Individual Contributions

Cloie Dale

Database:

- Initial high level database design
- Firebase Repository functions to allow basic CRUD functionality with the database
- Data services that implement the logic between the repository and the app (ex: getNumWords() calls the repository to get a child doc and returns the number of words)
- Database refactor to include inputs for both english and spanish for words and language selection field for both child and parent.
- Database anonymization to comply with IRB requirements

Spell Check:

- Util function that uses the wiki api to check if a given word exists in the selected languages and updates the word bank with the word's data (language, part of speech, definition)

Localization:

- Inable the users to switch to their preferred language through a localization service that translates the text in the app based on their selected setting
- Implement a Localization Service and Provider across the app to listen for changes to the selected language
- Use the firebase localization service to translate pages that came out of the box (log in page and user profile)
- Update settings page to allow parents to change their language preference and select their child's language preference on creation.

Video:

- Set up google cloud storage bucket to be used for media storage
- Create cloud functions to create signed urls for secure media upload and download
- Create video upload feature that works with add words to allow users to select videos from their device, have that video saved to their folder in the google cloud bucket, and tag their child's word tracker with the video

- Create a video streaming page to allow users to see videos. This was not fully implemented by the end of sprint 3 but the backend is functional, but the video playback window is not operational.

Gwynevere Deterding

UI Design:

- Designed the bottom navigation bar for the application
- Designed the styling for the application

Parent Home Page:

- Designed parent home page to display the number of words the child knows at the top, two cards with additional stats of “most recent word” and “words learned in the past week”, and the “add words” text field at the bottom of the page for word submissions
- Developed queries for the stats displayed
- Implemented word parsing for the text field that extracts words from the input, spell checks them with Cloie’s spell check function, and either displays an error message if the words are incorrect or adds the word(s) to the database if they are valid

Researcher Home Page:

- Implemented researcher homepage that displays a data table of all of the word utterances submitted by the users of the app
- Developed queries to get all of the data for the table
- Implemented filtering of the table with a text field that supplies autocomplete suggestions based on the selected field from the dropdown menu as the researcher types
- Implemented sorting of the table by any field
- Implemented functionality to download the data table as a CSV

Robert Powell

Statistics Page:

- Utilize database services to query data from a particular child in a specific format
    - Various query optimizations at points throughout the project
- Display queried data in readablee graphs using syncfusion
- Wrote adaptable graph switching function that should be easily explandable to display different statistics as needed by later implementers
- Utilized simple caching system to save queries when the same graph is requested twice in a single session
- Documented stats

Provider:

- Researched and selected Provider for use as our Dependency Injection library
- Provided config data to each page, allowing selection of language and current child.

UI:

- Allowed selection of current child through a drop down menu system in the top bar, displayed on every user-end page
    - Various optimizations for this menu with Provider help from Marcelo
- Various refactors to encapsulate the bottom bar and other elements needed on multiple pages

Settings:

- Added feature to add new child
- Added feature to add existing child to another parent via their email address

Marcelo Torres

Provider:

- Implement provider for services necessary throughout the app.

Database API:

- Implement model and service classes for database interactions that facilitate data retrieval and storage following Cloie's design
- Develop batch queries for Firestore

Authentication:

- Implement authentication pages using Flutter UI library
- Implement authentication service to track current user uid, email, and roles and provide them in real time throughout the app
- Implement user model service to keep up to date user database data and provide it in real time throughout the app

Custom Claims:

- Implement roles for users using custom claims
- Update Firestore rules to use custom claims for permissions
- Update authentication service to track user custom claims

Cloud Functions:

- Implement cloud functions to support custom claims (role) management
- Develop utils for role management and verification
- Implement a cloud function to get a users custom claims as a list
- Implement a cloud function to get a list of users' UIDs, emails, custom claims, and account status
- Implement cloud function to add a user's child to another parent