Darnell Domingo
Katherine Piniol
Karen Wong

Introduction

The creators in question are a group called “ProGrand.” The application being built is called “Moneycation.” Moneycation is a budget planner that will be used for personal reasons. This app would take inspiration from the traditional money envelope way of budget planning. Moneycation would allow the user to determine the allocation of an amount of money and whether or not there was an overall addition or subtraction of money. The application would be for a single user and requires a login and password. The data would be stored locally on the user’s device.

It will be coded in Javascript as a React application.

Requirements

Security and Privacy Requirements

One security requirement would be encrypted data. Data, like the username, the user’s password password, the user’s budget, and the user’s expenses would not be stored as plain text, and instead be encrypted. This would be so that the data can not be easily read and understood without using the application itself, which would be password-protected. Another requirement would be a log of all invalid logins. This log would help the user to take note of whether or not any unauthorized logins by the specified user are made.

A few privacy requirements would be login credentials and password-protected backup data. Having login credentials would make it so that only the user the data is associated with can access and make any changes to both the data and the profile itself.

Both requirements, with the encrypted data, encrypted backup data, and login credentials in mind, would allow for prying eyes to be unable to access the data.

In order to keep track of security flaws throughout the development process, a spread will be used to log down any flaws and changes made to fix them. To help

ensure any potential unauthorized changes will not happen, a “canary in the stack” will be used.

Quality Gates

It is vital that their information is protected sufficiently because of the use of personal information to accurately determine budget. These are the current situations that have been identified for Moneycation.

- Critical
    - Lack of data protection
      Ensure users can’t access personal information inputted by other users
    - Privilege
      Ensure there isn’t a way for someone to obtain an “admin” role that would otherwise allow them to view the personal information of other users
- Important
    - Lack of notice and consent
      Transfer of sensitive information from user’s end without user consent
    - Tampering
      Modification of data that would interfere with user’s ending/allotted budget
- Moderate
    - Lack of data protection
      Ensure users can’t see spending habits or subscriptions of other users

Risk Assessment

The program stores Personally Identifiable Information (PII) locally on the user’s device. This information includes:

- Full Name
- Personal Income
- Personal Savings

And other private data that includes:

- Personal Budgets
- Spending Amount

- Passwords

Users are able to plan their spending and income according to their monthly/weekly/daily/annual budget with the software features. The user will be able to keep track of what subscriptions and bills that need to be taken care of.

Because of this, a password is needed in order to access the application on their device. Users will be able to exit and log back into the app to re-access their financial information. The password and other sensitive information will be stored locally and will need to be encrypted. Threat modeling will be needed in order to access how to store the PII securely. Security reviews should be done to test if any user on the device is able to access the application data without the password. For example, another user on the computer should not be able to go to the app data and retrieve the savings amount stored in that user account.

Design

Design Requirements

A few design requirements in Moneycation are the login credentials, budget planner function itself, and having encrypted data. Storage wise, it's all stored locally on the user's device.

The user would first need to create an account to use Moneycation. This is so that no unauthorized personnel can access the account. For example, if the computer used is a shared desktop with family members or a friend, the login would prevent any person other than the intended user from gaining access. When stored as a file, the data would be encrypted so if any non-intended user accesses the data files, it would not be in plain-text. Having the data stored in plain-text would defeat the purpose of having a password-protected account, so the encrypted data would further provide more security. As for backup data, it would also be password protected and encrypted, with the user being encouraged to keep the backup data on a different location other than the user's device.

The budget planner is meant to be used as a “money envelope” way of budget planning on a month to month basis. This is so that the user knows what the budget is per section, per month. For example:

➢ User “John Smith” has a savings of $2,000 in the bank at the beginning of the month
+ John went under budget last month, so he has an extra $200
+ John earns $2,500 each month
− John’s rent is $1,500 each month
− John’s phone bill is $100 each month
− John wants to allocate $300 for groceries
− John wants to allocate $100 for his rainy day fund
− John wants to allocate the last 50% for savings, so he has $350 for recreation
➢ John has $2,350 at the end of the month, which is a gain

Moneycation would provide the function in the example above.

A log of invalid logins would also be made to notify the user of any potential unauthorized logins. This log would be a clear indication that another person that is not the intended user, attempted to log in.

