Friendzy

Akhil Kanagala, Manav Modi, Sashank Kannan, Charles Averill, Adrian Requena, Haley Wheatley, Andres Mancillas

1. Delegation of Tasks:

Andres: Will create UI/UX.

Adrian: Will research costs and feasibility requirements.

Charles: Will develop the backend for the app.

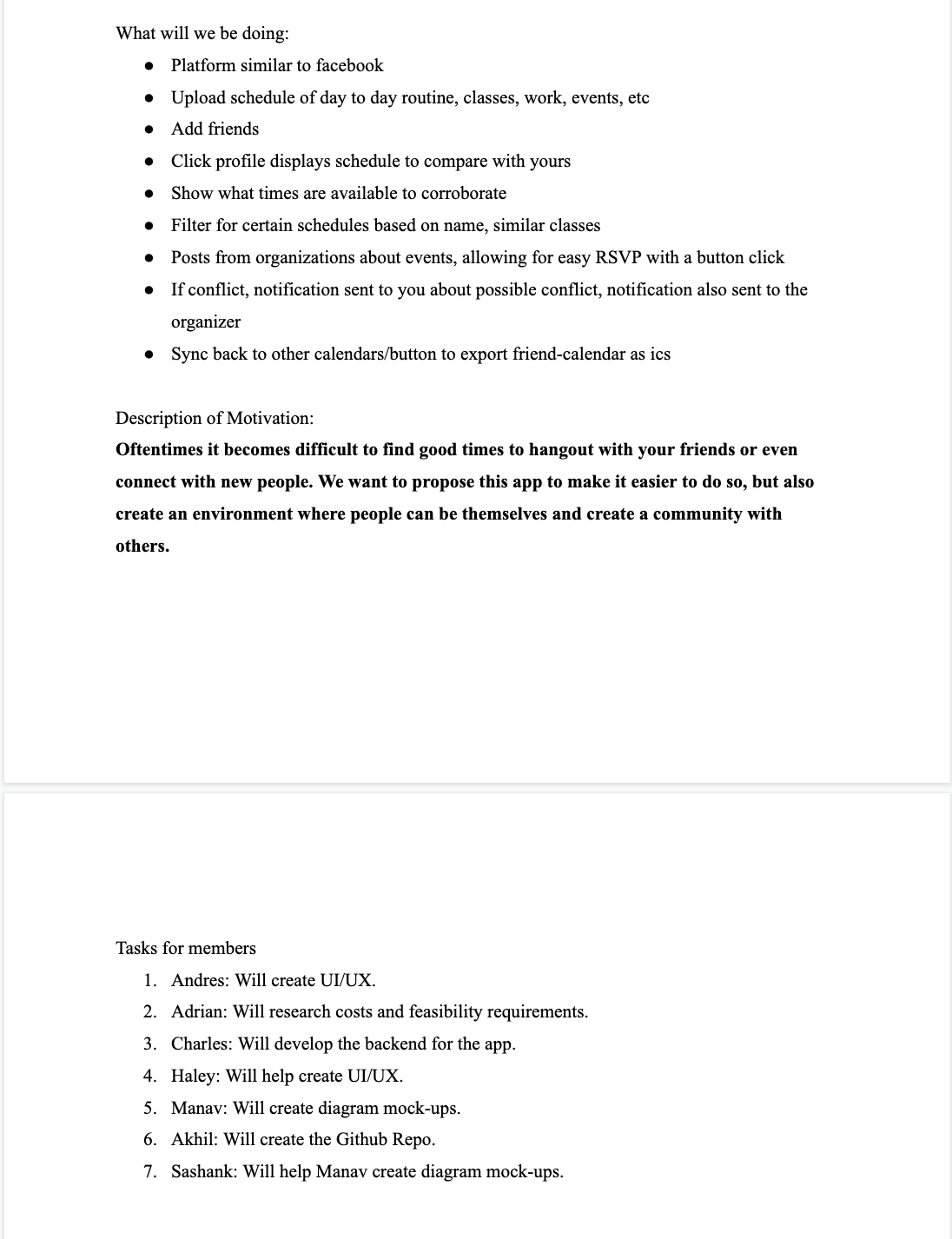
Haley: Will help create UI/UX.

