**Software Design Document**

**Description of project:**

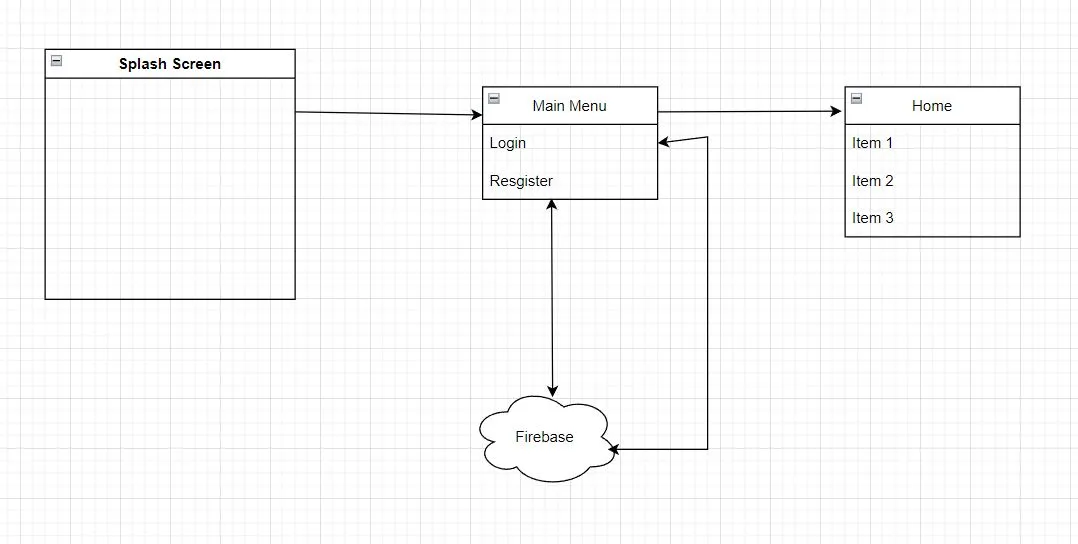
* **Purpose:** The primary goal of Seedling is to offer a streamlined, easy-to-use platform for learning essential words and phrases in various languages. It aims to simplify the language learning process using a flashcard system, designed for beginners and casual learners. The app will provide users with 625 essential words and phrases per language. It will be accessible across iOS and Android devices, ensuring a larger user base. The core functionalities include a landing page, a flashcard learning system, user authentication, and language selection options.

**Minimum Viable Product Features (MVP):**

1. **Landing Page:**
   * A simple, user-friendly interface that introduces the app and its features.
   * Ability for new users to sign up and returning users to log in.
2. **Functioning Flashcard System:**
   * Interactive flashcards that display words or phrases with the potential option to view translations, hear pronunciations, and flip for additional context or examples.
   * Progress tracking and review sessions based on user interaction and retention rate.
3. **User Login/Sign-Up:**
   * Secure authentication system allowing users to create and manage their accounts.
   * Options for authentication via email/password or third-party services (Google, Facebook).
4. **Language Selection:**
   * A feature to select a language to learn from the available options.
   * Ability to switch languages at any time, saving progress independently for each language.

**Diagrams:**

Class Diagram



**Architecture:**

The architecture that we are going to be using is a 2-tier server client architecture. We chose this architecture due to the nature of the application that we are developing. If we were to publish or expand the application, we would want to create a 3-tier version. Where one serves acts as a user requests, with 2 more database servers that contain the user data and languages for download.

The reason we are doing 2-tier is due to the simplicity that it gives us as beginner developers. We only need one server for user information storage. And we will be hard coding in the handful of languages that will be in the app. However, the design will allow for additional languages if we desire to add more.

How the 2-teir server client application is applied to the code. User data will be stored on the firebase server. The username, email, password, and progress will all be values stored on the firebase server. When the user performs a login, the client will call the server and retrieve the user data. When the user completed a task the progress will be saved, and the server will be updated to the new progress.

