Introduction

- Team Name: Big Banking Buddies (BBB)
- Team Members: Edwin Zheng, Nicholas Miyamoto-Pennywell, Ronnie Kauanoe, Yuhan Jiang.
- App title: Budget Banking Online (BBO)
  - Description: Banking/Budget Tracker, an online web application to organize your bank information including the current balance, past financial changes, and potential purchases. Regarding this project, our team aims to produce a website to manage the sensitive banking information of our users to help organize the data into a comprehensible and appealing format. The secondary goal of this project is to include a budgeting feature that can track items that have been purchased or are currently being considered for purchase by the user.
  - Functional Requirements:
    i. A sign-in page that allows users to monitor their financial records.
    ii. An opt-in option to allow users to track potential budgeting options. These options include saving money, watching wish list items, or item sale changes.
    iii. A basic financial report page for users to view their financial information.
    iv. A report page allowing users to report slight discrepancies, and the corresponding list of reports page for managers or administrators to view all report tickets.
    v. An action report page listing the recent actions taken by managers or administrators.
  - Type of Program: Web Application focused on an individual's banking information or spending habits.
- Development Tools:
  - Programming Language: Javascript, HTML, and CSS
  - IDE: Vscode(Visual Studio Code) or intellij.

Requirements

Security and Privacy Requirements

Users will be required to provide a valid email address, username, password, security question, and one or two authentication methods, such as images, additional security questions, or SMS/email two-factor authentication. The user’s email will be used to provide formal updates to the user (e.g. change of terms of service). The user’s password must be at least eight characters long, contain at least one capital letter, and contain at least one number. Only the user’s username and password are required for access to the site. However, the user will be prompted to answer their security question and additional authentication measures when attempting to make critical changes to their online account (e.g. changing their password).

Users will only be asked for their full legal name, birth date, and gender when creating their account to ensure that the user is connected to their unique account. Auxiliary information such as race, ethnicity, or religious affiliation will not be required but these fields will be available to fill in the user’s profile settings. However, this information may be requested at a later time to implement new features or to check a user’s qualification for specific promotions.

A database for tracking security issues will be implemented with each entry ranking from Level 3 (least important) to Level 1 (most important) security and privacy issues. Details of the issue severity levels are discussed in the next portion of the document. Each entry will contain the title of the issue, its level of concern, when it was posted, and when it was solved if applicable. Only the application developers will have access to this database.

Quality Gates

The level of privacy should be considered Level 1 if any personal information is either accessed or obtained without the customer's permission or stored in a way that puts customer information at risk. No personal information should be available or taken against the knowledge or request of the client. Furthermore, personal information should be kept in an environment with the utmost security to ensure clients' privacy. The level of privacy should be considered Level 2 if any non-sensitive personal information is accessed or obtained without a warning to the customer, the information can be easily traced back to the customer, or the information is stored in a way that is generally recognizable. Non-sensitive information in this sense can be considered any information that isn't inherently connected to a client but may be added by the client by either personal additions or requests. The level of privacy should be considered Level 3 if any customer information is available through file sharing, stored in a way that enables anyone to access it through a loophole, or at risk of deletion without the control or request of the customers.

The level of security should be considered Level 1 if any form of admin-level access is illegitimately obtained and used for malicious purposes. The level of security should be considered Level 2 if an attack can lead to a Level 1 issue but integrated security measures prevent it from progressing further. Denial of Service attacks, Elevation of Privilege exploits, target information acquisitions, spoofing, or brute force attacks are examples of issues that can fall under this level. These outcomes can be from either one or multiple threat points and should be treated with the utmost importance The level of security should be considered Level 3 if any information is randomly targeted and modified but the changes do not persist after a system restart.

Risk Assessment Plan for Security and Privacy

Every week the team will meet to attempt to use the current iteration of the software to identify any possible risks. Each team member will be in charge of reviewing code from another member from the past week's work. The team will identify the areas in code/software that deal with information and how easy it might be to access that same information from an illegal or unauthorized standpoint. The question of security will be determined upon the ease of access whereas privacy will be the amount of information that is readily available. Threat modeling will be used for all code in the program, since any vulnerability may result in unauthorized access, security reviews will be used mainly for specific data checkpoints where data may be transferred from host to client or vice versa.

