CS3354 Software Engineering
Final Project Deliverable 2

Budget Planner

Ethan Emmanuel, Om Hirpara, Alaa Jalali, Andrew Tran

1. **[5 POINTS]** Well described delegation of tasks, i.e. who did what in the project. Now that your project is complete, you are required to submit the delegation of tasks from beginning of the project until the end. Please make sure to fairly distribute tasks in the team and remember that at the end of the semester, each member of a team will receive the same grade. See grading policy below for more detail. If no/poor contribution by a member, please specify clearly so that we can grade each student fairly.
    a. Andrew researched the requirements and created the class diagram. He then estimated the cost of hardware and software products. He also created a test plan for our software with a sample unit code with defined test cases that show our results. Finally, he committed the final project to our GitHub Repository.
    b. Ethan initialized the GitHub repository, created sequence diagrams for the project, and chose our architecture pattern. He also calculated the estimated cost of personnel which includes the number of people to code the end project and training cost. He then viewed the competition and made a comparison of our product to what we are up against.
    c. Om selected our process model and created the use case diagram. He handled Project Scheduling by analyzing a feasible workload split among team members and estimating a date range based on this. Cost, Effort, and Pricing was also estimated by him using Function Point (FP). He then wrote the conclusion to wrap up the project.
    d. Alaa wrote the project draft and gathered a list of resources for everyone to look at. These resources will be cited in the proper formatted way. He put together the slides that contain bulleted points of info for our project. Task delegations were also planned and tracked by him.
2. **[10 POINTS]** Project Deliverable 1 content.

Project Description

The focus of our application is going to be options and accessibility. On the App Store, the most critically appraised Budget Planning app that is similar to ours is a financial management tracker called Mint. It boasts a secure system and high quality UI that shows visual indication of spending with circle graphs and bar charts, though feature wise, we believe this to be the bare minimum that

Budget Planner

Group members: Andrew Tran, Om Hirpara, Ethan Emmanuel, Alaa Jalali

Project Description: The Budget Planner is a tool that helps users manage their personal finances and track their spending habits. With this application, users can create a budget, set spending limits for various categories, and input their daily transactions. The application automatically tracks the user's spending in real-time and provides insightful reports and graphs to help users understand their spending patterns. Users can also set financial goals and track their progress towards those goals.

Motivation: As we become more and more independent, we have to become financially aware in our lives. Once we graduate, we will get jobs and start earning our own salary. Because savings is an integral part of retirement, we wanted to be able to track our expenses, such as rent, groceries, etc. Some other spendings not included under expenses is investments. Including how much of our salary we allocate into investments (stocks, Roth IRA, 401k) is a great way to narrow down net income. This will also allow us to notice whether we as individuals need to cut down on our spending habits.

Tasks:
Andrew will be tasked with researching the requirements for this project.
Ethan will be tasked with writing the documentation and designing for this project.
Om will be tasked with researching the economics/expenses
Alaa will be tasked with the testing process and procedures necessary for this project.

Scholar paper: We are not planning to write a scholarly paper in the end.

software like this should be capable of, much less our own app. Where we plan to stand out is by providing better accessibility and wider options for a wider range of users. A core requirement that most of the competition has is access to the user's checking and savings account, which allows for automatic tracking of spendings. While it's a needed feature that we don't plan to forgo, having it as a requirement lacks flexibility. It is narrow focused and assumes the user is set on categorizing and analyzing the funds within just one account. It also requires a connection to the user's respective bank in order to utilize the app's features. While Mint has built a trustworthy reputation that may merit this level of security, we believe that when viewed from the consumer standpoint, it can still be considered an unnecessary hurdle for a rather simple concept application. We will allow the user to access our features without having to connect their bank accounts, though the option to connect them remains. This means mandatory input from the user is a necessity, but no less should be expected if they forgo the bank connection. Once the info is entered, the app does all the work and hasoptions to display the info. Tracking, planning, comparisons, categorization, and consistent reviews can be generated on a daily, weekly, monthly, or any custom time interval basis. The goal isn't just to help the user plan their budget, but to achieve that plan with incentives on a smaller scale and a larger scale.

GitHub Repository

- Each member used their own personal GitHub account.
- Repository created named "3354-FrugalForce"
- All team members and TA added as contributors to GitHub repository.
- README.md has been committed to repository with "3354-FrugalForce" as content.
- "project_scope" file created and committed to repository.
- URL of repository: https://github.com/nartmobile/3354-FrugalForce

Delegation of Tasks

Question 1. AL
Question 2. Andrew, Ethan
Question 3. Team
Question 4. Om
Question 5. Andrew
Question 6. Om
Question 7. Ethan, Al
Question 8. Andrew
Question 9. Om

Previous Tasks (from team proposal):
Andrew will be tasked with researching the requirements for this project.
Ethan will be tasked with writing the documentation and designing for this project.
Om will be tasked with researching the economics/expenses

Alaa will be tasked with the testing process and procedures necessary for this project.

Software Process Model

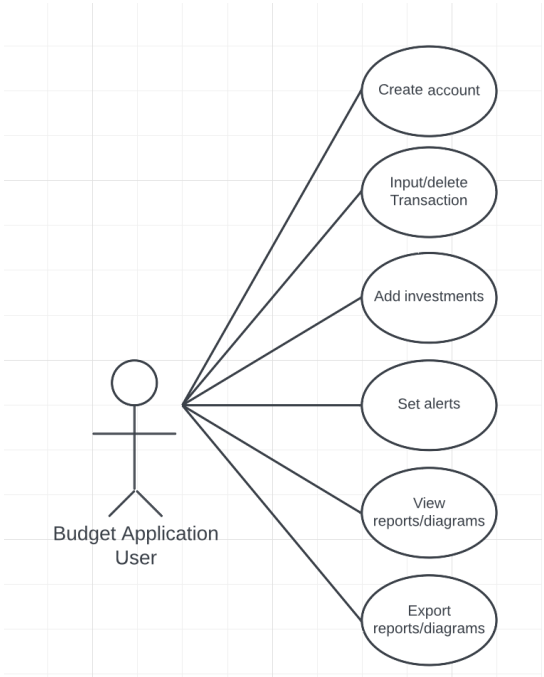
The prototyping model is employed in this project because it relies on feedback from users and clients on the current features that need improvement or certain features that need to get added in the next revision. When the new prototype is created using the user feedback, the cycle is repeated. The budget tracker application is going to be created specifically for users who have a difficult time managing their finances. However, new issues may arise for some users, which may need to be addressed through software features. For instance, if the user receives a biweekly paycheck, then they can set up a reminder using the alert feature on the budget tracker application to remind them to enter in their latest finances, such as profits and expenses(entertainment, groceries, etc.) within the past two weeks. If there is an issue with the application or a specific feature that needs to be included, a new prototype containing the fixes would be released, allowing the users to experience the latest installment of the application and provide their feedback on further improvements. Repeating this cycle of creating prototypes and receiving feedback on them can allow us to bring forth more likable features.