Attack Surface Analysis and Reduction

There will only be one level of privilege for users. The user should only be able to access their private data using their password. No administrative role is needed.

System User (Authorized):

- Will be able to access PII
- Will be able to access other private information

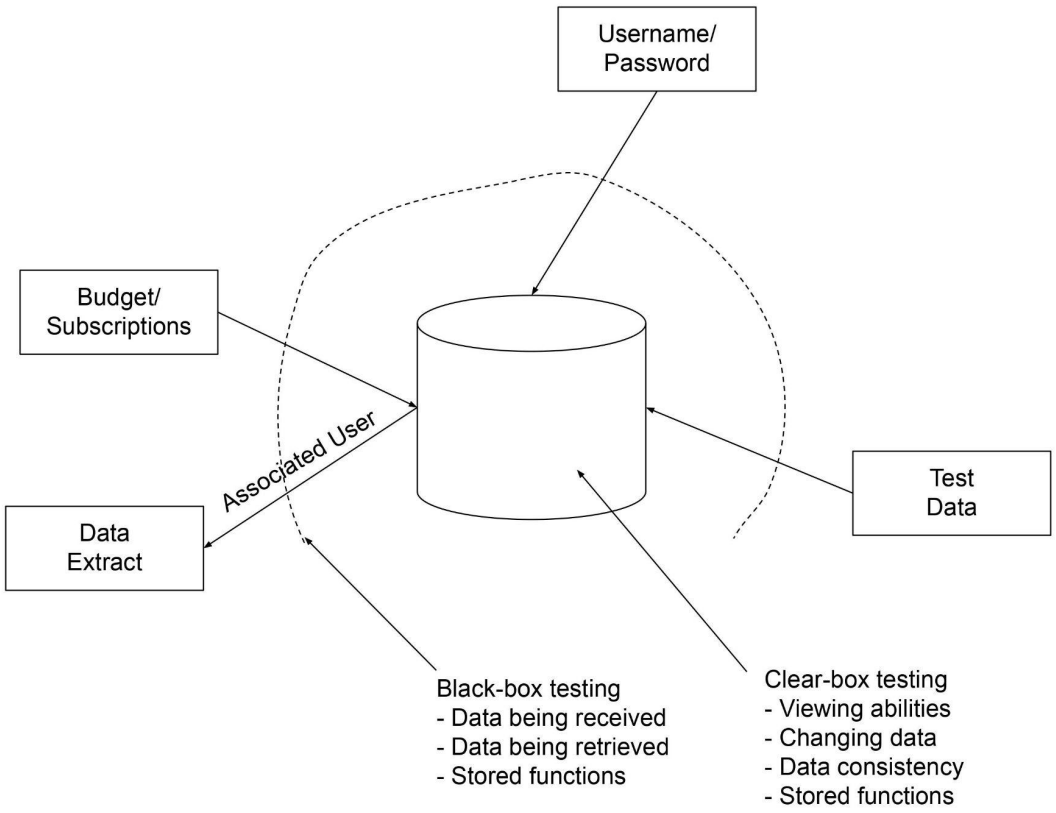
In order to get into the application, a password is needed. The log-in page will need to be tested for security vulnerabilities because it is a point of entry. Data entry forms may store information and caches. Because Moneycation is storing PII and data

on the local device, local storage would be another possible point of entry. High-risk areas of the code will include the page asking for the password, storing the personal data in local files, React App used to build the user interface, and security code (used for encryption and authentication).

Using similar applications like Goodbudget we can point out some vulnerabilities:

- Data is stored in an unsecure way
- Personal information can be accessed without logging in
- Personal information stored in the user account can be accessed through another user on the same device
    - If user does not log out on a public device, another user can view their information
- Injection where program accepts outside code as its own to be used to access secure areas and information

Threat Modeling


Username/
Password
Budget/
Subscriptions
Associated User
Data
Extract
Test
Data
Black-box testing
- Data being received
- Data being retrieved
- Stored functions
Clear-box testing
- Viewing abilities
- Changing data
- Data consistency
- Stored functions

