Institute for Research & Executive Education

COMPUTER SCIENCE AND ENGINEERING

RESEARCH & DEVELOP IT ENGLISH MOBILE APPLICATION

A project submitted in partial fulfillment of

the requirements for the Degree of

Bachelor of Computer Science and Engineering

from

VNUK Institute for Research & Executive Education

By

Nguyen Hoang Linh *18020004*

SUPERVISOR

Tran The Vu

Dang Thi Phuong Thao

Faculty of Engineering and Information Science

Da Nang, 2022

DECLARATION

I, Nguyen Hoang Linh, declare that this project title and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a BSc degree at VNUK Institute for Research and Executive Education.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at VNUK Institute for Research and Executive Education or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Nguyen Hoang Linh,

18020004 - 18CSE

CERTIFICATE

I do hereby declare that the research works embodied in this thesis/project entitled "English for IT" is the outcome of an original work carried out by Nguyen Hoang Linh under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of BSc in Computer Science and Engineering.

Dr. Tran The Vu **Dang Thi Phuong Thao**

Dedicated to

My Parents and my Family

ACKNOWLEDGEMENT

"First, I wholeheartedly dedicate this graduation project to my dear parents and my sister for their endless love, limitless support and continuous belief and encouragement.

Second, I would like to express my deep gratitude to my supervisors Dr. Tran The Vu and Mrs. Dang Thi Phuong Thao for their patient guidance, valuable and constructive suggestions, enthusiastic encouragement and useful critiques through the planning and the development of the whole project.

Last but not least, I specially devote the work of this graduation project to all the respectable and honorable teachers and colleagues who tremendously contributed to my academic professional progression and self-Development."

Da Nang, 2022 *Nguyen Hoang Linh*

ABSTRACT

This inspiration for the project came from passing the “English for IT” subject at Institute for Research & Executive Education. We wanted to put my newly acquired expertise to good use by turning it into a product, “English for IT”.

Motivation came from assisting juniors efficiently passing the subject and mastering all terminologies in IT in English. Students in Viet Nam wouldn’t be nervous when facing these strange terminologies in IT. Moreover, they can improve English skills through speaking, listening, reading, and writing skills, which would be good for them in the future. Last but not least, this app also addresses users that plan to improve English in IT as well as join an IT major, but own few English knowledge and skills.

In view of this, I want to initial an application to synthetic learning lessons and assist them to review learning knowledge through reviewing that knowledge by quizz, guess meanings, guess words, e.t.c.

Everything is continuously updated in the system. I used Firebase as my database, Dart as Backend and Flutter for the website.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

The English for IT Application project involved assisting students in the English for IT subject. This section of the report provides the students an online platform to learn this subject at the college. It also clarifies the project's objectives and provides a recap of the upcoming chapters.

1.1. PURPOSE OF THE SYSTEM

The project's goal is to assist undergraduates to master English in IT as well as major knowledge to improve English skills. The application is separated into two sections: the front end and the back end. Flutter is used to provide a user interface with which people may directly interact. Meanwhile, Dart is the programming language I chose and created for my back-end.

To deliver the greatest user experience, my app features a clean and straightforward UI. It is important to make content on an app available in the language of the clients in order to minimize misunderstandings and build their trust in the product.

User-centered design is critical. The goal of a user data platform site is to provide them with easy yet comprehensive information. The major goal of the Application should be to meet the expectations of the consumers.

1.2. SCOPE OF THE SYSTEM

One of the project's key aims was to initial an online platform for users to manage their study results. The system can be authenticated, saved lessons, learned new lessons, recommended lessons, learned throughout games and calculated score, managed profile, and chatbot. Following that, the user as a VNUKers may not only improve their English skills but also comprehend new jargon in IT. .

1.3. OBJECTIVES AND SUCCESS CRITERIA OF THE PROJECT

Objectives:

● Automatically – The system will automatically update, and save the results to the database system.

● User Experience - The system was created with the user's comfort in mind. I hope that the user-friendly design and clean look will meet your expectations.

● Free - Users can download and use my app without any fees.

Success criteria:

● As previously said, my primary goal is to come to client loyalty and ensure their experience. To do this, my framework must be user-friendly, accessible, and straightforward.

1.4. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

1.5. PROJECT PLAN

1.5.1. Gantt chart

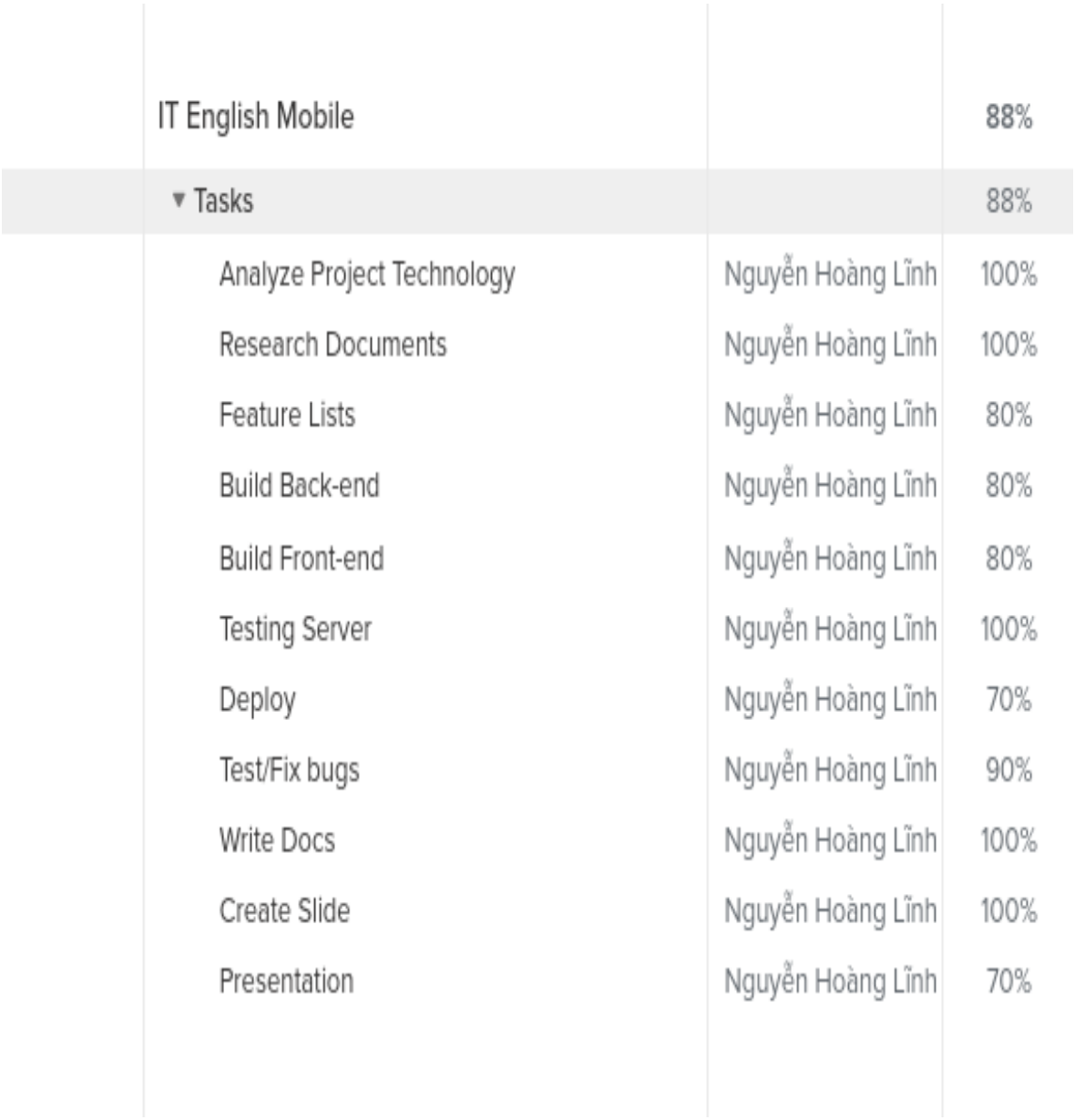

Figure 1.1. Gantt Chart

1.5.2. Project Phase Distribution

Planning & Analysis	Design	Implementation	Testing
30%	20%	40%	10%

Table 1.1. Project phase distribution

Planning & Analysis: This is my most crucial point of time in the syllabus. Therefore, I have to build a standard plan while analyzing my targets clearly and simultaneously. Moreover, time management must be logical and effective.

Design: My targeted users are un-graduates and a basement is on the application. Hence, my aim is to create a suitable design that fits with those requirements.

Implementation: I spent most of my time in this step on coding, fixing bugs and putting my system into operation.

Testing: I spent only 10 percent of my process on this stage for testing the system. Although I am not professional, I still try my best in this step.

1.5.3. Other Tables

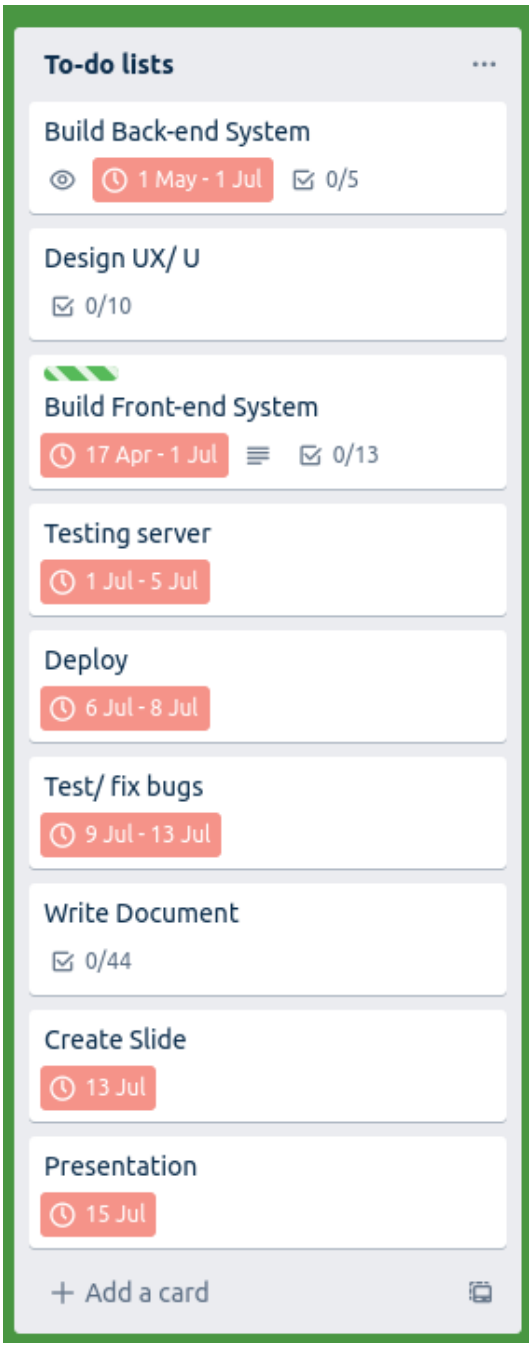

Figure 1.2. Trello Board

1.6. OVERVIEW

CHAPTER 2: METHODOLOGY & THEORETICAL BACKGROUND

2.1. METHODOLOGY

2.1.1. Software Development Life Cycle

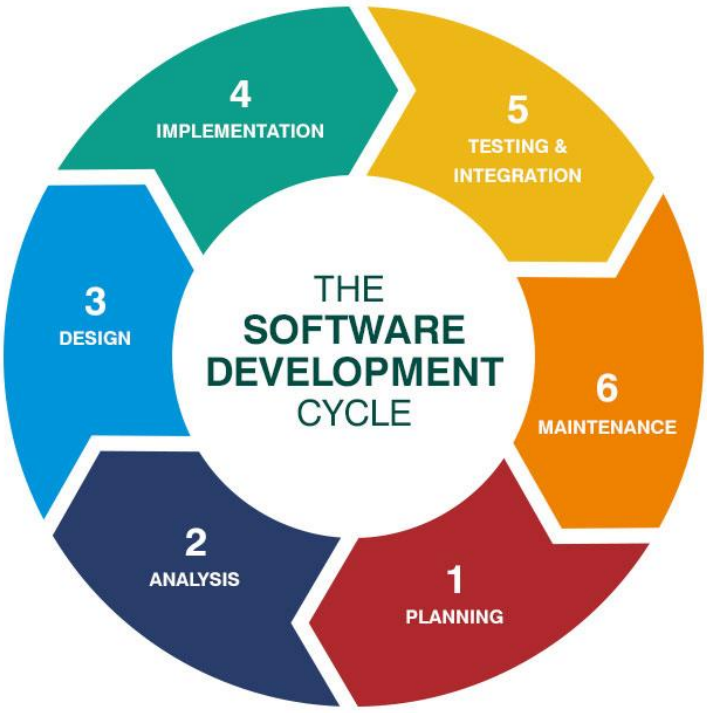

Figure 2.1. Software Development Life Cycle

Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time. SDLC includes a detailed plan for how to develop, alter, maintain, and replace a software system. SDLC involves several distinct stages, including planning, design, building, testing, and deployment. Popular SDLC models include the waterfall model, spiral model, and Agile model [1].