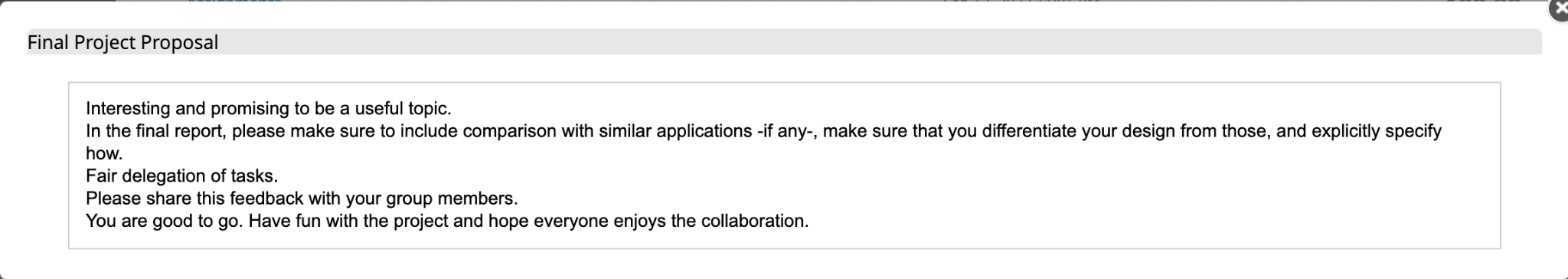
Manav: Will create diagram mock-ups.

Akhil: Will create the Github Repo.

Sashank: Will help Manav create diagram mock-ups.

1. DELIVERABLE 1 CONTENT





On GitHub: https://github.com/amancillas18/3354-Friendzy

Delegation of Tasks:

Andres: Will create UI/UX.

Adrian: Will research costs and feasibility requirements.

Charles: Will develop the backend for the app.

Haley: Will help create UI/UX.

Manav: Will create diagram mock-ups.

Akhil: Will create the Github Repo.