The discussion on the personal information at risk, name, age, gender, and banking information will be among that list. The user login part needs a threat modeling to make sure that users are accessing the correct user application session. Since the product contains the manager application, this session should take extra care to avoid the elevation of privilege. Besides that, the user application session also needs sufficient security checks to avoid potential errors.

Design

Design Requirements

This project will use Firebase to manage the user authentication, online database, and deployment. The web application should consist of the login page and the signup page that are directly linked to the Firebase authentication API. i) No anonymous users should be allowed to get access to the functionality pages. ii) The email address will be the UID. iii) No repeating email address shall be allowed. iv) Once the email has been recorded in the Firebase authentication system, the UID should never be changed.

Once the signup step has been completed, the user can set up their security questions and answers. These security questions and answers will be stored in a personal information document labeled with the user’s UID, and the document will be placed in a collection of the Cloud Firestore. When a user has to answer the questions, only the questions that match the UID will be given, and no answers should appear to the client-side.

Users cannot get access to any other pages besides the home page, login page, and signup page unless they have finished the login process. Only after that, users can view pages that include personal information. The functionality pages will collect the documents that match the UID from different collections and present these documents in a user-friendly way. All the functionality pages should listen to the Firestore and automatically update the pages when the documents are changed.

Attack Surface Analysis and Reduction

For this project, our group will be using four privilege levels for our users. Each level is associated with the amount of access that is granted to the website. These four levels can be described from lowest access to highest, starting with viewers, then clients, managers, and lastly administrators. Viewers are the first and lowest privilege level, they can be described as potential clients to the website with only access to the front page and sign-in features, no other information is presented or authorized for them to view. Clients are viewers who have signed up for the website, they now have access to the website's features allowing them to view only their sensitive information in an organized manner. It should be noted that clients have no access to any other sensitive information besides their own. Managers are the third privilege level and the first with access to management level features including editing certain features of the website. These features include responding to report tickets, editing the home page, reverting, and locking the home page. The last privilege level is the administrators who have all the management level features with the ability to make an emergency shutdown of any website feature. Additionally, the administrators will have access to view all of the site-wide actions taken by managers and other administrators. The last two privilege levels did not have explicit authorizations to view sensitive information, this is intended since by default these levels don't have complete access and should instead need to request it from the client along with the use of multiple authentication processes to view the sensitive information in question.