Software Requirements

- Functional requirements
    a. An user shall be able to manually add and remove transactions from their transaction history
    b. The system shall generate reports that can be categorized by expense type, time period, or a mix of both, on demand.
    c. The system shall generate projections towards user-defined goals each day.
    d. The system shall be able to import generated transaction data (in the form of .csv or .txt files) and add it to an user's transaction history.
    e. The system shall be able to export the user's transaction history (in the form of .csv or .txt files).
    f. The user shall be able to create custom filters for their transaction history (based on string matching).
- Non-functional requirements
    a. The system must be usable and accessible for all users, including those who suffer from disabilities such as blindness.
    b. The system's performance shall prioritize loading transaction data within 5 seconds, while graphs and reports shall be loaded within 30 seconds.
    c. The system must not use more than 300MB of RAM. It shall prioritize loading textual data first if user storage is insufficient.
    d. The system must be online and available 24/7. Any scheduled downtime shall not exceed 30 minutes.

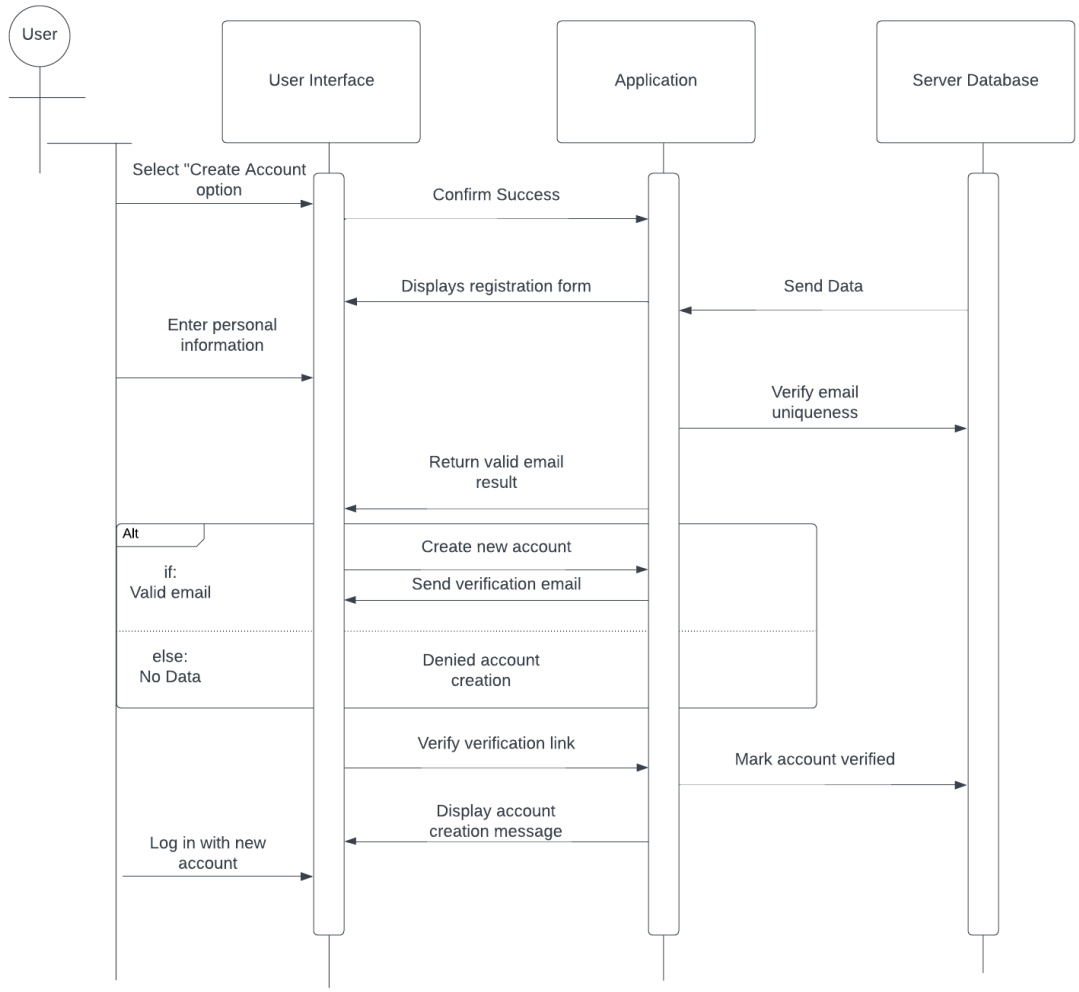
e. The system will require two-factor authentication in addition to industry standard encryption for user accounts.
f. The system will be hosted on the internet.
g. The system will be operated by an user that creates their own account using a unique username and password.
h. The system's database will be hosted using AWS (Amazon Web Services).
i. As a service hosted in the United States, the system shall maintain account security standards as determined by major American banks (Chase, Bank of America, etc.). [ASSUMPTION: There exists a guideline for account security that banks enforce in order to enter an agreement for their data to be formally given to the service.]
j. The system shall not share any of the users' transaction data or reports without the user's explicit consent.
k. The system shall be provided free of charge and as is, with its running costs funded by the creators of the product. [ASSUMPTION: The product runs on the goodwill of the creators]
l. The system's database credentials shall be entrusted only to the database architect (i.e. not all members of the development team).