Benefits of Software Development Life Cycle

The key reason behind adopting a clear SDLC is control over the development process. A working plan, conflict management between participants, and budget management are other big advantages of software development life cycle [2].

Thus, software development life cycle allows for:

- Having overarching control over. software development process.
- Improving resource management and cost-effectiveness.
- Gives teams a clear action plan.

- Improves cooperation between participants.

How SDLC Works?

SDLC works by lowering the cost of software development while simultaneously improving quality and shortening production time. SDLC achieves these apparently divergent goals by following a plan that removes the typical pitfalls of software development projects. That plan starts by evaluating existing systems for deficiencies.

Next, it defines the requirements of the new system. It then creates the software through the stages of analysis, planning, design, development, testing, and deployment. By anticipating costly mistakes like failing to ask the end-user or client for feedback, SLDC can eliminate redundant rework and after-the-fact fixes.

It's also important to know that there is a strong focus on the testing phase. As the SDLC is a repetitive methodology, you have to ensure code quality at every cycle. Many organizations tend to spend little effort on testing while a stronger focus on testing can save them a lot of rework, time, and money. Be smart and write the right types of tests [3].

Software Development Life Cycle Stages

Stage 1 — Planning

"What do I want?"

At the start of software development life cycle stages, the need to attract the most talented and experienced engineers inevitably arises. By considering the demands of the client, a skillful crew can create a reliable foundation for the rest of the software development life cycle phases.

This stage has a number of important steps. First, the team working on the project conducts a preliminary analysis to discover the aims and issues of their client. Based on the results, they propose a set of possible solutions, with a budget for each.

Stage 2 — Defining Requirements/ Requirements Analysis

"What are the current problems?"

The sequence of software development life cycle stages continues with a deep dive into the requirements, after the client has chosen a software solution. The team analyzes documents related to the project, evaluates the client's existing ecosystem. This is one the software development life cycle steps that some consider transitional, performing it

alongside planning and requirements analysis.

Stage 3 — Designing Product Architecture

"How will I get what I want?"

Having fully analyzed the client's requirements on previous software development life cycle stages, the developers create several product architectures and show them to the client.

The chosen architecture is then finalized in a Design Document Specification (DDS) and evaluated by all sides in terms of risks, operational reliability, universality, and cost-effectiveness. This architecture becomes the foundation for all next stages of the software development life cycle and the software product in question.

Stage 4 — Developing the Product

"Let's create what I want."

One of the most critical software development life cycle phases, this one aims at producing working code and showing results to the client. The development takes the majority of time in any project. Often, the project exceeds the initially estimated time: the client might consider adding something to the project scope in the course of the development process. In some SDLC models, the product can change in the process of the development.

Stage 5 — Testing the Product

"Did I get what I wanted?"

After all the preparations on previous stages of the software development life cycle are completed, quality assurance engineers start scouting for bugs. Testing is another crucial step among the SDLC life cycle phases as it allows for fixing critical problems before they will lead to critical losses.

Even though testing procedures may appear on other software development life cycle stages, as a separate one, testing provides a detailed map of breakdowns that need to be fixed.

Stage 6 — Deployment & Maintenance

"Let's start using what we got."

After the product is release-ready, the next action in the order of software

development life cycle steps is to display the finished software solution to the client.

To guarantee the proper work of the finished digital solution in the future, after all the SDLC life cycle phases are completed, the client can order general product maintenance to fix different issues (which inevitably emerge with any modern software product). This step completes the software development life cycle [4].

Wireframe and prototype

A wireframe is a layout of a web page that demonstrates what interface elements will exist on key pages. It is a critical part of the interaction design process. The aim of a wireframe is to provide a visual understanding of a page early in a project to get stakeholder and project team approval before the creative phase gets under way. Wireframes can also be used to create the global and secondary navigation to ensure the terminology and structure used for the site meets user expectations [6].

Wireframing is a practice used by UX designers which allows them to define and plan the information hierarchy of their design for a website, app, or product. This process focuses on how the designer or client wants the user to process information on a site, based on the user research already performed by the UX design team [7].

Software prototyping refers to the process of visualizing a software product before it has been created. Creating software from scratch requires a great investment in the form of time, money, and effort. That is why most clients prefer to have a visual prototype developed before work is put into the development of the actual product. The prototype acts as a 'model' closely replicating the appearance, and sometimes the functionality, of the product that the client has in mind [8].

Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation [9].

Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design [9].

I use Figma to design a prototype for my website.

Figure 2.2. Figma

Figma is a web-based graphics editing and user interface design app. You can use it to do all kinds of graphic design work from wireframing websites, designing mobile app interfaces, prototyping designs, crafting social media posts, and everything in between [10].

Source code control

Figure 2.3. Github

In this part, I'll discuss how I handle our source code and ensure that it's free of conflicts and errors.

Git is the version control system I am using in this project, and I've stored it on the GitHub service for collaboration. GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere [11].

I provide a set of guidelines to ensure that our team follows convention and that our source code is clean and safe:

- Create branches for production and development builds programs.
- Update code and check conflicts before editing code in the local device before

each commit.

- Can make very small commits, as long as they conform to the previous rule.
- Commit as soon as something is finished.
- Don't commit something that breaks the repository's code or makes it unusable.
- Provide commit comments with prefixes and meanings (ex. "+" is created, "-" is update).
- Commit code relevant to exactly one thing (ex. fixing a bug, implementing a feature).
- If there is something important to update, verbally interact with team members.

2.2. THEORETICAL BACKGROUND

2.2.1. Overview of web service technology

Our mobile application was built by:

- Flutter[12]

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase.

First described in 2015, Flutter was released in May 2017.

Every cross-platform framework is a dream of any business. Because if you separate two platforms native iOS and Android apps, it will be more expensive to develop and maintain. Just one single code base, Flutter allows to create attractive apps for both operational systems.

But, when comparing it with other competitors, Flutter is a bit different in some key aspects. Now, let’s have a closer look at the benefits of Flutter software development.

Why use Flutter?

- Faster code Writing[12]
  - *Hot reload.*

Typically, iOS and Android developers need to write code, then wait for it to compile, and be loaded on the device before seeing changes. But, with Flutter’s hot reload, they can check the effects immediately or without delay.

In other words, your development team, together with QA engineers and designers, can cooperate more effectively, make quick changes, and see outcomes as well. This is exactly what makes Flutter app development time faster than others.

○ *Ready-to-use widgets.*

Besides, a customizable kit of widgets make it easy to create applications of any complexity. No matter the screen size,w widgets are fast, reliable, and extensible. In addition, you can use ready-made widgets with a large set of Material and Cupertino. it is also possible to flexibly work with animation and gesture processing. Thus, the entire process is faster and simpler.

● Large Community support[12]

Maybe you don't know, Flutter is developed by tech giant — Google so that Google consistently supports Flutter developers with frequent updates and issue fixes.

Moreover, when using Flutter, you can notice a significant increase in performance compared to similar technologies. The Flutter app development community has been growing from strength to strength.

Since its first release, Flutter has 81,200 Stars on Github. Due to the friendly developer community, experts and beginners are ready to share their knowledge and experience. Moreover, Google organizes several events, helping both startups and established businesses to catch up, discuss, or learn how Flutter apps help to grow businesses.

● Similar to native app development[12]

The Flutter software renders use an internal graphics engine known as Skia. This software allows fast and well-optimized development than most other mobile app frameworks.

A Flutter application will be ultimately indistinguishable from the native app. Because Flutter doesn't rely on any intermediate code representations or interpretation.

Additionally, the Flutter development team can work seamlessly on both iOS and Android. With highly advanced and custom UI designs, it is an excellent choice for mobile applications.

● Own rendering engine[12]

Obviously, Flutter requires a pretty wonderful user interface. The problem with many cross-platform solutions is that they look the same on iPhone and Android.

But what about the companies that need to use Material Design for Android and Human Interface for iOS?

For such companies, Flutter is the most suitable solution. It is equipped with packages that contain a set of custom widgets for both operating systems.

- Simple Platform[12]

Flutter provides advanced OS features like GPS coordinates, sensor data collection, permission handling, Bluetooth, credentials, and other features in ready-to-use plugins that are supported by Google.

If your app is reliant on an OS-level feature not available as a plugin, Flutter can establish communication between its Dart programming language and the native code using platform channels.

This way, you can implement anything that a native app can do on a Flutter app, with just a little extra effort on the native side.

- Beyond mobile applications[12]

In 2018, the technology appeared as Flutter 1.0. But that is not all, at 2019 I/O conference, Google announced a technical preview of Flutter web. Without any change to source code, it is possible to run Flutter applications on browsers.

In 2019, at fluterInteract, Google announced Flutter Octopus, which allows debugging on multiple platforms simultaneously. From one single codebase, developers can create a solution that will work not only on mobile, web, but also on Mac.

- Good documents and resources[12]
  - Dart is very simple to get started with. If developers have any experience with JavaScript or other C-like languages, they will have no trouble picking up Dart within a day or two.
  - There is a robust plugin manager (pub.dev), which shows how to make use of styles, animations, and possible manipulations.
- Quicker time to market

As a business owner, you may think that one of the biggest Flutter's advantages is hiring one team for both iOS and Android platforms. However, another important benefit of Flutter is fast time-to-market.

For instance, it's hard to predict whether the iOS or Android version will be able to release a product on time. While Flutter works for cross-platform at the same time. Even though there are cross-platform solutions with the same advantages, Flutter still stands out from other frameworks.

● Beautiful UI[12]

This year, Flutter partnered with Google Assistant and Lenovo teams in the most recent contest: Flutter Clock. In fact, I am always amazed by what developers around the world are building with Flutter.

The framework offers a set of distinctive and eye-pleasing widgets. They also continuously improve the libraries with various interface components. As a result, a smaller development team can build a beautiful native app on both platforms much more quickly.

● Cost-effectiveness[12]

Cost optimization is an essential goal for every company. When developing native apps, business owners have to pay twice: one for the iOS version, one for the Android version. They must share the budget with two separate teams.

With the feature of Flutter, you only require one Flutter development team to develop and manage your apps. Hence, it significantly reduces expenses on human resources and shortens the time required to complete the development tasks.

● Dart[13]

Dart is a client-optimized programming language developed by Google and launched in the year 2011. The programming language syntax is similar to Java & C++ and you can use it to build mobile, desktop, server, and web applications. The Dart code compiles to native or JavaScript in order to run in the browser.

Dart's manifesto is that it uses Ahead Of Time (AOT), which compiles the code swiftly into native in a predictable way. Developers thus enjoy the breeze of developing

apps using Dart for Flutter. This feature makes it favorable for developers to code accurately and checks for the response, thereby delivering the program instantly.

Also, if developers want to develop animations and transitions, it is very easy and quick with Dart. These animations run at the speed of 60 frames per second. Also, Dart is capable of doing object allocation and garbage collection without any locks and does not need preemptive scheduling and shared memory. Also, when it comes to creating a bridge between realms, you do not need it in Dart, as it compiles Flutter apps to native code. The startup time is also faster and quick.

Why does Flutter use Dart? [13]

Flutter uses Dart as Dart allows Flutter to avoid the need for a separate declarative layout language like JSX and XML. The layout of Dart is declarative and programmatic, and it makes it easy for developers to read and visualize it very quickly and effortlessly. In addition to that, it makes it easy for Flutter to provide additional tooling as the layout is in one language and commonplace.

Another important reason why the duo is popular is that Dart, if required, uses the Just In Time compilation. This drastically reduces the time of development and responds faster.

Most of the Dart features are similar to that of static and dynamic languages, and it makes it easy for developers to learn and understand Dart easily and quickly.

If you see closely, not all the features mentioned above are extraordinary and superior. However, when these features are combined, they serve as the best implementer for Flutter. It is not an overstatement to say that Flutter app development would not be as smooth and streamlined without Dart for the Flutter app development company.

Let’s discuss some of the additional characteristics of Dart that explain a strong relationship between Flutter and Dart.

Compilation and Execution[13]

If you look at the history of web application development, there were a total of two types of programming languages: static languages and dynamic languages. Static languages in which variables are typed statically, such as C and Fortran, at compile time.

Dynamic languages are ones in which the variable can be changed at run time, such as JavaScript. Also, static languages were used to write native machine code that the hardware could run, whereas an interpreter executed dynamic languages, and it did not require any machine language code.