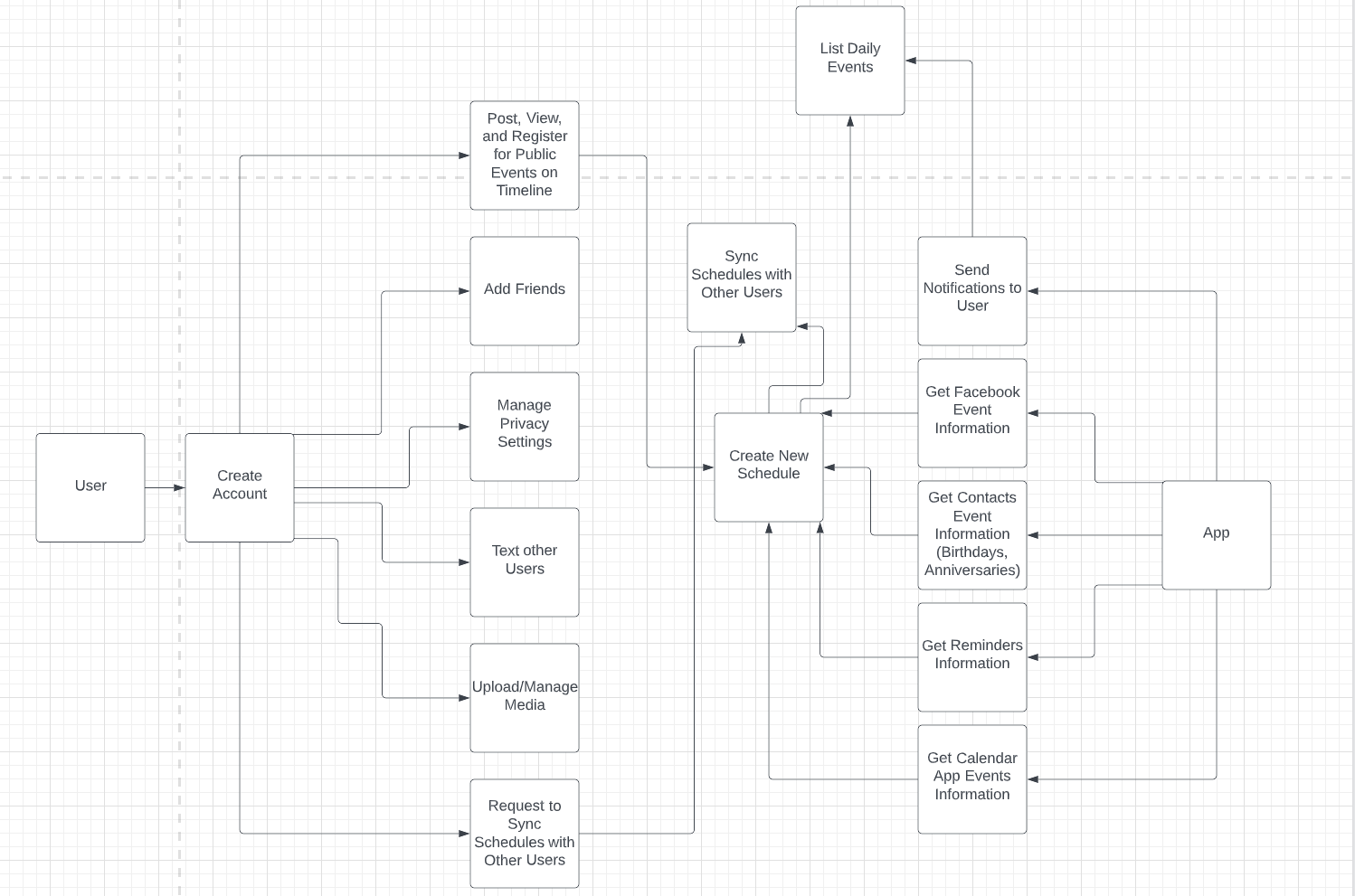
Sashank: Will help Manav create diagram mock-ups.

We would like to employ the Agile software process, using the evolutionary model, as we believe using Agile would help us complete the project the fastest and using the evolutionary model would help us update in the long run compared to the waterfall model.

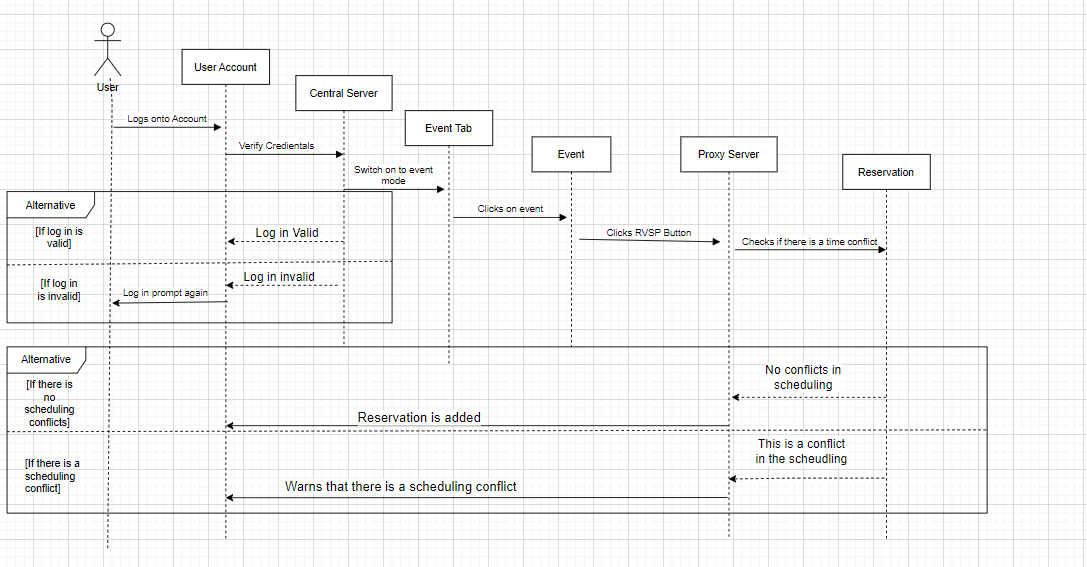
1. Create a schedule, view other schedules, find other people, friend other people, remove/block friend, chat with others

