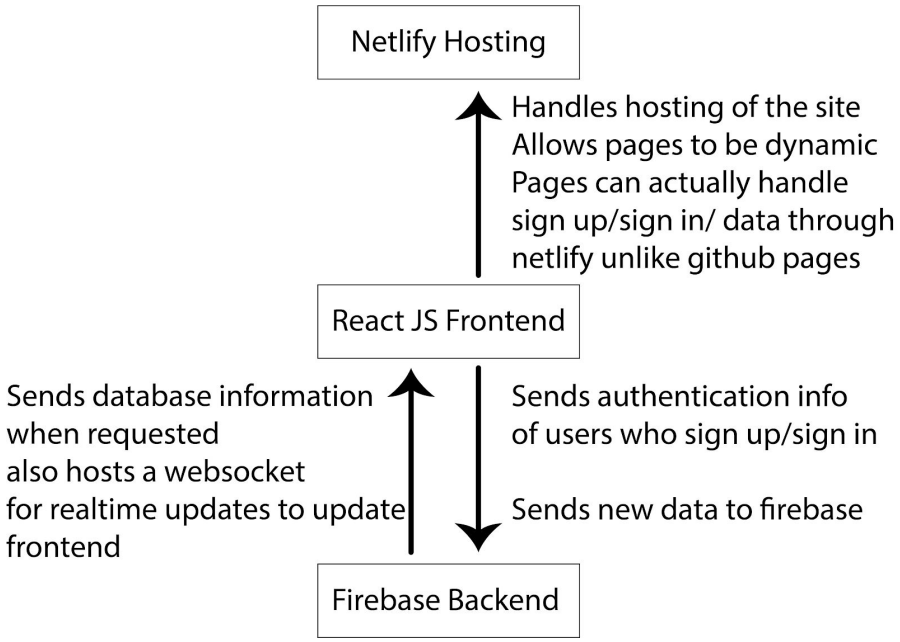
Milestone 4

Project Features List

- Users should have a personalized account and homes
    - Users will have their own accounts with a name, profile picture, email, and other extra data if need be. Users can also have set homes for various people (i.e. frat house, apartment, home with the family)
- Categories
    - Users will be able to see chores, supplies, and payments at a glance on the home page
- Assigned user tasks
    - Roommates can assign each other tasks, chores, supplies, etc.
- Calendar
    - Allow for easy organization of upcoming chores, maintenance, and payments.
- History
    - Users will be able to access the history of what's been completed in the house in the past for any reason - this includes payment transactions, chores completed, supplies bought, etc.
- Payment Plan
    - This has changed as we will not be working with venmo or cash app anymore. We will instead just focus on incorporating PayPal.
- Charts
    - Users will be able to have figures to see in detail various charts for who's done how many chores, total payment figures, etc.
- Notifications
    - Users will have the ability to receive notifications regarding the activity in their home (e.g. someone assigned them to an activity, someone paid you, etc.)