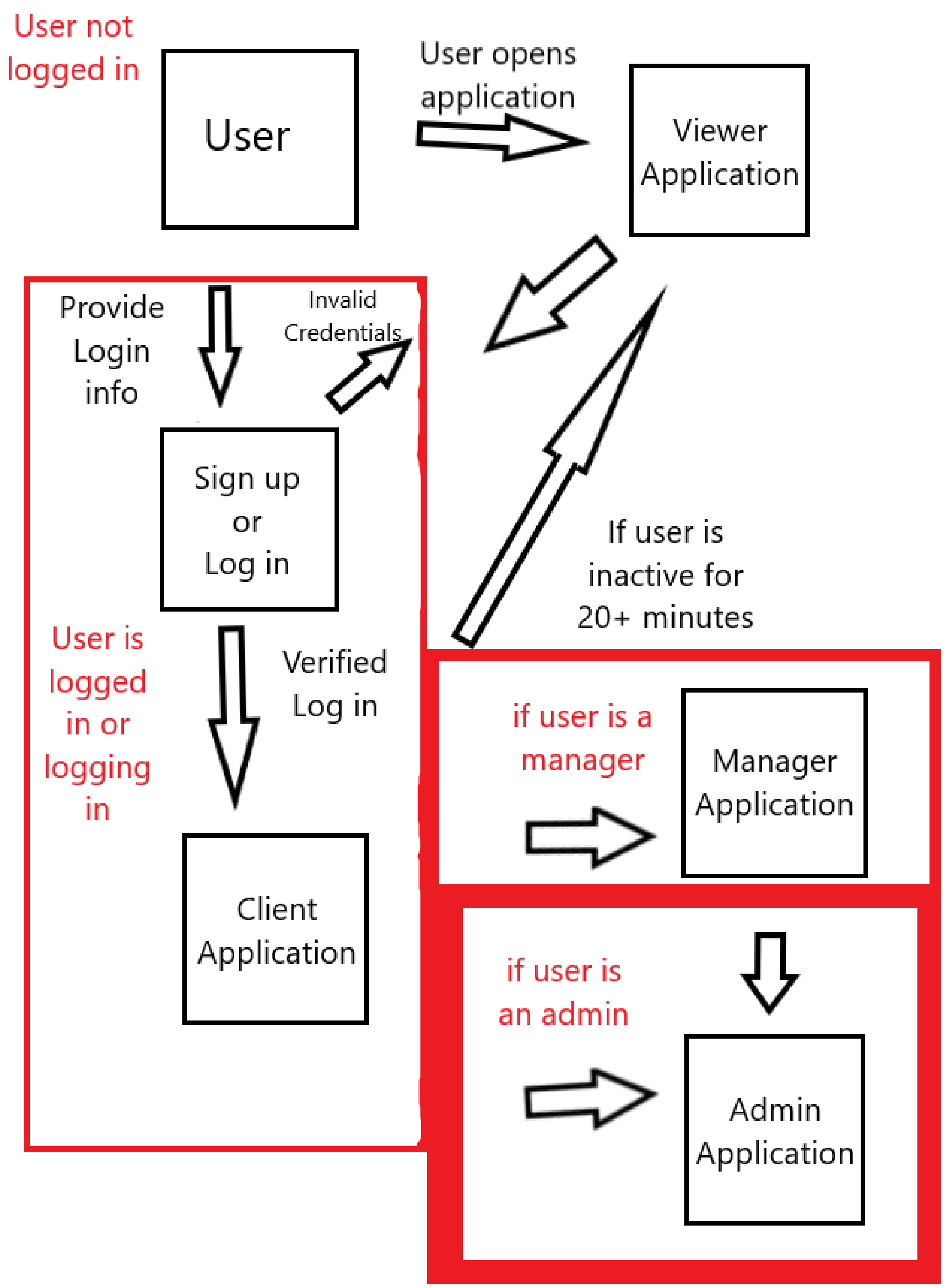
Potentially our attack surface should be quite large due to the rather rudimentary nature of this initial report process. The possible vulnerabilities the website may encounter include the general features, accounts, and databases we may use. The possible attack surfaces for our conceptual idea of this project include:

- General services, Services running by default, or Services running upon requests

- Enabled accounts, Enabled accounts in managers group, Enabled accounts in the administrator’s group
- Database security, Weak ACLs, or Hacking of the base code

Since this is the first attack surface analysis, we will start with the known surfaces that may be at the highest risks. Since this team will focus on the features of the website that transform the sensitive data, the accounts that link to it, and the databases that hold this information we aim to address these issues first. As we progress through this project the team aims to update this list along with focuses on solving the potential vulnerabilities.

Threat Modeling



Implementation

Approved Tools

Compiler/Tool	**Minimum Recommended Version and Switches/Options**	**Optimal Recommended Version and Switches/Options**	**Comments**
Browser development tools	Chrome 90.0 Firefox 78.6.1esr Microsoft Edge 86.0.622.69 Opera 68	Chrome 90.0 Firefox 78.7.1esr Microsoft Edge 88.0.705.63 Opera 68	Additional browser support may be added later.
Collaboration and version control	Github Github Desktop		
Database	Cloud Firestore		May change as needed.
Frameworks	Firebase React Semantic UI		

JavaScript HTML CSS	IntelliJ IDEA 2020.1.3 Visual Studio Code 1.53.0	IntelliJ IDEA 2020.3.2 Visual Studio Code 1.53.2	
Source code analysis	ESLint Visual Studio JavaScript Debug		
Web hosting service	Firebase		Using Firebase's hosting and deployment services.