Implementation

Approved Tools

Vendor/Source	Compiler/Tool/Language	Information
Oracle	Java SE 8 Platform	• JDK 8 (Java SE Development Kit 8) and JRE 8 (Java SE Runtime Environment 8) • Version should be "java version 1.8.0-bxx"
JetBrains	IntelliJ IDEA IDE	• Integrated developer environment for Java, JavaScript, HTML • Version 2021 and up
Oracle	Javac	• Compiler included in Java Development Kit and used by IntelliJ IDEA
Windows	Command Prompt	• Used to run application
Linux	Command Line	• Used to run application
MacOS	Command Line	• Used to run application
Meta	ReactJS	• React App • Free and open-source front-end JavaScript library for building user interfaces based on UI components • Uses http://localhost:3000/ to view and run app
Vercel	Semantic UI React	• Oficial React integration for Semantic UI • Provides React components, style sheets, themes, features and props
GitHub	Git/GitHub Desktop	• Code hosting platform to view issues and host easier collaboration
OpenJS Foundation	ESLint	• Static code analysis tool for identifying problematic patterns found in JavaScript code • Used to make code more consistent and avoid running into bugs when running the application

OpenJS Foundation	Node.js	• JavaScript runtime environment • Executes JavaScript code outside a web browser • Required to run ESLint
Meta	Jest	• JavaScript test framework that lets you access the DOM via jsdom
	JavaScript	• Programming language used to provide structure and style to web pages • Primarily used for client-side and server-side functionality
	HTML	• HyperText Markup Language used to structure the web page and content
	CSS	• Cascading Style Sheets used for describing the presentation of a document written in a markup language such as HTML

Deprecated / Unsafe Functions

componentWillMount

This is a function related to the React app tool. It's related to data retrieval from servers (Gray). It was often used incorrectly and used asynchronously when its purpose is synchronous. An alternative function to use is componentDidMount.

findDomNode

This is a function related to the React app tool. It allows the user to "access the underlying DOM node of a component" (Dopico). While not always directly typed by user, some libraries have been found to use this function in their methods. This function is only deprecated in StrictMode. A few alternatives is to either attach a "ref" to the DOM element or explicitly pass it as prop.

Date Methods

These functions are related to the javascript tool. A few functions are deprecated: getYear, setYear, toGMTString, and to LocaleFormat. The functions relating to the years

“are affected by the Year-2000-Problem” (MDN Web Docs). Instead, getFullYear, setFullYear and toUTCString should be used instead.

* *

This element is related to the HTML tool. It aligned content to the center of the page(Emekoma). An alternative is to use the css style “text-align: center” instead.

basefont

This is related to setting “the default font family, font size, and color of all the text in an HTML document” (Emekoma). An alternative is to directly assign the font family, font size, and color in css style.

Static Analysis

The static analysis tool that we will use for the development process is ESLint. What ESLint does is it makes sure the code is written and formatted properly. One thing it does is it checks if there are spaces added in between each class or function, which are indicated by “{“ and “}”. Another thing it does is it ensures packages and imports are at the top of the code. For example:

```
> import library.example;
>
> public static void main(String[] args) {
>       // Code goes here
> }
```

The above is what would pass the ESLint check, where as the following would not:

```
> public static void main(String[] args) {
>       // Code goes here
> }
>
> import library.example;
```

Verification

Dynamic Analysis

Vendor/Source	Compiler/Tool/Language	Information
Iroh.JS	Iroh.JS	• Dynamic code analysis for JavaScript • Allows you to do test cases, type checking, code visualizations, and more at runtime • Keeps track of the call stack so it is possible to view the code's flow and visualization

Iroh.JS Experience:

For this project we decided to use Iroh.JS as the dynamic analysis tool for Javascript. Unlike the other static analysis tools we use, it collects data which is only available at runtime. With Iroh, we attach listeners to parts of the code and when that code is reached during runtime, it will listen for calls, returns, loops, or any other supported code types.

Throughout the project's process we had trouble incorporating Iroh's testing abilities because most of the information and example of test cases are only what was given in their open source documentation (https://github.com/maierfelix/Iroh). However in the later stages, Iroh was used to inspect when functions are being entered and if the correct data was being passed.

