Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

Report

A. Introduction

Our team name is README and consists of the following members: Angeli Amascual, Matthew Kirts, Eric Rivera, and Jessica Tang. The application we have decided to create is called: Hobby Helper, which will allow the user to track their different hobbies such as books, TV shows, and video games. The user can track the status of different items within their selected hobby such as what books they have already read, what TV shows they are currently watching, and what games they want to play. In addition, the user will be able to see different statistical information relating to their hobbies. For example, they will be able to see how often a specific hobby has been updated and create ratings or reviews for a hobby item. Hobby Helper will be a Web Application created using JavaScript, React, Meteor, and IntelliJ Idea.

B1. Security and Privacy Requirements

The security requirements of our website are mainly to protect the private information given to the website from the user. The website will require the user's email address for account creation and as well for the user to create a password for the account. The plan to prevent logins from a person that is not the owner of the account is for the creation of security questions made during account creation. The idea is to have the user create security questions that only they should be able to answer. This will help to minimize potential account breaches. If the user's account becomes unavailable to them, the administrators of the website will be able to access the account's information if the user asks so. The website will never ask for any personal information other than email to be required. Administrators will also never ask for any login credentials or private information. In terms of user privacy, the user's information will be private. Only the user will have access to their personal information as well as the information regarding their hobbies.

There are potential security and privacy risks such as accounts getting hacked or information being leaked. Administrators will have to assist the account owners when this happens, but the administrators may need to confirm that they are the owner of their account

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

through their security questions. The website will only collect information that is required and store what the user has inputted into the database.

B2-1. Quality Gates and Bug Bars (Privacy)

Security is always of the utmost importance and being able to define the levels of access is always a thing to think about. Like with many websites created (whether through Meteor or any other means), there will be two primary levels of access to the site: user privileges and administrator privileges. For the user side of things, it is simple and straightforward; when signing in, the two major pieces of privacy will be the username and the password associated with said username. When signing up, however, you will have to provide some additional details such as first name, last name, and a security question and answer. The user profile itself will contain some basic information; a user profile image, the users name, and description. The admin will have similar privileges to the user, but can also view possible hidden information, passwords, page analytics, and can “ban” (remove) a user if the admin deems it appropriate.

B2-2. Security

On a simple level, the only real thing that people would have to worry about regarding security is just their passwords and possibly e-mail addresses. According to the Microsoft SDL Privacy Bug Bar, low level scenarios may include the collection of information without consent. This could include but is not limited to emails and passwords. Collection of information is typically used to send out notifications and possible items that could be related to a person’s hobby. On a more serious level, a user could end up with heightened privileges that they are normally not allowed to have, either due to a bug or tampering with the website. Examples may include but are not limited to somehow acquiring administrator privileges despite being unauthorized. PII on the local machine could also be collected without consent.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

B.3. Risk Assessment Plan for Security and Privacy

Security and Privacy will be assessed using a questionnaire.

Determining Your Privacy Impact Rating

The following behaviors apply to your software.

_X_ Stores personally identifiable information (PII) on the user's computer or transfers it from the user's computer (P1)

PII being stored/transferred:	Names, email addresses, IP addresses, activities, geographical indicators, demographic information.
User value proposition	Hobby Helper provides a convenient way to track and visualize progress of hobbies.
Describe your notice and consent experiences:	Notice and consent documentation is provided during sign up.
How users will access your public disclosure:	Public disclosures will be made available through an About page.
How users can control your feature:	User account settings and features will be available.
How you will prevent unauthorized access to PII:	Access to different types of information will be allowed through privileged accounts.

Parts of your program need threat modeling and security reviews include account credential authentication, and user data storage and access.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

C1. Design Requirements

The design requirements of Hobby Helper are based on both the security requirements and functional requirements we have designated for our web application. For the security requirements, a design requirement for the application is that it will need to be able to gather user information in a secure way. It also needs to maintain privacy between the different users as well as restricting information between the user and the database. Another requirement would be to allow the user to change their information securely and ensure that their account information stays safe. On the application functionality side, Hobby Helper's main goal is to track the user's hobbies and interests. Therefore, another design requirement will be for the application to showcase the tracking of users' hobbies and interests in a clear and concise manner.