Architecture Diagram

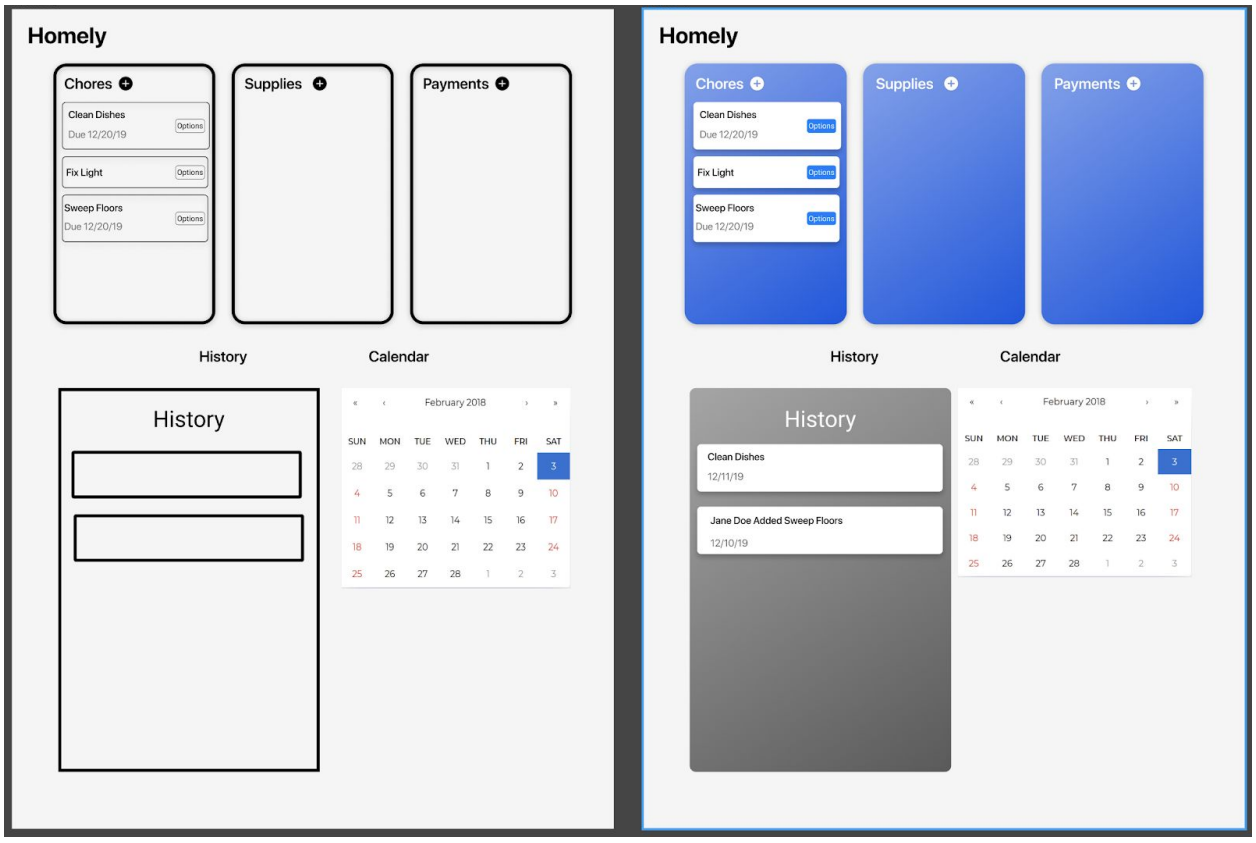


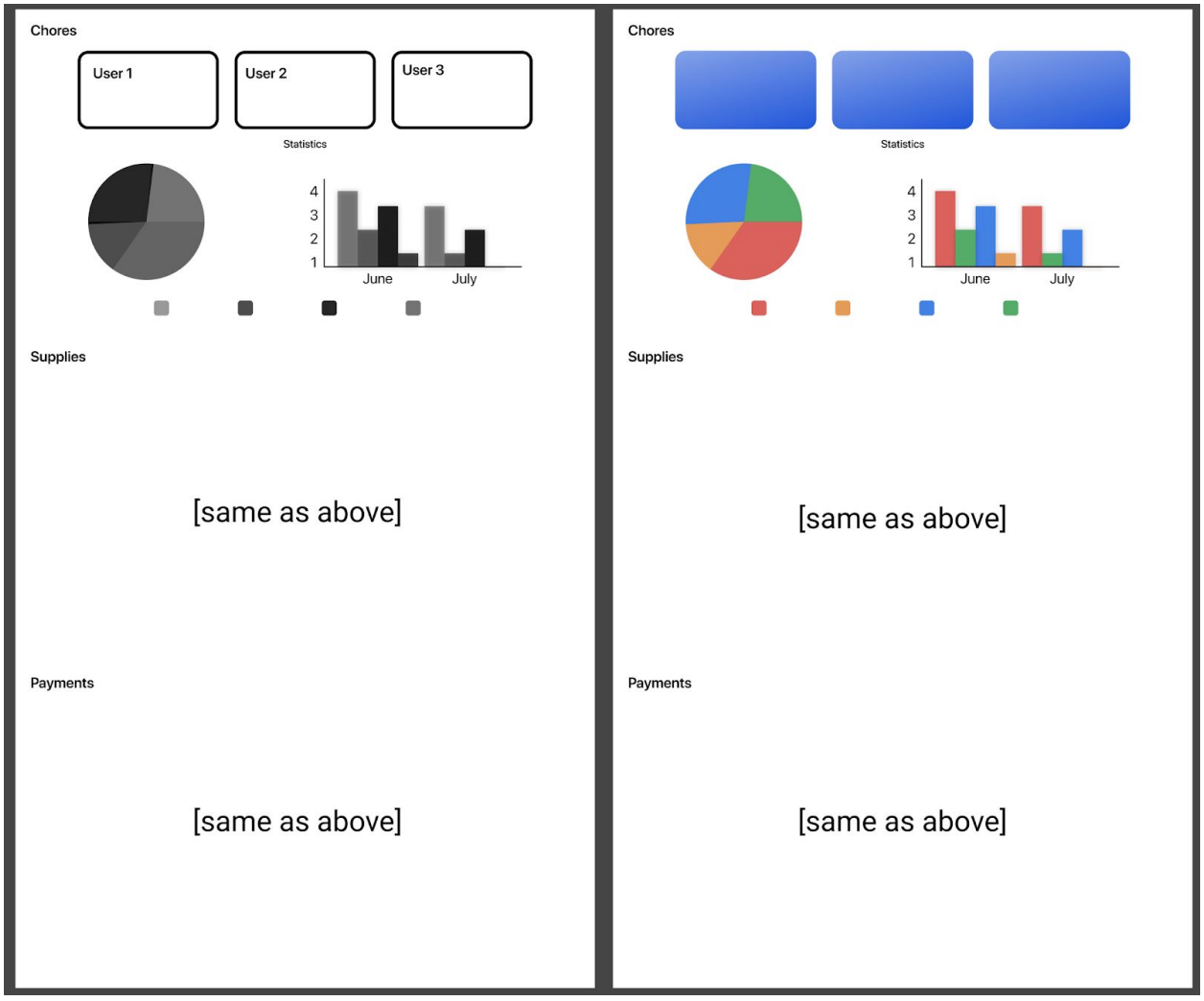
We use firebase to handle all information regarding handling accounts and databases. Firebase also allows for authentication and databases to be easily linked. ReactJS will render the data from firebase depending on if an account is signed in and what data is there.

