Team Working Agreement

Last Revised: 12/2
Fall Quarter

ChoreMates

Katherine Kosolapova - Product Owner
Maggie Heathcote - Developer
Nathan Nguyen - Developer
Annabel Truong - Developer
Victoria Ayala - Developer
Ethan Lin - Developer

Definition of Done

User Stories

- Functionality implemented fully
    - All acceptance criteria for a user story are met and functionality is implemented as needed by requirements.
- Code integrated in correct place, styled, and tested (manual or automated)
    - Code is merged with the main branch and there are no merge conflicts. Tests must pass successfully before being merged.
- Additional components removed
    - Anything that isn't necessary to functionality of code must be removed (i.e excess comments, files that aren't used, anything unaccessed)
- Testing console.logs removed

Sprints

- Demo sent (if applicable) and approved of by group (i.e. no objections or suggestions)
- All planned user stories in the sprint backlog have been completed and meet DoD for user stories
    - Any incomplete stories are moved to next sprint backlog
- Sprint retrospective is held to discuss what went well, what didn't, and areas for improvement
- Sprint artifacts (burndown chart, backlog) are updated to reflect progress

Style Guide

Our Styles

- Began styling with React Native Coding Standards and Structure
    - Primarily focused on
        - Renaming with camel case in JavaScript
        - Using snake case in MySQL
- Following Google's JavaScript Style Guide
- Hidden .env for database host, password, and API URL
- Encapsulate components in separate files for reuse/readability
    - Buttons, headers, themes, etc
- Commented code and gave functions/variables clear names to ensure readability
    - All comments are paired with the initials of the person adding that comment
    - That way, if there are bugs in the code or if someone needs help understanding a section of the code, they know who to go to
- Ensured even spacing between blocks of code, yet minimized unnecessary whitespace
- Imports and functions follow a consistent structure – useEffects at the top of the screen and rendering/styling at the bottom
- Created helper functions and Providers/Contexts to prevent code redundancy

Folder Structure

- api
    - tests – contains all written Unit Tests
    - db/connection.js
    - utils – contains helper functions and cron scheduling for recurrence
    - routes
        - contains all the GET/POST routes for backend api calls to the database
    - server.js – what gets called to start the server
- app_ui
    - components – contains app components such as blocks, buttons, and dropdown
    - contexts – contains UserContext, ThemeProvider, and other providers necessary for the app
    - functions/markCompleted.js
    - icons – contains all the files for the profile icons
    - screens
        - contains all the screens which get accessed on the tabs
        - NewChore, ChoreDetails, ManageGroup, etc are all considered screens and are treated accordingly
    - tabs – Home, Chores, Groups, and Settings pages
    - App.js – what gets called to start the expo app