In order to fulfill the design requirements, there are a variety of design features that we will add to Hobby Helper. First of all, the web application will ask for both required and optional information. The required information would include information such as the user's email, chosen username, and password. Optional information on the other hand will include things such as gender and age. To enforce security, another design feature is the presence of security questions to ensure the protection of the user's information when they want to change information such as their password or email.

In addition to design features relating to the web application's security requirements, there are also design features that will fulfill the functional requirements of the application. Hobby Helper will track the progress of the user's hobbies and interests. For example, it will keep track of what books a user wants to read, what they are currently reading, and books they have already read. Another feature will be to offer the user statistical information on their hobbies and interests such as ratings or reviews.

C2. Attack Surface Analysis and Reduction

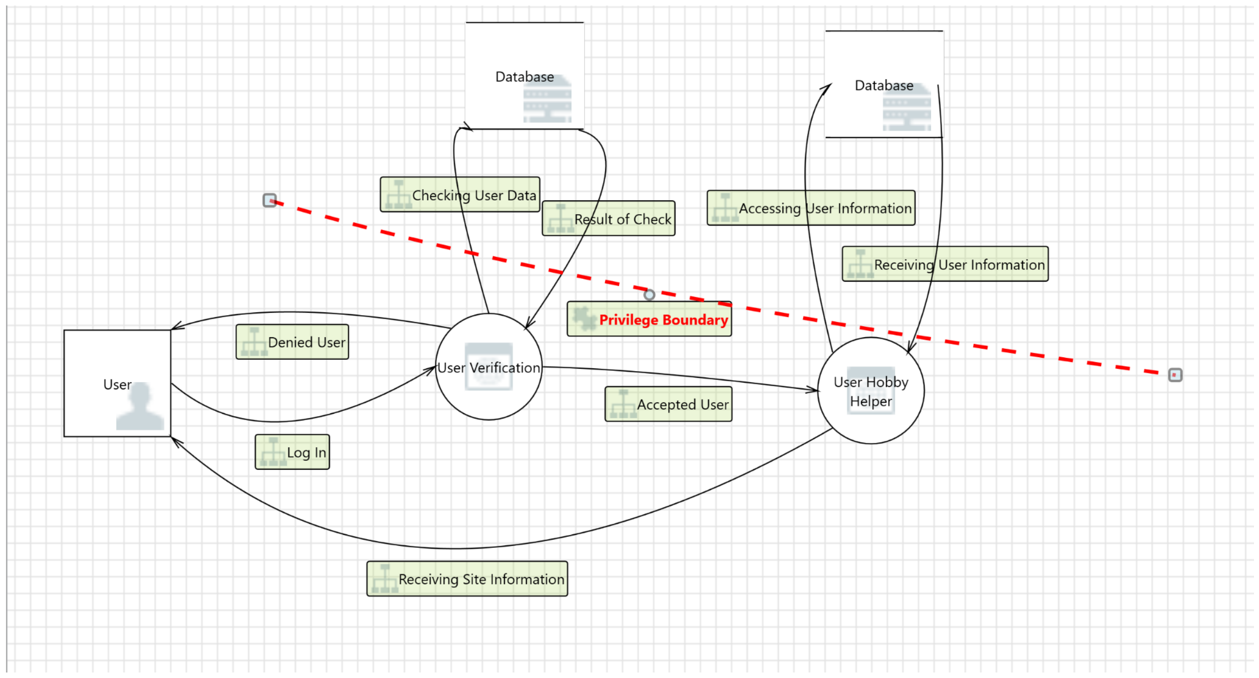
With this website, there needs to be certain details that need to be considered. As with any version one program, website, video game, etc., there will be bugs that might appear either