* Usability: Clear UI/UX
* Performance: Live service
* Space: With constant updates, clients need to be added to keep up.
* Accounting: Premium model, ad revenue
* Safety/Security: End-to-end encryption
* Environmental: Facilitates connections with other users, creating synergy between them
* Operational: Modifying the program overtime to maintain updates and more user interaction
* Development: Developers constantly working on the program to keep it functional
* Dependability: Maintain the a minimum of resources available so the server can run at all times
* Regulatory: Will follow the appropriate laws it would have follow
* Ethical: Not sell any schedule information to any companies that are interested in it

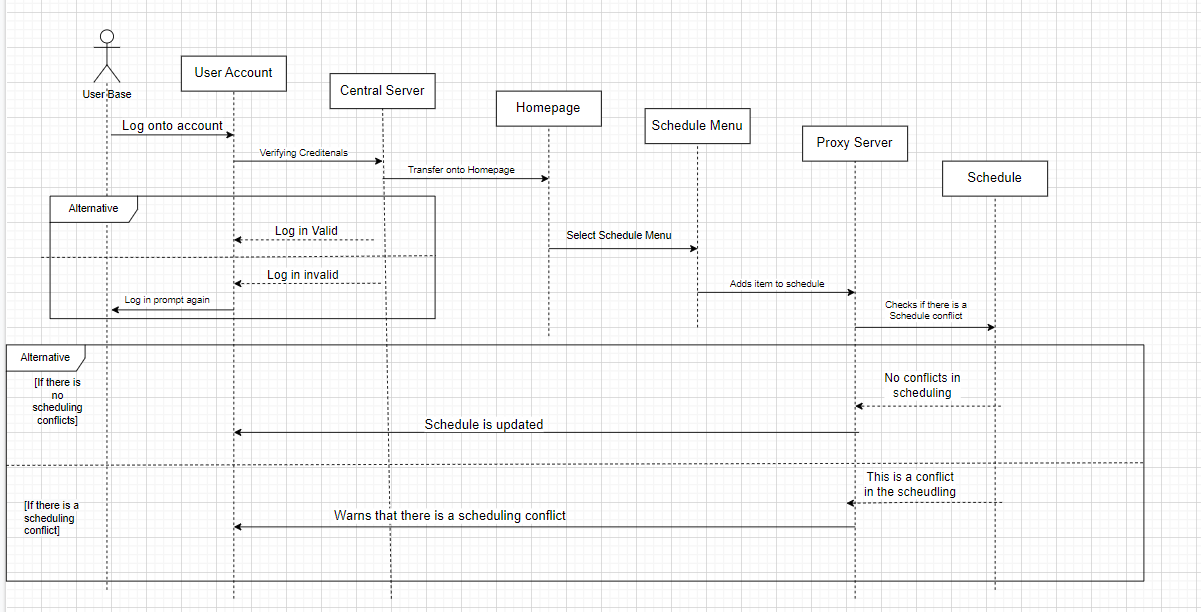
6. Use Case Diagram



7. Sequence Diagrams  
Sequence Diagram - Event



8. Sequence Diagram - Schedule

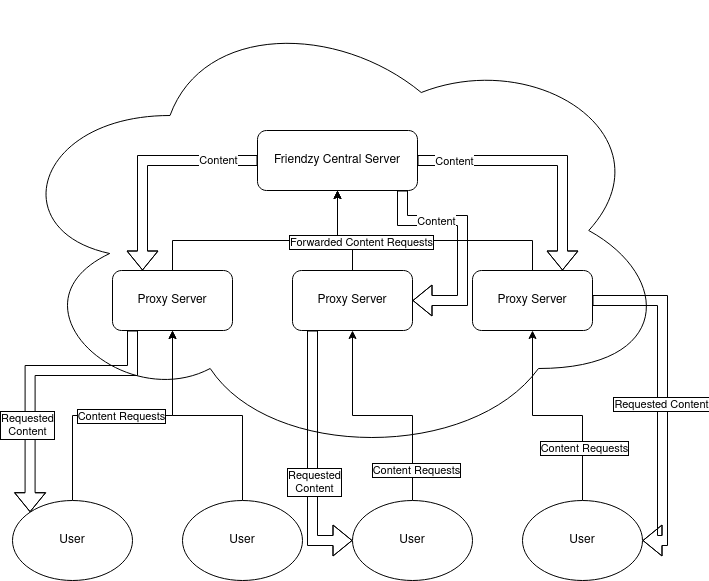


# 8. Class diagram

# 

# 

# 9. Architecture Diagram (Client-Server)



END OF DELIVERABLE 1 CONTENT

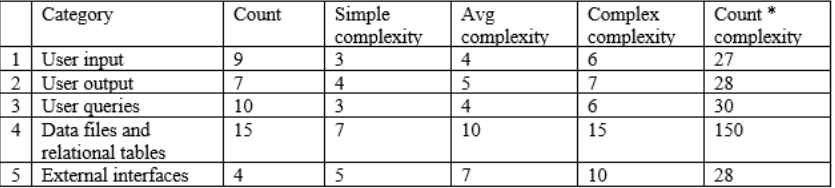
3.1)

When it comes to the development of social media apps, several factors can come into play when it comes to the scheduled delivery time of the product, depending on factors such as the technologies used, team size, total budget, and feature implementations.

Weekends would not be counted in the schedule, with eight hours being the standard number of working hours per day.

In order to effectively estimate the size and cost of the project, we used Function Point Analysis, a technique that is relatively reliable and proven to deliver relatively accurate estimates when applied properly. While there was indeed some consideration of the Application Composition technique, it was decided that the Function Point Analysis technique was better suited to our needs.

3.2)