Deprecated/Unsafe Functions

Deprecated/Unsafe Functions	**Alternatives**
componentWillUpdate() [React]	componentDidUpdate() [React] **OR** getSnapshotBeforeUpdate() [React]
componentWillReceiveProps() [React]	componentDidUpdate() [React] **OR** Use a memoization helper **OR** Use a fully controlled component [React]
<image> [html]	<img>

<big> [html]	Header tags [html] **OR** Font-size: [css]
<center> [html]	Text-align: center [css] **OR** Margin-left: auto + margin-right: auto [css]
<frame> [html]	<iframe> [html]
<font> [html]	<style> [html] **OR** Style attribute [html] **OR** CSS properties [css]
<nobr> [html]	White-space: [css]
<strike> [html]	<del> [html] **OR** <s> [html]
Align attribute [html]	Text-align: [css]
getYear() [js]	getFullYear() [js]
eval() [js]	Do not use eval(), extremely unsafe, no alternatives
escape/unescape [js]	encodeURI/decodeURI **OR** encodeURIComponent/decodeURIComponent [js]

Static Analysis

During this course semester, our group had decided upon utilizing two code compilers, Intellij, and Visual Studio Code. Frankly speaking, those members who chose to use Intellij had done so because of the access granted to them per separate computer science courses that were using the compiler and decided to continue to use it for convenience. Alternatively, those members who opted to use the Visual Studio Code compiler were based on both past experiences with the compiler and the college-friendly price tag. As for the preferred analysis tools for each compiler, those using IntelliJ have already had ESLint integrated while those using Visual Studio Code initially chose to use Auto-Formatter. After further discussions, we agreed to use the ESLint analysis tool but haven't completely integrated it in all our member's compilers and aim to do so soon. Our group has recognized the fact that there may be possible issues that arise running two separate static analysis tools and have started to plan ahead. One such potential solution, for the time being, is holding group code review sessions using the ESLint analysis tool until all members have integrated it into their compilers.

Looking at the ESLint analysis tool it's quite clear how it aims to identify potential issues in code for the user. The analysis tool uses color-coordinated tilde's under sections of code that it identifies as errors, warnings, or weak warnings which are associated with the colors red, orange, and grey respectively. Along with the color-coordinated underlining of code a brief description is given describing the issue. These issues that ESLint identifies can change depending on the user, this tool is described as "flexible and configurable" by their documentation page and can pose quite a few complications if everyone isn't using the same setup for the tool. In this case, using the default setup as described in previous classes will be the basic setup for our group. The default setup may include weak warnings as formatting issues, warnings as unused code, and errors as nonfunctioning code.

As for our experience with ESLint, all of the members within this group have had past experiences with it and can agree upon the convenience it brings when coding. The use of tildes to underline problem areas in code is quite apparent and easy to use to distinguish the different issue levels. The brief descriptions are clear and concise making it an opportune tool to quickly understand and correct any mistakes. We as a group can agree we find this tool quite useful with the only difficulties that we could think of involving the post-setup for the Visual Studio Code users who are now switching over to it.

Implementation & Verification

Dynamic Analysis

For the intents and purposes of this stage in the project, our group researched two dynamic analysis tools, Iroh.js and React Development Tools. The latter of which has been used by multiple members of the group in the past. The goal for this section is to determine a dynamic analysis tool that fits the general needs of the project while remaining convenient to use for the group. For informative purposes, a dynamic analysis tool is a type of code review that occurs during code execution. In the case of this project, we aim to use it to determine vulnerabilities when our web page is being used.

The first Dynamic analysis tool the group researched was Iroh.js. Iroh.js is the first google search recommendation for dynamic analysis tools for javascript which it is specifically developed for. Iroh.js can be easily added to any code or browser as described by the developer on his website with the main function of this tool being to include console messages describing specific functions called during the code execution. To find these messages the user would need to open the developer's tool window which can be opened by right-clicking the web page and choosing the inspect option or by hitting the Function 12 button on your keyboard then selecting the console tab.

The React Development Tool is an additional easy-to-use dynamic analysis tool that the group considered alongside Iroh.js. Since this tool was a Google Chrome web browser extension it was quick and easy to download for the majority of the group who already used Chrome. Once downloaded it adds two new tabs to Chrome's Developer Tool window(which can be accessed as described above by right-clicking and selecting inspect or pressing the Function 12 key). Unlike Iroh.js, these two new tabs that are labeled “Components” and “Profiler,” allow the user to interact, view, and record information relating to the web page in question.