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

on our end or on the users end. Lower security risks that might occur may include the accidental leakage of P.I.I. that the user himself/herself did not authorize to be shown. Moderate levels of risks include leakage of passwords or possibility of specific windows locking up, due to internet connection (minor) or a problem within the code itself (moderate). DDoS attacks may also lock up controls or specific screens, which may require a full restart of the website server side. On the server side, things that administrators will have to worry about includes items like data leakages, both minor and severe (i.e., P.I.I. on a smaller level, extremely sensitive information such as addresses and passwords on a more severe level). Additional care needs to be taken to prevent unintentional data leakages from happening.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

C3. Threat Modeling



The web application Hobby Helper has some potential threats that should be noted throughout the creation process. Using the Microsoft Threat Modeling Tool I was able to see different examples of threats that could occur from our program. These threats can be classified using the STRIDE model. There are two possible Spoofing category threats. The database could be spoofed by an attacker when a user attempts to log in. If this were to happen the data that the user is inputting could be sent to the attacker's chosen destination instead of the database for user verification. Another spoofing related threat is that the attacker may attempt to spoof User Verification or User Hobby Helper in order to try and access the database through them. Another threat category is Denial of Service and one such threat within that category would be an attacker doing a resource attack when accessing the database to lead to a deadlock or timeout on the web application (Microsoft Threat Modeling Tool, 2020).

The next threat category in the STRIDE model is Information Disclosure. One potential threat from this category that applies to Hobby Helper is that if the database is not protected properly an attacker could read information from the database that they should not have access to. In addition, another threat is that data flowing from the database could also potentially be read by an attacker. Repudiation is another STRIDE threat category and a possible threat under that category is that information from the database may end up not being either received or written due to an attacker interfering. The last threat that could possibly occur in Hobby Helper is data flow into the database being tampered with which could lead to the database being corrupted (Microsoft Threat Modeling Tool, 2020).

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

D. Implementation

D1. Approved Tools

Tool Name	Version
IntelliJ IDEA	2020.3
React	17.0.2
Meteor	2.1
Mongo	4.4
ESLint	7.18.0
Iroh.js	0.3.0

D2. Deprecated/Unsafe Functions

- Possible unsafe/deprecated Functions
  - IntelliJ IDEA
  - React
    - Deprecate javascript: URLs as a common attack surface
      - Solution/Fix: https://github.com/facebook/react/pull/15047
    - Deprecation warning for React.createClass. Points users to create-react-class instead
      - Solution/Fix: https://github.com/facebook/react/pull/9771
  - Meteor
    - DDP's connection.onReconnect = func feature has been deprecated

        - DDP.onReconnect(callback) method should be used to register callbacks to call when a connection reconnects. This is used by the accounts system to re-login on reconnects without interfering with other code that uses connection.onReconnect
- Mongo
    - DeprecationWarning: collection.findAndModify is deprecated.
        - Use findOneAndUpdate, findOneAndReplace or findOneAndDelete instead
    - The MongoDB driver's remove() function is deprecated
        - Use deleteOne() and deleteMany()
    - Update() function is deprecated
        - Use updateOne(), updateMany(), and replaceOne() functions. You should replace update() with updateOne(), unless you use the multi or overwrite options.
- ESLint
    - If you use deprecated globals in the node, browser, or jest environments
        - You can add a globals section to your configuration to re-enable any globals you need

D3. Static Analysis

ESLint is a static analysis tool that our team members are all familiar with due to ICS classes such as ICS 314 and ICS 414 which used it to maintain coding standards while developing web applications. Due to that familiarity, we chose ESLint to help us maintain our code throughout our development process. ESLint itself serves as a static code analysis tool. It functions primarily as a way to spot errors within the code while also providing recommendations based on how the code is written. Through its usage we are able to ensure that each member’s code is correct and follows the same coding style.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

From working with ESLint we have been able to see both its advantages and disadvantages. ESLint has great functionality. It will spot any coding errors the user makes and also provide users with an explanation on what the error is. ESLint will also often provide recommendations and shortcuts while coding such as syntax suggestions and alternatives for code lines. Through these functions, ESLint has assisted our team in creating neater code that is easier to review. With this tool, code quality is improved and coding standards can be maintained among multiple coders.

