Fetch: The Tinder for Doggy Playdates
Software Design Specification

Ronny Fuentes, Kyra Novitzky, Jack Sanders, Stephanie Schofield, Callista West
Fetch Team
https://github.com/JackSanders1998/CIS422Proj2
J. Flores (jjf) - 02-15-2021 - v0.91

Table of Contents

1. SDS Revision History

Date	Author	Description
2-15-2021	Steph S.	Added title of app to top of document.
2-15-2021	Kyra N.	Added remaining group information to title page
2-15-2021	Steph S.	Added modules of software architecture
2-16-2021	Kyra N.	Added a system overview, suggested/added new ideas for software architecture, and added layout for software modules
2-17-2021	Kyra N.	Added Primary Role and Functionality for User Account Information, Login Screen, and Deck. Created Static Model for Deck.
2-17-2021	Kyra N. Ronny F.	Adjusted module break down
2-17-2021	Ronny F.	Added information to Log-In/Sign-Up and User Account Information
2-17-2021	Kyra N.	Added information to Deck, Homepage, and Profile
2-18-2021	Ronny F.	Finished all up to date information in Log-In/Sign-Up and User Account Information
2-18-2021	Ronny F.	Finished Matches module and is all up to date.
2-18-2021	Kyra N.	Adjusted Software Architecture section and finished Homepage and Deck modules. Removed previously existing Profile module and renamed User Account Information module to new Profile module
3-11-2021	Callista W.	Made additions and edits to match final version of our program
3-11-2021	Kyra N.	Revised all Static/Dynamic diagrams to match final project.

2. System Overview

For dog owners everywhere, it is sometimes difficult to find other canine companions for their dog to consistently play and bond with, especially when on the go. Anyone that owns a dog of their own knows that the dog community is a special one, but it is sometimes hard to become a part of. The intention behind Fetch is to act as the Tinder for doggy playdates. Owners of dogs download Fetch, make a profile, and from there, go through various doggy profiles to find the

perfect match for their own beloved pup to play with, whenever and wherever. Fetch will not only allow dogs to form friendships, but it will expand and unite the canine community as well.

Fetch consists of four main components: User account information, a login screen, a deck of other profiles, and the match page, to see all of the matches you have made. User account information and the login screen are meant to set up profiles, whether that be for existing or new users. User account information is meant for new users who need to add their dog's information such as a profile picture, personal details, etc. While the login screen is for users who would like to access their previously created account, including previously stored information, data, and activity. The Deck and matching features are components of the system that allow for the logistics behind matching to occur, or in some cases to not occur. Matching with a user will provide the current user with their match's email to allow for communication outside of the application.

3. Software Architecture

3.1 *Components and their functionality*

- Client side (What the user sees)
    - Log-In/Sign-Up
        - Allows users to log-in to an existing account or create a new one. This will typically be the initial page a user sees when opening the application.
        - After signing up, users can immediately create a profile by signing into the application using the log-in page.
    - Profile
        - Where a user updates and stores all of their information. It must be filled out upon initial sign-up, however, if a user decides they would like to change anything they will always have the ability to do so.
        - Users have the ability to upload a profile picture from their camera roll or take a picture within the app. (Not in simulator, only phone)
    - Homepage
        - Deck of dogs
            - This is the card-like deck that consists of all dog profiles. Users will start viewing the deck by pressing the center button to populate the deck. From here, users will be able to see other profiles to make the decision to match or decline.
        - Match/Decline buttons
            - These buttons send profiles to the match or decline entities within the FireBase database. Matches will appear within

the match tab (bottom right corner of the application), where users can then contact which matches they wish to connect with.

- Matches
  - All currently existing matches will exist under this page.
  - The is no unmatch functionally at this point in the development of Fetch.
- Server side (What the programmer sees)
  - Database
    - Existing User Profiles
      - Public
        - Name
        - Dog's info
        - Profile picture
      - Private
        - User ID
        - Password
        - Email
    - Matches
      - User IDs
        - Array of matched dogs corresponding to each user
    - Declines
      - User IDs
        - Array of declined dogs corresponding to each user

3.2 *Module Interaction*

Within Fetch there is no single module that holds functionality of higher importance in comparison to the others, each component is equally crucial to the development of the application. The four main modules consist of AuthenticationView (Login/Signup), ProfileView (Profile), HomeView (Homepage), and MatchingView (Matches).