The main difference between the two dynamic analysis tools, Iroh.js and React Development Tools, was that the latter required the javascript program to run React. Despite this glaring restriction the tool also provided more functionality, allowing the user to edit and record information as the program runs. Both are open source with access to their software on GitHub, but it is interesting to note that the React Development Tools explicitly exclaims that it doesn’t transmit any data remotely. In practice, the group has primarily used the React Development Tools and has done that on three separate occasions. Since the primary focus for the first two coding milestones up until this point has been primarily focused on front-end visual changes our group was able to utilize the Components tab to identify key react components that may not be displayed as we had envisioned. It should also be noted that we have not run into any glaring issues from either analysis tool up until this point in the report and we aim to continue using both tools as our project progresses past this halfway mark.

Attack Surface Review

Compiler/Tool	**Minimum Recommended Version and Switches / Options**	**Optimal Recommended Version and Switches / Options**	**Comments**	**Changes and/or New Vulnerabilities**
Browser development tools	Chrome 90.0 Firefox 78.6.1esr Microsoft Edge 86.0.622.69 Opera 68	Chrome 90.0 Firefox 78.7.1esr Microsoft Edge 88.0.705.63 Opera 68	Additional browser support may be added later.	No changes or vulnerabilities

Collaboration and version control	Github Github Desktop			No changes or vulnerabilities
Database	Cloud Firestore		May change as needed.	Need to update security rules
Frameworks	Firebase React Semantic UI			No changes or vulnerabilities
JavaScript HTML CSS	IntelliJ IDEA 2020.1.3 Visual Studio Code 1.53.0	IntelliJ IDEA 2020.3.2 Visual Studio Code 1.53.2		No changes or vulnerabilities
Source code analysis	ESLint Visual Studio JavaScript Debug			No changes or vulnerabilities
Web hosting service	Firebase		Using Firebase's hosting and deployment services.	No changes or vulnerabilities

There have been no major development changes or updates in the approved tools used. The Firestore database needs to have its security rules updated to ensure no unauthorized access or tampering can occur. This database will be responsible for holding sensitive information for all users and will be the first location for threat actors to exploit. The security rules will be updated in the upcoming milestones.

Verification

Fuzz Testing

The first attempt to affect our site involved brute-forcing our way into an account using a list of common passwords on NordPass, a service designed by the creators of NordVPN.The list can be found at https://nordpass.com/most-common-passwords-list/. We set up a dummy account foo@bar.com and attempted each password using an AutoIt script and once it terminated, we discovered that the script had successfully logged into the account. We don't know how many attempts were made to access the account but in any case, we did not have a measure in place to lock the account after too many unsuccessful attempts were made. With the current lack of security features, this hacking method succeeded 100% of the time and if the site was to go live, numerous accounts would be vulnerable to the same exploit.

The second attempt to affect our site involved inspecting returned webpages and source files. The goal of this exploit was to find sensitive information that was not hidden or encrypted to gain access to an account. Fortunately, searching for known passwords or email addresses did not return any results. Furthermore, if a user is signed in, the current user's information cannot be found through this method. Keeping it unavailable while signed in keeps threat actors from scraping data while a user is signed in so they can obtain access at a later time. In the end, this hacking method succeeded 0% of the time.

The third attempt to affect our site involved manipulating the web parameters. The goal of this exploit was to view unprotected data or execute commands either through Firebase or Firestore, the database system used in our Firebase application. We attempted to go to pages we knew couldn't be accessed unless a user is signed in (e.g. /dash, /profile, /wish-list) but were automatically sent to the login screen. Furthermore, when inputting Firebase or Firestore commands in the URL field, the commands were read as plain text and were unrecognized web pages, thus leading back to the login screen. The same actions were taken when a user is logged in and we were still unable to access any sensitive data. Thus, this hacking method succeeded 0% of the time.