With time, things have advanced and become somewhat complicated too. The developers introduced virtual machines that can mimic a hardware machine in software. They are called advanced interpreters. They make it easier for developers to port a language to new and advanced hardware platforms. And the input language executed on a virtual machine becomes an intermediate language.

Also, the time is now for Just In Time compilers and Ahead of Time compilers that run during the execution of the program and compile it on the fly.

You measure the development cycle by the time between making a change or modification to the program. Despite that, the time taken to execute the program also counts to see the result of the change. When it comes to AOT compilation, development cycles go slow. You might find it a limitation for AOT compilation. But the accuracy and the program can run predictably without pausing for analysis and compilation at runtime. In addition to that, it is quick and easy to execute compiled programs.

On the other hand, when it comes to JIT compilation, it offers much more development cycles. But the execution is slow and jerkier. Also, the starter time for JIT compilers as it has to give some time for analysis and compilation of the program. The main issue is user experience here. Users will abandon the app if it takes more time to execute and load.

Compiling and Executing Dart[13]

Dart developers understood this issue before they started working on Dart Development. Hence, they have done unparalleled work on advanced compilers and virtual machines for dynamic and static languages. The result was awesome. Dart is phenomenally flexible when it comes to compilation and execution.

If you are looking for a language that is excellent at compiling both AOT and JIT, Dart is your answer. The reason why Flutter with Dart makes a great pair is the significant advantages Dart brings by being capable of supporting both types of compilation.

During the development, Dart performs JIT compilation with its fast compiler. Besides, when the app is ready for release and deployment, it is compiled AOT. Dart helps achieve the two most crucial aspects- 1) superior and rapid development cycles; 2) quick and fast Execution and startup times.

Not just that, Dart can also be compiled into JavaScript, and it allows Flutter developers to use the code between mobile apps and web apps. Using the Dart compiler, you can compile the Dart program into native code.

It is a standalone and highly effective virtual machine that uses Dart language as its intermediate language and acts as an interpreter. Also, you can compile and transpile Dart into other languages too. It is not just flexible and versatile but extremely fast too.

Additionally, Dart has the potential to prevent jank, caused because of a variety of reasons. The predictableness of Dart gives you better control over the application. So, you can easily avoid Janks in the Flutter code.

Eliminates XML Files[13]

Because Dart's layout is easy to read and visualize, Flutter doesn't require a separate language for its layout, like XML. Flutter also has advanced tooling as everything is in one language and in one place.

Eliminates the JavaScript Bridge[13]

Dart's direct compilation and execution into native code without an intermediary bridge (e.g., JavaScript to Native) allows for hitchless animations on a user's gadget, as well as smoother user interfaces running at 60fps. Additionally, Dart is able to perform object allocation and garbage collection without the need to acquire locks.

Scaffolding[13]

The scaffold in Flutter is noticeably different from the scaffolds in iOS or React Native or Android UI. One key difference is that it stretches to fill in the space available. This typically results in the scaffold taking up the entire screen of the device window.

The scaffold has APIs for an app bar, drawers, floating buttons, and the bottom sheets so that the primary content design's graphic interface can be enforced.

Incorporates HTTP[13]

The Dart programming language uses an abstraction called Future to host HTTP resources from the internet. The http.get() function returns a Future that contains a Response. The Future class is used in Dart to deal with asynchronous functions, and a future object reflects a possible value or error that will be visible in the Future at some point.

Flutter and Dart are friends forever![13]

As you saw in the post, there are some excellent benefits Dart offers to the developer. Developers can use it as a compiler and executor for developing superior, comprehensive, robust, and fast apps. Besides, Flutter and Dart are both Google's products, which makes them the deadliest duo, extending support for other Google products as well. So, companies and entrepreneurs, if you want to build something unique, hire experienced Flutter developers before someone else hires them.

Firebase[14]

Firebase is a popular BaaS platform from Google. It provides a broad range of features and components that help mobile and web application development. Firebase is a NoSQL database platform program that follows the JSON protocol for data storage actions.

2.2.2. Web service architecture

- Clients

Flutter is a Google framework used to build natively compiled applications for mobile, web, desktop and embedded devices with a single codebase. Like React/React Native projects, I work with Atomic System/Components to make our widgets more reusable and organized.

When I develop an app, I usually need to communicate with an API or some external resource, such as a local database and sensors. So, the interfaces folder is the place where people store the files that will help them with this communication. Middlewares are responsible for providing an extension to an event or action from your application. This can be an HTTP request interceptor, a logger or something similar. Here, I'll use an hexagonal architecture approach in a way that allows us to separate our application layer from the domain layer and the infrastructure layer. With this architecture, the code will be

more organized, but I also need to modularize it to make it easy to add a new feature, modify the old ones and make the code more reusable. For example, my app is separated into independent pieces which are able to work individually. Moreover, this approach supports testability, which is a good point for building a reliable project and improving code quality.

2.2.3. Understanding Monolithic applications

In software engineering, a monolithic application describes a single-tiered software application in which the user interface and data access code are combined into a single program from a single platform.

A monolithic application is self-contained and independent from other computing applications. The design philosophy is that the application is responsible not just for a particular task, but can perform every step needed to complete a particular function.[1] Today, some personal finance applications are monolithic in the sense that they help the user carry out a complete task, end to end, and are private data silos rather than parts of a larger system of applications that work together. Some word processors are monolithic applications.[2] These applications are sometimes associated with mainframe computers.

In software engineering, a monolithic application describes a software application that is designed without modularity. Modularity is desirable, in general, as it supports reuse of parts of the application logic and also facilitates maintenance by allowing repair or replacement of parts of the application without requiring wholesale replacement.

Modularity is achieved to various extents by different modularization approaches. Code-based modularity allows developers to reuse and repair parts of the application, but development tools are required to perform these maintenance functions (e.g. the application may need to be recompiled). Object-based modularity provides the application as a collection of separate executable files that may be independently maintained and replaced without redeploying the entire application (e.g. Microsoft "dll" files; Sun/UNIX "shared object" files). Some object messaging capabilities allow object-based applications to be distributed across multiple computers (e.g. Microsoft COM+). Service-oriented architectures use specific communication standards/protocols to communicate between modules.

In its original use, the term "monolithic" described enormous mainframe applications with no usable modularity[3]. This, in combination with the rapid increase in computational power and therefore rapid increase in the complexity of the problems which could be tackled by software, resulted in unmaintainable systems and the "software crisis".

2.2.4. Understanding and Visual Studio

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1. CURRENT SYSTEM

English for IT is one of the core skills of software engineers. Because the error code displays in English, if people don't have a solution to handle tasks, the first thing they should do is to copy the error to Google Search to find a solution.

However, most of them are also in English, so if they are not fluent in English, it will be difficult for the developers.

Tools like Google Translate are also good, but they don't translate closely and cause confusion, so people should still be proficient in English.

3.2. PROPOSED SYSTEM

3.2.1. Overview

The system was designed and developed with the goal of providing solutions to other systems' issues.

The system focuses on the market for student's content. The system primarily updates English for IT data, which is from Oxford English for Information Technology for the junior, so that they can pass English for IT subject at the university and improve their English skill.

Furthermore, the app is developed in a simple manner with all important information. The system is also designed to automatically update data on schedule, minimizing administration workload.

3.2.2. Functional Requirements

#	Features	Description
1	**Authentication**	
	Register	Users must register an account with their email and password.
	Login	Users use registered account to login to the system
	Logout	Users can log out after their current account after login.
	User Profile	Users must add their profile with some info: first name, last name, and their photo.

2	**User Management**	
	View User Profile	Users can see their profile after login to their account
	Edit User Profile	Users can edit their profile after login to their account
3	**Skills for Learning**	
	Listening	Users can listen and answer the question
	Writing	Users can write a paragraph and submit
	Reading	Users can read a paragraph and answer the question
	Speaking	Users can speak to the phone a sample sentence on the screen, the system will automatically check their voice.
	Score	After submitting the answer, the system will calculate users' score.
4	**Review Skills**	
	Quiz	Users can make a quiz with some mode: 5 minutes, 15 minutes, 30 minutes, 45 minutes, and 60 minutes.
	Practice	Users can guess meaning words with the following words
	Review	Users can guess the words with its images

Table 3.1. Functional Requirement

3.2.3. Nonfunctional Requirements

	Detail
Externals Interfaces	User Interfaces: - GUI should be simple, clear and have recognizable elements. Software Interfaces: - Application: works well with Android phones.
Quality Attributes	Usability: - The interface must be simple and easy to use. - App's UI is fit for each Android device - Data is clearly displayed. - Links and buttons can be pressed easily. Reliability:

	- The application should be used whenever the internet is on. - The information stored on the database is accurate
Performance	On average, the system takes 1 minute to update the app's data.
Dependability	Security: - Only Admin has full access to the system - Only user logins get their own profile and compare with competitor channels. - Use Firebase Authentication Safety: - The password should be hidden
Support Document	The SRS should be provided for developers which are responsible for implementing and maintaining.
Design Constraints	Time constraints: Must be completed in a certain time Cost constraints: Must be completed with a specific budget Technical Constraints: Limits of technology or available technology

Table 3.2. Non - Functional Requirement

3.2.4. System Models

My system will be described following the below scenarios:

- The system will also optimize the data structure to accommodate a large number of queries.

- Users accessing the app can log in by Firebase Authentication.

- Users may access the app and learn English for IT by choosing the categories clearly presented on the homepage and checking the answer of every lesson exercise.

- Users can access the profile page and update every time and everywhere.

- Users can practice and review vocabulary with a quiz test in many modes: 5 minutes, 15 minutes, 30 minutes, 45 minutes, and 60 minutes.

3.2.4.1. Scenarios

The dotted horizontal lines define some clear application layers.

In the diagram above, data flows from the outside world, into the services, view models, and all the way down to the widgets.

The call flow goes into the opposite direction. Widgets may call methods inside the view models and services. In turn, these may call APIs inside external Dart packages.

View models do not reference any widgets objects (or import any UI code for that matter). Instead:

Widgets subscribe themselves as listeners, while view models publish updates when something changes.

This is known as the publish/subscribe pattern, and it has various implementations in Flutter. You have already encountered this if you used ChangeNotifiers or BLoCs in your apps.