With AuthenticationView, otherwise known as the sign-up and login module, new accounts may be created and existing accounts may be re-accessed. This component interacts directly with the authentication component of the Firebase database. When signing up, AuthenticationView will connect with Firebase in order to privately store a user's email, encrypted password, and their unique user ID. When a user would like to log back in, AuthenticationView allows for the user to do so by interacting with the previously stored authentication information. Whether logging in or signing up, the AuthenticationView module automatically reroutes a user to the Profile module.

The Profile module continues to carry out the remaining steps needed before being able to access and interact with other modules. Again, Profile directly communicates with

Firebase in order to store information. In this case it accesses the user's unique ID from the Login/Sign-up module and then proceeds to store that unique id as a key in the realtime database, along with the submitted profile information as values. Once this is complete, a user is free to interact with the remaining modules, or go back to previous ones, in any way they please.

The Homepage module's interaction is primarily with the user and the database. User's direct the Homepage module on where to store another user's information within the database, either to the Matches' Entity or the Declines'. However, the Homepage module's interaction with Firebase is two-sided. It also grabs information from the Users Entity of the database in order to display another profile's information within the Deck.

Lastly, the Matching module is much like the previous modules in that it interacts directly with the database, grabbing the current user's matches from the Matches Entity.

3.3 Rationale for Architecture

Fetch's current, fundamental goal is to allow users to maintain an account that will let them submit and edit personal information, make decisions on other dog profiles, and view the matches they have made. In order for this to happen Fetch needed to have the 4 basic modules that exist: Login/Sign-up, Profile, Home, and Matches.

Login/Sign-up could have been separated into two different modules completely. However, Fetch uses the same authentication functionality provided by Firebase in order to make this happen. Combining them into one module made the management of authentication more simplistic, limited room for errors, and allowed for users to switch from one to the other in an easier fashion.

The rest of the modules remained separate from each other for better organization within the program code and because their functionality is independent of one another.

It is important to note that within each module, the front-end and back-end components are typically separated from one another.

4. Software Modules

4.1. Log In/Sign-Up

4.1.1. Primary Role and Function

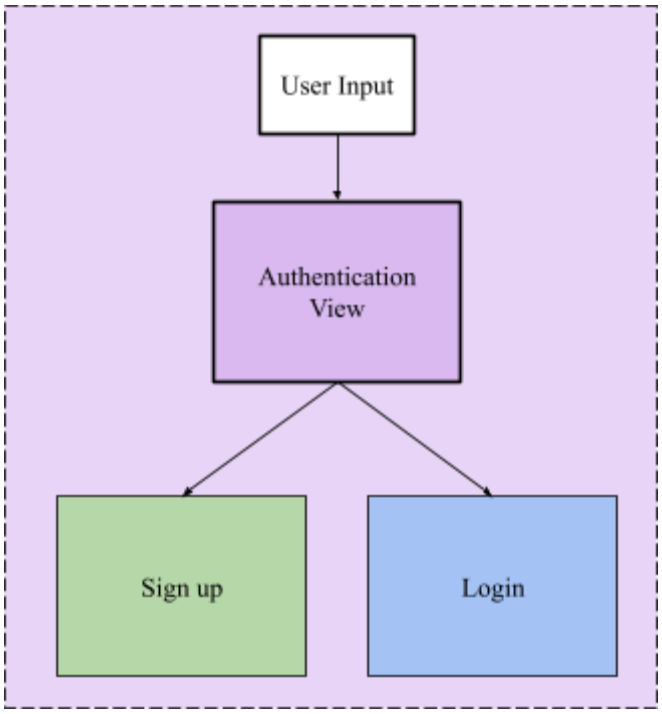
The login or sign-up page is where users will have the option to either create a new account with Fetch or return to their existing account. If the user is new, they'll be asked to supply a valid email address and password to gain access to the application. Otherwise, if the user has an account, they'll return to it, where they'll have access to all their history within the application.

Incorrect email and password will give the user a notification that their credentials do not match an existing user.

