CS3354 Software Engineering
Final Project Deliverable 2

Digital Lifestyle Habit Tracker

Rhea Bhatia
Cameron Meyer
Mark Mondt
Jackson Nestelroad
Andrew Sylvester
Prathyusha Thiruvuri

Delegation of Tasks

Our project was split up evenly between our six members prior to beginning work on both deliverables. Task delegation was decided at in-person and online team meetings. All team members were present at team meetings and were successful in meeting soft deadlines.

Below is a table outlining each task and who completed it.

Task	Team Member
Deliverable 1	
Software process model	Andrew Sylvester
Functional requirements	Everyone (in-person meeting)
Non-functional requirements	Everyone (in-person meeting)
Use case diagram	Mark Mondt
Sequence diagrams	Everyone except Jackson Nestelroad
Class diagram	Jackson Nestelroad
Architectural design	Cameron Meyer
Deliverable 2	
Project timeline	Jackson Nestelroad
Function Point estimations	Rhea Bhatia, Mark Mondt
Other cost estimations	Rhea Bhatia, Cameron Meyer
Test plan	Andrew Sylvester
Comparison to similar products	Prathyusha Thiruvuri
Conclusion	Everyone (online meeting)
Presentation	
Introduction	Rhea Bhatia
Objective	Rhea Bhatia
Comparison to similar products	Prathyusha Thiruvuri
Functional requirements	Cameron Meyer
Non-functional requirements	Andrew Sylvester
Use case diagram	Mark Mondt
Sequence diagram	Rhea Bhatia
Class diagram	Jackson Nestelroad
Architectural design	Prathyusha Thiruvuri
Test plan	Andrew Sylvester
Project timeline	Jackson Nestelroad
Function Point estimations	Mark Mondt
Other cost estimations	Cameron Meyer
Conclusion	Rhea Bhatia

Project Deliverable 1 Content

CS3354 Software Engineering
Final Project Deliverable 1

Digital Lifestyle Habit Tracker

Rhea Bhatia
Cameron Meyer
Mark Mondt
Jackson Nestelroad
Andrew Sylvester
Prathyusha Thiruvuri

CS 3354.001 — Final Project Proposal

Title

Digital Lifestyle Habit Tracker

Members

- Rhea Bhatia
- Sophia Maloney
- Cameron Meyer
- Mark Mondt
- Jackson Nestelroad
- Andrew Sylvester
- Prathyusha Thiruvuri

Description

We are developing a mobile application to track time spent in other applications with an emphasis on accountability. Users will be able to set limits on certain apps and monitor how they are using their device throughout the day with a variety of data visualization tools. Users can invite others to view and monitor their activity as well, allowing friends and family members to hold each other accountable in lowering screen time. Our application offers a new way to decrease screen time, because it's easier to seek goals together rather than alone.

High-level features:

- Track time on apps
- Limit time on apps or app categories
- Data visualization
- Invite others for accountability (mutual or not)

Reasoning

We find ourselves wasting so much time through social media and other distracting apps on our phones. Furthermore, it's even easier to neglect times of productivity for distractions when we are alone or when we know nobody else will know. Our goal is to provide a time-tracking application that will be more effective in decreasing screen time compared to other applications, and we believe increased accountability and transparency will achieve this goal.

Task Delegation

Tasks will be further delegated as more research is conducted and as the parts of our application solidify in the requirements gathering process. All team members will be responsible for contributing to each deliverable, including requirements, system design, and business estimates. Team communication will be conducted using GroupMe and in-person meetings will be scheduled as needed.

✓

A loose delegation of tasks for the deliverables are as follows:

- Deliverable 1
  - Domain Research (Mark, Jackson)
  - Functional requirements (Andrew)
  - Non-functional requirements (Sophia)
  - Use case diagram (Rhea)
  - Sequence diagrams (Cameron)
  - Architectural design (Prathyusha)
- Deliverable 2
  - Project Scheduling (Prathyusha, Mark)
  - Cost, Effort, and Pricing Estimation (Rhea, Cameron)
  - Test plan (Andrew)
  - Competition research (Jackson)
  - Conclusion (Sophia)
  - Prototype/test code (all)
  - References (all)
  - Presentation slides (all)

ok

Addressing Feedback on Project Proposal

There was no specific feedback given to us regarding our initial project proposal. The word "loose" is circled in our delegation of tasks because we only assigned tasks for the deliverables with limited knowledge on where each team member would work best. This flexibility has been removed for this deliverable, and the delegation of tasks is now much more strict.

Delegation of Tasks

- Andrew Sylvester – Software process model
- Everyone – Functional requirements
- Everyone – Non-functional requirements
- Mark Mondt – Use case diagrams
- Everyone – Sequence diagrams
- Jackson Nestelroad – Class diagram
- Cameron Meyer – Architectural design

GitHub URL

https://github.com/prathyushathiruvuri/3354-Digital_Lifestyle_Habit_Tracker

Software Process Model

The system will be developed using an iterative process model. Specifically, we will use the Scrum method to incorporate agile methodologies into the development of the application. This agile process will always be centered on the initial requirements established within this report and the project scope document. However, it is very possible that the wants and needs of potential users will differ from our initial requirements, so an iterative model will give us the flexibility to react to changing user requirements discovered in later iterations.

An iterative process model is ideal for our application because it produces business value earlier in the development lifecycle. Our team can create an application with the essential features and build and improve the application over time. Furthermore, this model allows our team to focus on customer research and testing earlier in the software lifecycle, preventing us from investing in big features, such as exporting data to a variety of formats, when they may provide no value for users at all. Project issues and changes can also be detected earlier. Altogether, an iterative process model provides us with more flexibility in the development of our application that will hopefully avoid major time and money losses in the later stages of implementation.

Functional Requirements

While there are a wide variety of functional requirements for this application, this report is restricted to a range between five and seven. Below are the core functional requirements for the application.