One disadvantage of ESLint is the lack of information provided with some errors. On occasion, the static analysis tool would point out errors but not explicitly state the error type or provide recommendations on how to fix it. These errors can be coding style errors or errors relating to the functionality of the code. In the case of the former, it is a minor inconvenience but with the latter it is frustrating for the coder to figure out how to fix it.

Overall, ESLint performs its job as a static analysis tool well. It is able to reliably spot coding errors, provide coding recommendations, and is built-in for both IntelliJ IDEA as well as Github. Due to this, despite some minor issues and struggles with ESLint our team will continue to use it as our static analysis tool in order to ensure quality code throughout our development process.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

E. Verification

E1. Dynamic Analysis

A dynamic analysis tool can be used to analyze code during runtime. This can help the user spot problems while the application is running as well as assist the user with debugging them. Since our web application uses Javascript we encountered difficulty while searching for potential dynamic analysis tools. A majority of dynamic analysis tools found by our team were for programs made with Java, C, or Python. In addition, some dynamic analysis tools were out of data and unusable. Through our research, our team determined that "Iroh.js" would be our best possible option. Iroh.js is a dynamic analysis tool made by Maier Felix and that has various features such as runtime type checking, test cases, and enabling the user to measure performance.

Working with Iroh.js was both difficult and frustrating. Before being able to use Iroh.js and experiencing what it was like, our team encountered problems with getting it set up. In order to set up Iroh.js into our web application we had to first import it and then apply it to parts of our code. However, the documentation for it and the examples provided did not provide sufficient enough guidance which led to confusion amongst the team members. This was also due to the fact that the examples provided by Iroh.js used pure javascript while our team's web application used Javascript and React. In addition, since Iroh.js is a relatively outdated and unknown dynamic analysis tool there was also very little outside help we could turn to for answers. Eventually one of our team members was able to determine what the problem was which finally enabled us to proceed with using it.

After getting Iroh.js setup there were still some difficulties with making it do what we wanted it to. As stated earlier, our program is not pure javascript and uses React to help build the web application. Although we are not completely certain, it is possible that our usage of React has affected how Iroh.js works. One way we tried to use Iroh.js is through monitoring code flow in our web application. Iroh.js can detect the execution of different parts of the code such as function calls, if statements, loops, etc. This can then be displayed on the console for the user to review and analyze. We were able to get Iroh.js to work partially but some of the listeners we created to monitor the code failed to work properly despite following the examples provided on the Iroh.js GitHub repository.

Although our team faced difficulties while working with Iroh.js, there were still some successes with using Iroh.js. One success involved using Iroh.js to measure the performance time of a function. Through Iroh.js it is possible to record the amount of time it took for specific functions to complete. This information can be retrieved by the user through the console. The message displayed on the console can also be customized however works best for the user. In addition, as stated previously there was some moderate success with monitoring code flow. We were able to review part of the code flow through this method. However, due to the issues we faced we were unable to use Iroh.js to its fullest extent.

Our team plans to continue to experiment with Iroh.js despite the difficulties we have encountered. We hope to further improve our understanding of the dynamic analysis tool in order to use it more efficiently and effectively while we wrap up code development.

E2. Attack Surface Review:

After reviewing our approved tools and the current updated versions, our team noticed that some of our tools were not up to date. In addition, through this assignment Iroh.js was added as an approved tool for our application.