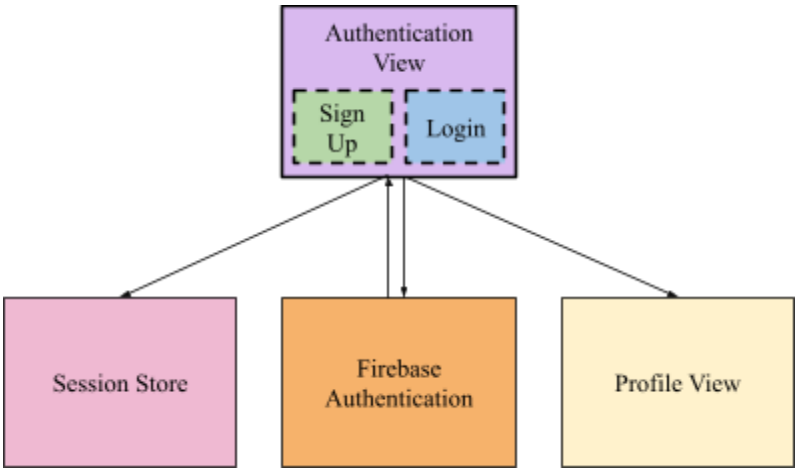
4.1.2. Interface

Regardless if the user is new to Fetch or not, they will be directed to the login screen, where they will be able to take one of two routes. They will have the option to create an account or log in to an existing account.
If the user happens to be new, they will be directed to the account information page (upon providing a valid email and password), where they will be asked to fill out information about their dog in order to utilize the application to its fullest. This information will include the owner's name, their dog's name, their dog's breed, dog's age, and dog's personality. They will also have the opportunity to add a profile picture from their camera roll, or take a picture on the spot.
If the user has an existing account with Fetch, upon login, they will be directed to the homescreen.

4.1.3. Static Model



4.1.4. Dynamic Model



4.1.5. Design Rationale

As a way of making the application simple, conceptually (along with a separation of concerns), the consensus was to create two routes (navigation links) depending if the user was new or not. The new user route is solely responsible for directing the user to a profile page where they will be able to enter their corresponding information, submitting that information, and be directed to the home page. The existing user can navigate to the home page upon login. This format is user-friendly, with navigation tabs on the bottom of the application to allow users to easily navigate to the profile, home page, and matches page.

4.1.6. Alternative Designs

An alternative approach/design would be to potentially get rid of one of the navigation bars. Specifically, directing a new user to the profile page. If this were the case, this means that creating an account would take place on the login screen, which could potentially make the process easier, negating the use for a new page. We would populate the form fields in order for the user to enter their dogs' information or simply direct them to the home page. The primary goal of this is to keep the design and functionality simple and elegant for the user.

4.2. Profile

4.2.1. Primary Role and Function

The profile is initially visited if a new user has just created an account. Before they can begin to find compatible matches, they must first enter all of their dog's information. The information entered will be appended into the FireBase entity "Users". Their information will be stored into separate arrays to separate specific information such as dog breed. Users will always be able to access, and/or edit their information (profile) at any point in time, provided they've made an account. By pressing the update button on the profile page, the information will be updated in the FireBase.

4.2.2. Interface

Once the user has successfully supplied an email address and password, they will be required to fill out a form in order to complete their account (registration). The account information is as followed.

Account Information:

- ❏ Email and password

Dog Information:

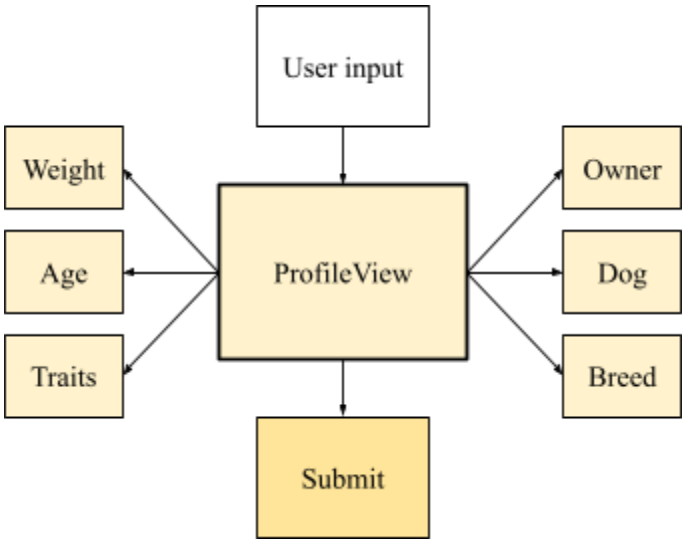
- ❏ Owner's Name
- ❏ Breed
- ❏ Dog's Name

Optional Information:

