End-Term Project Report

Mobile Application Development

Faculty name: **Ms. Aarti Sangwan**

Student name: **Ritesh Dhiman (21CSU332)**

Ayushi Goyal (21CSU410)

Piha Gupta (21CSU473)

Ritik Pawar (21CSU420)

Deepak (21CSU408)

Semester: **6th**

Group: **FS-B**

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2023-24

Project Title

LEARNIT

Table of Contents

1. Project Description

The objective of this project is to create a language learning app called **LearnIt** in Flutter with modules for vocabulary, grammar, and pronunciation practice. The application Include flashcards, quizzes, and gamification elements to boost user engagement. Additionally, the app integrates with inventory management APIs for finding synonyms, antonyms, definition of the word provided by the user and uses state management for real-time updates.

2. Problem Statement

Language learning has become increasingly important in a globalized world. Traditional methods often lack engagement and flexibility, which can hinder progress. To address this, we aim to create an innovative language learning app called LearnIt, leveraging modern technologies and interactive methods to enhance the learning experience.

LearnIt (Education): Create a language learning app in Flutter with modules for vocabulary, grammar, and pronunciation practice. Integrate audio resources or third-party language learning APIs (if applicable).Include flashcards, quizzes, and gamification elements to boost user engagement. Employ Provider or Bloc for state management in handling user progress and scores.

3. Analysis

3.1 Hardware Requirements

- Processor: Intel Core i3 or higher
- RAM: 8 GB or higher
- Storage: 50 GB of free space
- Smartphone: Android 6.0 or higher / iOS 11.0 or higher
- Internet Connection: Stable broadband connection

3.2 Software Requirements

- Operating System: Windows 10 or higher / macOS / Linux
- Flutter SDK
- Dart Programming Language
- Integrated Development Environment (IDE): Android Studio / Visual Studio Code
- Firebase (for backend services)
- API Services: RESTful APIs for inventory management
- Version Control: Git

4. Design

4.1 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps

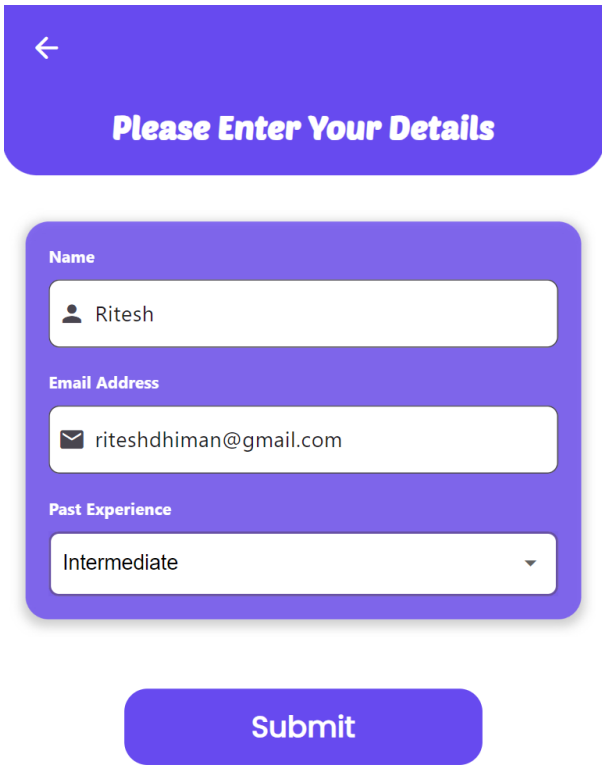
Get Started Screen

The **Get Started Screen** is the main page of the application. It contains a simple Elevated Button to proceed to the next screen that is the User Details Screen. This page serves as the entry point to the application.

User Details Screen

The **User Details Screen** is aimed to get the user details which are name, email and past language experience. The name is stored in with the help of Provide to be used again in the proceeding pages.

Form Validation is also used to validate the entry made by by the user and only allow continuation to the next page after proper validation.

