily send many types of requests to our API, including reasonable calls that we expect to succeed or reasonable calls that we expect to get an error status code of and also unreasonable calls where we feed the request random data and see how it behaves with our API.

iii. Our API was also pretty robust and specific when it came to handling requests such as sending back a more specific HTTP status code rather than just a generic 4xx HTTP status code. Using Postman, we attempted to break into our program using specific HTTP requests that we know should send back the correct HTTP status code that we implemented in our API.

b. Property-based Testing

i. Property-based testing is a testing technique where it is designed to test the aspects of a property should always be true. This allows for a range of inputs to be programmed and tested within a single test. For example, given that an input takes a string, the property of that input would be a string and property-based testing would test all the possible range of inputs that satisfies that property.

ii. We used a popular Javascript library called fast-check which is a property based testing framework. Rather than having to write multiple tests that take on a range of different possible values, this framework allowed us to write a single test that would test a function or input and give it all the possible values. In our case, our inputs in our app were only numbers. Our actual implementation would limit numbers that weren't valid such as negative numbers or numbers that were too high. Rather than having to make an assertion case for each limitation and also test a range of different values (which would be the fuzzing part), this framework allowed us to write a single test case that inputted all the possible range of number inputs into our functions and from there we can see if our code properly handles the invalid inputs properly.

iii. This technique is essentially accomplished in automating fuzzing of our app in a very easy and minimal lines of code way. Fuzzing our program using this technique would prove to be a lot more

reliant in making sure our app handled edge cases properly and imitated a hacker breaking into our program with a range of different inputs.

c. **Manual End-to-End (E2E) Testing**
    i. E2E Testing is a technique that tests the entire software from beginning to the end to ensure the application flow behaves as expected. It stimulates a real user scenario and validates the system under test and its components.
    ii. We didn't have automated E2E testing set up for this assignment, so we just did the typical manual E2E testing where we go through our app, input expected and unexpected values into any forms, and observe the changes manually. In terms of fuzzing or trying to hack our system to see how it would behave, this is a lot less effective than the property-based testing technique for the obvious reason that it isn't automated. However, compared to property-based testing which tests code in isolation or groups of a few code blocks, manually doing E2E testing was still helpful in observing our front-end specifically and making sure it behaved as expected. We manually inputted expected values that the app would expect and produce good output and also unexpected or random values that the app should throw an error to. In that sense of testing the front-end, this technique did accomplish its goal.

2. **Static Analysis Review**
    a. We still use ESLint as our only tool for static analysis for this assignment. Now that we are working with HTTP calls, specifically Promise-based calls (Promises in Javascript are deferred and async computations), ESLint has

actually been really useful for us in detecting code that would fail before we even ran our code. Since we're inexperienced when it comes to handling requests on the server side and the client side, we aren't really sure how to handle requests/responses properly or know exactly how the requests/response bodies look like. In those cases, ESLint has been really productive in warning us if we tried to access a value of a response body that doesn't exist. This has really helped us making our application more bug-free and helped us in debugging our code.

3. **Dynamic Review**
    a. Our dynamic analysis for this assignment differs from the previous assignment. Previously, we used a framework that specifically did dynamic analysis on source code. For this assignment, we just leveraged unit tests and integration tests to do our dynamic analysis. The techniques we discussed in the Fuzzing Testing section above are essentially the same techniques and tools that we use for dynamic analysis of our code. Postman tested our server-side code and API. Our unit and integration tests utilized property-based testing to accomplish fuzzing and monitoring our code behavior in runtime.

Assignment 5 - Release

1. **Incident Response Plan**
    - **Privacy Escalation Team**
        - **Escalation Manager:** Responsibilities include ensuring that the organization's incident response procedures are followed through promptly and efficiently and that the necessary individuals and representatives are included in order to resolve the problem. This individual is in charge of evaluating the incident and assigning incident response tasks to other representatives respective of their duties until the problem is resolved and completed. At the end of the procedure, this individual is responsible for evaluating the effectiveness of the response team and working with the reporting party and any other contacts or representatives involved to close and remedy any issues.
        - **Legal Representative:** Tasked with resolving any legal matters concerning the organization and helps to establish an attorney-client privilege surrounding the incident. This individual should be able to provide legal documentation and information required in compliance with a court order or warrant and should be responsible for the handling and collection of any data or information to be used in a legal proceeding. Ideally, this representative should be accessible around the clock for guidance and advice surrounding legal affairs.

    - **Public Relations Representative:** In charge of helping to resolve any public facing concerns throughout the incident procedure and is responsible for the planning and execution of public internal and external communication. This individual is also responsible for publicly representing and promoting the reputation of the organization throughout the process.
    - **Security Engineer:** The main focus of this individual should be directed at ensuring that the organization's computer system is running securely and analyzes the network to try and foresee any possible security relation issues. In relation to an incident procedure, this person is tasked with identifying the source of the escalation and the possible impact or risks involved surrounding the issue. As directed by the Escalation Manager, this individual will be tasked with fixing any such breach or disruption to the network.
- **Emergency Contact**
    - In the event of an emergency, a user can reach our team via email stated in our Contacts area of our web app (info@barbellbuddy.com).
- **Incident Procedure**
    - The incident response plan will go into effect immediately once notified of the problem. Any such e-mail or message relating to a possible breach, fault, or issue will be promptly forwarded and handled by the Escalation Manager for evaluation.
    - During evaluation, the Escalation Manager will be responsible for examining the content of the report and determining what the next steps of the process will be and gathering more information to decide what will be required to resolve the incident.
    - The Escalation Manager has the authority to contact and include the Security Engineer, Legal Representative, and the Public

Relations Representative on as needed basis depending on the nature of the incident.