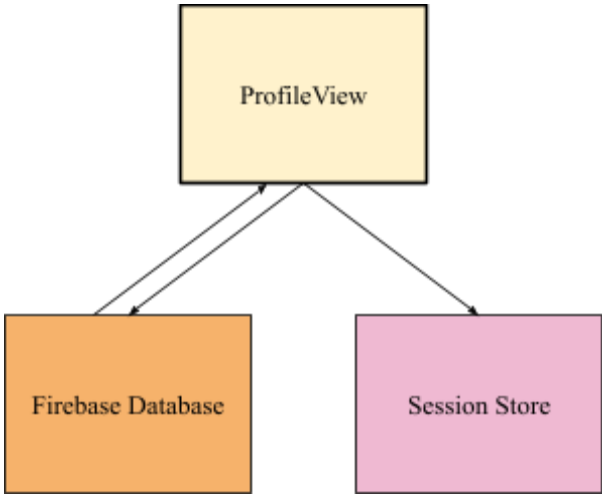
- ❏ Weight
- ❏ DoB/Age
- ❏ Personality (social, shy, aggressive, etc.)

Note: The user (after registration is complete) will be able to edit this page at any point in time by changing information and selecting the update button on the profile page.

4.2.3. Static Model



4.2.4. Dynamic Model



4.2.5. Design Rationale

The information displayed on this page will be utilized by Fetch in order to display the profile to other users. If a user fails to provide any of the required information, they will essentially enter empty strings into the database, which will be displayed as an empty category when other user's view their profile within the deck. This may discourage users from matching with other users due to lack of information.

Since SwiftUI, FireBase, and XCode were new to all members of our group, we decided to split each module into groups, to all for better support while trying to learn new languages, databases, and development programs. This proved to be very beneficial, with two groups tackling different aspects of the application. Within each module group, we split into front end and back end development, to be time-efficient. Three members tackled the login/profile pages with the addition to FireBase to create user authentication within the login/sign up page.

4.2.6. Alternative Designs

A previous approach to the user information, specifically for new users, was just to populate the form (user/pet info) on the login screen. However, the issue we ran into was determining whether the user was new or not. There is no toggling option such as, "are you a returning user or a new user", and including that would be too redundant and take away from the simplicity we'd like to offer. We're currently utilizing the idea that existing users will navigate to the homescreen after providing their credentials, rather than incorporating more costly functions (respect to time), and new users will navigate to their own module.

4.3. Deck

4.3.1. Primary Role and Function

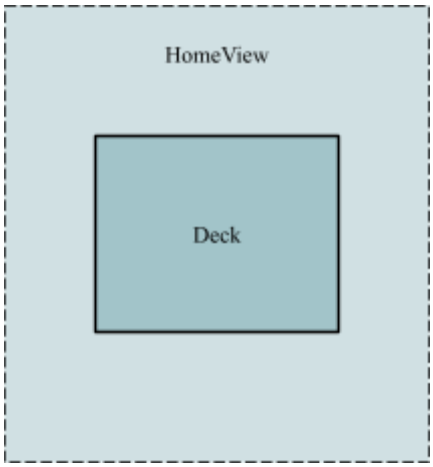
The Deck is the core functionality that allows for Fetch to create matches for various users. The operation can be visualized similar to a deck of cards that are all facing up. A user will go through each card, or in this case dog profile, 1 by 1 and determine whether they think this owner and pup will make a good fit for a future playdate. Once a user has made a decision on a profile by accepting or declining, that profile will leave the deck and enter one of two database entities: Matched and Declined..

The *Matched* pile will hold the current user's matches to be viewed on the matches page. This will present the user with the key information about their match, as well as an email to contact their match. Declined users will be stored in a separate entity within the database. This information will not be displayed to any user, so no user has to worry about declining a user and receiving backlash.

4.3.2. Interface

Users have control over their Deck and piles through "buttons" that will appear on their Homepage as well as within their Matches pile. The leftmost button containing a heart will be how users will match with a profile. The right button containing a red "X" will be used to decline a profile. The middle button containing an arrow is used to populate the deck and must be pressed before matching/declining can occur. The middle arrow button can be used to repopulate the deck as well to change a decline to a match for a profile.

4.3.3. Static Model



4.3.3. Dynamic Model

Not applicable, for the Deck is simply an important attribute of the HomeView module and doesn't interact with other modules.