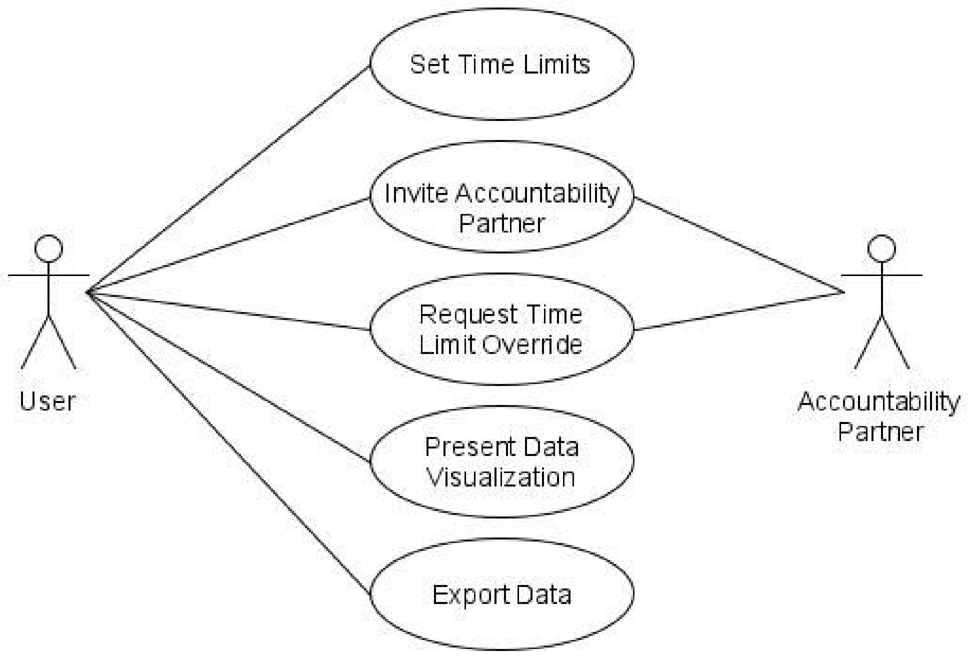
1. The application will track and save the amount of time spent on an allowed application from the moment it is opened to the moment it is either put in the background or it is closed.
2. The user can set daily or weekly time limits for specific applications, categories, or application groups.
3. The user can invite another user by email or phone number to overview and manage certain app usage data, time limits, and restrictions as an accountability partner.
4. The user can request application time limits and restrictions to be overridden by an associated accountability partner.
5. The application can present visualizations for data in the form of a table, bar chart, pie chart, line chart, or word cloud.
6. The user can export daily, weekly, monthly, yearly, or all-time data to a .csv, .xls, .pdf, .json, or .xml file.

Non-functional Requirements

1. **Product Requirements**
    a. **Usability Requirement** — All tutorials or help pages within the application will be no more than five steps. A simple tutorial will be available to the user on initial startup and in the help menu. Training time for the application should be no more than ten minutes.
    b. **Efficiency Requirements**
        i. **Performance Requirement** — The application should have a minimal impact on the performance of the device it is running on. The difference between performing a set of tasks with the application running and performing the same tasks without the application running should be no more than 3%.
        ii. **Space Requirement** — The application download size should not exceed 50 MBs.
    c. **Dependability Requirement** — The application shall succeed in retrieving the proper app usage data for 99.99% of user-initiated requests. The database server shall fulfill 99.99% of requests.
    d. **Security Requirement** — User passwords shall be stored in the database only after it is passed through a secure hashing function. All database requests shall be conducted over HTTPS.
2. **Organizational Requirements**
    a. **Environmental Requirement** — The application shall run on devices running one of the following operating systems: iOS 13 or higher; or Android 10 or higher.

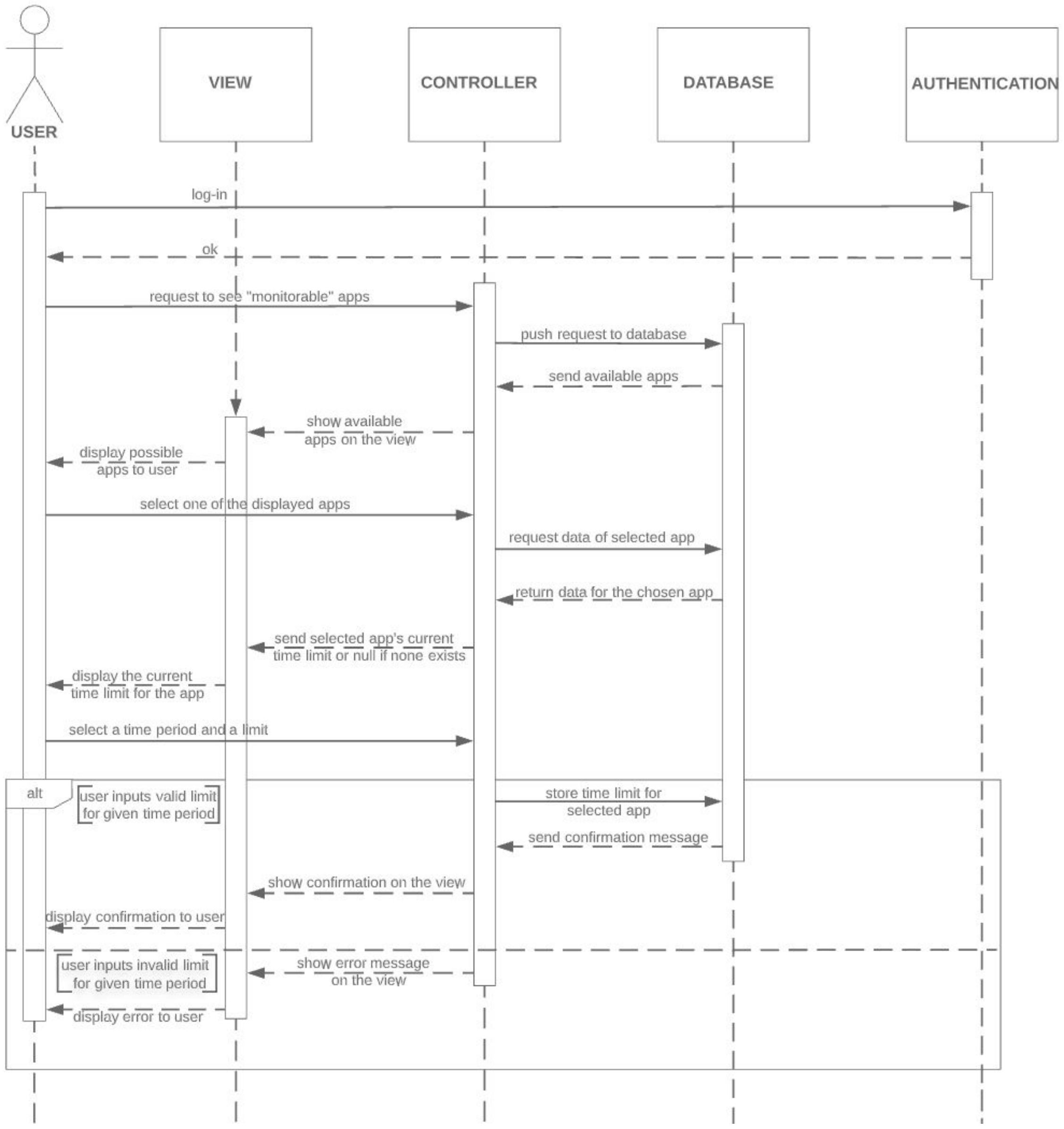
   b. **Operational Requirement** — The application shall be updated with every major release of the supported operating systems described above.
   c. **Development Requirement** — The application shall be developed for cross-platform mobile devices using the Dart programming language and Flutter UI development kit.
3. **External Requirements**
   a. **Regulatory Requirement** — A user may request for all screen time data and account information to be deleted at any time. This requirement is in compliance with the European Union's General Data Protection Regulation.
   b. **Ethical Requirement** — No screen time data or personal information will be given to third parties of any kind. If data is to be sold to third parties in the future, explicit permission from the user is required to share their screen time data with external sources. No personal information of any kind will ever be shared with external third parties.
   c. **Legislative Requirements**
      i. **Accounting Requirement** — The total cost of development and maintenance for our app should not exceed $50,000 per year.
      ii. **Safety/Security Requirement** — The application shall provide an accessible link to the company's privacy policy outlining what information is gathered, how it is shared with other parties, and how users can review and delete data. This requirement is in compliance with the California Online Privacy Protection Act of 2003.