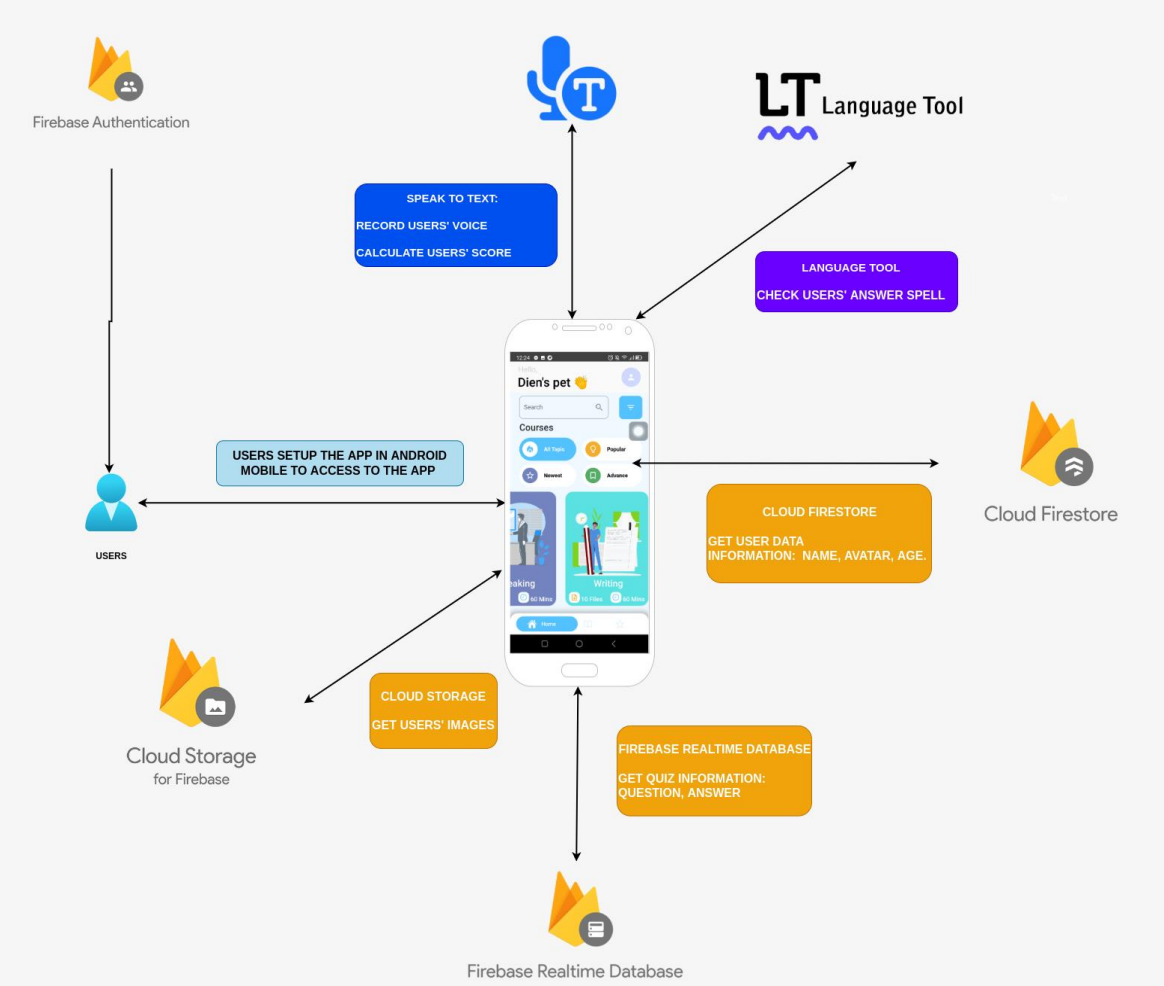

Figure 3.1. Scenarios

3.2.4.2. User case model

3.2.4.2.1. User case diagram

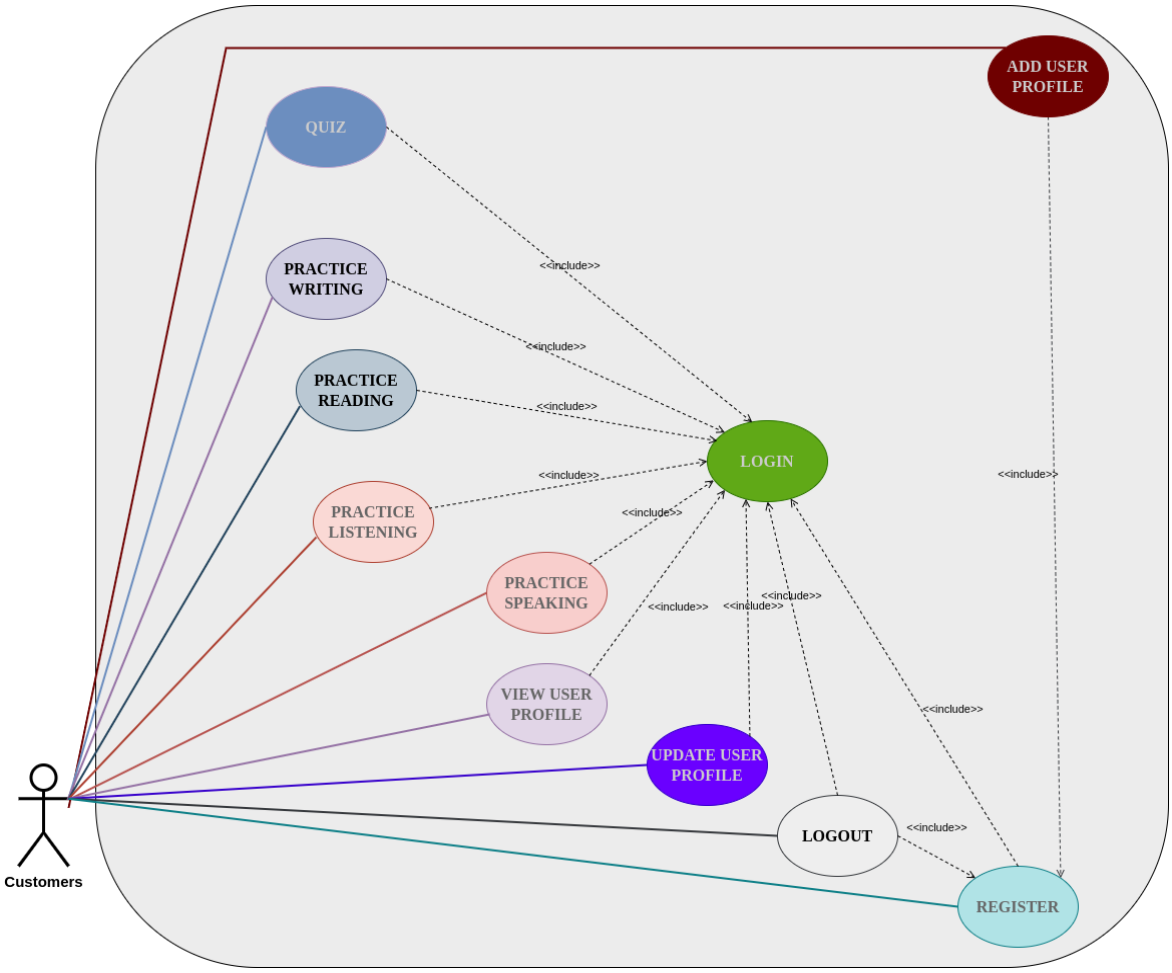

Figure 3.2. User Case Diagram

3.2.4.2.2. User case table

Name	Login
Actor	User
Entry Conditions	- The Internet is connected. - The app is running.
Flow of Events	1. Users access to the app. 2. Users click on the “Get Started” button. 3. The system will display the Login screen. 4. Users type email and password that they registered in the text field respectively.

	5. Users click the “Login” button. 6. Email and password is verified.
Exit Conditions	Users login successfully

Table 3.3. Login User Case Table

Name	Register
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running.
Flow of Events	1. Users access to the app. 2. Users click on the “Get Started” button. 3. The system clicks on the “Create New Account” button. 4. Users type email and password. 5. Users click on the “Continue” button. 6. The system will navigate to the Profile screen. 7. Users type first name and last name in the text field respectively. 8. Users click on the “Continue” button. 9. The system will navigate to the Profile screen. 10. Users type Age in the text field. 11. Users click on “Choose an image button”. 12. The system will navigate to the image from the phone. 13. Users choose an image. 14. The system will return back to the Profile screen. 15. Users click on the “Create new account” button. 16. The system will save users’ information in database
Exit Conditions	Users register successfully

Table 3.4. Register User Case Table

Name	View User Profile
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running. - Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. Users access the Profile screen in the bottom bar. 4. The system will display a list. 5. Users click on the "My Account" button.
Exit Conditions	The system will display users' information, such as: name, age, password.

Table 3.5. View User Profile User Case Table

Name	Update User Profile
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running. - Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. View User Profile. 4. Users update first name, last name, age, password, and image respectively. 5. Users click on the "Save" button. 6. The system will save first name, last name, age, password, and image respectively in the database.
Exit Conditions	Users' profile updated successfully.

Table 3.6. Update User Profile User Case Table

Name	Practice Writing
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running. - Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. The "Home Screen" displays 4. Users click on "Writing" button 5. The system displays a list of unit will display 6. Users click on each unit 7. The system displays a question 8. The system displays a text field for users to enter an answer 9. Users can type text in the text field 10. Users click on the "Submit" button
Exit Conditions	Submit users' answers successfully.

Table 3.7. Practice Writing User Case Table

Name	Practice Listening
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running. - Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. The "Home Screen" displays 4. Users click on "Listening" button 5. The system displays a list of unit will display 6. Users click on each unit

	7. The system displays a question 8. The system displays a text field for users to enter an answer 9. Users can type text in the text field 10. Users click on the "Submit" button
Exit Conditions	Submit users' answers successfully.

Table 3.8. Practice Listening User Case Table

Name	Practice Reading
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running. - Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. The "Home Screen" displays 4. Users click on "Reading" button 5. The system displays a list of unit will display 6. Users click on each unit 7. The system displays a question 8. The system displays a text field for users to enter an answer 9. Users can type text in the text field 10. Users click on the "Submit" button
Exit Conditions	Submit users' answers successfully.

Table 3.9. Practice Reading User Case Table

Name	Practice Speaking
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running.

	- Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. The "Home Screen" displays 4. Users click on "Speaking" button 5. The system displays a list of unit will display 6. Users click on each unit 7. The system displays a question and a button 8. Users can click on the button to say something 9. The system will record users' speech.
Exit Conditions	Submit users' answers successfully.

Table 3.10. Practice Speaking User Case Table

Name	Quiz
Actor	Users
Entry Conditions	- The Internet is connected. - The app is running. - Users logged in to the account.
Flow of Events	1. Users access to the app. 2. Login. 3. The "Home Screen" displays 4. Users navigate to Quiz screen 5. The system displays a list of modes for users to choose. 6. Users click on each mode. 7. The system displays questions, options and calculates time. 8. Users click on 1 option.
Exit Conditions	Users answer the quiz successfully.

Table 3.11. Quiz Case Table

3.2.4.3. Object Model

3.2.4.3.1. Activity Diagram

3.2.4.3.1.1. Activity Diagram Quiz

If the user is not logged in, they cannot make a quiz.

Customers will make a request to the server after confirming that they logged in to the app. The server then creates a session and forwards it to access to the Home Screen.

After navigating to the Home Screen, the system will display the bottom navigation bar. Users will need to click on the middle button.

After that, the system will display 5 modes for customers to choose: 5 minutes, 15 minutes, 30 minutes, 45 minutes, and 60 minutes.

After clicking on each question, the system will display a question and the answer, and users can click on 1 answer for a question. If it is true, the score will be plus one, and the next button will appear.

Users click on the next button, the screen will navigate to the next question until the end.

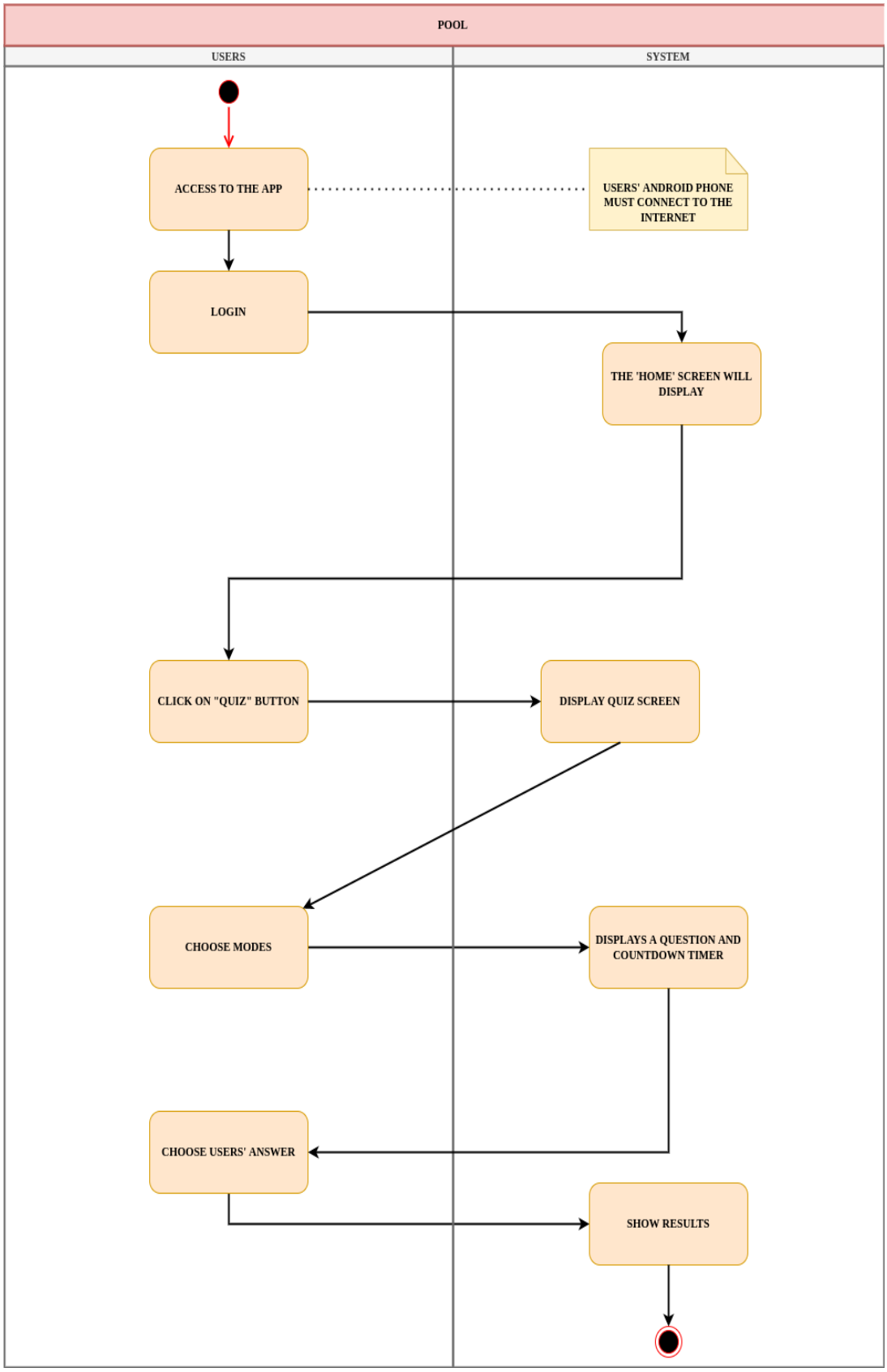

Figure 3.3. Activity Diagram Quiz

3.2.4.3.1.2. Activity Diagram Practice Listening

If the user is not logged in, they cannot practice listening.