- The first focus should be directed at finding the source of the escalation and its impact on the system. The Security Engineer would be essential for determining the validity of the incident and any possible risks associated with the escalation.
- At the stage, employee outreach would be vital for gathering statements and creating a timeline of events and expectations.
- Once a clear summary of the known facts are collected and any impending risks are negated, the Escalation Manager can assign each member of the response team with individual tasks in relation to their duties.
- Legal and Public Relations Representatives may also be contacted if appropriate to seek resolution should the Escalation Manager deem it necessary.
- Once the Escalation Manager has ensured that the incident has been resolved, it may also be necessary to implement further incident response management training or product/service changes or updates based on the incident and how it was concluded in order to prevent future issues.
- Employee and public relations outreach may also be necessary depending on the nature of the incident.

2. **Final Security Review**

Grade: Passed FSR with exceptions

After conducting a final security review, our team has decided to give a grade of “Passed FSR with exceptions” to our program. First, we discuss the areas of our

security review that led us to the conclusion that our program has reached an acceptable compromise about the SDL requirements.

Our RESTful API consists of two resources: User and WeightInventory. All of the endpoints that these two resources expose are thoroughly tested using both unit and integration tests. These tests make sure that our API sends the appropriate status codes and responses. They also validate against both legal and illegal requests sent to the endpoints. This includes testing against injection attacks and making sure that endpoints that require authorization only send back a proper response body if the sender of a request is authorized to access those endpoints. Our team has concluded that this portion of the security review passes the requirements.

On the client side, our team has tested the functions that calculate the appropriate output for the frontend of our program. We also have both unit and integration tests that validate these functions so that they return the correct outputs for all cases of inputs. These functions were also tested against both cross-site scripting inputs and random inputs. Our tests have shown that our program can negate these kinds of malicious inputs and also not let them compromise the user. For this portion of the security review, our team has concluded that it passes the requirements.

Now we discuss the exception to our final security review. In addition to testing both the client and server side separately and concluding that both of these two sides of our program passes the security requirements, we also did end-to-end (E2E) testing that tests how the program would behave if we made both client and server calls at the same time. However, what makes our E2E testing process an exception to the final security review is the fact that unlike the separate testing processes for client and server, our E2E testing process was done manually and not automatically through automated tests. Unlike the automated tests for client and server separately, we obviously could not test as much edge cases in our manual E2E testing process. The primary reason that we could not automate our E2E testing process is because our team encountered difficulties setting up automated E2E tests. We encountered many

bugs that we could not resolve when trying to write our E2E tests. However, we still believe that even with the manual E2E testing process, we have tested it thoroughly enough that we reached an acceptable compromise about passing security requirements for this part of the security review.

3. **Certified Release & Archive Report**
    - **Release Version**
        - **Link:** **https://github.com/ICS427JFam/BarbellBuddy/releases/tag/1.0.1**
        - **Version Number: 1.0.1**
    - **Summary Report**
        - **Features**
            - BarbellBuddy is a web application for gym users and fitness afficanadios alike. This app is aimed at people who utilize a barbell whether at the gym or in the comfort of their own home and allows the user to focus and concentrate on their workout instead of doing arithmetic calculations in their head. This app will handle all the math needed to effortlessly calculate which weighted plates to add to the barbell based on your available equipment.
            - The application allows a user to create their own account. Under this account, they can save their inventory of available equipment to use in the calculator.
            - BarbellBuddy has three main pages that a logged in user can access:
                - **Barbell Calculator Page**: Based on the user's saved inventory of weights and barbell, the user can enter a number that is the weight input into the calculator.

The calculator then proceeds to display which weight plates and how many of them the user needs to load up to the bar based on the weight input entered. The specific weight plates and the amount of weight plates that the calculator outputs is based on the user's inventory which is saved and can be modified in the Weight Inventory Page.

- **Weight Inventory Page**: The user has the ability to customize the equipment inventory used for calculations based on the weighted plates available to the user at any given time.
- **Conversion Calculator Page**: This page allows a user to input a number, either kilograms or pounds, and the calculator automatically converts one to the other unit (kilograms to pounds and vice versa).

- **Future Development Plans**
  - User forum: A place where users can communicate with each other to ask and give advice, post personal records, etc.
  - A workout and rep plan generator based on user input and desired goals available to set and customize.
  - A "Personal Record" keeping page to note the amount, type of lift, etc. along with dates and other information.

- **Technical Report:**

Links

- **Online Repository:** https://github.com/ICS427JFam/BarbellBuddy
- **Release Version:** https://github.com/ICS427JFam/BarbellBuddy/releases/tag/1.0.1

Technical Notes

Environment Requirements:

- Node (https://nodejs.org/en/) Version 12.16.1+

How to install and run:

1. Clone the repo from https://github.com/ICS427JFam/BarbellBuddy
2. `cd` into the root directory
3. Install the dependencies by doing `npm install` in the root directory
4. Install the server-side package dependencies by switching to the “server” directory by doing `cd server && npm install`
5. Go back to the root directory using `cd ..`
6. Install the client-side package dependencies by switching to the “client” directory by doing `cd client && npm install`
7. Go back to the root directory using `cd ..`
8. To run the API server only: `npm run server`
9. To run the whole program (client and server): `npm run start`
10. When running the whole program, it should automatically open a new tab in your web browser that directs to the web app hosted on localhost on the initial start. Otherwise, to access the running program, go to `http://localhost:3001/`.