Use Case Diagram

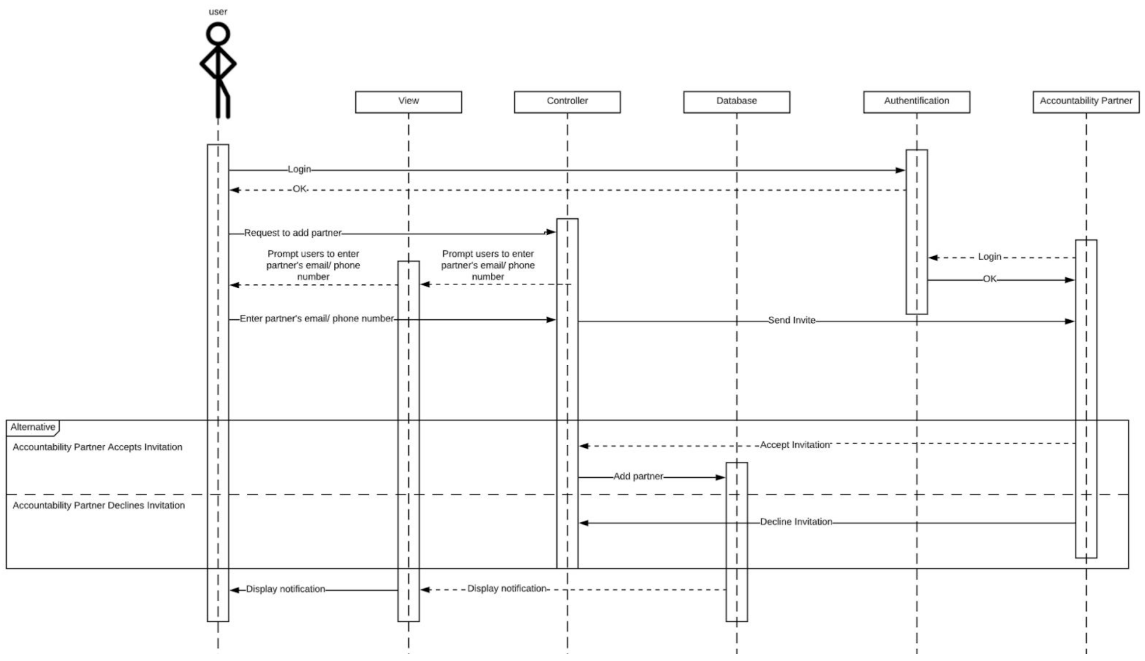

Set Time Limits
Invite Accountability Partner
Request Time Limit Override
Present Data Visualization
Export Data
User
Accountability Partner

Sequence Diagrams

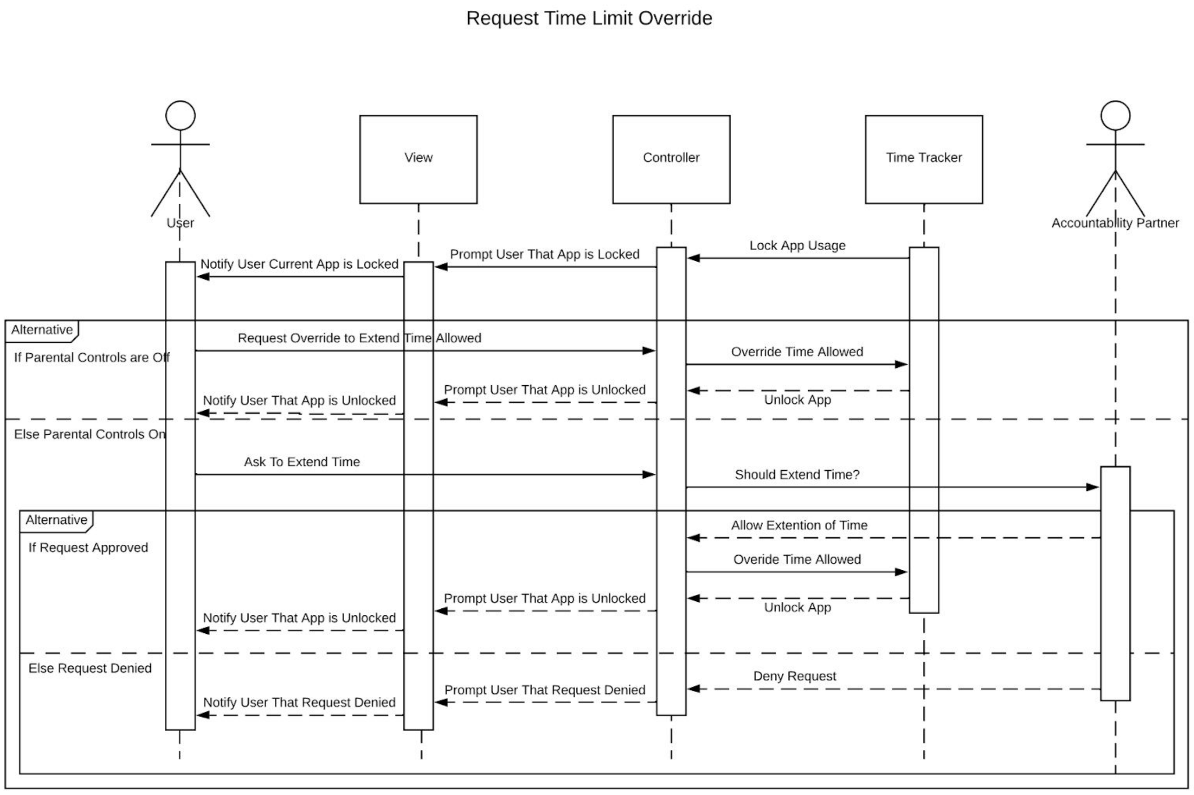
Set Time Limits


USER
VIEW
CONTROLLER
DATABASE
AUTHENTICATION
log-in
ok
request to see "monitorable" apps
push request to database
send available apps
show available apps on the view
display possible apps to user
select one of the displayed apps
request data of selected app
return data for the chosen app
send selected app's current time limit or null if none exists
display the current time limit for the app
select a time period and a limit
alt
user inputs valid limit for given time period
store time limit for selected app
send confirmation message
show confirmation on the view
display confirmation to user
user inputs invalid limit for given time period
show error message on the view
display error to user