Customers will make a request to the server after confirming that they logged in to the app. The server then creates a session and forwards it to access to the Home Screen.

After navigating to the Home Screen, the ‘Listening’ button will display. Therefore, if users click on the ‘Listening’ button, the system will navigate to a list of units.

Users can click on each unit, then a list of questions and text fields for users will display. Users can type users’ answers to the text field and click submit. The system will automatically get users’ answers.

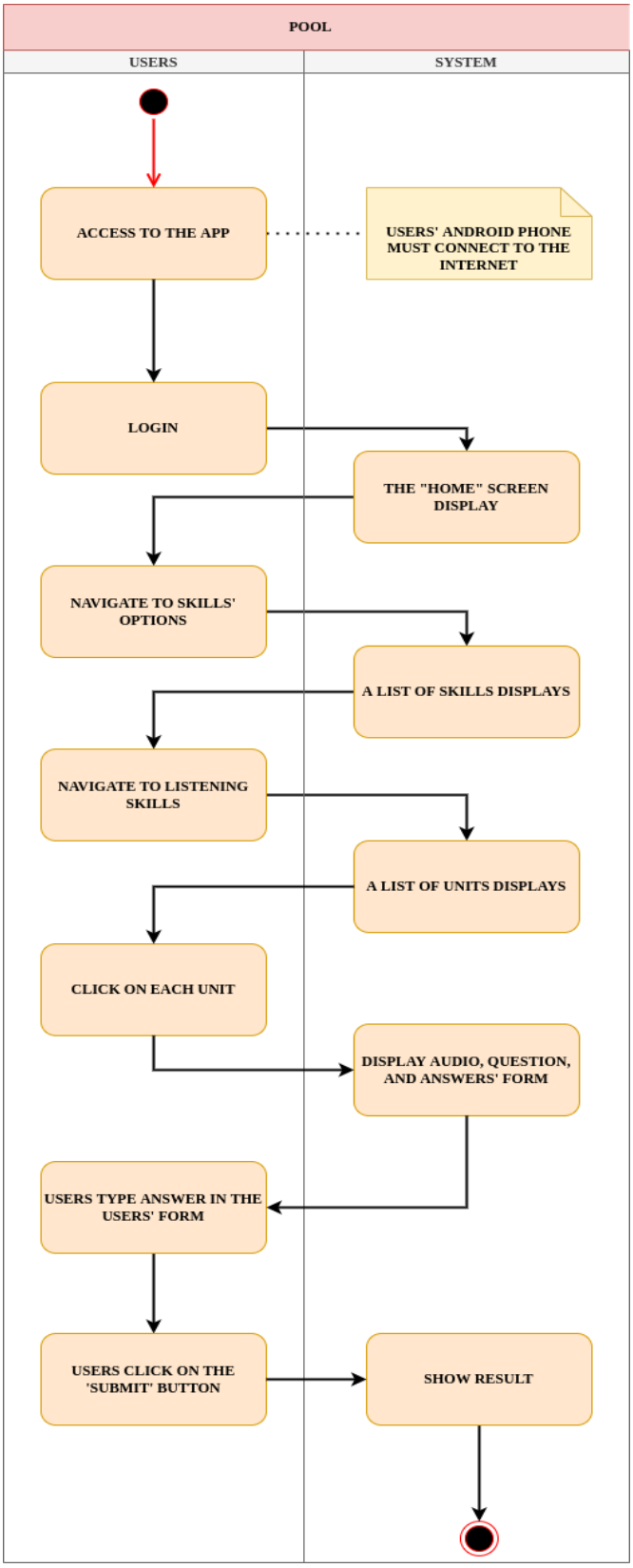

Figure 3.4. Activity Diagram Practice Listening

3.2.4.3.2. Class diagram

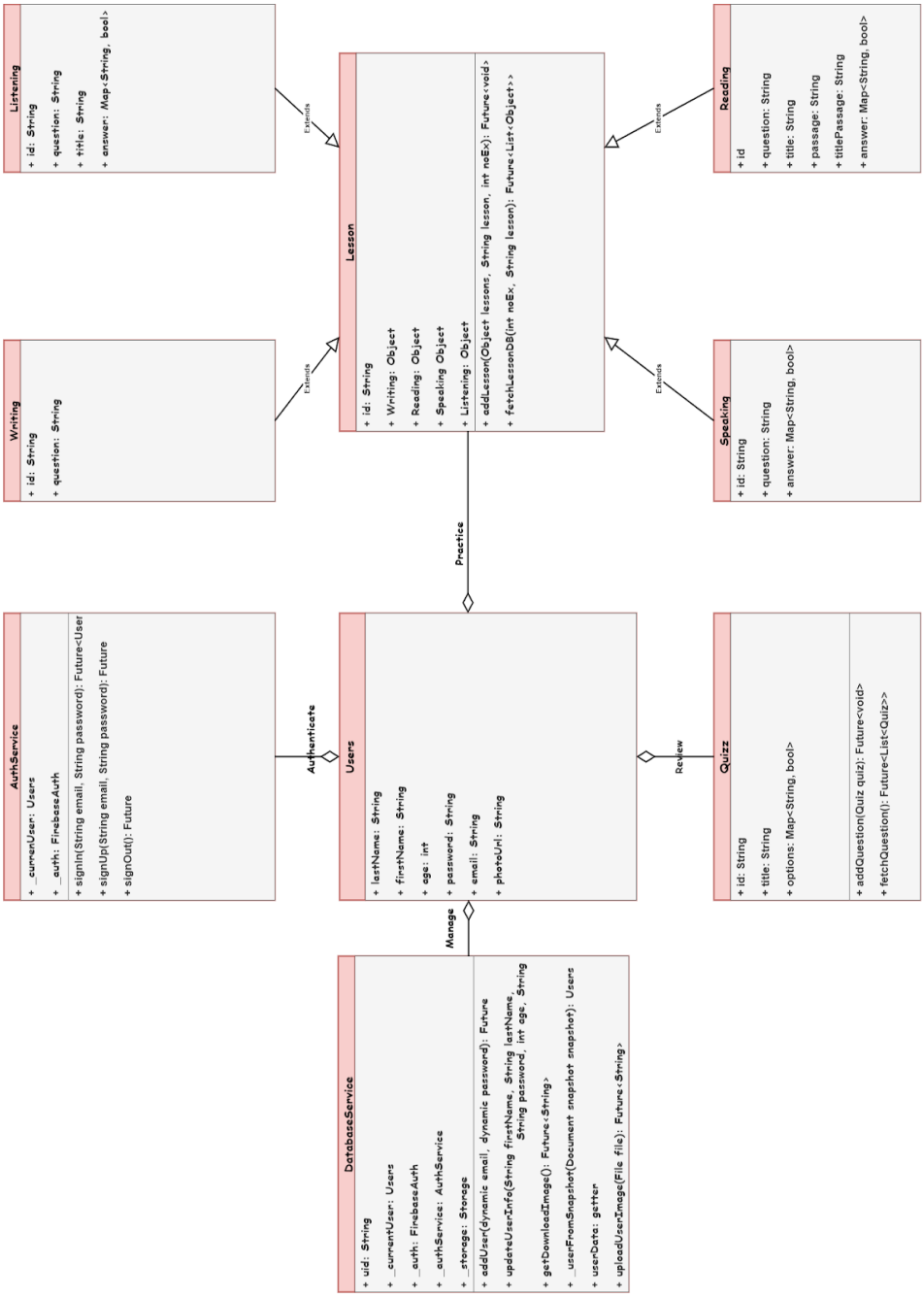

Figure 3.5. Class Diagram

3.2.4.4. Database Design

The database design of the system is illustrated in the following figure.

