WordBuds

Sprint Planning Document (Sprint 1)

Sprint Goal Backlog (Sprint 1)

Chase Compton, Nick Pucci, Joe Schmith, Caitlin Seaborn, and Zimuzo Ernest-Eze

High-Level Project Overview

Project Mission:

WordBuds is a mobile app allowing parents to track their child's early language learning over time. Parents can log words their child learns through text, audio, or video, and view graphs showing their child's learning. The data logged in the app will also be used by the University of Alabama's Language Education and Cognitive Science Lab for linguistic and child development research.

Motivation/User Groups:

- **Parents:** Parents can track their child's language learning
    - This allows parents to be more engaged with their child's development
- **Researchers:** UA researchers can use the data entered by parents for continued research
    - Research into first language learning is currently limited
    - This type of dataset for early language learning is essentially nonexistent
    - This data could lead to major progress in linguistic and child development research

Project Overview:

- **Mobile Application**
    - **Sign-in:** Users should be able to sign in using [sign-in methods]. Users have access to different portions of the app baked on their role (Parent, Researcher, or Admin)
    - **Sign-up:** When opening the app for the first time, the user should be prompted to create an account and accept terms and conditions. An app walkthrough and introductory survey may also be added
    - **Profile:** Contains information for user and any added children
        - Each parent account has an email/contact information
        - Each child added to a parent has a name and birthday, and is linked to the input data for that child
        - Parents can share their child's data with another parent
    - **Input Page:** Parents can input a word, phrase, or sentence uttered by their child and each valid, unique word will be added to their child's data. They can also upload an accompanying video or audio clip.
    - **Statistics Page:** Parents can view a variety of graphs generated based on their child's entered words.

    - **Video/Audio Page:** Parents can view and download/stream the video and audio files they have uploaded
- **Researcher Dashboard**
    - **Homepage:** Researchers can view all data entered by parents in tabular form, and download the data as a CSV file
- **Admin Page**
    - Admins can modify user roles and other settings from the admin page
- **Backend Services**
    - Firestone database
    - Authentication (Firebase)
    - Localization/language options
    - Spell check/dictionary check functions
    - Cloud functions (firebase)
    - Video and audio upload (Google Cloud storage)

Existing Features:

- **Add Words**
    - Located at bottom of landing page
    - Text - user can input string of words, phrases, or sentences in text box
        - Input is parsed for unique, individual words that are valid in English or Spanish dictionary
        - Words sent to spell check (check_and_update_words)
        - Words that are new to project-wide word bank
    - Video - user can select a video from their device
        - Only allows .mp4 files
        - Uses file_picker library
    - After words are verified, app fetches a signed URL that is used to upload the video to the parent's google cloud storage bucket
    - For each unique word, a word tracker will be created or updated in the child's word tracker collection
    - Current child can be identified by calling the current child service - this is also used to display the current child's name in the app
- **View Stats**
    - View Stats/Learning Summary page of app displays graphs about child's learning
        - Syncfusion is used for graph display
        - Graphs can be selected using drop down
        - Time horizon can be modified using numerical input box
    - The page is stateful
        - Graph type being displayed is GraphType Enum

                - There is a global final list of GraphTypes (graphsWithLength) that indicates which graphs have adjustable length
                - Both should be updated when new graph types are added
            - TextEditingController allows user submission of graphLength
            - graphCache element is a map from (GraphType, int, String) to dynamic to prevent re-querying same data during a single session (reduce database hits)
    - Functions in Build
        - wordsKnownFeature - wrapper for database call to get total words known by a child
        - graphSwitcher - wrapper for a switch statement that selects proper graph to be displayed (needs to be expanded for new graph)
        - lengthChangeFeature - wrapper for text box and submit button to change graph length
        - graphTypeSelectDropdown - dropdown to select graph
    - Every graph is a FutureBuilder - function depends on results of asynchronous function that depends on results of Graph Data Function
        - Graph Data function makes queries and organizes them into a list, then uses the list to create a graph
    - Note - graphs/queries for a full learning history can be large and expensive - previous team suggests adding caching or better database end statistics
- **Authentication**
    - Handled by firebase - UI is provided by package firebase_ui_auth
    - Users are given a randomly generated userID stored in firebase and parent role by default
    - SIgn in methods can be enabled/disabled in Firebase console - by default email/password sign in is enabled
    - Authentication status/user roles are checked and stored by AuthenticationService class - can be used to access user email, id, displayname
    - User database data is stored and synchronized using UserModelService class - can be used to receive user specific info like child IDs, should be accessed using consumer widget and provider
- **User Roles**
    - Admin page is available to users with admin role
    - User roles can be managed in admin page - accessible from researcher home screen
        - Admin can enter a specific email address to modify
        - “Get Custom Claims” - gets roles for the email currently selected

        - Assign/remove user roles for selected email (uses cloud function)
        - UID-Email Table - fetches table that contains email address, user ID, custom claim/roles, and account status of all users
- **Add Child**
    - Uses utility function
    - GUI has user input a name, birth date, and set of languages
    - If parent exists, uses childDataService to create a new child and add to database with current parent
    - Also modifies parent's list of children using parentDataService
- **Share Access to Child**
    - Add existing child to another user
    - Passing in a valid email
    - Uses addChildToOtherParent
- **Video Function**
    - Video upload process: file selection, get signed url, upload file
    - Videos are not currently compressed
    - Video download process: user selects word, corresponding signed url is generated and used to request the file, file is written to temporary directory and should be available for playback
        - Previous team states this features didn't work properly
- **Researcher Page**
    - Only accessible to users with researcher role
    - Displays a scrollable table of all word utterances submitted by all users
        - Each entry contains child ID, child age, word, language, part of speech, and date of first utterance
        - Table can be filtered using dropdown menu
    - Current data table can be downloaded as CSV file
- **Cloud Functions**
    - Run on firebase hosted server, used for secure code or adminSDK is needed
    - Assign/remove roles
    - Get roles/custom claims
    - Get email-uid table
    - Generate video upload/download signed URL
    - Updating cloud functions: change index.js file in functions folder and run firebase tools command
        - Modify and save index.js
        - From firebase_project directory run "firebase deploy –only functions"