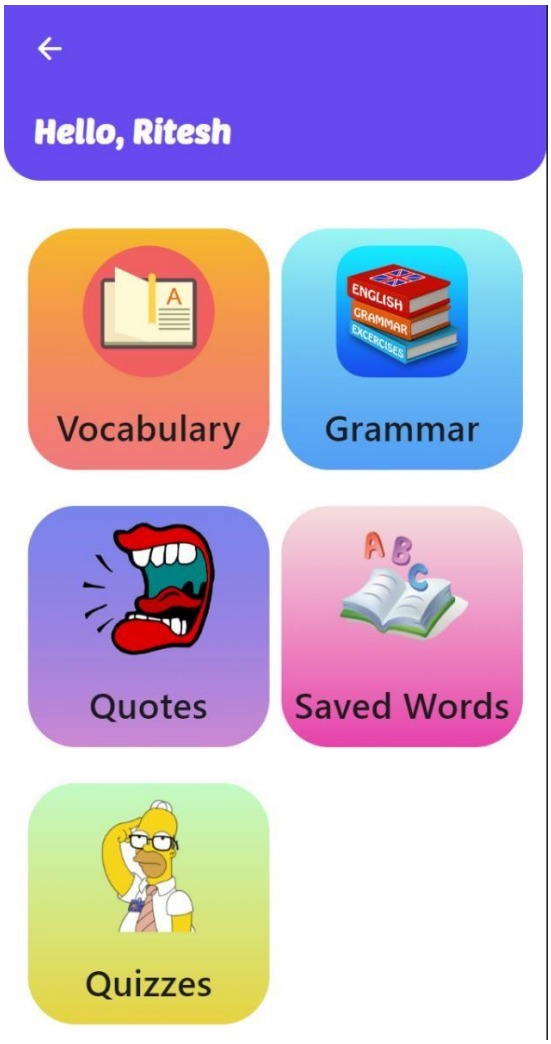


Main Screen

The **Main Screen** is the page from where the User can navigate to other pages. This page essentially contains all the core functionalities of the application.

Main screen contains cards to different functionalities which are :-

1. Vocabulary Page
2. Grammar Page
3. Saved Words
4. Quotes Page
5. Quizzes Page



Vocabulary Screen

The **VocabularyScreen** displays **Word of the day** which is obtained as a response of **Random Word API.**

An option is also provided to save the word for future reference. Provider State Management is used for this where the word is added in a List variable which has global accessibility.

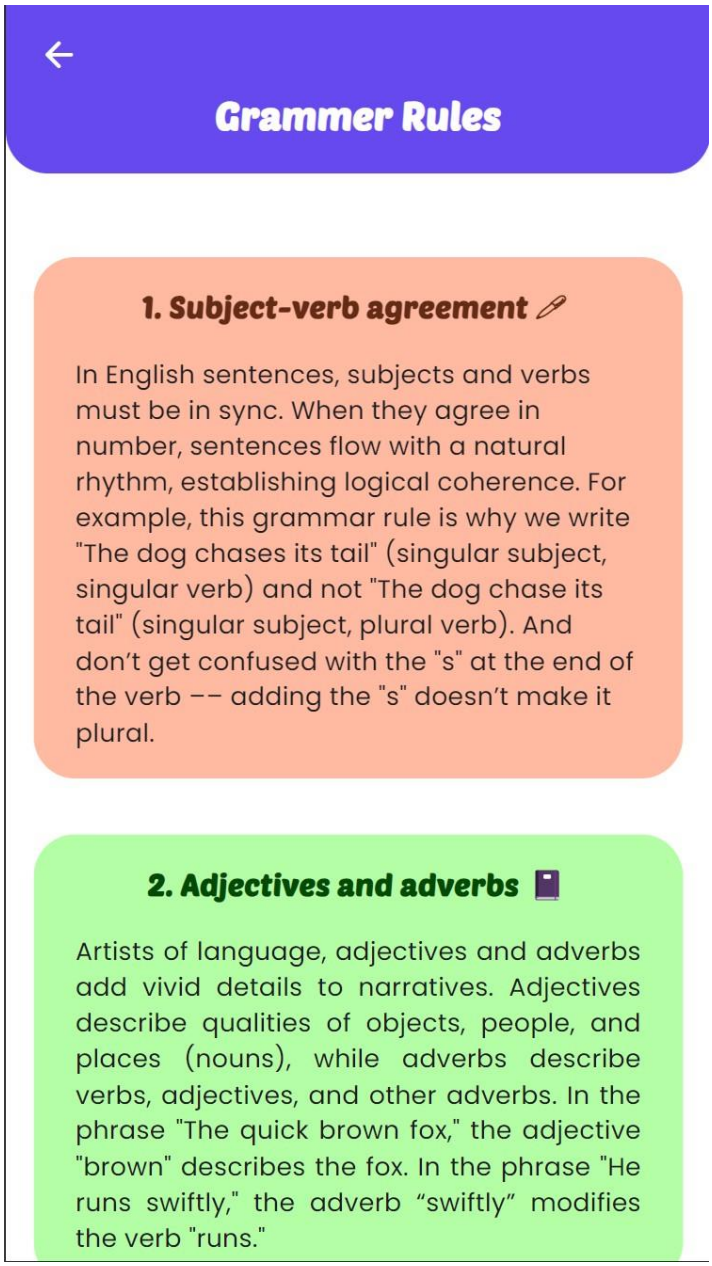
Search Functionality is also provided which gives 3 results :-