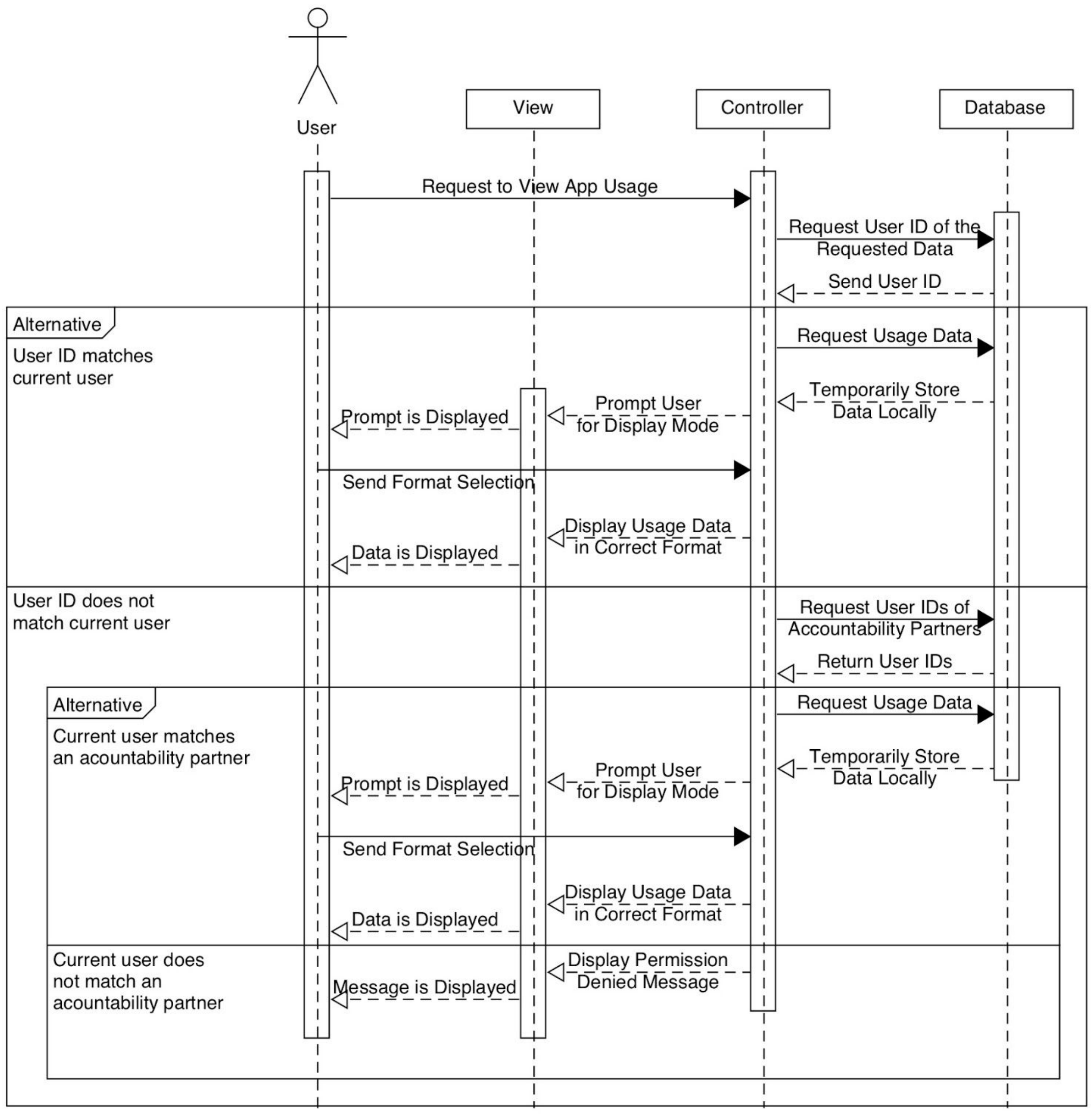
Invite Accountability Partner


user
View
Controller
Database
Authentification
Accountability Partner
Login
OK
Request to add partner
Prompt users to enter partner's email/ phone number
Prompt users to enter partner's email/ phone number
Login
OK
Enter partner's email/ phone number
Send Invite
Alternative
Accountability Partner Accepts Invitation
Accept Invitation
Add partner
Accountability Partner Declines Invitation
Decline Invitation
Display notification
Display notification