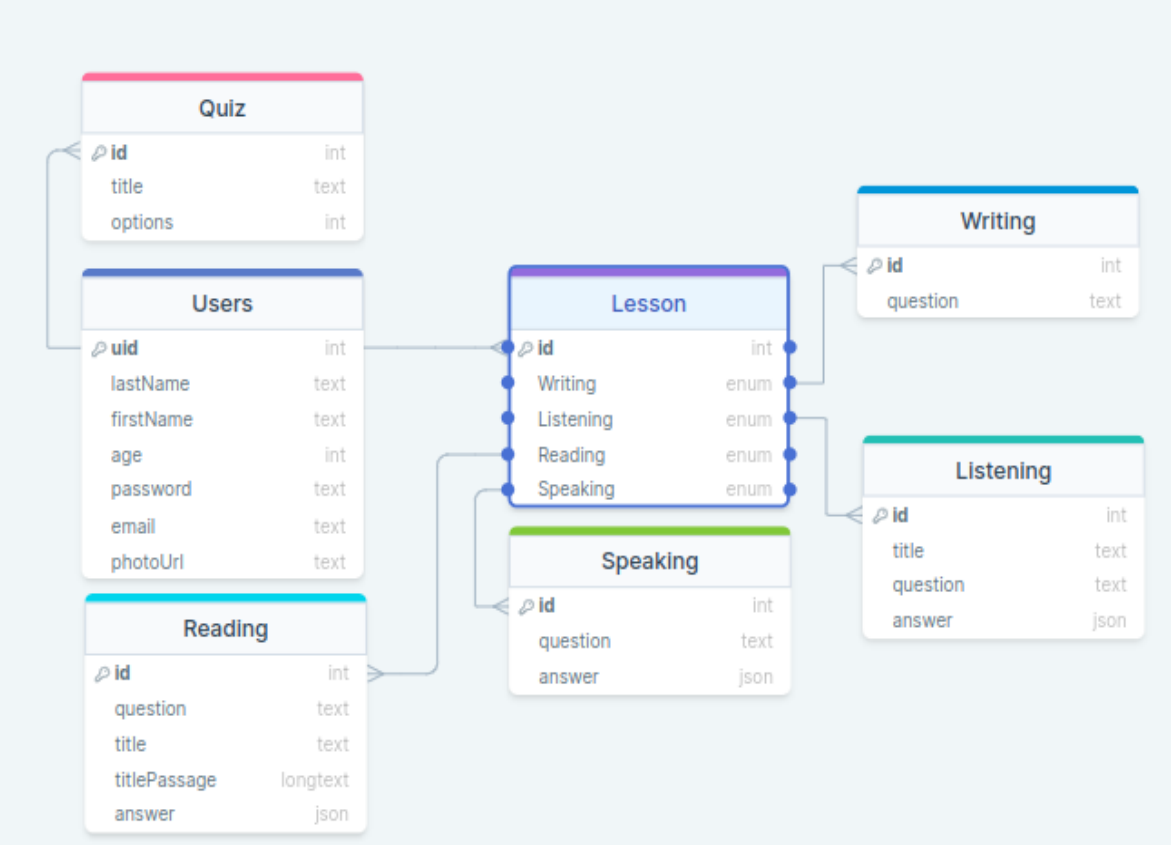

Figure 3.6. Database Design

3.2.4.5. User Interface – Navigational Paths and Screen Mock-Ups

3.2.4.5.1. Website sitemap

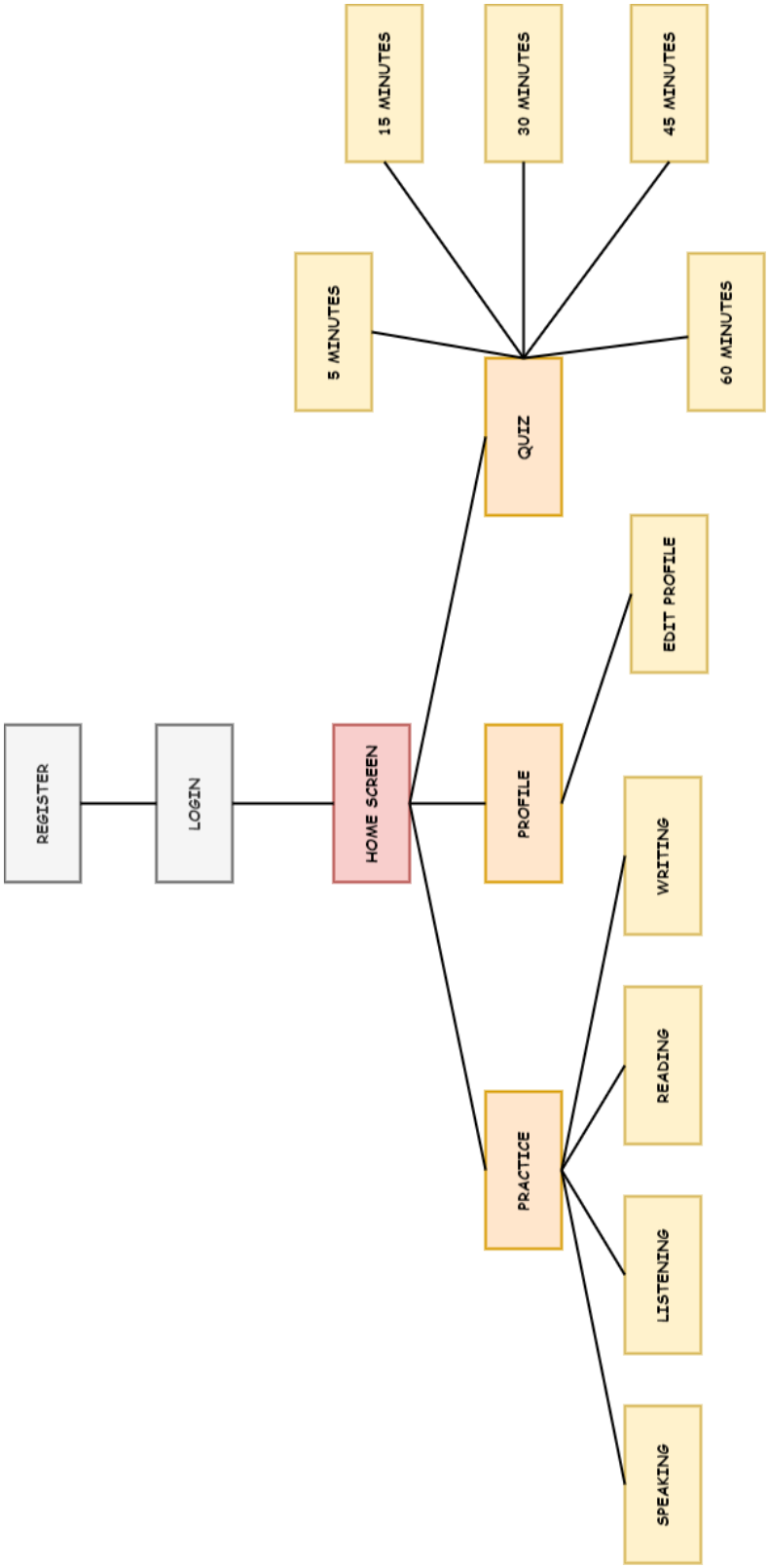

Figure 3.7. Website Sitemap

3.2.4.5.2. Details

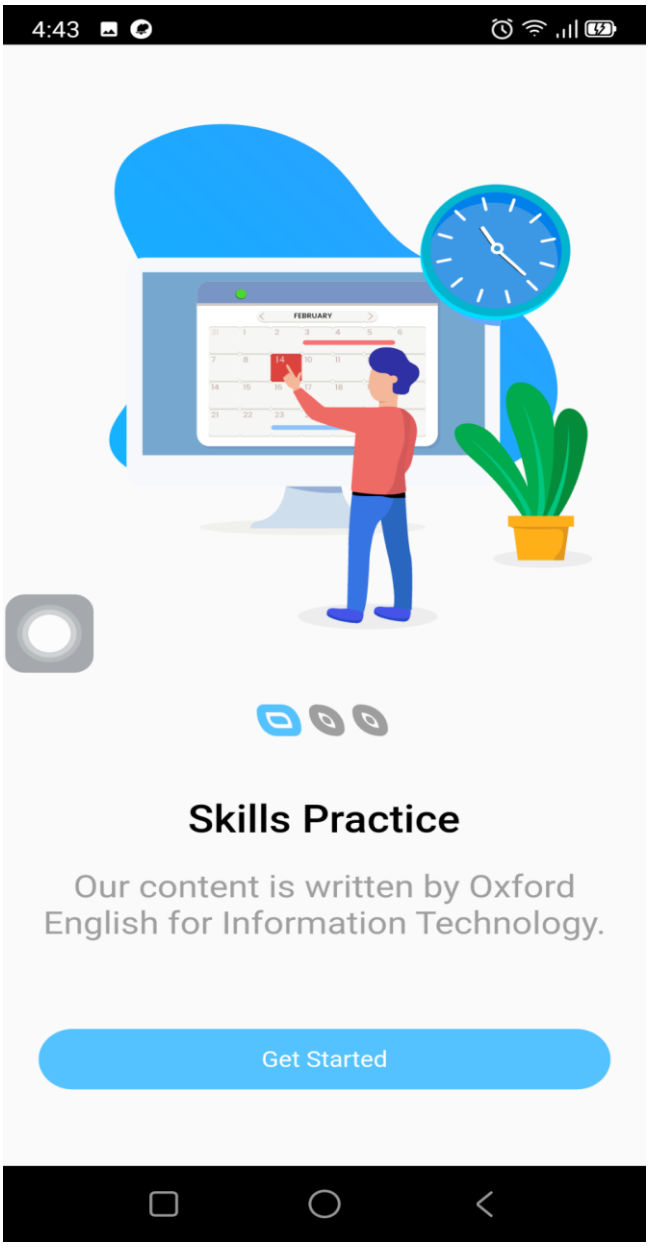

Figure 3.8. Splash Screen 1

On Slide	Display Splash Screen 2
On Click “Get Started” button	Display Login Screen

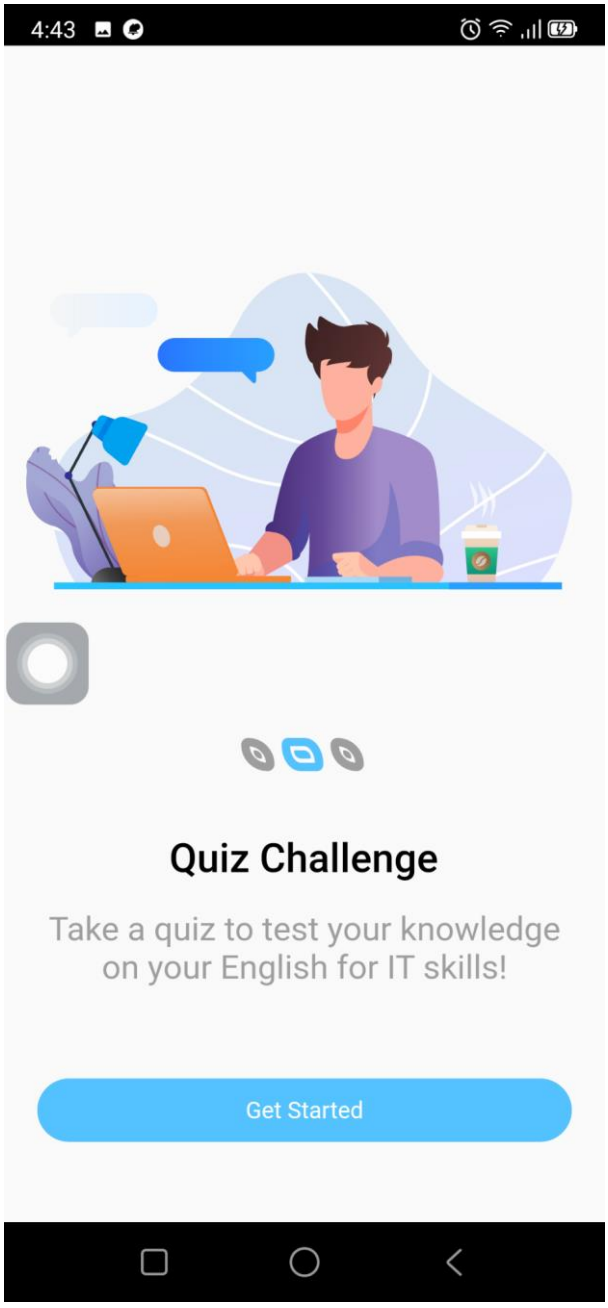

Figure 3.9. Splash Screen 2

On Slide	Display Splash Screen 3
Reverse	
On Click “Get Started” button	Display Login Screen

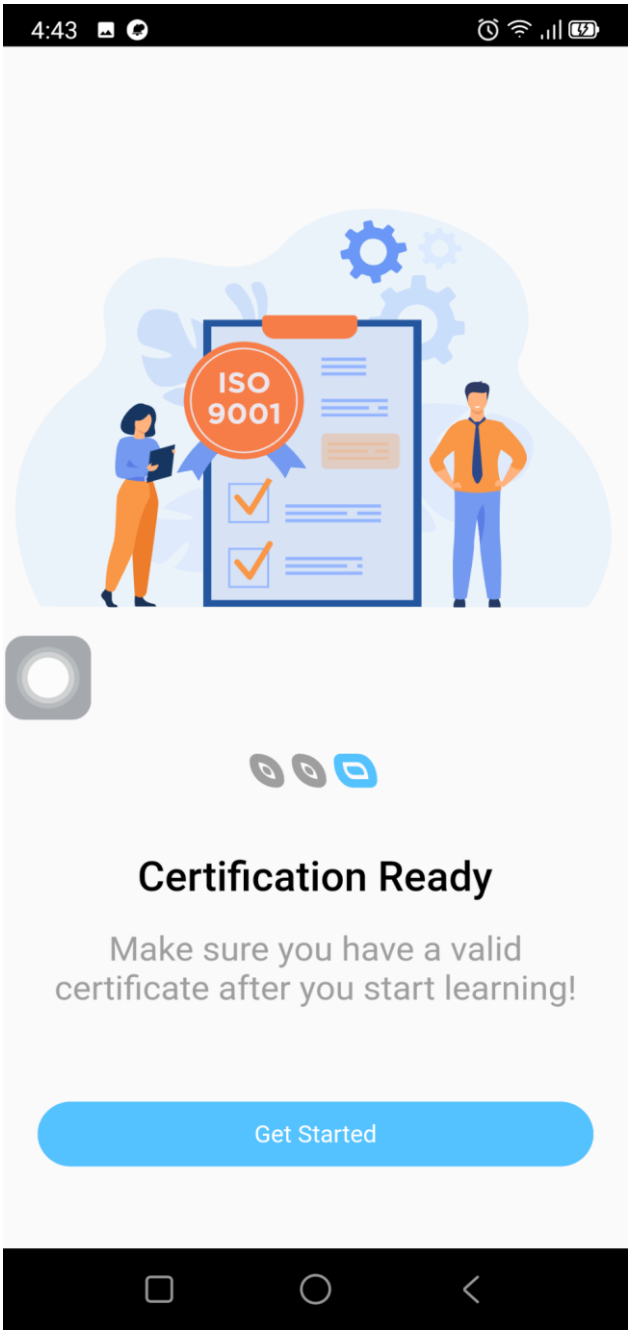

Figure 3.10. Splash Screen 3

On Slide back	Display Splash Screen 2
On Click “Get Started” button	Display Login Screen

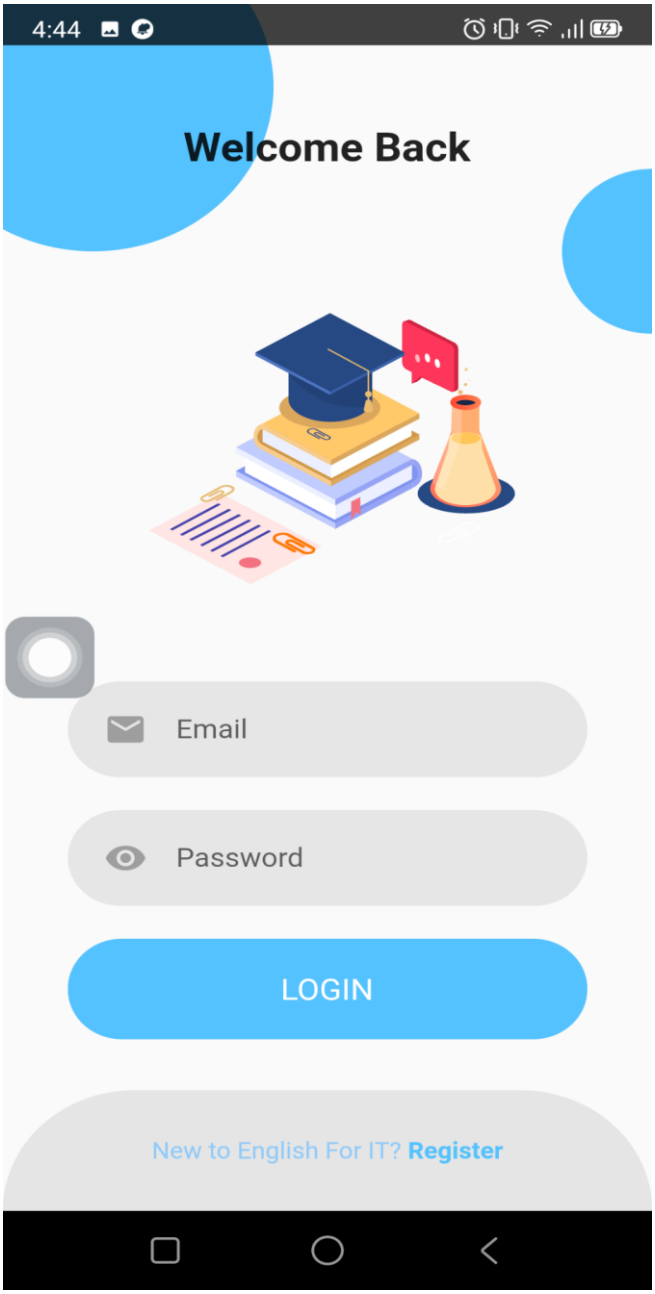

Figure 3.11. Login Screen

Fill Right Information	
On Click “Login” button	Display Successful Screen
Fill Wrong Information	
On Click “Login” button	Display Error Screen