1. Word Definition
2. Synonyms
3. Antonyms

By default the definition, synonyms and antonyms are displayed for word of the day.

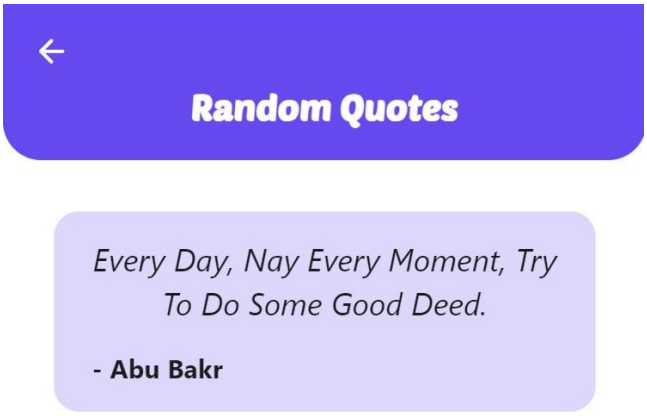
Grammar Screen

The **Grammar Screen** provides the rules of grammar that one should be aware and always follow as the course of learning the language.



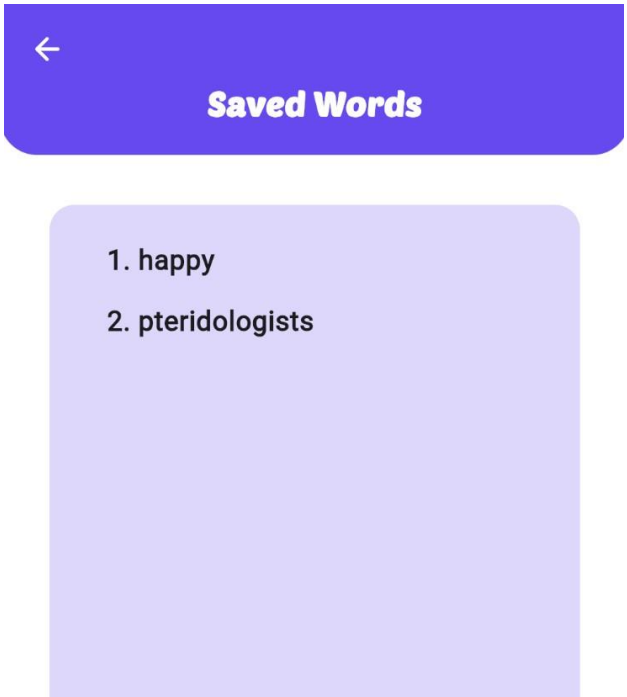
Quotes Screen

The **Quotes Screen** displays a random quote every time along with the author of that quote.