Netlify will handle hosting of the site and will let reactJS render as it needs to. Netlify also works well with dynamic data that changes per user because it handles it like a normal server rather than a static page like github.

Front End Design

We have developed both WIREFRAME and a Colorized version of what we expect our website to look like.

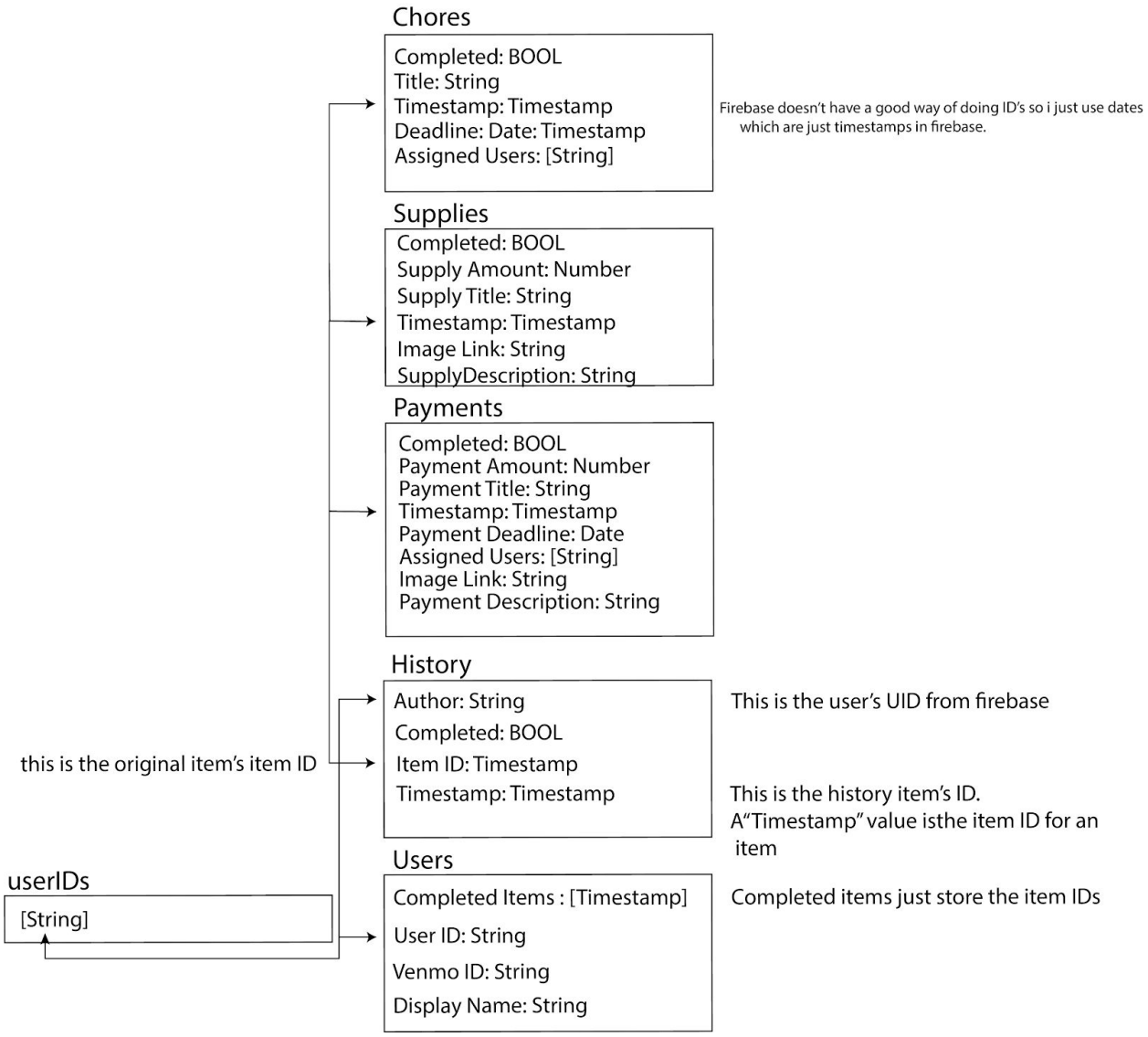




Web Service Design

We use firebase as our primary web service API. Firebase doesn't really do a REST form of their api, rather it has its own methods from the library. So rather than a GET request to retrieve data and a POST to send new data, it just uses a firebase.get() and firebase.update(). Firebase asks for a collection (in our case 'Homes') and a document (a randomly generated string is grabbed on launch) and will use those to get the data and set any new items. The data that is received from our initial get is the homes that the user is added to, and we set new data as a javascript object to the home.

Database Design



Our database model is centered around the user data that is created, along with a generated ID based on the time it was created. Firebase uses a *Timestamp* data type which is their form of dates. Chores have a few things, which is the Title, whether the chore was marked as completed (on the first time a chore or any item is made, Completed is false), the Timestamp, which is the ID, the deadline date, and the assigned users, which is an array of user IDs that are assigned.

Supplies and payments follow a similar model, just with more options, as they can support images, descriptions, and require a quantity or amount.

History items are unique. They are generated at the same time as an item is created. Their model is just an item ID that is the same is the original item's ID. They also have their normal Timestamp, which is just its own way of identifying itself (also convenient for sorting by date

when representing them!). History items also record the author so that we can say “Jane doe created X Chore). Lastly, History items have a completed key which will represent if someone marked it as complete versus they created the chore.

The users record has 4 keys, but given our feature changes to remove venmo and support paypal, some keys may change. In the Users record, there exists an array of Timestamps for completed items. When a user completes an item, the completed item’s ID is added to the array of timestamps. The user also has a User ID, which is used in history to lookup who created/completed an item. There is a venmo ID that would allow for venmo payments, but unfortunately venmo isn't supported so that key is useless for now. Could be useful if someone in the home was looking for their roommate’s venmo though! Lastly, the user’s display name is there. History items use the User ID to find the user record, then it grabs the display name from there. The display name key is needed to represent users when looking them up by a readable name

Lastly, there are the userIDs records. It is just an array of strings that are the User ID’s. This is used in the security rules. When a user attempts to read from a home or add to a home, firebase will check if they are in the userIDs array. If they are, they are allowed, and if not they get an error.

We use firebase instead of postgres because of its close integration with its authentication stack. Firebase allows for easy sign in so we don’t have to implement our own solution as we wouldnt make it as secure due to lack of experience. It made more sense to use firebase to handle sign in as it would also allow for easy permission handling so people can only read their own homes. With postgres, we would have had to handle signin ourselves and that includes its own set of features.