4.3.5. Design Rationale

The mechanics behind Fetch was inspired by the app Tinder where all users are displayed in the deck, except for the current user. They will not see their own profile in the deck, to avoid matching with themselves. For Fetch to accomplish that, the current user's ID is stored and checked with profiles in the deck. there needs to be discard piles consisting of the outcome of a series of user decisions. Even if a user exits the app, their decisions on profiles will remain within their respective piles so that future visits to Fetch will continue to present relevant profiles on their Homepage. Additionally, all profile information will remain within the users account to prevent the user from having to reenter information every time they open the app.

Fetch only requires one user to match with a profile for the profile to appear on the matches page. This differs from Tinder in this aspect.

4.3.6. Alternative Designs

Another possible architecture that could have been carried out is changing the "one-way street" design aspect to be "two-way". In other words, if one user accepted another profile, they would have to wait for that user to also match with them. This would follow the Tinder matching algorithm. This would then create a Pending pile aspect of the Deck architecture, where users would wait for pending profiles to become declined or accepted. This algorithm would be similar to friend requests on platforms like Facebook or Instagram.

4.4. Matches

4.4.1. Primary Role and Function

The deck essentially holds all the functionality for pairing matches or scrapping them under the hood. The matches functionality specifically works with the Accepted pile (array) and visually displays the array of matches (accepted) for the user.

Matched users will appear on the matches page, with the email of the profile the user matched with. This will allow users to easily navigate to the matches page when wanted to contact a user regarding a playdate for their dog. Matched users will continue to stay stored to be viewed at a later time.

4.4.2. Interface

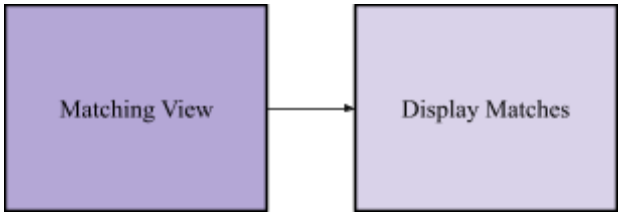
A visual representation of all matches with said user will be displayed to them (vertically aligned), along with some key pieces of information such as

- ❑ Picture of dog

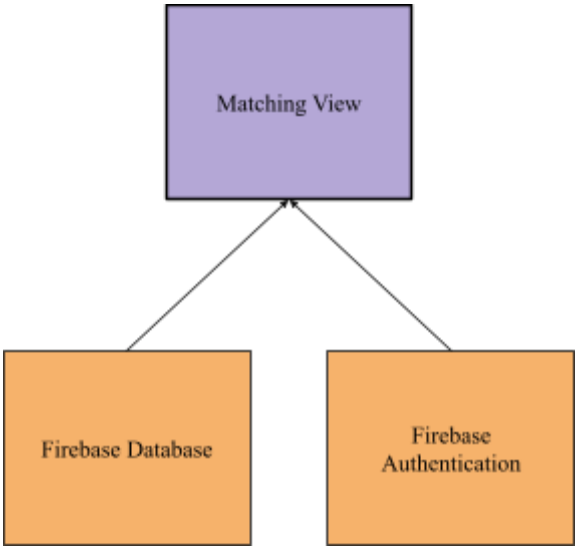
- ❏ Name of dog
- ❏ Name of owner
- ❏ Dog's Breed

Users will have the option to match with a user using the match button (heart) or decline a user using the decline button (red "X"). A picture will appear on the deck page, however, it will remain the same image of one dog. This will be a future project to have each profile display their own profile image for other users to view.

4.4.3. Static Model



4.4.4. Dynamic Model



4.4.5. Design Rationale

Since the logic behind matching is kept under the hood, there's no way for a user to access all the potential matches they've encountered. That's why the overall design for the Matches page/module is to visually represent every user contained within the matches array, so they can start arranging playdates.

For this specific module, it was imperative to separate the logic from the homescreen to avoid collision of any sort. A majority of the logic and functionality is contained within the home screen. Which is why we're essentially branching off the homescreen, extracting the matches array, and adding more functionality & complexity on a separate module.

4.4.6. Alternative Designs

The previous design for the matches was a bit ambiguous from the beginning. We knew we wanted to display all possible matches, but thought, what else would a user want to have access to? The most obvious attributes needed to be a picture of the dog, dog name, and the owners name. The other issue we ran into in regards to the matches array was, exactly how would the deck be populated? We'd consider similar breeds, age, reactivity/social levels, and gender, which sounds simply. However, when that was said and done, a sketch was drafted with a trivial deck, and quickly realized we had no location attribute, which is needed to filter the deck locally. That way someone from Oregon wouldn't be matched up with someone from New York (or anywhere else).