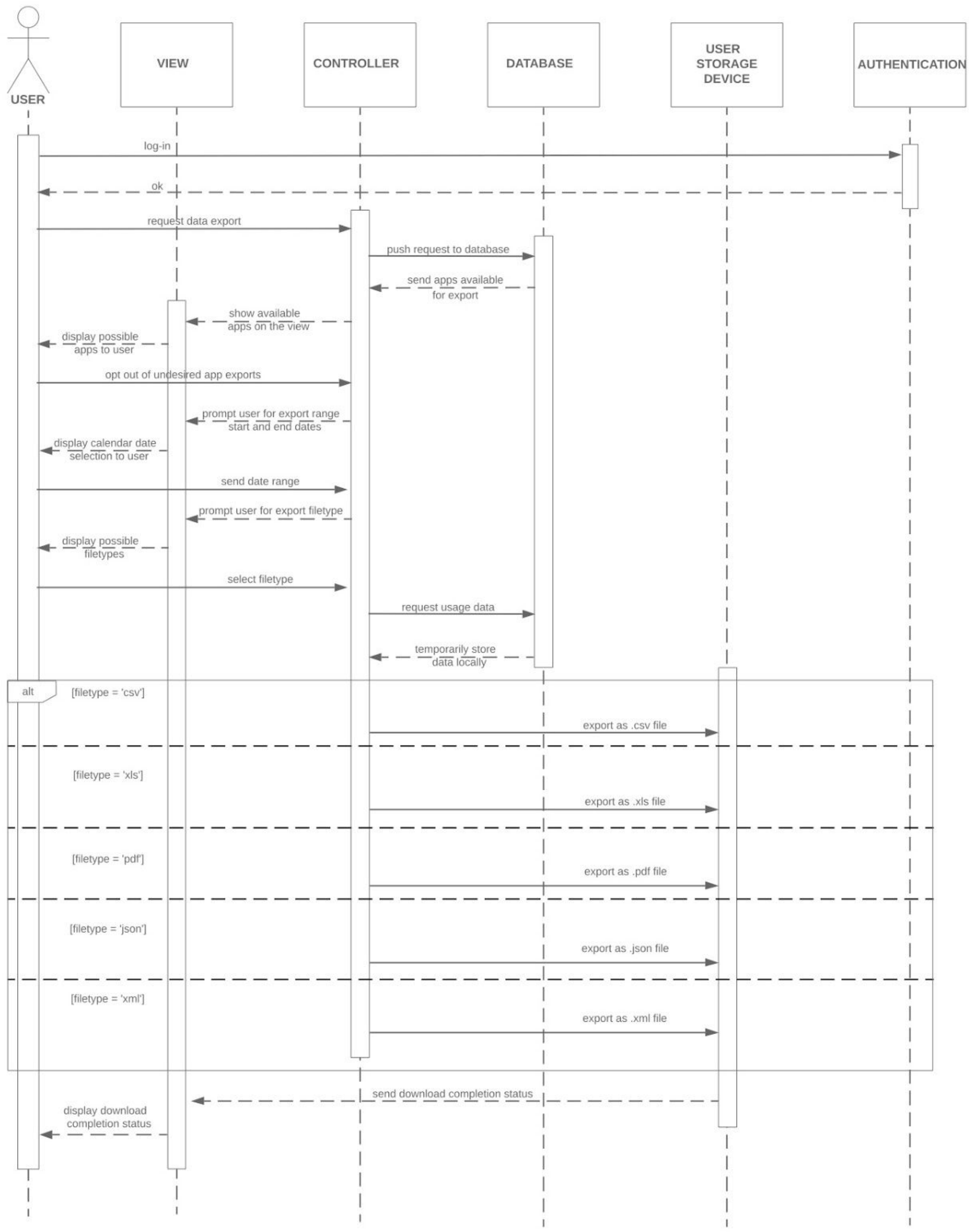
Requesting Time Limit Override Sequence Diagram


Request Time Limit Override
User
View
Controller
Time Tracker
Accountability Partner
Lock App Usage
Prompt User That App is Locked
Notify User Current App is Locked
Alternative
If Parental Controls are Off
Request Override to Extend Time Allowed
Override Time Allowed
Prompt User That App is Unlocked
Unlock App
Notify User That App is Unlocked
Else Parental Controls On
Ask To Extend Time
Should Extend Time?
Alternative
If Request Approved
Allow Extention of Time
Overide Time Allowed
Prompt User That App is Unlocked
Unlock App
Notify User That App is Unlocked
Else Request Denied
Deny Request
Prompt User That Request Denied
Notify User That Request Denied

Present Data Visualization

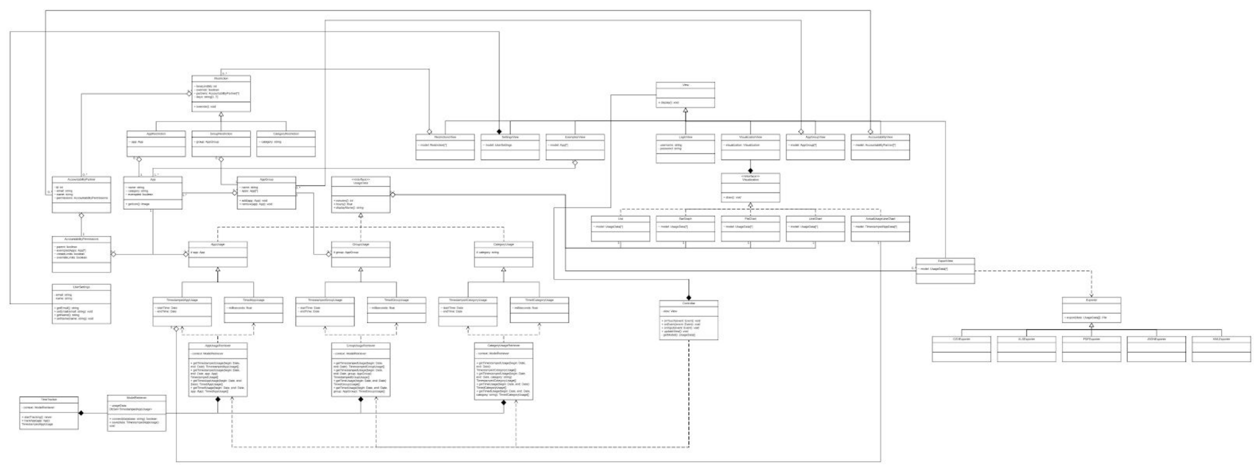

User
View
Controller
Database
Request to View App Usage
Request User ID of the Requested Data
Send User ID
Alternative
User ID matches current user
Request Usage Data
Temporarily Store Data Locally
Prompt User for Display Mode
Prompt is Displayed
Send Format Selection
Display Usage Data in Correct Format
Data is Displayed
User ID does not match current user
Request User IDs of Accountability Partners
Return User IDs
Alternative
Current user matches an acountability partner
Request Usage Data
Temporarily Store Data Locally
Prompt User for Display Mode
Prompt is Displayed
Send Format Selection
Display Usage Data in Correct Format
Data is Displayed
Current user does not match an acountability partner
Display Permission Denied Message
Message is Displayed