Iroh was used in AccountsContext.js. Listeners were attached to localStorage item variables. This was to ensure account information like passwords and authentication were being changed properly when the function was called. Example of a listener code can be seen below:

```
stage.addListener(Iroh.VAR).on("after", function(e) {
   console.log("Account Added", e.name, "=>", e.value);
 }
);
```

Attack Surface Review

Vendor/ Source	Compiler/ Tool/ Language	Latest Version
Oracle	Java SE 8 Platform	Java SE 18
JetBrains	IntelliJ IDEA IDE	2021.3.3
Oracle	Javac	JDK 18
Windows	Command Prompt	
Linux	Command Line	
MacOS	Command Line	
Meta	ReactJS	17.0.2
Vercel	Semantic UI React	2.1.2
GitHub	Git/GitHub Desktop	2.14.2
OpenJS Foundation	ESLint	8.12.0
OpenJS Foundation	Node.js	16.9.0
Meta	Jest	27.5.1
	JavaScript	ECMAScript 2018
	HTML	HTML5
	CSS	CSS3

Fuzz Testing

For the project, one fuzz tester that was used was BooFuzz. It can be found on github, instructions for installation found in the github repository. In order to run BooFuzz on windows, a virtual environment using python was required, including other things like pip were needed to run it.

However, at this stage, functionality is still not fully implemented yet, so testing was run on the functions in the planner page. With the current inputs, it's limited to the

appropriate variable type. For example, if the input needs to be a number, it will not take anything that is not a number. And as for potential max integer value, numbers are displayed as an infinity sign in the case that a number gets too big.

Allowing the wrong type of input would break the function. On the planner page for example, it keeps track and calculates the end result of “amount used” and “max budget.” If it were a string instead of a number, then there would be an error. And in the case of a max integer value error, it has the potential of crashing the application.

Static Analysis Review

For our project, we are using ESLint to problematic patterns in the project’s JavaScript code. This tool works offline when the project is not running and uses the config rules as shown below. We have used ESLint to check syntax, find problems, and enforce code style.

Style Guide: AirBnb
Plugins: React
Rules:

```
"max-len": ["error", 120],
"no-console": "off",
"prefer-arrow-callback": "off",
"no-plusplus": "off",
"linebreak-style": "off",
"func-names": "off",
"no-restricted-syntax": "off",
"object-curly-newline": ["error", {
 "ObjectExpression": "always",
 "ObjectPattern": { "multiline": true },
 "ImportDeclaration": "never",
 "ExportDeclaration": { "multiline": true, "minProperties": 3 }}],
"react/jsx-filename-extension": [1, { "extensions": [".js", ".jsx"] }],
"react/jsx-tag-spacing": 0
```

ESLint problems can be marked as Warnings or Errors depending on the severity. After making changes and saving the file, these errors will appear below in the IntelliJ IDE

Problems window to notify us and can appear in the console window when we run the project.

Examples of the possible errors received:

"ESLint: 'useContext' is defined but never used. (no-unused-vars)"

"ESLint: Missing semicolon. (semi)"

"ESLint: The closing bracket must be aligned with the line containing the opening tag (expected column 13 on the next line) (react/jsx-closing-bracket-location)"

Full ESLint configurations can be found under '*my-app/.eslintrc.json*' file.

Dynamic Review

When using testcafe, all the pages worked and loaded fine. The test showed that the NavBar correctly navigated to the appropriate pages, and the links correctly matched the page it was on. Also, when working in different branches and working on pages that weren't linked to in the NavBar, the pages were showing up properly. An example of this was the SignUp page. The SignUp page wasn't linked to in the NavBar, but when changing the link appropriately, the page showed up and worked fine.

Other than the minor ESLint errors, there were no other errors that showed up when doing the testcafe tests. The pages showed no errors when they were actually run and being viewed.

Release

Incident Response Plan

This incident response plan is implemented in the response of a security or privacy escalation. The following incidents are included:

- Privacy related inquiries
- Contact of a privacy incident
- Privacy comitments failure
- Data Breaches

Privacy Escalation Team

A team will be set up consisting of an escalation manager, legal respresentative, public relations representative, and the security engineer. In case of an emergency, the privacy escalation team can be contacted by email at *moneycationtech@gmail.com*.

The **escalation manager** will be responsible for evaluating and determining the escalation process. They will be working with the reporting party to determine the following:

- If the incident is valid
    - Who reported the incident
- Team members involved with the incident
    - The escalation manager should work in close contact with those involved
- The impact of the escalation
    - Which parts of the application is affected
- The source of the escalation
    - Where the incident occurred

They should ensure that all parts of the escalation are resolved.