Saved Words Screen

The **Saved Words Screen** displays the words that were saved by the user on the vocabulary page. The saved words are fetched using the provider the Provider which were saved in a List variable.



Vocabulary Quiz Screen

The **Quizzes Screen** offers a quiz format to test users' vocabulary knowledge. Each question presents multiple-choice answers, with the design featuring:

- A title bar matching the app's color theme.
- Question and answer options displayed within cards for a clean, modern look.
- A button to move to the next question, updating the user's score accordingly.
- A dynamic scoring system to track user progress.

State Management

The app employs the Provider package for state management, ensuring efficient handling of user progress and scores. The **VocabularyQuizProvider**, **SavedWordsProvieder** and **GrammarProvider** classes manage name of the user, email of the user, the list of questions, track user answers, saved words and update scores.

VocabularyQuizProvider

Manages vocabulary quiz questions, tracks the current question index, and updates scores based on user answers.

GrammarProvider

Handles grammar quiz questions, maintains the current question index, and calculates the score.

SavedWordsProvider

Used to keep a list of saved words as a List on the basis of user input.

External API Integration

Random word API is integrated to generate random words which are displayed as word of the day. This random word received as the response from the API is used to fetch more details by passing it to another API called **Datamuse API**.

The **Datamuse API** is integrated to fetch synonyms, antonyms, related words, and definitions for vocabulary learning. This integration allows the app to provide rich language data without needing an API key, making it accessible and easy to use.

The **Dummyjson API** is integrated to fetch random quotes along with the author of the quote.