Remaining Features:

1. Video Download/Playback OR change to streaming
2. Compressing videos
3. Terms and Conditions page
4. User intro survey
5. App tutorial
6. Option to convert a user to researcher
7. Complete process for Apple Store
8. Complete Process for Google App store
9. UI/UX improvements
10. Additional graphs/features for researchers
11. Additional graphs/features for parents
12. Add audio upload
13. Improve language selection

Sprint 1 Planning

Sprint 1 Goal:

The primary goal for Sprint 1 is to fully onboard the new team to the existing Baby Words Tracker codebase and infrastructure. This involves a comprehensive discovery phase to understand the current architecture, features, and previous design decisions. Key objectives include establishing a consistent development environment for all team members, investigating and analyzing the app's data efficiency to plan for future cost-saving optimizations, and implementing initial "quick win" UX/UI enhancements to gain familiarity with the front-end codebase. Furthermore, we'd like to tackle some of the non-technical work in preparation for App Store publishing.

Sprint 1 Deliverables:

1. Conduct Project Discovery and Codebase Audit.
2. Establish and Document Development Environments.
3. Investigate and Analyze Data Efficiency Issues.
4. Research TestFlight and set up beta for users.
5. Add terms and conditions page
6. Add an introductory survey page
7. Improve home page UI/UX
8. Improve statistics page UI/UX
9. Word entry system error testing

Sprint 1 Deliverables Detailed:

1. Conduct Project Discovery and Codebase Audit

- **Assigned:** All Team Members
- **Description:** This foundational task involves a thorough review of all existing project materials to build a comprehensive understanding of the application.
    - Review the feature documentation left behind by the last team to understand the intended functionality and implementation of each feature.
    - Analyze the project's file structure, including the Flutter App and the Firebase Project, which contains Cloud Functions.
    - Examine the data architecture, including the Firestore database structure, data models, repositories, and services.
    - Investigate the implementation of core systems like authentication, spell checking, and video handling. The goal is to understand the previous team's decisions and identify potential limitations or areas for improvement.

2. Establish and Document Development Environments

- **Assigned:** All Team Members
- **Description:** Ensure every team member has a fully functional and consistent local development environment to prevent configuration issues.
    - Set up the Flutter development environment for the application.
    - Configure the Firebase project environment, including installing dependencies for Cloud Functions
    - Successfully connect the local application to the Firebase emulators for testing database rules and functions offline.
    - Document the complete setup process and any project-specific configurations to streamline onboarding for any future team members.
    - Get up to speed with Dart and the Flutter frameworks.

3. Investigate and Analyze Data Efficiency Issues

- **Assigned:** Chase & Nick
- **Description:** A key concern for scalability and cost management is the number of database reads, particularly on the stats page.
    - Analyze the stats page, specifically the Graph Data Functions that make database queries to populate charts.
    - Evaluate the current caching mechanism, which is designed to prevent re-querying data during a single session on the stats page.
    - Based on the previous team's notes, assess strategies to reduce database reads, such as enhancing the caching system or creating database-end aggregate statistics. The final deliverable for this task will be a planning document outlining the sources of inefficiency and a proposed technical strategy for optimization in Sprint 2.

4. Research TestFlight and set up beta for users.

- **Assigned:** Chase & Nick
- **Description:** As the primary goal is App Store publishing, this task involves researching the process for beta distribution. The team will investigate the requirements for Apple's TestFlight, including developer account setup, Xcode configuration, and the process for creating and uploading a build to App Store Connect. This will prepare the team to deploy a beta version to test users in a subsequent sprint.

5. Add “Terms and Conditions” Page

- **Assigned:** Joe and Zimuzo
- **Description:** To meet App Store requirements, the application must include legal documentation. This task involves creating a new, accessible page within the Flutter application to display the Terms and Conditions. The team will implement the UI for this page, which will later be populated with text provided by the project sponsor.

6. **Add “Intro Survey” Page**

- **Assigned:** Caitlin and Zimuzo
- **Description:** Based on the previous team's plans, a demographic survey is needed for research purposes. This task is to create the user flow for an introductory survey presented to new users after account creation. The team will implement a page that can either host a simple, built-in form or integrate a WebView to display an external survey from a tool like Qualtrics.

7. Home Page UI/UX Improvements

- **Assigned:** Joe and Zimuzo
- **Description:**
    - Make adding words more intuitive
    - Add a “custom word” button
    - Improve child summary visuals (child name, number of words, most recent word)
    - General visual improvements by establishing a consistent style
    - Fix screen shake glitch

8. Statistics Page UI/UX Improvements

- **Assigned:** Nick
- **Description:** The statistics page can be confusing for new users. This task involves improving the experience for a user who has not yet added a child or logged any words. The team will implement an "empty state" for the page, providing a more welcoming and instructional view instead of a blank or broken-looking graph area.

9. Word Entry System Testing

- **Assigned:** Caitlin
- **Description:** This task is to improve the robustness and user-friendliness of the core word input feature. The team will systematically test the system under various conditions, such as offline status or submitting words with punctuation. The goal is to identify and document failure points and improve how the application communicates errors to the user when a word cannot be processed by the spell check API.