Tool Name	**Previous Version**	**Updated Version**
IntelliJ IDEA	2020.1.4	2020.3
React	16.14.0	17.0.2
Meteor	1.9.3	2.1
Mongo	1.10.1	4.4

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

ESLint	7.18.0	7.18.0
Iroh.js		0.3.0

E3. Fuzz Testing

The first method of fuzzing our team tried out was to manually go into the Hobby List and Kanban Board pages and conduct numerous forms of random input. This ranged from characters to numbers to various symbols (i.e. Japanese Kanji characters, Chinese/Mandarin characters, etc.). Due to the fact that the inputs for these two areas of the application dealt with strings, there were no problems with getting the data properly inputted or updated to the database. However, a problem noted from this method involved the character length of inputs. A long number of characters would cause the formatting of the application to collapse. In the case of the Hobby List page, a long hobby name inputted or edited would cause the statistics, edit, and remove features to be pushed down to the bottom of the page. While this is not a serious problem, it is less than ideal. One way to fix this would be for our team could be to limit the number of characters that can be put into the text fields. Doing this will prevent the possibility of overflow of characters or input on each of the specified categories. Another way to fix this problem would be to adjust the formatting that displays the data so that the formatting would not collapse due to a high character count. Another problem noticed by a team member was hobby items supposedly not appearing in their respective Kanban Board. This was because when adding a hobby item to a Kanban Board it included a “Hobby” field that asked the user to type out the name of the hobby to add the item to. Therefore, even if the hobby was slightly off such as “TV Show” as opposed to “TV Shows” the hobby item would not be registered in the correct board. In order to fix this our team changed the input from user input to a dropdown selection. Through this method, hobby items can only be added to the user’s already created hobbies.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

The second method our team used to try and break our application also involved using manual input but this time with the Sign In and Register pages. Similar to the previous method, due to our existing code there was some prevention in regards to false inputs. For example, the email inputs for both the Sign In and Register pages required an actual email format. However, the “First Name” and “Last Name” inputs on the Register pages allowed both numerical and symbols in their input. Although this would not necessarily break the application it is not ideal. Another result we noticed from this method was that similarly to the previous method, there are no character limits for the inputs. In order to address these issues our team will implement character limits appropriate to the inputs and adjust the requirements accordingly to only accept data that fits the given input name.

Given that Meteor also has Testcafe support, the final method that we used was Testcafe tests. Testcafe helps ensure that a program is doing what it is supposed to do by providing several inputs and default values, while also coding in set parameters for the program to follow. To test fuzzing, several base tests were made along with other test parameters that the program can take but should not. This is the same method as before, though requires some extra hardcoding and preset parameters. Base tests included basic input and followed through everything that a normal user would input; type of hobby, name of the hobby, whether it is in progress, and the hobbies description. After concluding these tests, a new set of tests were created with parameters that one would not expect or put in; from random gibberish to overly complex sentences. The result was the same as manual input; a huge text block that would push other hobbies further down the list and several test failures due to foreign input. While the dropdown tests failed (since there is no selection for “fasdfasgeqr”), the only problem at that point simply involved a less-than-ideal amount of characters in the names and descriptions. The solution to it was the same as before, a character limit. If a character limit is placed, i.e. twenty-five characters, then the test would fail if the input exceeded twenty-five characters.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

Two additional tests were conducted on the sign-up page. One test involved input from an already existing user, and another which involved input with a large amount of characters. The first test worked as intended. Since a user already existed in the system with the same name, the test failed when attempting to submit the form. The second attempt was similar to the Testcafe tests on the Hobby List. Large input and possibility of overflow could occur especially on the navbar. Once again, character limits is the solution to this problem. Email character limits should be set to around thirty or thirty-five characters as well as first and last names. That way, the text from the users email will not flood the navbar.

E4. Static Analysis Updates

The static analysis tool choice for our application has remained unchanged. The Hobby Helper application still utilizes ESLint and no new static analysis tool has been considered. Regarding ESLint, our team continues to use it to manage our code syntax and help prevent errors in our code. Frequent uses of ESLint include using it to notice typos or warnings associated with the code. ESLint has been a simple yet effective tool to use when spotting errors within the code, whether it is immediate or through command prompt/shell commands.