Use Case Diagram

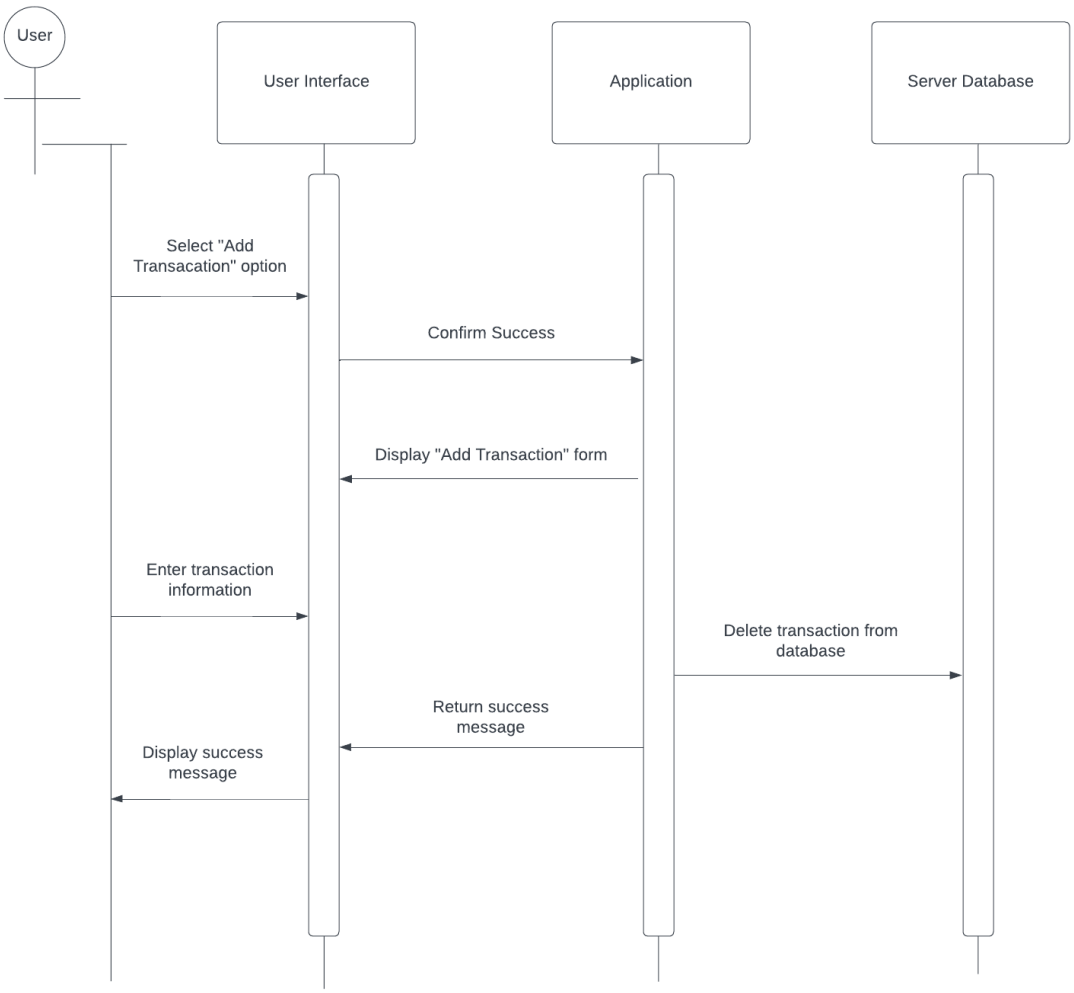


Sequence Diagrams

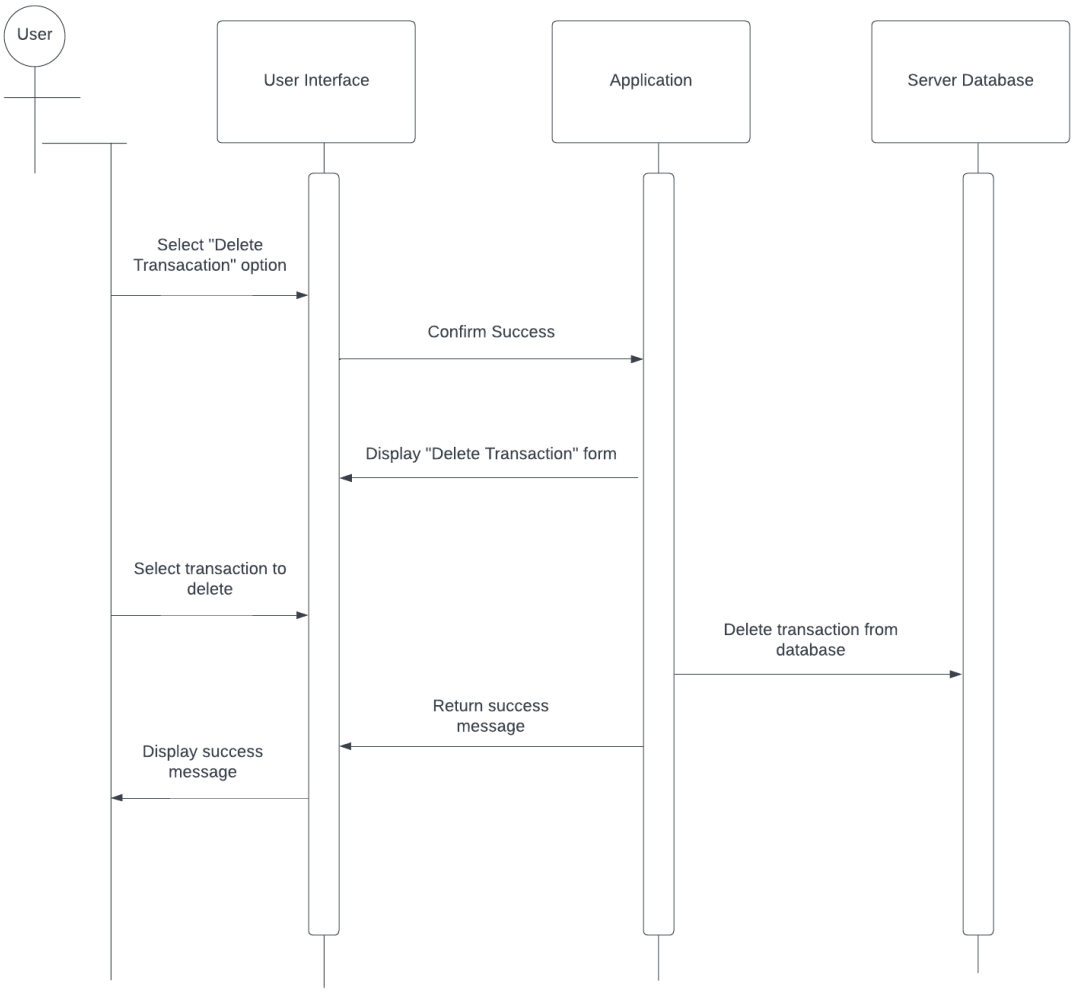
Create Account



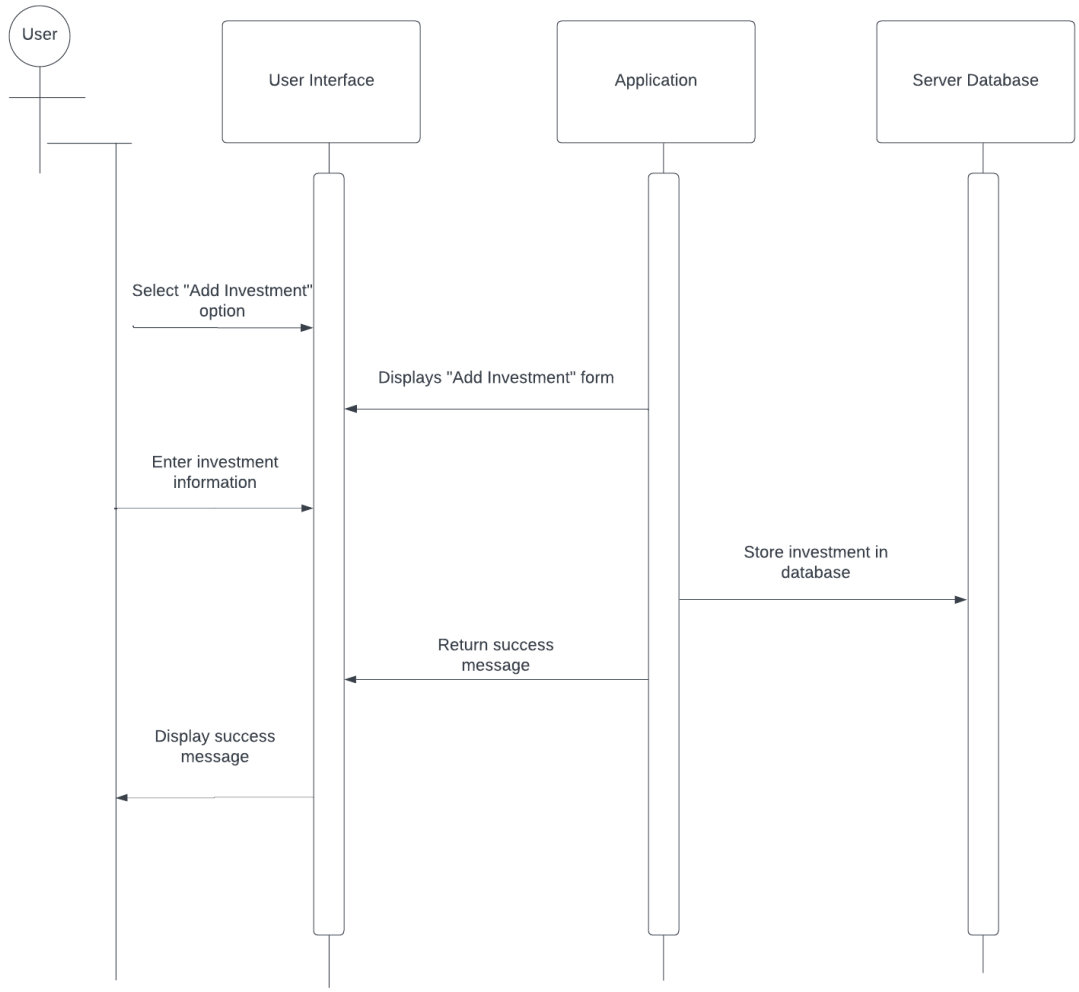
Add Transaction



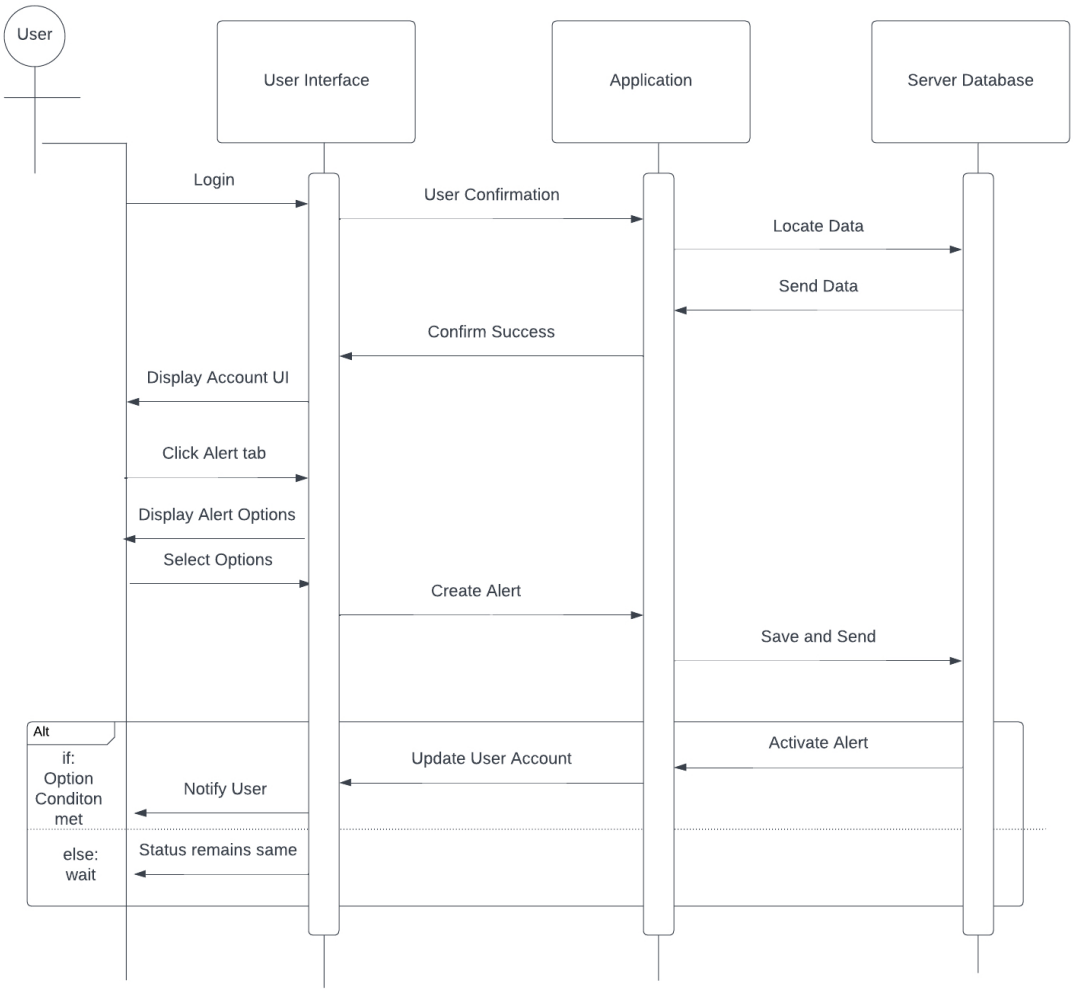
Delete Transaction



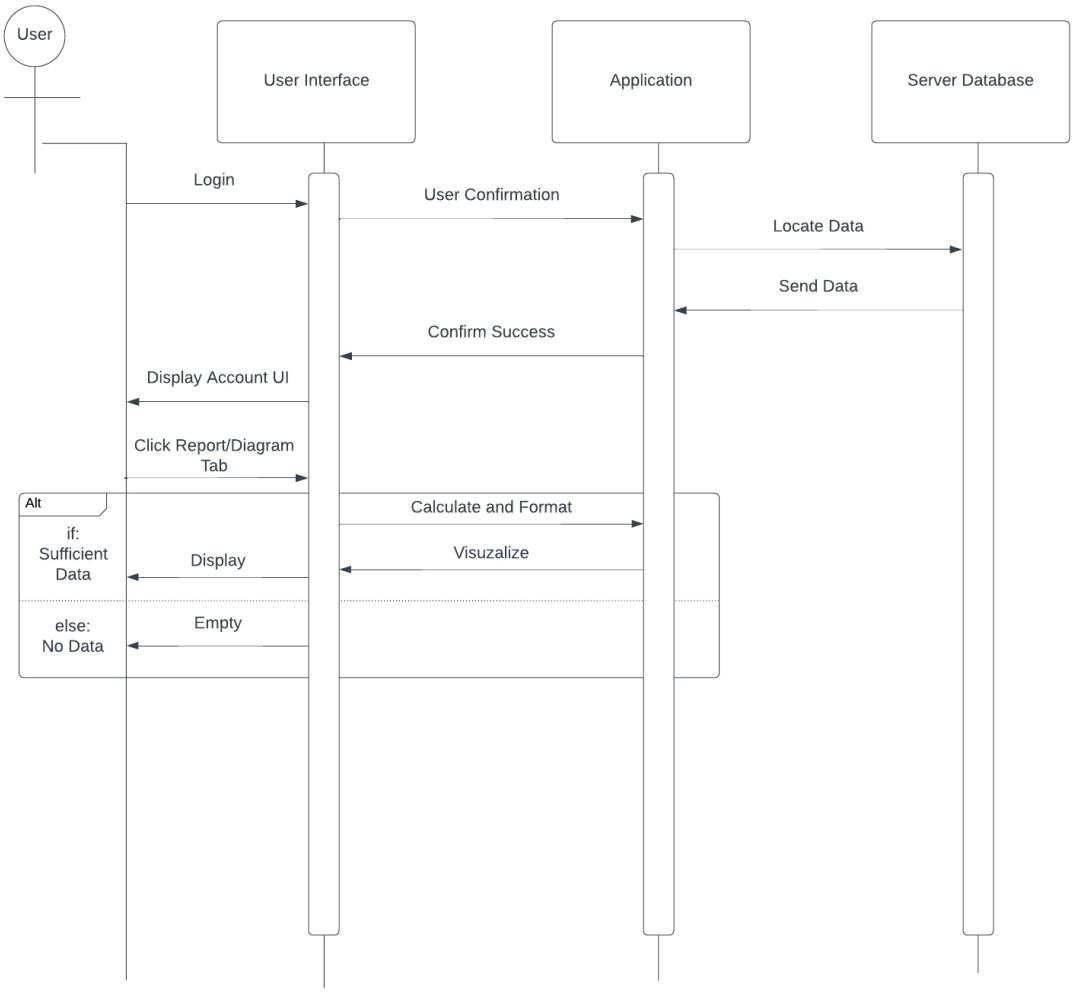
Add Investment



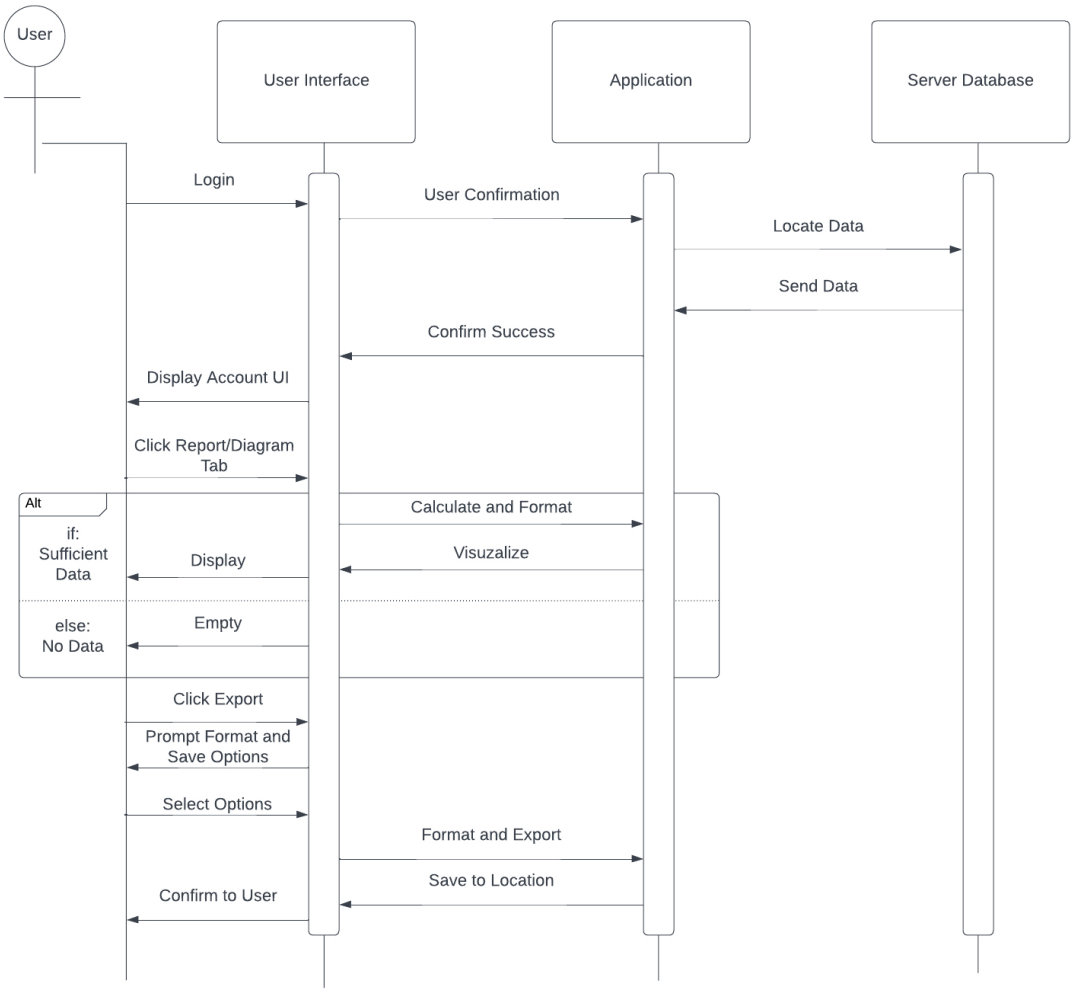
Set Alerts


User
User Interface
Application
Server Database
Login
User Confirmation
Locate Data
Send Data
Confirm Success
Display Account UI
Click Alert tab
Display Alert Options
Select Options
Create Alert
Save and Send