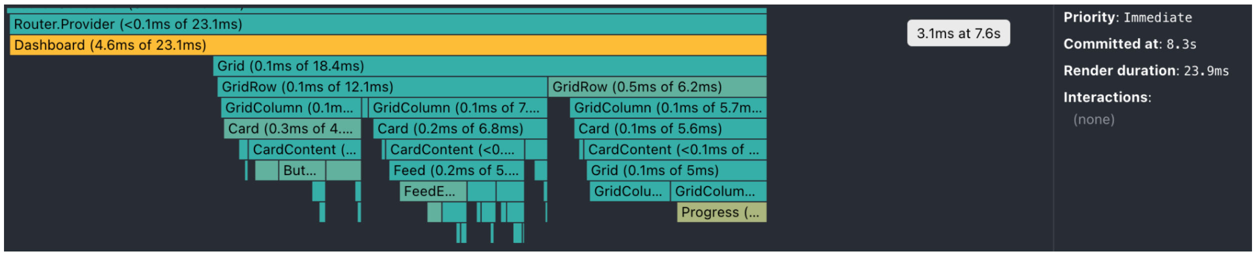
Static Analysis Review

ESLint has now fully become used across all members of the group. There has been little difficulty in getting all members set up with ESLint, and so far no issues with the code checker have been encountered. As a result, the group's code has been able to maintain a uniform style. This uniform styling has allowed manual code reviews to be swift and efficient, as all members know what to expect out of the code's format, cutting down on time spent trying to follow how a person formatted their code. Overall, the auto-formatting and the error checking that ESLint provides an overall clean look to the code, as well as saving development time for all developers during coding and code reviews.

Dynamic Review

At this point in the project, the group has almost finished up the front-end changes for our code, with few back-end implementations remaining. That being said, after further reflection amongst ourselves the concern for more insightful feedback regarding the back end changes from a dynamic analysis tool was highlighted. To be one hundred percent transparent, this issue will be another focus for our project moving forward. These concerns for an additional dynamic analysis tool for back-end changes arose when testing our application and running into errors. For example, an infinite re-render warning due to a useState updating in a loop caused the page to not load and the React Development Tool had no feedback to assist in that issue. The current dynamic analysis tool provides little feedback for anything besides the react components, so a suggestion for an additional dynamic analysis tool was the Chrome Canary developer tool. Like the React Development Tool, it is a Chrome extension but our group aims to research more into it and other options before committing to this additional tool.

With our group's new concern out of the way, the React Development Tool has been helpful in every step to accomplish the front-end changes. The components tab, which has been stated in the previous section, has helped to determine the specific react components that are on the page allowing us to pinpoint specific changes that need to be made. The profiler tab has also been used a bit more, giving us insight into the render time of certain actions, for example, the main action of users logging in, which would take on average ten seconds to accomplish. The profiler also went into detail regarding the different react components and how long each took to render. One such attempt looked as follows:



Priority: Immediate
Committed at: 8.3s
Render duration: 23.9ms
Interactions:
(none)

Potentially this information could be used at the end of the project in an attempt to lower render times. While our project is small, the actual success in something like this would be useful for future projects that would be larger and would require reducing the render times as much as possible.

Release

Incident Response Plan

As the team pushes toward the final release of our project we have decided to address the potential issues we may encounter with an incident response plan comprised of three main components. These three components consist of a privacy escalation team, a team project email, and a set of procedures our team will follow when dealing with an incident. The purpose of the privacy escalation team is to respond to reported or known breaches in software in a timely and efficient manner. The team project email would allow community members or good samaritans to notify our team of any issues with the live project. Lastly, the procedures are the steps the privacy escalation team should be taking in response to an incident. We'll begin by describing the members of the privacy escalation team and their different roles.

Our privacy escalation team will consist of four main roles as described by the Microsoft SDL documentation, these roles include the escalation manager, legal representative, public relations representative, and general core team member. It should be noted that the size of our team is rather humble and as such, each member is technically a core team member, but one will be solely responsible as the general core team member to allow the other members to focus on their other individual tasks. Nicholas is our team's escalation manager and is responsible for organizing and informing the team of any new or current incidents that have not been resolved. Along with this his main role is to assist the other members and follow up with them periodically to ensure that the proper responses are given to any concerned party. Yuhan is our team's legal representative and will deal with any legal issues that may arise. He will be responsible to review each member and their work to ensure that they are held up to our team's standards. He will also be required to review and write any legal documentation for the tools we use and any changes we make. Ronnie will be our team's public relations representative and will focus on crafting a favorable image for our team in light of all incidents the team has or needs to solve. This role includes communicating with each team member to assist in producing specific responses to the public. Lastly, Edwin is our team's general core member and will be responsible for working on solutions to each incident brought up. While all members of our team share this role, Edwin will be required to lay the groundwork for the rest of the team to follow up on when they have finished their tasks. This also requires him to be independent and at times he'll also need to be a quick thinker as certain incidents won't allow him the luxury of time.