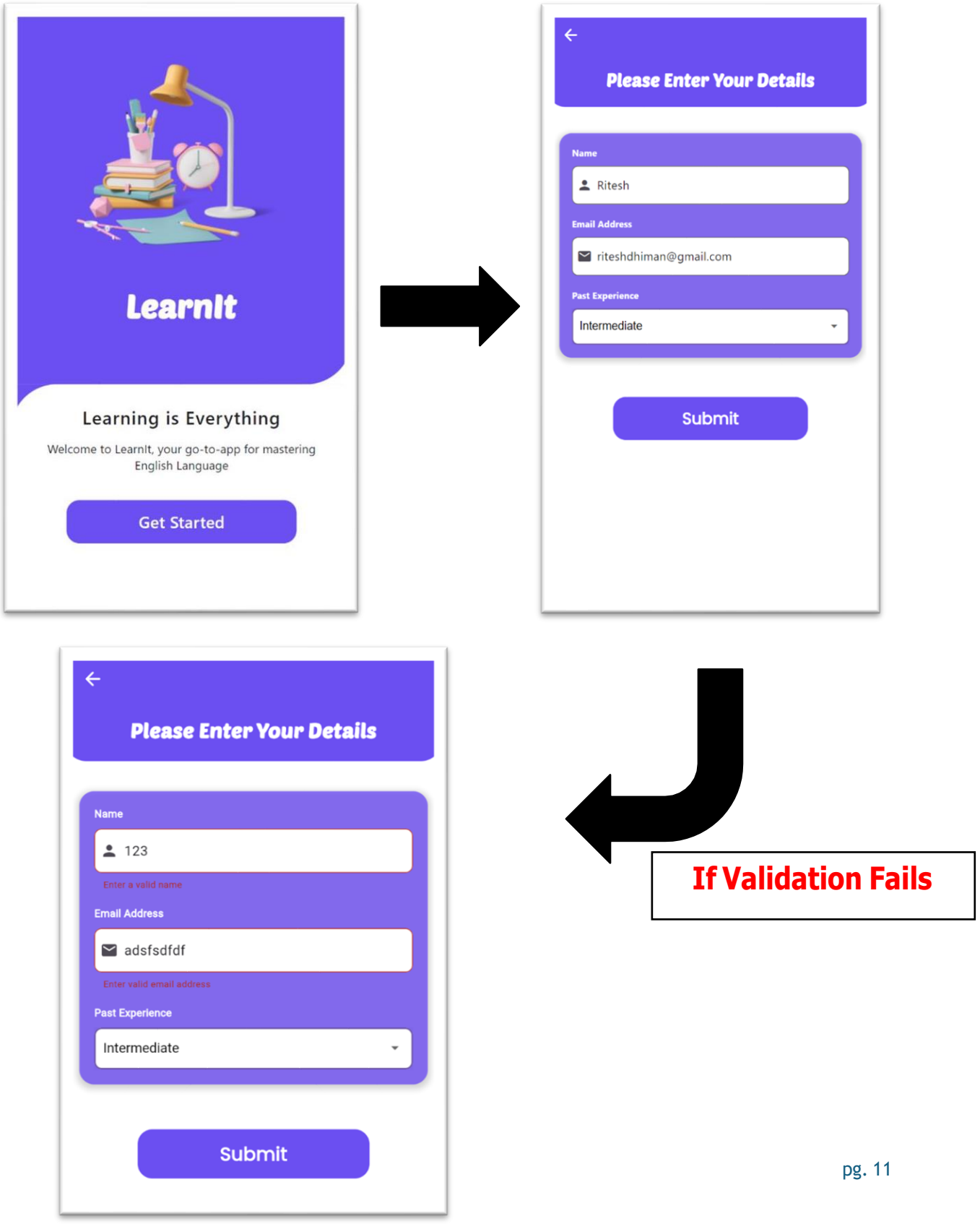
Algorithmic Approach

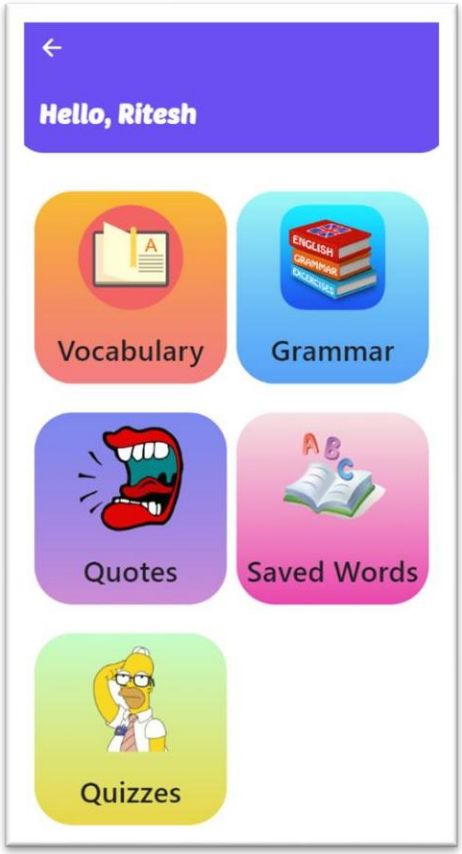
The Language Learning App utilizes several algorithms and systematic approaches to handle different functionalities such as fetching data, managing user progress, and displaying quizzes. Here is an overview of the algorithmic approaches used:

1. Fetching Vocabulary Data**:**
    - Fetch synonyms, antonyms, related words, and definitions from the Datamuse API.
    - Store the fetched data locally for display.
2. Managing Quiz Questions**:**
    - Store quiz questions and options.
    - Track the current question index.
    - Check user answers and update scores accordingly.
3. User Progress Management**:**
    - Use Provider for state management.
    - Update user scores based on their answers.
    - Reset scores and question indices as needed.