**Front End**

Technology:

* Flutter/Dart for a cross-platform application ensuring uniformity and performance on both iOS and Android.

Components:

* User Interface: Simple and intuitive design to facilitate easy learning.
* Local Storage Management: Efficient handling of user data and preferences locally.

**Backend**

Technology:

* Firebase, which provides a full suite of backend services including database, authentication, and cloud functions, simplifying backend setup and scalability.

Components:

* User Management: Firebase Authentication for handling user sign-ups, logins, and profile management.
* Data Management: Utilizing Firebase for real-time data storage and management.
* Authentication Services: Managed through Firebase to ensure security and seamless integration.

**Database**

Technology:

* Firebase Firestore, a NoSQL database ideal for handling varied data types and facilitating real-time operations.

Structure:

* User Profiles: Data related to user identities and preferences.
* Flashcard Data: Storage of all flashcards used within the app, organized by language.
* User Progress Data: Tracking user learning progress, preferences, and interactions.
* Language Data: Information about the different languages offered in the app.

**Containerization and Orchestration**

The current application does not utilize Containerization, Virtualization, or Orchestration. However, future iterations may incorporate these technologies. Here are the potential needs for these future iterations:

Technology:

* Virtualization for running the database.
* Virtualization and Containerization with Orchestration to host a user-friendly web interface.

Components:

* User data and language information will be stored on a virtual server.
* The user web interface will operate within a container, with orchestration to manage load balancing.
* Both the database and containers will run on virtual machines to ensure redundancy and support dynamic scaling.

**DevOps Approach:**

1. **Continuous Integration and Continuous Deployment (CI/CD):**
   * Try to use CI/CD to automate testing and deployment phases of the development cycle. This ensures that any updates to the app's codebase are automatically built, tested, and deployed, maintaining high development speed and quality.
   * Use tools like GitHub Actions for CI/CD processes.
2. **Monitoring and Logging:**
   * Implement monitoring to ensure that the app performs well and remains reliable across all platforms.
   * Use logging to track user activities and system errors, which will be important for debugging and understanding user interactions with the app.

**Testing:** Discuss the testing strategies and methods that will be used to ensure the quality of the software. This should include unit, integration, system, and acceptance tests, along with any test automation frameworks employed.

**Design Patterns:** The primary objective of using the prototyping design pattern is to help with the creation of multiple objects or classes that represent a word. This approach enables a standardized method for integrating these classes into flashcards for quizzing users. Prototyping also supports the stable addition of new words or entire languages and allows for class-based categories, enhancing user preparation for specific interactions in the language. Although the specifics of how data storage and word classification are not set in stone, we may also consider using adapter or strategy design patterns to handle phrases or grammar.

Iterators will be essential for managing the sequence of flashcard lessons. They can enable features such as moving incorrectly answered flashcards to the end of the lesson, providing users another opportunity to answer correctly. An iterator designed to handle this functionality, without repeating correctly answered flashcards, will enhance the users learning process, and ensure efficient progression through lessons.

**DevSecOps:** Explain how security practices are integrated into the development and operations processes. This includes the use of automated security testing tools, regular security assessments, and strategies to ensure secure coding practices and secure deployment environments.

**Databases/Data Strategy:** Detail the database systems used, the database structure, and how data flows throughout the application. This section should also cover data storage, retrieval, management strategies, and data integrity and security considerations.

**System Design:** Provide an in-depth look at the system’s design specifications, detailing how the software will be structured and how its various parts will interact. This should include a thorough description of the system's components, responsibilities, and interactions.

**Cloud Computing:** Describe how cloud resources are utilized in the project, including the choice of cloud service provider, types of services used (e.g., SaaS, PaaS, IaaS), and the architecture of the cloud environment. Also, discuss how cloud computing benefits the project in terms of scalability, reliability, and cost-effectiveness.