E5. Dynamic Analysis Updates

Utilizing the dynamic analysis tool found has continued to prove difficult. Iroh.js has remained our team's dynamic analysis tool for Javascript as it was the only viable and readily available tool that could be used in the context of the code provided. As mentioned earlier, getting Iroh.js to work has been met with mixed results. Trying to implement Iroh.js in ways different to previous methods still remains difficult. This has to do with our application using React. Due to that, most of the components for Hobby Helper are created as .jsx files which aren't standard JavaScript files and are part of the reason why we have encountered difficulty using Iroh.js.

F1. Incident Response Plan

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

In the case of the application having potential data leaks, privacy problems, and/or potential bugs, an incident response will be put in place to resolve the issue(s) in question. While each person in the group is able to help resolve the issues, the organization of who does what is as follows: Matthew Kirts (public relations representative), Eric Riveria (escalation manager), Angeli Amascual (security engineer), and Jessica Tang (legal representative).

Matthew Kirts (mkirts@hawaii.edu) will be responsible for the public relations response, providing regular status updates on the application while also providing public assurance. Eric Riveria (erivera9@hawaii.edu) will be responsible for the escalation process, laying out the groundwork for any and all steps that are necessary to proceed with resolving the incident at hand. Angeli Amascual (aamascua@hawaii.edu) will find the issue(s) at hand and is responsible for resolving the issue. Additional help may be syphoned from the rest of the team depending on the severity of the situation. Jessica Tang (jktang@hawaii.edu) will serve as the legal representation, taking care of negotiations with other group(s) and taking lead of the legal department. She will also communicate frequently with all members, providing legal updates on the situation, especially with the escalation manager in order to adjust the escalation schedule and add/remove steps in the escalation process.

When an incident arises, a set of organized steps will be taken in order to resolve the problem(s) at hand:

- Inspect the source of the incident.
- Determine the severity of the incident.
- Details regarding the incident and what type of problem it is (received either through the user or any member of the team).
- Establish a timeline.
- Execute the incident response.

Depending on the incident, additional steps may be taken to resolve the issue. These additional steps include but aren't limited to; research regarding similar problem(s) and outside help i.e. online articles and/or public relations. Frequent communication among the team is imperative to help resolve the situation, and is therefore recommended that everyone involved remains in close communication on any and all work done to minimize the severity of the incident.

F2. Final Security Review:

The threat model for Hobby Helper has undergone changes since its first conception. Previously, our team sought to include a social aspect to the web application but due to difficulties with implementing it the social function was removed. Therefore, our threat model had to be adjusted accordingly. The current threat model was used to review the running of Hobby Helper and how data is accessed throughout the application. Hobby Helper employs the usage of Meteor to store its databases in the form of collections. Our web application's code is also formatted in a way such that the only information from the database that can be retrieved by the user is their own. The only exception to this is for administrators who have access to all information within the application.

The static analysis our team has used throughout development was ESLint. Through ESLint, our team members were able to have a consistent coding style throughout all of our pages. In addition, ESLint helped to prevent minor errors within our code such as misspelling. For Hobby Helper's release, our code has completely passed the ESLint test. As for dynamic analysis, Hobby Helper used Iroh.js to minimal success due to the barriers we faced from the conflicts between pure javascript and ReactJS.

After thoroughly running Hobby Helper as if we were an end user, our team determined its grade as Passed FSR. The previous issues discovered while conducting Fuzz Testing were resolved and our code passes the ESLint guidelines. In addition, reviewing the threat model in accordance to how data is moved and kept in the application shows that Meteor handles the security of the databases and access to information well.

F3. Certified Release & Archive Report:

The following is the link to the release version of the program: https://github.com/ics-427-team-readme/hobby-helper/releases/tag/1.0.