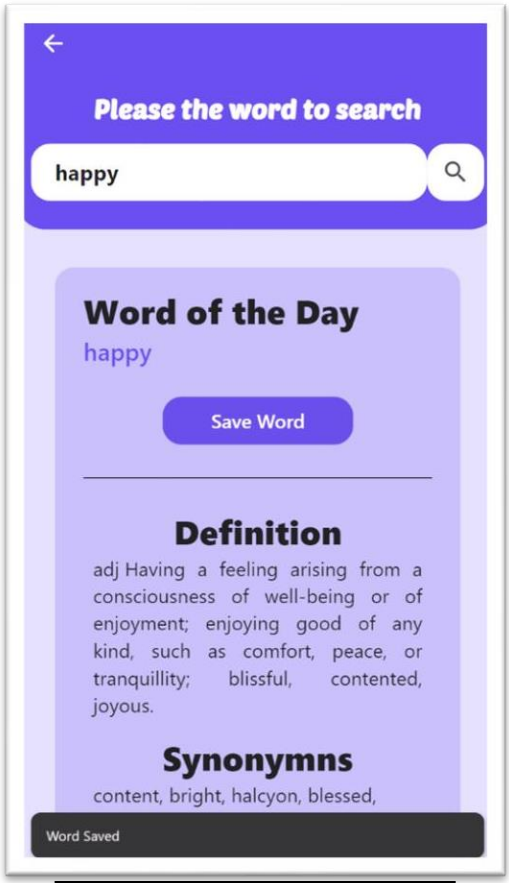
5. Output (Screenshots)