Alt
if:
Option
Conditon
met
Activate Alert
Update User Account
Notify User
else:
wait
Status remains same

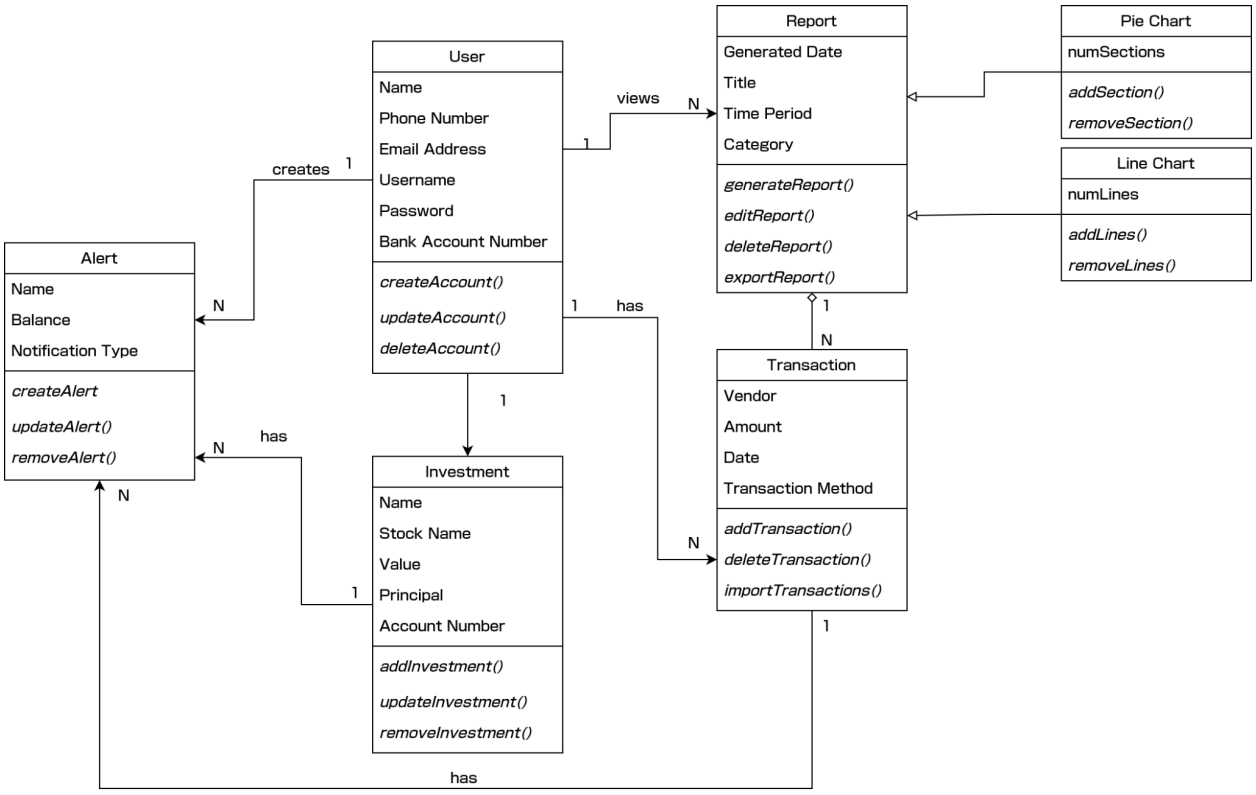
View Reports/Diagrams



Export Reports/Diagrams



Class Diagram



Architecture Pattern

We have decided to use the MVC software architecture model because the MVC offers data management as well as allowing the presentation of data in various manners. The Model component would contain the user's budget data, such as monthly income and expenses (rent, groceries, entertainment, etc.). There would also be yearly data that would contain yearly income and expenses (taxes, yearly subscriptions, etc.). The View component is where the data is rendered for the user to see. For the user to view historical trends, there could also be graphs and charts of their personal spending in certain categories of spending. The Controller component would contain the behavior of the application, such as calculations. This means that if the data changes, such as at the end of the month when the user enters their income and expenses, the data would be changed and presented in the View component. The repository model would not be suitable if a failure occurs with the repository because the data would not be accessible. The client-server model is not ideal as it is mainly utilized when data is shared by more than one individual and in the case of our idea, only one person will have access to their own data. The pipe and filter pattern does offer processing of data, but it doesn't allow the data to be presented in various manners. Finally, a downside of the layered architecture pattern is that differentiating between the layers can become difficult, ultimately creating issues.