While the privacy escalation team has been outlined, the necessity of community feedback and assistance has been recognized. As such the manager and public relations representative have been allocated an email address that can be used for this purpose. The team's contact email is bbo@contact.us. The team agreed that the manager and public relations representative should share access to this email as it will be used to receive and organize incident reports while also sending or responding to these concerns from the community.

As for the procedures our team will follow, regardless of whether the incident was discovered by one of the team members or community members, we will follow these six steps:

1. Confirmation, as a team we will need to confirm that this incident is a real issue. This first step will always include our team manager Nicholas, and at least one other core team member to investigate these incidents. Depending on the initial investigation an incident might force its way to the identification step to be addressed sooner than later.
2. Identification, without a proper understanding of what an incident discloses to the public we cannot properly respond to them either. An understanding of the threat level needs to be distinguished to decide how to proceed. Every member will be involved in this step to ensure that every angle is covered when identifying an incident. Each incident will be categorized as one of five threat levels:
    1. Shutdown, this level requires the team to shut down the project to stop anyone from exploiting this incident as it risks all clients or community members.
    2. Immediate Attention, this level is severe as it risks some of our client's sensitive information and requires immediate action and if it cannot be solved within a designated time or any additional incidents are reported or identified it should be promoted to Shutdown.
    3. Focus, this level is an issue that requires our attention as it risks some of our client's personal information, this can be promoted to Shutdown if the amount of personal information is drastic enough to be used against them or if the method used to take this information is publicly available.
    4. General Fixes, this level is for more general issues that might not be low level nor involve client or community risks.
    5. Community Requested Changes, this is the lowest level and considered the standard threat level for small issues or bug fixes like user interface mistakes that might be generally reported.

3. Response, depending on the severity of the incident no more than twenty-four hours for a Shutdown level incident, to two weeks for a Community Requested Change should have elapsed after an incident's identification, without a public response to our community, and community member(s) who may have discovered this. The goal is to honestly communicate with our community about the issues we may face to ensure their trust is never misplaced. This step mostly deals with the public relations and legal representative who will need to deal with how to respond to the community, the manager will also be involved but to a lesser extent.
4. Documentation and Solution Testing, the team should document and research the incident to produce a solution that is ready for the public. This step should be accomplished concurrently with the previous, Response, step and take anywhere from one to seven days to accomplish. Once a solution has been discovered, this step shouldn't be concluded, instead, the team should split up to proceed to deliver the solution while the rest finish up the documentation of such an incident. Generally, the core member will focus on solving the incident issue and assist in its release while the remaining members assist him or document everything else.
5. Solution Release, like step three, depending on the severity of the incident the solution should be released accordingly. For threat levels one through three, the solution may be released immediately or during a period that the project sees less traffic from the community. Otherwise, the release may be held for a scheduled update. The core member will be responsible for the solution release scheduling and return to solving other issues afterward.
6. Review, after the incident, has been solved and the team can analyze the documentation. This step might take place immediately after the solution was released and documentation has concluded or further into the future. The purpose of this step is to recognize and prevent another incident of this type from occurring again in the future. This may include adding new rules, tools, or processes the development team needs to use to ensure safer coding practices. The team may also decide to explore new training or increased public relations in

response to this review. The team manager will organize this information and follow up with any decisions made during the Review step.

Final Security Review

Our threat model specifies three types of users of increasing capability: client, manager, and admin. The initial concept was to have a manager-role user be responsible for handling issues with client-role users and administrators would be responsible for handling direct changes with the application itself. Due to time constraints, we weren't able to implement the manager role. This being stated, the client role only has control over their account and the admin has control over the entire site. Only the developers have the ability to grant admin permissions, attempting to manipulate the URL to access the admin console redirects the client to the home page.

The entire development team is still using ESLint to ensure code legibility and reduce redundancies. When tested at the end of our latest release, no errors were detected. As development progressed, we decided to only use React Development Tools for our dynamic analysis as it was specifically suited for the framework we were using. Since this tool only represents the overall structure of our application, it does not reveal any sensitive information and does not pose a significant security risk.

Based on our previous fuzz testing and another round performed at the end of our final release, we are still encountering a Level 1 exploit where a threat actor could access an account by attempting the most common passwords. At this time, we have not created a reliable way to handle this vulnerability but a future release will be made to address it directly.