Another complex feature that we've drafted up (if complexity isn't an issue), and want to implement is an unmatched feature, which would be implemented by a sliding feature. WIth experimenting with the sliding feature, we ran into difficulties quickly and let the match and decline options as buttons.

4.5 Homepage

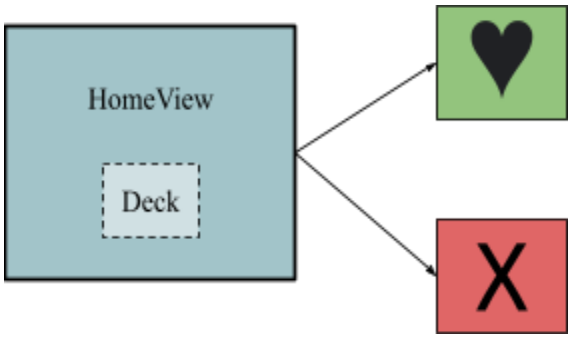
4.5.1. Primary Role and Function

While the Deck module holds the functionality for how profiles will be moved into a specified user's discard piles, the Homepage's role is simply a way for Fetch to display the profiles remaining in the Deck and allow for users to interact with new profiles. It will display profiles 1 by 1, the red "X" and green "♥" buttons, the middle arrow button to populate the deck, and the tab bar. After pressing the arrow button, the deck will be populated and will begin to display the user's profiles including the user's name, their dog's name, and their dog's breed.

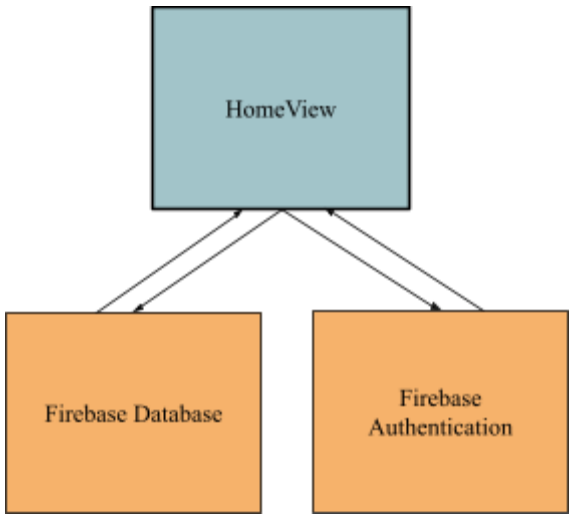
4.5.2. Interface

Users will have the opportunity to interact with the Homepage in various ways: A user may make a decision on a profile by selecting decline ("X"), match("♥"), relocating to the Matches page or relocating to the Profile page.

4.5.3. Static Model



4.5.4. Dynamic Model



4.5.5. Design Rationale

In order for Fetch to attain sufficient organization, there needs to be separation of client and server side architecture. Because the architecture and design behind the Deck module is relatively complex, the Homepage provides a means for visualization of the Deck and the component that allows users to shift to the other modules, Matches and Profile. The Deck and homepage containing only three buttons was a design decision to simplify the app for the user's sake. The red "X" and green heart were designed to make their purpose clear without flooding the page with words.

4.5.6. Alternative Designs

Another approach that was taken into consideration was combining Homepage and Deck into one module, since they are essentially the same thing with the exception of Homepage being client side and Deck being server side. This would have made the backend of the homepage and deck difficult, separating entities that correlated nicely together.

5. References

Faulk, Stuart. (2011-2017). CIS 422 Document Template. Downloaded from https://uocis.assembla.com/spaces/cis-f17-template/wiki in 2018. It appears as if some of the material in this document was written by Michal Young.

IEEE Std 1016-2009. (2009). IEEE Standard for Information Technology—Systems Design—Software Design Descriptions. https://ieeexplore.ieee.org/document/5167255

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Commun. ACM, 15*(12), 1053-1058.

Google, Google, firebase.google.com/.

6. Acknowledgements

We would like to thank Professor Juan Flores and Joseph McLaughlin for being supportive and gracious “clients” on this project, and for providing resources for us to continue this work.