Export Data

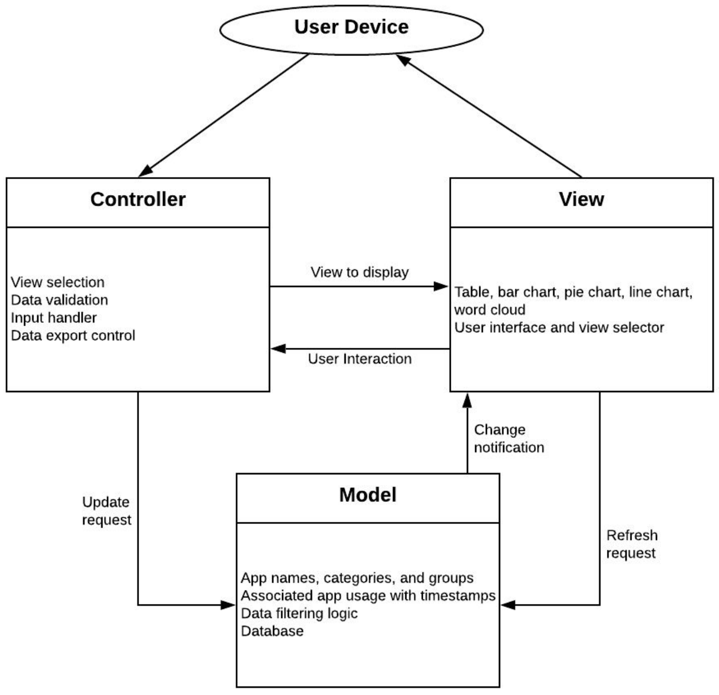

USER
VIEW
CONTROLLER
DATABASE
USER STORAGE DEVICE
AUTHENTICATION
log-in
ok
request data export
push request to database
send apps available for export
show available apps on the view
display possible apps to user
opt out of undesired app exports
prompt user for export range start and end dates
display calendar date selection to user
send date range
prompt user for export filetype
display possible filetypes
select filetype
request usage data
temporarily store data locally
alt
[filetype = 'csv']
export as .csv file
[filetype = 'xls']
export as .xls file
[filetype = 'pdf']
export as .pdf file
[filetype = 'json']
export as .json file
[filetype = 'xml']
export as .xml file
send download completion status
display download completion status

Class Diagram

Full-scale image on project GitHub repository in Diagrams folder.



Architectural Design



The application architecture follows the model-view-controller (MVC) pattern. The application is centered around the data collected from how the client uses their mobile device throughout the day. Our application also focuses on presenting this data to the user in a variety of ways, such as charts, graphs, and spreadsheets. Since all of these presentations, or views, will be acting on the same data, or model, the project fits very nicely with the model-view-controller architecture as the user is viewing and managing the way they use their mobile device.

For our application, the model component will be dynamically generated from the application database. The model and database will work together in the application to represent the user's screen time data.

End of Deliverable 1 Content.

Project Scheduling, Cost, Effort, Pricing Estimation, Project Duration & Staffing

Project Scheduling

Our project will be developed using an iterative software development model. A single set of requirements will be arranged and documented prior to the start of development, and every development iteration will be based on these initial requirements. Iterations will progressively build on top of each other to eventually form the final product. Early iterations will focus on the application's essential features so that user feedback can be collected as soon as possible. Later iterations will add non-essential features defined by the requirements and address user feedback gained from previous iterations. Each iteration reflects a partition of the project's requirements.

A project timeline based on these ideas is given below. We estimate that the project will take six months to develop iteratively, assuming there are no major delays in iteration cycles.

Month 1: Gather Requirements. The first month of the project is dedicated to market research of existing applications in the project domain, requirements brainstorming and gathering with user research, and project cost estimations. A good amount of time is needed for this initial stage because it is the foundation for the rest of the development cycle. Requirements will be clearly documented and available through every future iteration. Requirements are only subject to change if user feedback in later iterations greatly conflicts with the assumptions and requirements produced from this stage.

After this first stage, application development begins. Every iteration of the application is allocated a one month period to review the previous iteration's successes, failures, and user feedback; review the goals for the iteration and what requirements will be met; develop the new iteration's features; test the new iteration's features internally (unit testing); and collect real user feedback for the new features and improvements. We recommend the following timeline for each iteration: one week for iteration review and a detailed design of the new features for the iteration, two weeks for development with simultaneous unit testing, and one week of user testing and feedback collection.

If the deadlines for each iteration become too tight and some goals are missed, the current iteration will not be delayed and will still end after one month. In iteration review, any missing features will be given priority for the next iteration. Project managers and developers should decide if all goals for the current iteration can be met within the one month period due to the additional work pulled over from the previous iteration. If not, the lower priority features should be put on hold for the next iteration. An iteration may need to be added to the end of the development cycle, delaying the project.

Month 2: Back-end and UI Design. The first iteration primarily focuses on the back-end logic and database server for the application. A server should be deployed with endpoints for saving and retrieving data, registering and authenticating users, and connecting to the database server. Internal data representation must also be carefully modeled for use in the database, server, and application model. While the application is being crafted from the back-end, the overall user interface design for the mobile application should be designed in preparation for front-end development.

Month 3: Front-end Iteration 1. The first front-end iteration focuses on the most essential features in the application. At the end of this iteration, the application should be able to track screen time for individual apps; create, enforce, and override time limits for applications, and present data to the user as a sorted list.

Month 4: Front-end Iteration 2. The second front-end iteration will add accountability partners to the application flow. Users should be able to invite and remove accountability partners, allowing partners to view their data and override their time limits. Furthermore, exempting applications from data collection as well as creating time limits and viewing data by application category or custom application group should be implemented.

Month 5: Front-end Iteration 3. The third front-end iteration will focus on data presentation to users and accountability partners in the form of graphs, charts, and timelines. Development should also begin on data export modules for .csv and .json files. If possible, user feedback will also be addressed here.

Month 6: Front-end Iteration 4. The fourth and final front-end iteration will produce data export modules for .xls, .pdf, and .xml files. However, development on these modules may be delayed if work from previous iterations overflows into this one. This iteration should primarily focus on user feedback from previous iterations, perfecting the user experience.

After the sixth month, the application should be ready for publication and purchase. At this point, the application enters a constant maintenance cycle where developers receive user feedback, analyze their merit alongside the initial project requirements, and make fixes or changes to the application accordingly.

<u>Cost, Effort and Pricing Estimation</u>

Explanation of Function Point Table

Number of User Inputs – 7. Account creation requires 4 inputs, 1 each for the fields of username, password, email, and phone number (even though the last two are optional). Setting a time limit also requires an input. Inviting an accountability partner can either use email or phone number, so that would add 2 inputs.

User Input is low complexity because all inputs are either just strings or numbers.

Number of User Output – 19. The main sources of user output would be the data visualization on screen, and the different files that can be exported. There are 5 different forms the data can be visualized in, and 5 different file types for export. The other outputs are prompts for user input or selection. Account creation and login both require prompts for their corresponding input. Data visualization and data export both require a prompt to select the type. A notification must be displayed prompting for an override if a time limit has been reached. Setting a time limit requires a corresponding prompt, as well as inviting an accountability partner. The menu and tutorial will each need their own output functions too.

User Output complexity is high, since the data visualization and file output are very involved and dominate the complexity, even though the other outputs are just simple prompts.

Number of User Queries – 12. Login would involve 1 query. Requesting override requires a query, as well as selecting the option to invite accountability partners. Both data visualization and data export would require 2 queries, since they need to be chosen, and then a specific option must be chosen after that. Searching for accountability partners in the app takes 1 query. Manipulating the menu requires another query, and interacting with the tutorial may involve up to 3 queries.