3. **[35 POINTS]** Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing: Include a detailed study of project scheduling, cost and pricing estimation for your project. Please include the following for scheduling and estimation studies:

a. **[5 POINTS]** Project Scheduling. Make an estimation on the schedule of your project. Please provide start date, end date by giving justifications about your estimation. Also provide the details for whether weekends are counted in your schedule and what the number of working hours per day are for the project.

Start Date: 7/1/2023
End Date: 1/31/2024
The start date is August 1st because we need time to gather information about this project. Although we have most of the requirements, we need to research and take time to learn about the technologies we could incorporate in order to create the best version of this application. The end date is January 31st of the following year because we would like to take the final month to do the last rounds of major testing where we test each and every functionality of the application. Weekends are not counted in the schedule, and the working hours per day of the project are 4 hours. However, within the final month (1/1/2024 - 1/31/2024), weekends will be utilized and the working hours per day may increase depending on if the integration testing is on schedule.

b. **[15 POINTS]** Cost, Effort and Pricing Estimation. Describe in detail which method you use to calculate the estimated cost and in turn the price for your project. Please choose one of the two alternative cost modeling techniques and apply that only: **(Function Point)**

The method we are using to calculate the estimated cost and price for the project as a whole is the Lines of Code experience based method. This is because we would like the pricing to reflect the amount of work that actually goes into creating the application (i.e lines of code). The function point method is used to calculate the effort.
Estimated lines of code needed for the application: 15,000 lines
Cost per line of code is approximately $13
$13 x 15,000 = $195,000

	Function Category	Count	Complexity			Count x Complexity
			Simple	Average	Complex	
1	Number of user input	6	**3**	4	6	(3)(6) = 18
2	Number of user output	8	**4**	5	7	(4)(8) = 32
3	Number of user queries	6	**3**	4	6	(3)(6) = 18

4	Number of data files and relational tables	25	**7**	10	15	(7)(25) = 175
5	Number of external interfaces	5	**5**	7	10	(5)(5) = 25
	Total					**268**

Determine Processing Complexity (PC)
(1) Does the system require reliable backup and recovery? **5**
(2) Are data communications required? **3**
(3) Are there distributed processing functions? **2**
(4) Is performance critical? **4**
(5) Will the system run in an existing, heavily utilized operational environment? **4**
(6) Does the system require online data entry? **3**
(7) Does the online data entry require the input transaction to be built over multiple screens or operations? **2**
(8) Are the master files updated online? **5**
(9) Are the inputs, outputs, files, or inquiries complex? **2**
(10) Is the internal processing complex? **4**
(11) Is the code designed to be reusable? **2**
(12) Are conversion and installation included in the design? **0**
(13) Is the system designed for multiple installations in different organizations? **4**
(14) Is the application designed to facilitate change and ease of use by the user? **0**

Compute PCA/FP

PCA = 0.65 + 0.01(5 + 3 + 2 + 4 + 4 + 3 + 2 + 5 + 2 + 4 + 2 + 0 + 4 + 0) = **1.05**

FP = GFP * PCA = 268 * 1.05 = **281.4**

E = FP / Productivity = 281.4 / 60 = 4.69 **= approx. 5 person weeks**

Team size = 4 so,

D = 7 person weeks / 4 people = **1.75 week**

c. **[5 POINTS]** Estimated cost of hardware products (such as servers, etc.)
    i. Work space for development team - $1600
        1. Desk - $200
        2. Computer - $800
        3. Monitor - $200 (x2) = $400
        4. Peripherals - $100
        5. Chair - $100
    ii. Sever Rental - $70/mo
d. **[5 POINTS]** Estimated cost of software products (such as licensed software, etc.)
    i. Figma (Prototyping and design software) - $12/mo * 5 (# of devs) = $60 / mo
    ii. Trello (Timeline management) - $5/mo * 5 (# of devs) = $25 / mo
e. **[5 POINTS]** Estimated cost of personnel (number of people to code the end product, training cost after installation)

i. Project Manager: Oversees the entire project, communicates with the client, and manages the development team. Average hourly rate: $150.
ii. Software Architect: Designs the software architecture and ensures that the system is scalable and maintainable. Average hourly rate: $150.
iii. UI/UX Designer: Designs the user interface and user experience. Average hourly rate: $100.
iv. Front-end Developer: Develops the client-side code of the application. Average hourly rate: $125.
v. Back-end Developer: Develops the server-side code of the application. Average hourly rate: $150.
vi. Quality Assurance Engineer: Tests the application for bugs and ensures that it meets the requirements. Average hourly rate: $125.
vii. Technical Writer: Creates technical documentation for the application. Average hourly rate: $75.
viii. Total cost: $875/hr
ix. Training cost after installation: no training as the application can be maintained by the same team, and the application is used by everyday people

4. **[10 POINTS]** A test plan for your software: Describe the test plan for testing a minimum one unit of your software. As an evidence, write a code for one unit (a method for example) of your software in a programming language of your choice, then use an automated testing tool (such as JUnit for a Java unit) to test your unit and present results. Clearly define what test case(s) are provided for testing purposes and what results are obtained (Ch 8). Include your test code as an additional document in your zip file submitted.

a. We would develop a suite of tests that would ensure the functionality of our program. This would be a form of bottom-up testing, where we seek to establish working components and modules and build our way up towards a main program. Attached is a sample test that would check to see if we can successfully obtain the most expensive purchase from a given list of transactions. The module is written in JavaScript and is tested using Jest. The test returns valid if, in the sample data, it correctly returns the id of the most expensive item (the speakers).