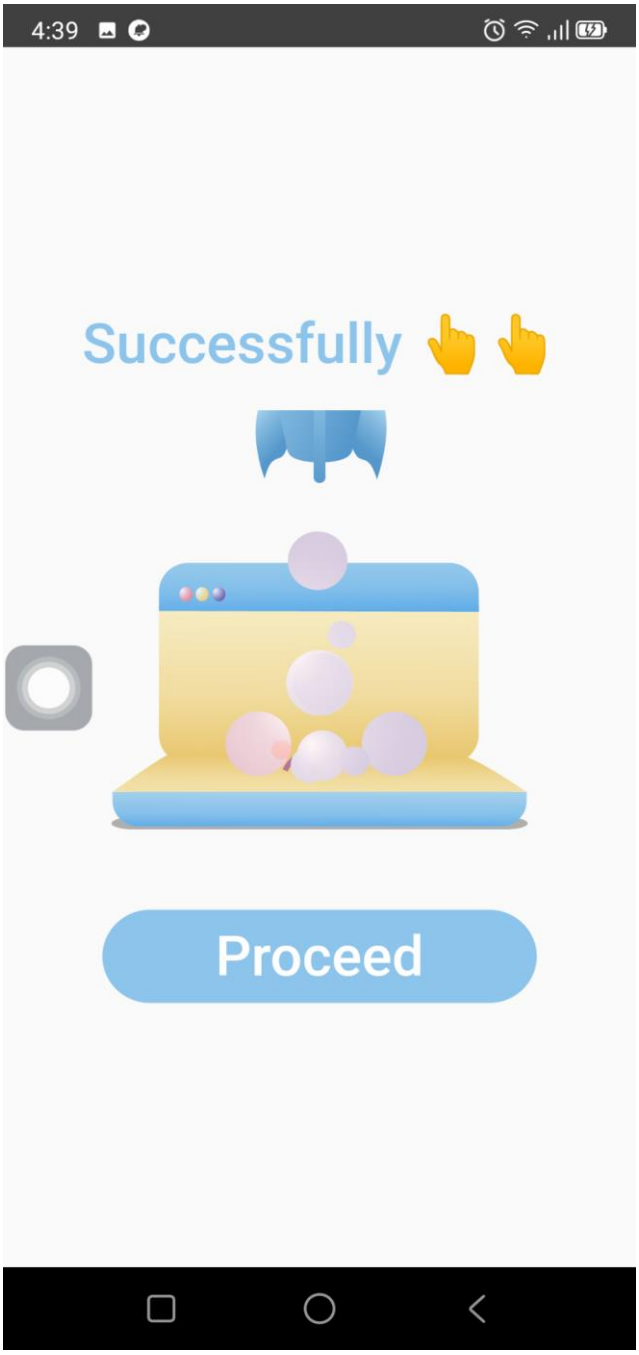

Figure 3.12. Successful Screen

On Click “Proceed” button	Display Home Screen

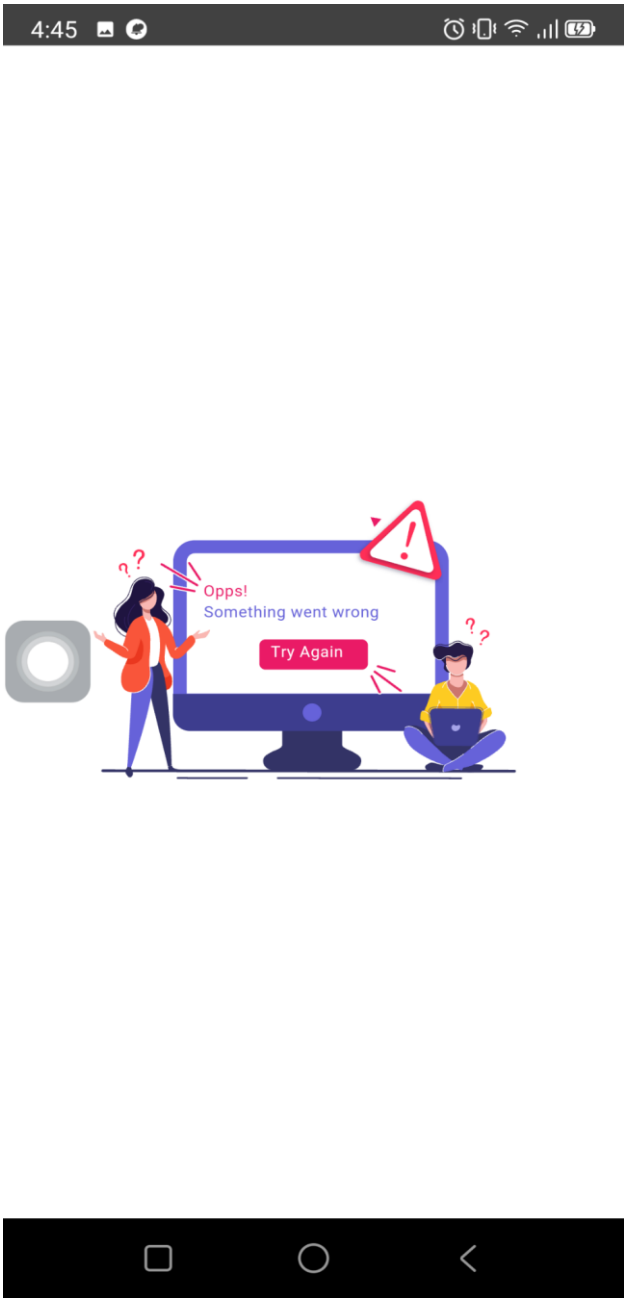

Figure 3.13. Error Screen

On Click	Display Login Screen

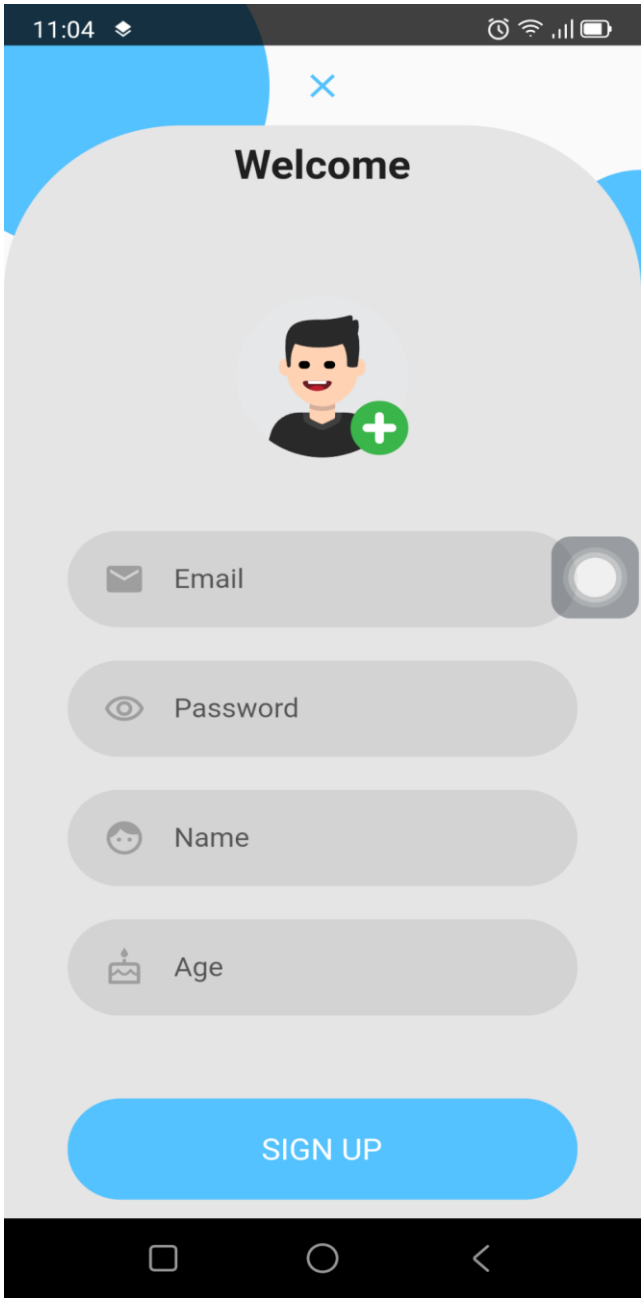

Figure 3.14. Register Screen

Fill Right Information	
On Click “Sign up” button	Display Successful Register Screen
Fill Wrong Information	
On Click “Sign up” button	Display Error Screen

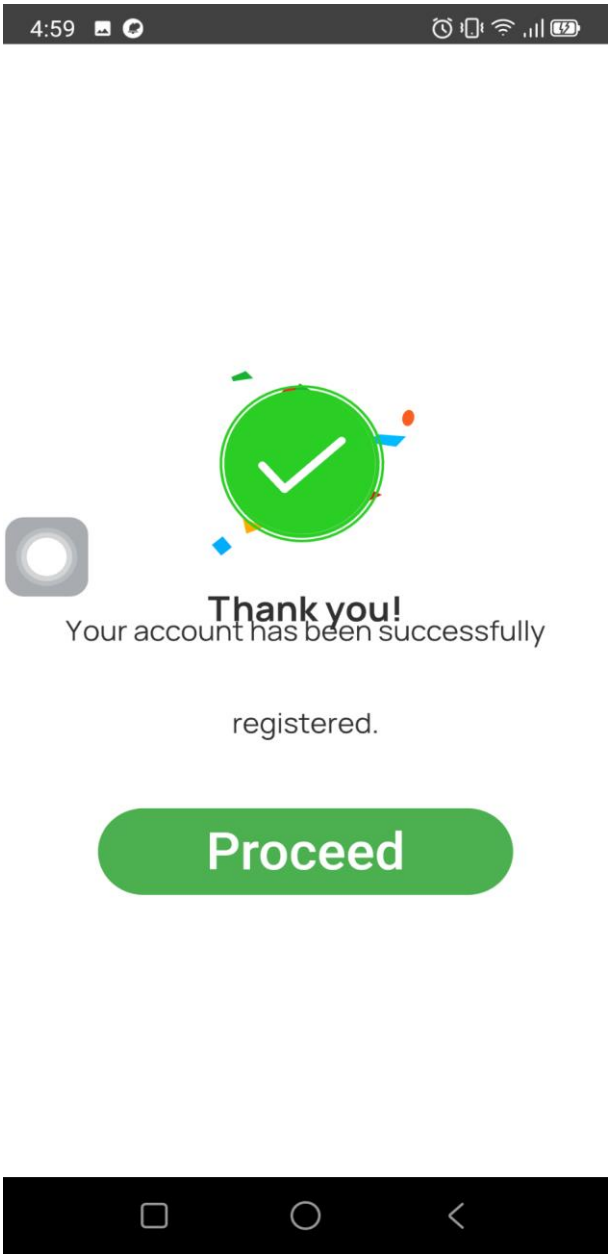

Figure 3.15. Successful Register Screen

On Click “Proceed” button	Display Home Screen

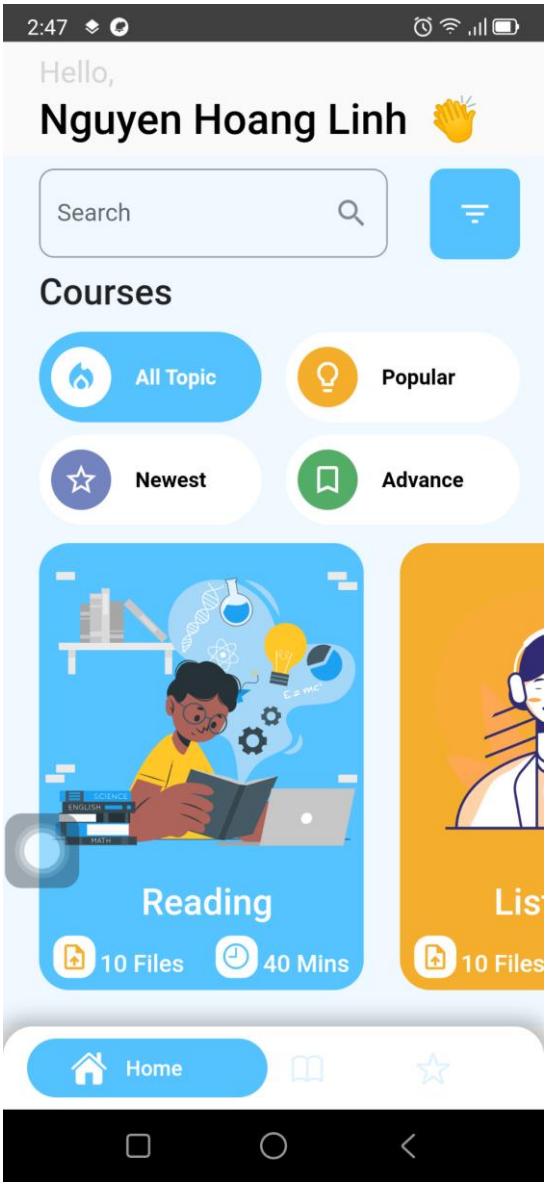

Figure 3.16. Home Screen

On Click “Reading”	Display Reading Enrollment Screen
On Click “Listening”	Display Listening Enrollment Screen
On Click “Speaking”	Display Speaking Enrollment Screen
On Click “Writing”	Display Writing Enrollment Screen
Bottom Navigation bar	
On Click	Display Course Screen
On Click	Display Games Screen
On Click	Display Settings Screen

