WordBuds - Testing Plan and Test Cases

1. Introduction

Wordbuds is a dual-platform system designed to monitor and analyze the linguistic development of children. The primary part of the system is the mobile application for data entry by the end user, the other portion is an Administrative Web Dashboard used for high-level data and account management with role-based access for various actors who may use the site for analytics oversight. Testing for the mobile app consists of manually verifying data entries, UI functionality, and data integrity. The testing workflow for the Web Dashboard is automated via *Playwright*

2. Testing Strategy & Methodology

The project employs a hybrid testing methodology to maximize coverage across different technological stacks.

- **Automated Regression Testing (Admin Web Dashboard):** Utilizing the Playwright framework to execute headful and headless browser tests. This ensures that new features or deployments do not break existing critical infrastructure like authentication and component rendering.
- **Manual Functional Testing (Mobile Application):** A hands-on approach to simulate the end-user journey. This focuses on data integrity, UX flow, and the verification of the logic regarding child development metrics.

How to run existing Playwright tests

Run the following commands from a single terminal in the “webapp” directory

```
npm install

npm run build

npm test

npx playwright install

npm run start -- --host 127.0.0.1 --port 4200
```

In a second terminal, run the Playwright tests using

```
npx playwright test
```

3. Administrative Web Dashboard (via *Playwright*)

Playwright scripts are configured to validate Chromium, Firefox, and WebKit engine based scenarios listed below.

Test Case Title	Detailed Objective	Procedure	Expected Result
Initial Page Load & Latency Check	Verify that the application is reachable and that core CSS/JS assets load without 404 errors.	Navigate to the staging/production URL and await “networkidle'” state.	Page title is correct, and no console errors are present
Negative Auth: Unauthorized Access	Ensure the security perimeter prevents accidental entry via invalid credentials.	Input a non-existent email and "password123". Click 'Login'.	System triggers a '401 Unauthorized' and displays a user-friendly error toast.
Role-Based Access Control (RBAC)	Validate that different permission levels see only the data they are authorized to view.	Log in as "Admin", “Researcher”, and “Parent” role and attempt to navigate to the data view.	System redirects the user to a restricted dashboard view.
Component Presence & Interactivity	Confirm that critical UI nodes (Sidebar, User Table, Export CSV) are present in the DOM.	Use CSS selectors to verify visibility of key UI elements (selectors) and visual components (Graphs, User Data)	All tracked selectors return a visible status and respond to click events.

MANUAL **Database Testing**	Ensure account updates are properly tracked to the database after making edits on the webpage.	Create a test account and modify fields such as role, name, etc. and confirm the account changes.	Firestore database reflects all changes made to the test account,

4. Mobile Application (Manual Functional Testing)

Test Case Title	Detailed Objective	Procedure	Expected Result
Child Profile Instantiation	Ensure the database correctly creates a unique identifier for a new child record.	Navigate to 'Add Child', input Name/DOB/Gender, and save the profile.	The profile appears in the 'Child List' and persists after a fresh application restart.
Vocabulary Input & Linking	Verify that words entered into the input field are correctly associated with the active child profile.	Select "Child A", click 'Add Word', type "Banana", and submit.	The word is added to the database with the foreign key corresponding to "Child A".
Data Integrity: Word Norms	Check that the app's internal logic correctly assigns developmental norms to specific words.	Add the word "Ant". View the word details.	The app displays the correct Average Age of Acquisition based on provided norms.
Chronological Metadata Verification	Ensure that the system accurately timestamps the moment of acquisition for language mapping.	Add a word and immediately check the 'Date Added' field in the word history.	The timestamp matches the device's system date and time.

5. Environment & Tools

- **Operating Systems:** Windows 11 (Web Testing), macOS, iOS / Android (Mobile Testing).
- **Testing Tools:** Playwright (v1.4x), Node.js environment.
- **Reporting:** All manual failures are to be reported, Playwright generates documentation in the case that any automated tests fail.

By adhering to this test plan, the WordBuds team ensures that both the Web app and the mobile app is functional and reliable.