The **legal representative** will be responsible for understanding and recognizing law . When handling privacy related inquiries or privacy commitment failures, the legal representative should determine what laws to abide to and which regulations are affected in the privacy or security escalation process.

The **public relations representative** manages the images of the organization and the communication with media outlets or the reporting party. They are responsible for resolving any PR concerns that may rise in the privacy or security escalation process.

The **security engineer** is responsible for managing the security systems. This may include implementing and testing security features, troubleshooting, and planning application upgrades. If new security features are implemented, testing must be done for those features. In response to a privacy incident, a troubleshooting plan should be implemented. Because our team consists of three people, the role of the security engineer is shared between all members. The issues the security engineer face will be split between the privacy escalation team.

Privacy Incident Response Process

Once a privacy or security incident is reported to the team, information should be acquired from the reporting party. Additional information may be acquired from members responsible for working on this application. This information should include:

- The reporting party contact information
    - To determine the validity of the escalation
- Parts of the application involved
- What privacy or security feautre is being violated
- Log/Warning/Error documentation and messages
- The date and time incident is found and reported
- Additional information on the application performance

Once the incident is evaluated, the appropriate people should be contacted and a resolution will be discussed with the rest of the privacy incident response team. Depending on the information given, the resolution may include:

- Pirvacy or security commitment change
- Short-term/Long-term application changes
- Users will be notified of the data breach
- Users will be notified of the security breach
- Public relations outreach
    - PR release of information concerning the incident
    - Incident details and incident solution being taken
- Update on documenation

The solution the privacy incident should resolve all concerns of the reporting party and ensure this event and similar events will be prevented from happening again in the future.

Final Security Review

Final Threat Model and Analysis Review

The **threat model** was pretty accurate in terms of how we wanted the program to set up. There were usernames and passwords associated with each account. Information such as budget and income were inputted by the user, and they would be shown or extracted based on the *associated user*. In other words, each user had their own data associated with them, so they could not access what another user inputted, which was the aim to prevent any threats of outside access.

The **static analysis tool**, ESLint, worked great for the program. It was able to tell us even immediately if there was an error on how the code was formatted or if imports and libraries were missing. It was a great tool to use *during* development of the program

and even *after we* had to go back and check on the program to see if everything was fine. There were some slight issues with it at first though, with one of them being the installation and setup of the tool. We used ESLint before in other classes and programs, and it was weird, in the sense that ESLint didn't properly register the program at first. To fix this, we had to reinstall and redirect it. In the end, ESLint did its job in doing a static analysis and overview of the project.

The **dynamic analysis tool**, TestCafe, did its part just like ESLint. It was able to assess if pages loaded up properly and if parts of the pages function correctly. For example, if the NavBar and the links in the NavBar didn't redirect correctly, feedback was given and we could go in to fix it. This was not the case, however, as even throughout development, we were loading up our pages to make sure we could access them and load up whatever we needed to in them. In a sense, we confirmed before using TestCafe, but it's always good to do an after check as some bugs or errors may occur after the "final" push or commit to the program, allowing us to do one more final push.

Final Run and FSR Grade

In the final run of the program, it was found that page accessibility and functionality worked fine. The landing and every other page displayed what we originally wanted to display, so there were no issues there. So, after conducting a final run and review of the threat models, static and dynamic analysis tools, the final grade to be given is **passed FSR** (Final Security Review). According to the guidelines set by the FSR process, the threat models and tools used for static and dynamic needed to be reviewed, which was done. There were some vulnerabilities found in previous versions, mainly due to ESLint not being configured properly. However, once it was, we were able to identify what needed to be fixed, in which fixes were applied to to fix any vulnerabilities found.

Certified Release & Archive Report

Moneycation version 1.0.0 currently allows the user to input their income and add different budget topics. For each budget topic, the user is allowed to name it as the user pleases. The user must also set a max allowed budget for each topic and add how many dollars was used for each topic. For each topic, there is a colored bar indicating how much was used. For example, the user may have one labeled “Food,” with money set as “$90/$100,” in this case, it would be red. Red means that almost the entire allowance for this budget is used, or that it is over budget.