Stepwise Output

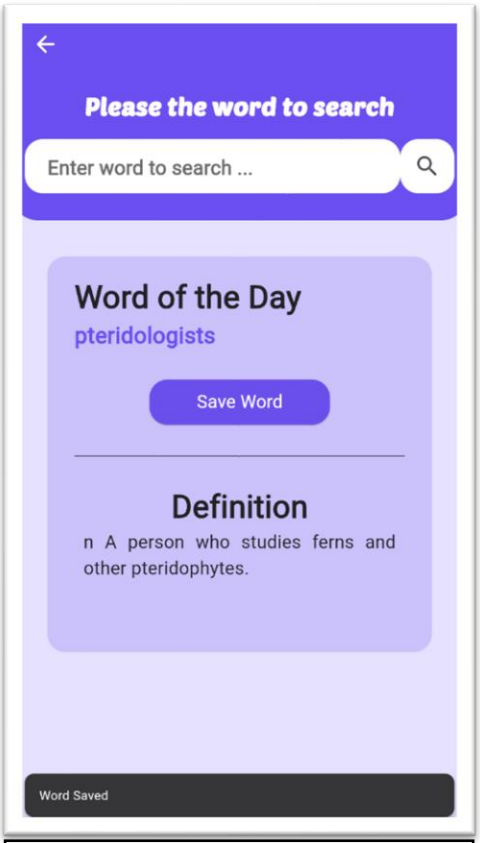




Vocabulary Screen

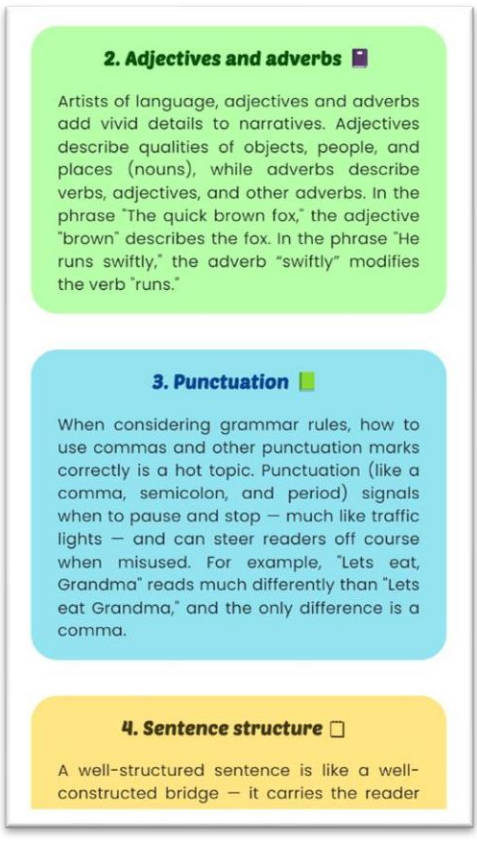


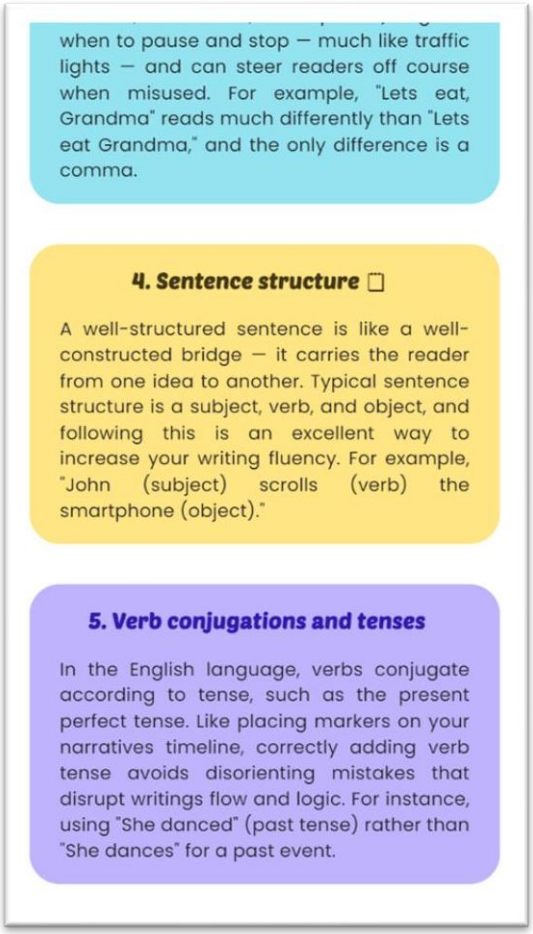
Saving Word



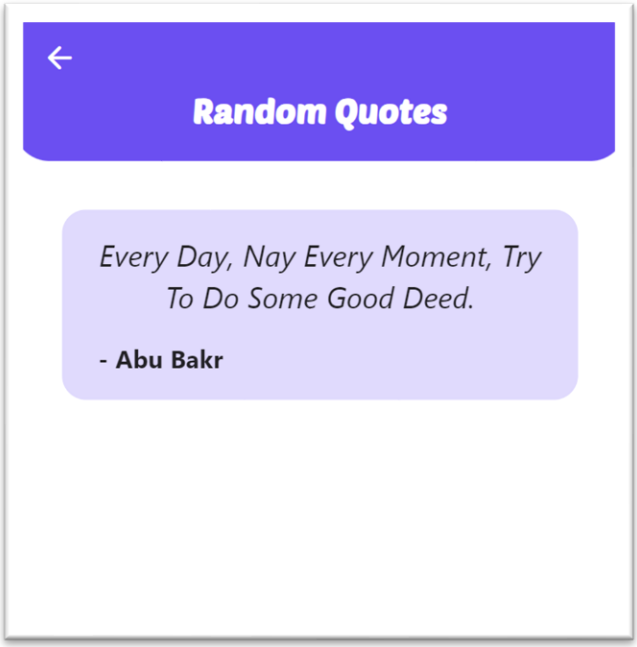
Saving Another Word

Saved Words Screen

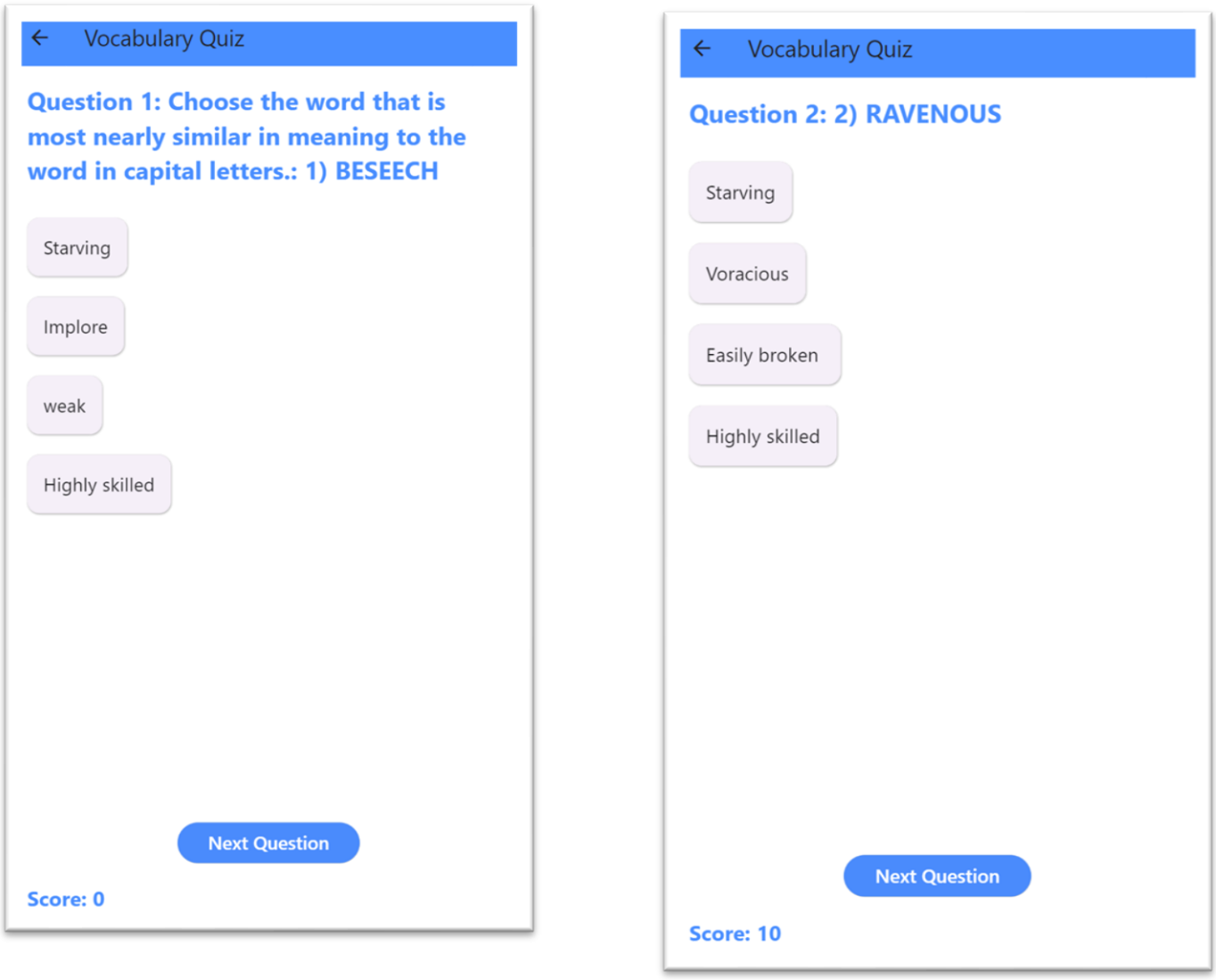




Grammar Screen



Saved Words Screen



Quizzes Screen

6. Conclusion and Future Scope

The Language Learning App is a comprehensive tool designed to enhance vocabulary, grammar, and pronunciation skills. By integrating quizzes, and gamification elements, the app provides an engaging and effective learning experience. The use of the Datamuse API enriches the app with extensive language data, while state management with Provider ensures smooth tracking of user progress. Overall, the app combines educational content with interactive features, making language learning both fun and effective.

Future Scope

1. Enhanced Pronunciation Practice:
    - Integrate speech recognition for real-time feedback.
    - Add more audio examples and exercises.
2. Expanded Vocabulary Database**:**
    - Include additional language resources and APIs.
    - Provide more contextual examples and usage scenarios.
3. Adaptive Learning:
    - Implement machine learning to personalize the learning experience.
    - Dynamically adjust the difficulty level of quizzes.

GITHUB REPOSITORY LINK

https://github.com/ncu-piha/Flutter-Projects/tree/main/Learnit-App