Hobby Helper has currently released version 1.0. In this release version, the web application allows users to create an account in order to manage their hobbies. After first signing

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

up for Hobby Helper, the user is directed to create a profile. This will allow the user to input their name, a bio description, as well as choose a profile icon. After creating their profile, users can access the “Hobby List” page to create a hobby. Once a hobby has been added to the Hobby List page the user can click on it to access the relevant kanban board. In addition, users are also able to edit and remove hobbies from the Hobby List page. Another feature included in Hobby Helper is that the user is able to view statistics regarding each individual hobby. The statistics page will allow the user to view all hobby items associated with the given hobby as well as view or adjust the ratings and reviews for each item.

The kanban board featured in Hobby Helper has three columns: “Backlog”, “Current”, and “Completed”. These three columns help the user keep track of the hobby items within the kanban board. From the kanban board, users can create a hobby item and assign it to whichever column it fits in. Users are also able to move hobby items between each column to show changes in the hobby item’s status. Hobby items can easily be edited or removed from the kanban board as well.

Hobby Helper also features a security question and answer system. When creating a profile, in addition to filling out their information the user will be asked to provide a security question and answer. These two inputs can be used to reset their password in the event they forget. If the user forgets their password, they can access the security page by clicking on the link provided in the “Forgot your password?” section on the “Login” page. It will ask the user to provide the email address associated with their account. If the email address exists within the database, the user will then be prompted to provide the answer to the associated security question. If the answer is correct, the user will then be asked to provide a new password for their account.

There are additional features that our team would like to implement as part of our future development plans. A future feature to include in Hobby Helper is the social functionality of the web application that was previously removed due to being too difficult and time consuming. With this new functionality, users would be able to view other users’ hobby lists, kanban boards, statistics, and profile if given permission. In order to implement this feature, Hobby Helper

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

would also need to implement additional privacy features to track and control what data is available for other users to view. Another possible feature would be upgrading the profile page to include information about the user's hobbies and their relevant statistics. For example, users would be able to list their top hobbies or hobby items and their ratings or reviews.

Another feature that could possibly be added to Hobby Helper is a friend system. This also ties into the social functionality stated earlier. This could allow users to set their privacy settings to not allow strangers to view their hobbies and information but allow their friends to view it. Adding this feature would require changes in the current collections to track friend lists as well as friend requests and the status of those requests. Finally, during the initial stages of brainstorming our team discussed the idea of a recommendation system for users' hobbies. This feature would provide users with recommendations on possible hobby items depending on the items they have already added and rated. If this is added in conjunction with the friend system, it could be possible to allow for friends to recommend hobby items to each other as well.

Hobby Helper is a web application built with ReactJS, Meteor, and MongoDB. In order to run this web application you can download release version 1.0 from the following link: https://github.com/ics-427-team-readme/hobby-helper/releases/tag/1.0. It will give you the option of either downloading the .zip or .tar.gz file. Either option works and once the download is complete then extract the files. Afterwards, open up the command prompt on your computer and move to the directory where the extracted files are located. This can be done using the "cd" command. After reaching the directory, "cd" once more into the "app" directory. From here, enter the command "meteor npm run start" to start running the application. Enter "http://localhost:3000/" on your desired web browser to view Hobby Helper's landing page. In order to exit this application, enter "CTRL C" on your command prompt and close the web browser's window containing Hobby Helper. There are no specific requirements in order to run Hobby Helper besides having a computer and web browser installed.

Angeli Amascual
Matthew Kirts
Eric Rivera
Jessica Tang
ICS 427
4/11/2021

Works Cited

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

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

Maier, Felix. Iroh, maierfelix.github.io/Iroh/

Microsoft. "Microsoft Security Development Lifecycle Threat Modelling." *Microsoft*, 2021, www.microsoft.com/en-us/securityengineering/sdl/threatmodeling.

2020. Microsoft Threat Modeling Tool. Microsoft Corporation.

Pluggable JavaScript linter. (n.d.). Retrieved February 21, 2021, from https://eslint.org/