The table above gives an accurate representation of our calculations when it comes to this project. The FPA technique uses 5 main function categories, those being user input, user output, user queries, data files and relational tables, and external interfaces.

The Complexity numbers were taken from the sources provided in class and were left unchanged. After calculating and adding the numbers in the count complexity column, we received a final number of 263 function points, We then calculated the PCA number, which came out to a total of 1.05. By multiplying the total number of function points by the PCA number we came up with a total of 276.15 function points, and after dividing that by the productivity value of 60, we ended with a final person-week total of 4.6025, which we rounded up to 5 total person-weeks. The duration of this project would be equal to the calculated effort value divided by the team size. 5 Person-Weeks divided by 7 people would return a value of 0.71, which we then rounded up to one total week.

3.3)

Developing Friendzy would require higher-powered computers with specialized hardware and multiple peripherals, such as monitors, docking stations, keyboards, and mice. To supply each developer, we would require around $4,750 dollars per machine at the current market rate [4], for a total of $33,250. In addition, we may also require additional spare components in case of hardware failure or breakdown, and therefore bought an additional two systems, for another $9,500.

3.4)

Our product would employ AWS Cloud computing in place of traditional IT, allowing for on-demand self-service, variable expenses in place of capital expenses, massive elasticity, and increased agility and speed. Using the AWS Price Calculator, The estimated monthly costs for the services that we selected was $3,123.41 per month, with a total annual cost of $37,480.92. Some examples of these services include Amazon Cloudfront, Amazon API Gateway, Amazon VPC, and many more.