This app requires a login to use to ensure no unauthorized personnel can access the information. In the settings, the user can change their password and delete their account, also along with all the budget they have set, so in the case someone else would like to use the app, they can’t access the previous user’s data.

Future development plans would allow the user to both import and export data. For the planner section, the user would be able to have separate forms for each month. Each month, the end result income would carry over to the next. And there would be an automatic carry over for each budget topic so the user would not have to set it again each month. For the security part, all the data would be encrypted. Another thing is to code it in something other than a react app so it would not show up through “localhost:3000/” and instead as an app on your desktop.

Read Me File

A web / desktop application for ICS 427 Software Quality Assurance.

Completed Tasks

- Nav Bar Html
- Page for Sign In / Sign Up [Sign In/Sign Up]
- Planner HTML
- Button for import/export data [Settings]
- Button to delete account [Settings]
- Button to change password [Settings]
- Create logic for Sign In / Sign Up

- Import functionality for deleting account [Settings]
- Import functionality to change password [Settings]
- Editing functionality for budget
- Login specific pages

Assignment 5: What each team member worked on within the project so far.

Katherine Piniol (https://github.com/427-ProGrand/Moneycation/tree/piniolk-work)

- Nav Bar HTML (Complete)
- Nav Bar Functionality (Complete)
- Settings Page HTML (Complete)
- Login/Signup Functionality (Complete)
- Settings Page Functionality (Complete)
- Editing functionality for budget
- Start wiki homepage

Darnell Domingo (https://github.com/427-ProGrand/Moneycation/tree/branch-login) (https://github.com/427-ProGrand/Moneycation/tree/branch-signup)

- Login/Signup Page HtML (Complete)

Karen Wong
(https://github.com/427-ProGrand/Moneycation/tree/karenwon-budget-form)
(https://github.com/427-ProGrand/Moneycation/tree/karenwon-view-budget)

- New Budget Planner Page HTML (Complete)
- New Budget Planner Page Functionality
    - Add Budget (Complete)
    - Delete Budget (Complete)
    - Add Income (Complete)
    - Date Selector (Complete)
- ESLint Config (Complete)
- Login specific pages

Links

Repository Release Version 1.0.0 Wiki Page [Documentation]

Technical Notes

Specifications

- @semantic-ui-react/css-patch@1.0.0
- @testing-library/jest-dom@5.16.4
- @testing-library/react@12.1.4
- @testing-library/user-event@13.5.0
- eslint-config-airbnb@19.0.4
- eslint-plugin-import@2.26.0
- eslint-plugin-jsx-a11y@6.5.1
- eslint-plugin-react-hooks@4.4.0
- eslint-plugin-react@7.29.4
- eslint@8.13.0
- iroh@0.3.0
- react-datepicker@4.7.0
- react-dom@17.0.2
- react-router-dom@6.3.0
- react-router@6.3.0
- react-scripts@5.0.0
- react-select@5.3.0
- react@17.0.2
- semantic-ui-css@2.4.1
- semantic-ui-react@2.1.2

How To Install

1. Install Node
2. Download the zip file and extract to designated folder.
3. Open up command line and move into the Moneycation folder, then move into "my-app."
4. Run "npm install"
5. Run "npm start"

Closing Thoughts

There were many challenges working on this. We had no experience when it comes to security and analysis, so we were going in completely blind. Some unexpected challenges were using React, the most updated version we used had bugs that required very specific patches and workarounds just to get it to work. Some disappointments we had is not being able to implement everything we initially wanted to, both planner functions and the security side of things.

Resources

1. Gray, Josh. React: componentWillMount to be deprecated! Northcoders. [cited 2022, February 20]; Available from https://northcoders.com/company/blog/react-componentwillmount-to-be-deprecated

2. Dopico, Clara. Getting rid of findDomNode in your React application. Medium. [cited 2022, February 20]; Available from https://medium.com/trabe/getting-rid-of-finddomnode-method-in-your-react-application-a0d7093b2660

3. MDN Web Docs. [cited 2022, February 20]; Available from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Deprecated_and_obsolete_features

4. Emekoma, Hafsah. Deprecated HTML elements (and what to use instead). LogRocket. [cited 2022, February 20]; Available from https://blog.logrocket.com/deprecated-html-elements-and-what-to-use-instead/#center