The final grade we are giving our application is Passed FSR with exceptions. While our codebase does not expose sensitive information, there is still the glaring issue of a threat actor simply brute-forcing their way into our site. Given this grade, we would like to remind our clients that their passwords need to be secure and should not be found on any list of common passwords to avoid this scenario. We would also direct clients to this Google article (https://support.google.com/accounts/answer/32040?hl=en) stating the basic necessities of a strong password.

Certified Release & Archive Report

Certified Release Link: https://github.com/Big-Banking-Buddies/bbo/releases/tag/1.0

Version: 1.0

Features

- Creation and storage of user accounts
- Password Hashing via Firebase's innate hashing system
- Ability to modify user's own profile
- Ability to reset passwords
- Different permission levels for users
- Budget Tracking
- Transfers between different accounts
- View history of account actions

Future Development Plans

- Ability to turn on a pin for secondary verification
- Email verification of user accounts
- Multi-factor authentication
- Mobile application
- More administrative actions for admins and managers
- Integration of wishlist items into a budget feature
- Link bank APIs to application

Technical Portion

Install Instructions

1. Download the project https://github.com/Big-Banking-Buddies/bbo/releases
2. Open the terminal and run "npm install" to get the dependencies.
3. Open another terminal and run "npm run database" to start the local database for the wishlist function (You may skip this if you don't want to use the Wishlist).
4. Run "npm start" to start the localhost.

Work Cited

Ambler, Scott W. *Data Flow Diagram (DFD)s: An Agile Introduction*, Agile Modeling, www.agilemodeling.com/artifacts/dataFlowDiagram.htm.

Ambler, Scott W. “Security Threat Model.” Security Threat Models: An Agile Introduction, Ambysoft Inc., 2020, www.agilemodeling.com/artifacts/securityThreatModel.htm.

“Appendix C: SDL Privacy Questionnaire.” *Microsoft Docs*, Microsoft, docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307393(v=msdn.10)?redirectedfrom=MSDN.

“Appendix E: Required and Recommended Compilers, Tools, and Options for All Platforms.” *Microsoft Docs*, Microsoft, docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307395(v=msdn.10)?redirectedfrom=MSDN.

“Appendix K: SDL Privacy Escalation Response Framework (Sample).” *Microsoft Docs*, Microsoft, docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307401(v=msdn.10)?redirectedfrom=MSDN.

“Appendix M: SDL Privacy Bug Bar (Sample).” *Microsoft Docs*, Microsoft, docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307403(v=msdn.10)?redirectedfrom=MSDN.

“Appendix N: SDL Security Bug Bar (Sample).” *Microsoft Docs*, Microsoft, docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307404(v=msdn.10)?redirectedfrom=MSDN.

“Configuring ESLint.” *ESLint*, eslint.org/docs/user-guide/configuring/.

“Create a Strong Password & a More Secure Account.” *Google Account Help*, Google, support.google.com/accounts/answer/32040?hl=en.

“Deprecated and obsolete features.” MDN Web Docs Moz://a, Mozilla and Individual Contributors, 2021, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Deprecated_and_obsolete_features.

“Fending Off Future Attacks by Reducing Attack Surface.” *Microsoft Docs*, Microsoft, docs.microsoft.com/en-us/previous-versions/ms972812(v=msdn.10)?redirectedfrom=MSDN.

“HTML Elements Reference.” MDN Web Docs Moz://a, Mozilla and Individual Contributors, 2021, developer.mozilla.org/en-US/docs/Web/HTML/Element.

“Iroh.js.” *Iroh*, Meierfelix, 21 Aug. 2017, maierfelix.github.io/Iroh/.

“Phase Five: Release.” *Phase Five: Release | Microsoft Docs*, docs.microsoft.com/en-us/previous-versions/windows/desktop/cc307420(v=msdn.10)?redirectedfrom=MSDN#EB.

“React.Component.” React, Facebook Inc., 2021, reactjs.org/docs/react-component.html#unsafe_componentwillupdate.

“React Developer Tools.” *Google*, Google, 2 July 2013, chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en.

“Top 200 Most Common Passwords of 2020.” *NordPass*, nordpass.com/most-common-passwords-list/.