5. **[10 POINTS]** Comparison of your work with similar designs. This step requires a thorough search in the field of your project domain. Please cite any references you make.

a. Empower is a web-based budgeting and financial management application that allows users to track their spending, create a budget, and manage their investments. Empower's primary focus is on investment management, so users who have a significant amount of investments may find it to be a useful tool. One unique feature of Empower is that it provides users with access to a team of financial advisors who can provide personalized investment advice. However,

unlike some of the other applications on this list, Empower does not offer bill payment functionality.

b. EveryDollar is a web-based budgeting application that helps users create a budget and track their expenses. EveryDollar uses the "zero-based budgeting" approach, which means that users assign every dollar of their income to a specific budget category. EveryDollar's user interface is simple and easy to navigate, making it a good option for users who are new to budgeting. However, EveryDollar does not provide investment tracking or bill payment functionality.
c. PocketSmith is a web-based budgeting application that allows users to create a budget, track their spending, and forecast their finances. PocketSmith's forecasting feature is a unique selling point, as it allows users to see how their financial situation will look in the future based on their current spending habits. PocketSmith also provides users with personalized insights and advice based on their financial data. However, PocketSmith's user interface can be overwhelming for some users, and it does not offer investment tracking or bill payment functionality.
d. BudgetBakers is a web-based budgeting application that helps users track their expenses, create a budget, and set financial goals. BudgetBakers offers a "smart assistant" feature that provides users with personalized financial advice and insights based on their spending habits. BudgetBakers also offers investment tracking functionality, making it a good option for users who want to manage their investments alongside their budget. However, BudgetBakers does not offer bill payment functionality.
e. Buxfer is a web-based personal finance application that allows users to track their expenses, create a budget, and split expenses with friends and family. Buxfer offers social features, such as the ability to split bills and expenses with other users, making it a good option for users who share expenses with others. Buxfer also offers investment tracking functionality. However, Buxfer's user interface can be confusing for some users, and it does not offer bill payment functionality.
f. Mint is a web-based budgeting application that helps users track their spending, create a budget, and pay bills. Mint offers a wide range of features, including investment tracking and personalized financial advice based on users' spending habits. Mint's user interface is intuitive and easy to use, making it a good option for users who are new to budgeting. However, Mint's bill payment functionality can be limited, and some users have reported issues with syncing their financial accounts.
g. FrugalForce has the potential to stand out by offering better accessibility and wider options for a wider range of users. The app plans to provide features that go beyond the bare minimum, such as tracking, planning, comparisons, categorization, and consistent reviews, which can be generated on a daily, weekly,

monthly, or any custom time interval basis. The option to connect bank accounts remains, but users can also access the app's features without connecting their accounts. Unlike some competitors that require access to the user's checking and savings account, FrugalForce allows for more flexibility and a broader focus, assuming the user is not set on categorizing and analyzing the funds within just one account. Incentives on a smaller and larger scale are also offered to help users achieve their budgeting goals. Overall, FrugalForce aims to differentiate itself by providing better accessibility and options for a wider range of users, as well as offering incentives to help users achieve their budget plans. It also allows users to access features without connecting their bank accounts, which may be more flexible for some users. However, it is important to note that some of the previous web-based budget planning applications offer more advanced features such as forecasting, scenario planning, and investment tracking, which FrugalForce may not offer.

6. **[10 POINTS]** Conclusion - Please make an evaluation of your work, describe any changes that you needed to make (if any), if things have deviated from what you had originally planned for and try to give justification for such changes.

Some of the changes we had to make had to do with the scheduling and testing portion of the project. Regarding the testing, we initially were planning on implementing a form of top-down integration testing. However, we found that starting with the main control module would not allow us to have the availability of subordinate components. Therefore, we switched to bottom-up testing, which does provide that as subordinate components are combined into clusters to perform a specific software functionality. Another aspect of our project that we had to alter was the scheduling. We initially planned on starting the application on 7/1 and ending on 12/31. However, anything that can go wrong will go wrong. Consequently, we gave ourselves an extra month for final rounds of major testing where we test each and every functionality of the application. Therefore, the development of this application will now end on 1/31.

7. **[5 POINTS]** References: Please include properly cited references in IEEE paper referencing format.

[1] E. Eldon, "Buxfer is giving Mint, other personal finance sites, a run for their money," *VentureBeat*, 04-Jan-2008. [Online]. Available: https://venturebeat.com/business/buxfer-is-giving-mint-other-personal-finance-sites-a-run-for-their-money/. [Accessed: 21-Apr-2023].

[2] J. Biberdorf, "Is Pocketsmith Legit?," *Modest Money*, 16-Mar-2023. [Online]. Available: https://www.modestmoney.com/is-pocketsmith-legit/. [Accessed: 21-Apr-2023].

[3] K. Payne, “Mint budgeting app review,” *Forbes*, 07-Mar-2022. [Online]. Available: https://www.forbes.com/advisor/banking/mint-budgeting-app-review/. [Accessed: 21-Apr-2023].

[4] M. Georgiou, “Mobile app development: How much does it cost? (2023 guide),” *Imaginovation*, 21-Feb-2023. [Online]. Available: https://imaginovation.net/blog/how-much-does-it-cost-to-build-mobile-app/. [Accessed: 21-Apr-2023].

[5] R. Farrington, “Everydollar Review: A budgeting app that gets back to basics,” *The College Investor*, 07-Nov-2022. [Online]. Available: https://thecollegeinvestor.com/35086/everydollar-review/. [Accessed: 21-Apr-2023].

[6] S. Mathis, “How do companies protect customer data?: TechTarget,” *Customer Experience*, 12-Jul-2022. [Online]. Available: https://www.techtarget.com/searchcustomerexperience/answer/How-do-companies-protect-customer-data. [Accessed: 21-Apr-2023].

[7] StartupYard, “Michal Kratochvil: Budgetbakers CEO talks profitability and pivots,” *StartupYard*, 21-May-2019. [Online]. Available: https://startupyard.com/michal-kratochvil-budgetbakers-ceo-talks-profitability-and-pivots/. [Accessed: 21-Apr-2023].

[8] T. Fesenmyer, “Should you link your bank accounts to third-party apps?: FCW law,” *FCW Legal*, 22-May-2018. [Online]. Available: https://www.fcwlegal.com/link-bank-to-third-party-apps/. [Accessed: 21-Apr-2023].

8. **[10 POINTS]** Slides
9. **[OPTIONAL 10 POINTS]** Program code implementation
10. **[5 POINTS]** GitHub requirement. Commit deliverable 2, test code, implementation code (if applicable), and slides.