In addition to enterprise cloud computing software for the actual development, we would also need special Identity and Access Management Software in the form of RSA SecurID. A hardware token is distributed to users by RSA, which generates a random 6-digit number at 60-second intervals. This code is then linked to a computer. The computer requires a user attempting to log into a network resource to enter both their credentials as well as the number currently displayed on their token. Using their ID Plus E3 plan, which according to their site, is “Purpose-built for security-first, highly complex IAM environments that require future-proofed security,” we would pay $6 a month for each user, resulting in a total cost of $42 a month and $504 a year.

3.5)

Based on the function point alternative cost modeling techniques, it is estimated that it would take 7 person weeks and with 7 personnel it would take 1 week to code the product. Based on the median pay of a software developer which is $109,020 per year [5], each worker would be paid 2090.79 for the week of coding with a total of 14635.56.

4. **Turned in on eLearning and see GitHub repo**

5.

Some designs that are similar to Friendzy are Calendar, Facebook, UpTo, and Howbout. On Calendar, you can share your calendar in an email or an embedded link with other people. Then when users find a date and time that works for them, the event is added to everybody’s calendar [1]. Calendar is also able to integrate with other calendars such as Google Calendar, Outlook, and iCloud [1]. On Facebook, you can add people as your friends and RSVP to events. For example, to RSVP to an event the user can select from one of three options: Going, Maybe, or Can’t Go. The three options will apply if the event is a private event. For any public events, the user can select from one of two options: Interested and Going [2]. If the user has a conflict, then they can change their RSVP status. UpTo allows tracking individual activities in real-time for friends and colleagues, and “the app can also sync with your existing calendars from Google Calendar, iCloud, Outlook, and Facebook Events, as well as calendars for sports teams, concerts, and TV shows” [1]. On Howbout, you can plan events with your friends, find out your friends’ availability, send event invites to your friends, and share calendars with friends. To find out your friends’ availability you can use the polls feature. The polls feature can also be used to vote on any decisions involving the event [3].

6.

Friendzy is a scheduling app that we are designing that will make it easier to collaborate with other people to create a group schedule and be able to see when others are free in order to plan events. There are some scheduling apps such as Google Calendar, but they do not provide a clean way to view the group calendar without looking very clustered and confusing to the user. This makes it harder to be able to plan events with your friends or be able to see the schedule of other people who are in the shared calendar. We did not have to make any changes to our design or the ideas that we had originally thought of. We have created the diagrams to help view the process from a planning point-of-view and be able to see how the components are able to connect. We had decided, as a group, to utilize the client-server architecture diagram as we would require one server for the server with the calendar and the clients would be the people that are able to use the server and edit the information. We have decided to use AWS to be able to access the people and their information in the database to be able to be in the calendar and make changes and save the events under their name in the database for easy access to the app and their information.

7.

[1] H. Jones, “What is the best app for sharing a calendar?,” *Calendar*, 17-Feb-2022. [Online]. Available: https://www.calendar.com/blog/what-is-the-best-app-for-sharing-a-calendar/. [Accessed: 08-Nov-2022].

[2] Facebook, “How do I respond to an event?: Facebook help center,” *How do I respond to an event? | Facebook Help Center*. [Online]. Available: https://www.facebook.com/help/201140859928818. [Accessed: 08-Nov-2022].

[3] Howbout Ltd., “Howbout: Social event planner,” *App Store*, 30-Nov-2019. [Online]. Available: https://apps.apple.com/gb/app/howbout-social-event-planner/id1477248221. [Accessed: 09-Nov-2022].

[4]PCPartPicker, “PCPartPicker Builds for Friendzy Developers.” [Online]. Available: https://pcpartpicker.com/list/KkbZDq. [Accessed: 18-Nov-2022].

[5] U.S. Bureau of Labor Statistics,“Software developers, Quality Assurance Analysts, and testers: Occupational outlook handbook,” *U.S. Bureau of Labor Statistics*, 09-Sep-2022. [Online]. Available: https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm. [Accessed: 18-Nov-2022].

8.

Link to presentation:

https://docs.google.com/presentation/d/1iRvb4h-UIHQYKI6m-G9dOTGeXJUZ0Kke0ADg7rwmM0I/edit?usp=sharing

9. **Skipped**

10. **See Github repo**