User Query complexity is low due to the fact that most involve simply selecting an option from a menu, although some involve searching and comparison to a database.

Number of Data Files – 3. There is very little data stored, rather than directly accessed from the phone. One file would contain user information, including the information necessary for login. One contains all the app usage data for the specific device. A third file contains data for different apps, such as what category they belong to.

Data file complexity is low, reflecting that the files are mainly lists.

Number of External Interfaces – 4. Device I/O and device background processes are the two most important. Some sort of texting and emailing interface must be used to invite accountability partners that aren't already users of the app. The usage data of all the apps on the device is the last interface.

External interface complexity is high, since mobile operating systems have some idiosyncrasies, and the application may be developed for multiple environments

	Function Category	Count	Low Complexity	Average Complexity	High Complexity	Count * Complexity
1	Number of User Input	7	**3**	4	6	7 * 3 = 21
2	Number of User Output	19	4	5	**7**	19 * 7 = 133
3	Number of User Queries	12	**3**	4	6	12 * 3 = 36
4	Number of Data Files and Relational Tables	3	**7**	10	15	3 * 7 = 21
5	Number of External Interfaces	4	5	7	**10**	4 * 10 = 40
						GFP = 251

PCA Calculation

Does the system require reliable backup and recovery? ⇒ 4

Are data communications required? ⇒ 5

Are there distributed processing functions? ⇒ 5

Is performance critical? ⇒ 0

Will the system run in an existing, heavily utilized environment? ⇒ 4

Does the system require online data entry? ⇒ 5

Does the online data entry require the input transaction to be built over multiple screens or operations? ⇒ 3

Are the master files updated online? ⇒ 5

Are the inputs, outputs, files, or inquiries complex? ⇒ 3

Is the internal processing complex? ⇒ 1

Is the code designed to be reusable? ⇒ 1

Are conversion and installation included in the design? ⇒ 2

Is the system designed for multiple installations in different organizations? ⇒ 0

Is the application designed to facilitate change and ease of use by the user? ⇒ 3

Sum * .01 = .41

PCA = .65 + .41 = 1.06

FP = GFP * PCA = 266.06

Productivity = 8.75 FP / person-weeks

This assumes 4 hours to complete a "Function Point" and 35-hour work weeks.

Effort = 266.06 / 8.75 = 30.41 person-weeks

Time = 30.41 / 4 employees = 7.6 weeks

The development proper will take 8 weeks.

These 8 weeks predicted by the Function Point estimate will take place in months 2 through 6 of development. For each month, 2 weeks were budgeted strictly for development, so the project allocates 10 weeks in total for code development. Most of the remaining time is used testing, receiving feedback, reviewing feedback, and modifying requirement specifications. The extra 2 weeks serves as a reasonable buffer in case the estimate of 8 weeks was flawed or unexpected complications occur.

Hardware Product Cost

To develop full functionality for the app, it requires the use of a cloud storage solution. After researching multiple options, our team decided to utilize both Google's App Engine and the Google Cloud SQL for SQL Server. We decided to begin using these services with minimum instances, storage capacity, RAM, etc. with the understanding that we will likely have to upgrade our plans in the future after release and there is increased usage of our cloud storage. The combined monthly cost for these Google services is $453.42 [1]. These services are not necessary during the planning phases of development, so we only intend to purchase them in month three of our development timeline, when application construction officially begins. The four-month total for these services is $1813.68.

Additionally, our team decided it would be best to purchase four smartphones for development purposes. Since the application will be released on both the Apple App Store and Google Play Store, our team will receive two iPhone 11's and two Google Pixel 4's. The Google Pixel 4 base model costs $799 [2], totaling $1598 for the pair. A base model iPhone 11 costs $699 [3], coming out to $1398 for both of them. For the entirety of the development cycle, the combined hardware costs amount to $4809.68.

Software Product Cost

For technical development of the application, our team plans to work with the Dart programming language within the Flutter UI toolkit. Flutter is a free and open source product [4], so it wouldn't cost the team anything to use it for development.

Upon completion of the application, we plan to publish it to the Google Play Store and the Apple App Store. Each of these digital storefronts require fees from developers. For the Play Store, there is a $25 required fee upon registration [5]. For the App Store, there is a $99 annual developer fee [6]. Both of these fees make up the entirety of our software costs, so the total cost during our development cycle amounts to $124.

Personnel Cost

Our company intends to hire four skilled software developers to complete this project. In order to compete with other software developer salaries in the industry, we intend to compensate each of our employees with a $75,000 yearly salary. For ease in our calculations, this salary comes in at $6250 per month. So, $4\ \mathit{developers} * \$6250/\mathit{month} * 6\ \mathit{months\ of\ development}$ equals a total of $150,000 in personnel costs over our development cycle.

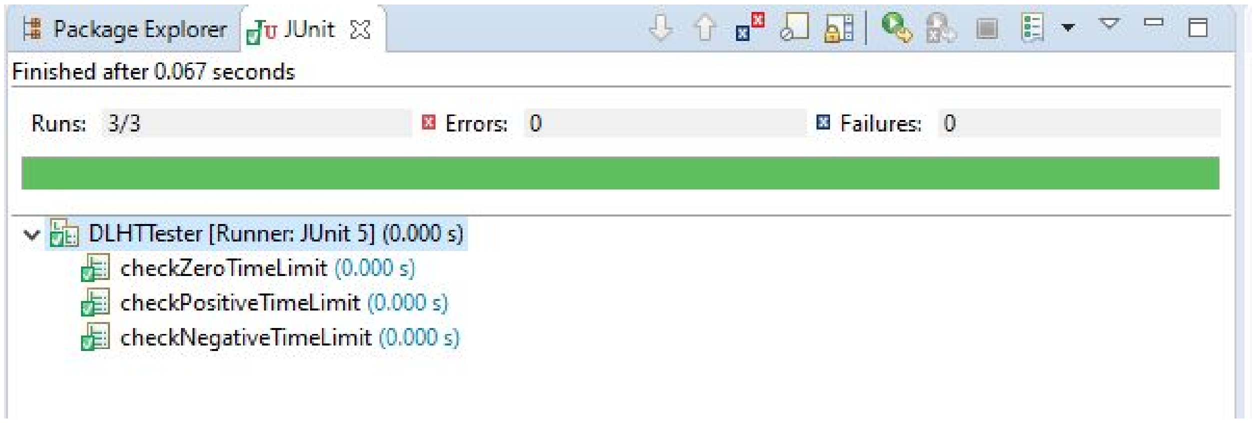
Test Plan

There will be two main types of tests during our application's development. Unit testing of the software code will be conducted alongside development, so developers will be responsible with writing unit tests that validate their code. User testing of the requirements and any new features will be conducted at the end of each iteration as described in the project schedule above. This user testing phase will use real, interested people in our application to collect real user feedback that can guide development for future iterations.

One of the units of our software is the setting of a time limit for an app. We can test that the app is setting valid time limits with the use of equivalence partitions. Valid input is any integer positive or negative. Positive integers are time limits set for the app, zero automatically locks the app, and a negative integer means there is no time limit. Broken into partitions:

- **Partition 1 (Time Limit > 0):** A time limit is set. In code this will return the string "Time limit set".
- **Partition 2 (Time Limit = 0):** No time limit is set and the app is locked. In code this will return the string "App locked".
- **Partition 3 (Time Limit < 0):** No time limit is set and the app stays unlocked. In code this will return the string "App unlocked".

This unit and tests were programmed in Java (Can be found in the zip folder). By using JUnit the three partitions were tested with the following results:



```
Finished after 0.067 seconds
Runs: 3/3        Errors: 0        Failures: 0

DLHTTester [Runner: JUnit 5] (0.000 s)
    checkZeroTimeLimit (0.000 s)
    checkPositiveTimeLimit (0.000 s)
    checkNegativeTimeLimit (0.000 s)
```

<u>Comparison to Similar Design</u>

It is always important to scout similar designs in your project's field to find what is missing and create a product that can capitalize on what the market is lacking. Social Fever is an Android application which enables you to limit the usage of any type of app you have on your phone [7]. It allows you to set daily time limits and see your overall time spent on individual apps that you have set restrictions towards [7]. However, where our app exceeds the capabilities of Social Fever is the added ability to have a user be locked out of the problematic app. There is a way to override your original time limit, but by having the problematic app be unable to access helps to curb smartphone addiction.

Another available app to limit your cell phone usage is called Space and is available for Android and Apple products [8]. In addition to the features introduced by Social Fever, Space also has the capability to kick you off your phone when you exceed your limit and it also tracks how many times you have unlocked your phone [8]. You can also allow your friends to see your usage and use this feature as a type of competition to see who is on their phone the longest [8]. Although friends have access to your usage, they cannot be invited to help you track it like our app provides. The biggest downfall of this app is that it prevents you from setting specific limits for problematic apps and just tracks the usage of your entire phone as a whole. Our app will allow you to set times for particular apps and in turn lets you fix addiction to certain applications.

Next we have AppDetox which is available as an Android app. AppDetox's main draw is the fact that it is used to set parental controls on kids' devices [9]. This is one of the ways that the Digital Lifestyle Habit Tracker aims to use the accountability partner feature. In the case where parents want to set definite restrictions on their children's usage [9]. There is even an additional feature of AppDetox where you can view all the times that you went overtime on your limits [9]. You can even block certain apps, but it has the same downside as Social Fever where you don't get kicked off the app, meaning that it lacks in making the user accountable.

Lastly, it is important to see what a smartphone's built in features are to see if there are similarities there as well. Apple products have a built-in feature called Screen Time which tracks your daily usage of your phone as well as individual apps. You have the opportunity to set limits for individual apps and you can choose if you want to be kicked off the app when you reach your limit. There is a way that you can just override being shut out, however. You can also set times that you want to be off your phone in general. Android phones have a built-in feature as well called Digital Wellbeing & Parental Controls. In Android's version you have ways to track most apps. Basic apps like the settings, Play Store, and the actual phone cannot be limited. You are also able to simply ignore your preset limits like in Screen Time. Additionally, there is no feature to get kicked off your entire phone which differs from Apple's Screen Time.

In researching, it is obvious that there is currently no single app that is able to achieve all the requirements that the Digital Lifestyle Habit Tracker seeks to incorporate into its features. By designing a smartphone application which has multiple features like setting parental controls, being unable to override preset time limits, and inviting accountability partners will allow people to be more thoughtful of how they are spending their time on their devices.

Conclusion

Our project's scope, requirements, and goals have stayed very consistent over its lifespan. One small change we made from Deliverable I to Deliverable II is the change of our costs to develop our application. Originally, we thought that the cost would be \$50,000 but now that we have done more research we see that the monetary cost comes closer to \$155,000. We are now accounting for the cost to pay developers.

Our team believes the application detailed in this deliverable serves a realistic purpose that improves on all of the other options in the problem domain. We are living in a world where technology is surrounding us, and it is easier than ever to get addicted to scrolling mindlessly through a screen. Screen-time tracking applications can help us stay away from our phone, but it is too easy to push away time limits with existing applications. Our application provides a layer of accountability that motivates users to stick to their goals to stay more connected to reality than to technology.

References

[1] “Google Cloud Platform Pricing Calculator,” Google Cloud Platform Pricing Calculator. [Online]. Available: https://cloud.google.com/products/calculator#id=d03e9055-3cde-4b6d-bfa5-4c9a4be64519. [Accessed: 14-Apr-2020].

[2] “Google Store - Pixel, Chromecast and more,” Google. [Online]. Available: https://store.google.com/us/config/pixel_4. [Accessed: 14-Apr-2020].

[3] “iPhone 11 64GB White,” Apple. [Online]. Available: https://www.apple.com/shop/buy-iphone/iphone-11/6.1-inch-display-64gb-white-unlocked. [Accessed: 14-Apr-2020].

[4] “Flutter,” Flutter. [Online]. Available: https://flutter.dev/. [Accessed: 14-Apr-2020].

[5] “Google Play Console,” Google. [Online]. Available: https://play.google.com/apps/publish/signup/. [Accessed: 14-Apr-2020].

[6] Apple Inc, “Purchase and Activation,” Support - Apple Developer. [Online]. Available: https://developer.apple.com/support/purchase-activation/. [Accessed: 14-Apr-2020].

[7] *Social Fever - Stop Smartphone Addiction.* (2019). SYSTWEAK SOFTWARE PRIVATE LIMITED. Accessed: Apr. 10, 2020. [Online]. Available: https://play.google.com/store/apps/details?id=com.systweak.social_fever&hl=en_US

[8] *SPACE - Break phone addiction.* (version 14.1.2). Mrigaen Kapadia. Accessed: Apr. 10, 2020. [Online]. Available: https://apps.apple.com/gb/app/space-break-phone-addiction/id916126783

[9] *AppDetox - App Blocker for Digital Detox.* (2019). AppDocs. Accessed: Apr. 10, 2020. [Online]. Available: https://play.google.com/store/apps/details?id=de.dfki.appdetox&hl=en