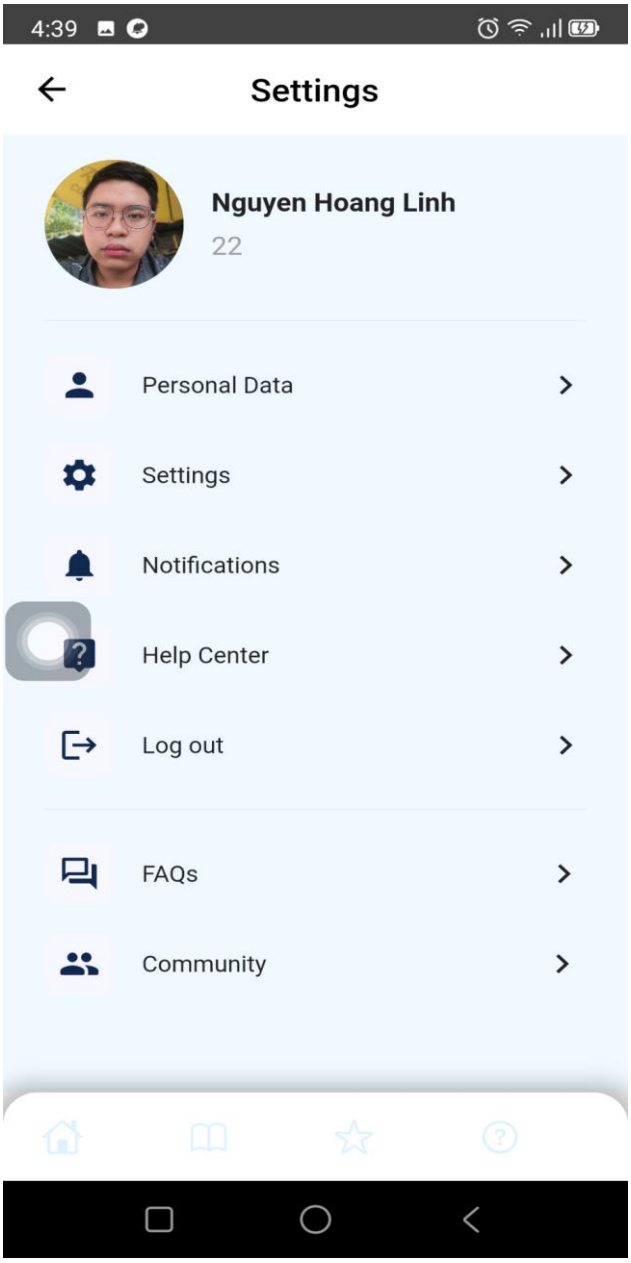

Figure 3.17. Settings Screen

On Click ‘Personal Data’	Display Personal Data Screen
On Click ‘Log out’	Display Login Screen

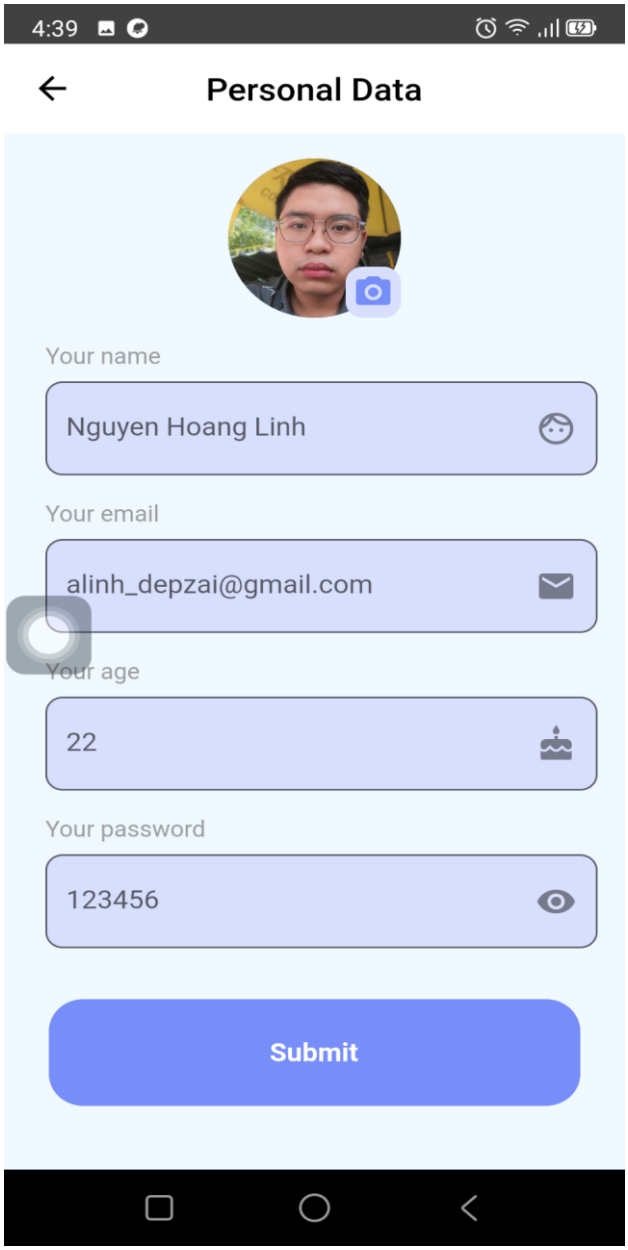

Figure 3.18. Personal Data Screen

On Click “Submit” button	Display Users’ Personal Data Changed Successfully And Navigate To Settings Screen

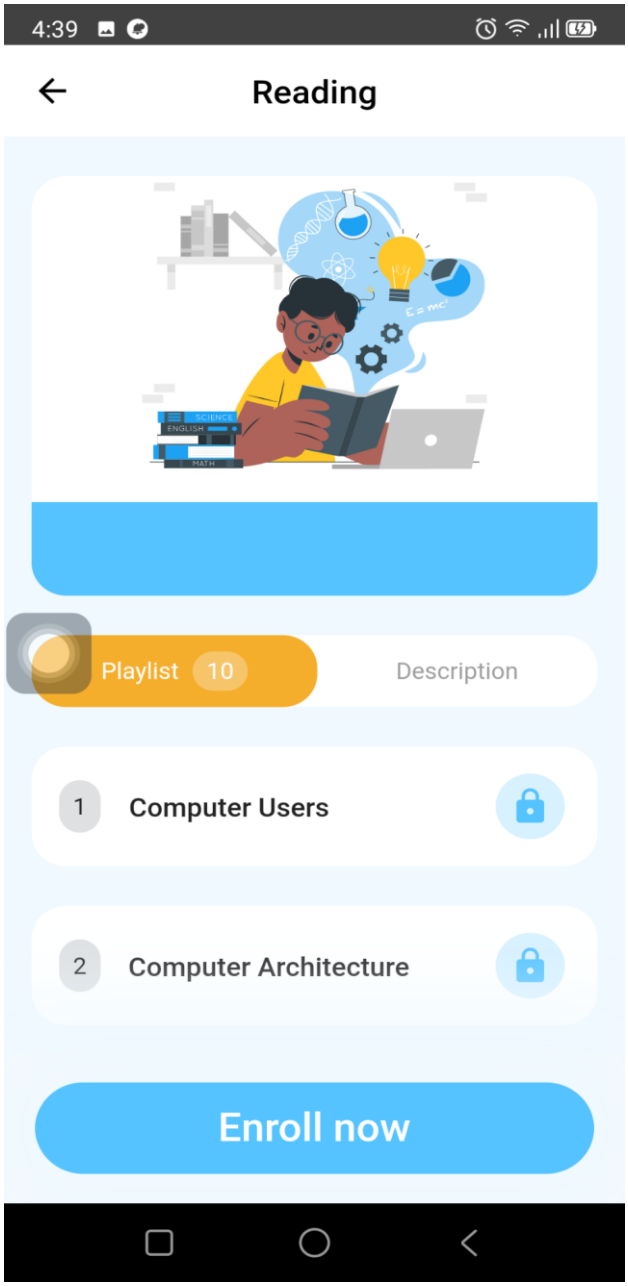

Figure 3.19. Reading Enrollment Screen

On Click “Enroll now” button	Display Unlocked Reading Lesson Screen

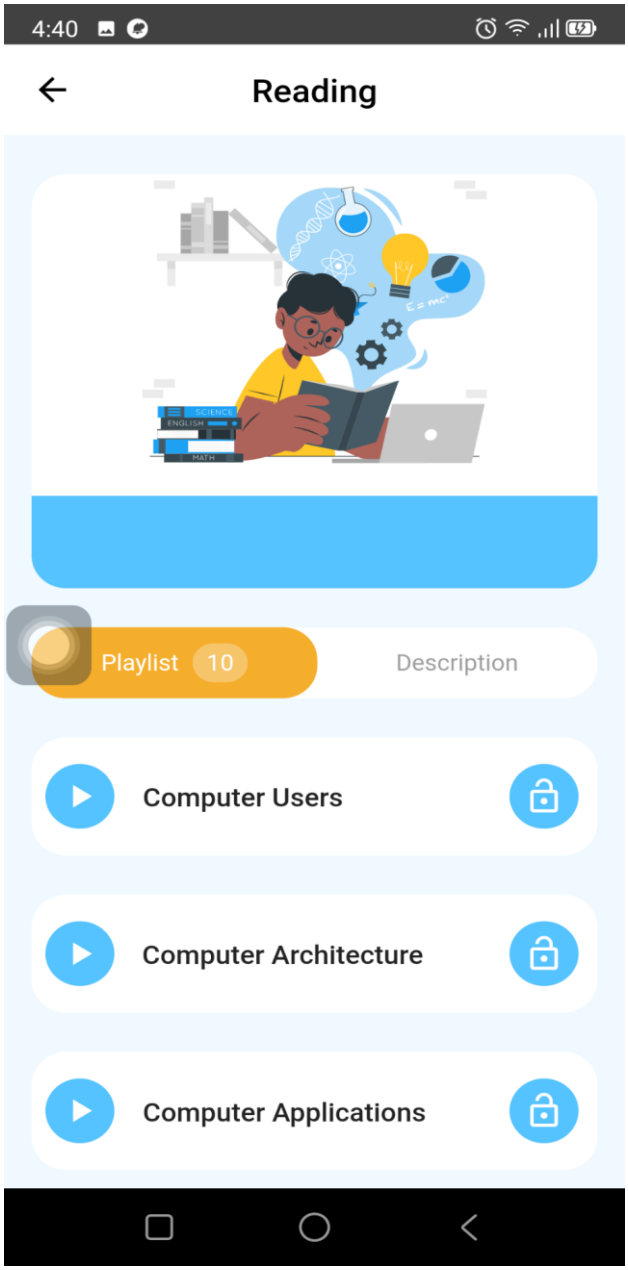

Figure 3.20. Unlocked Reading Lesson Screen

On Click Each Lesson	Display Reading Contents Screen

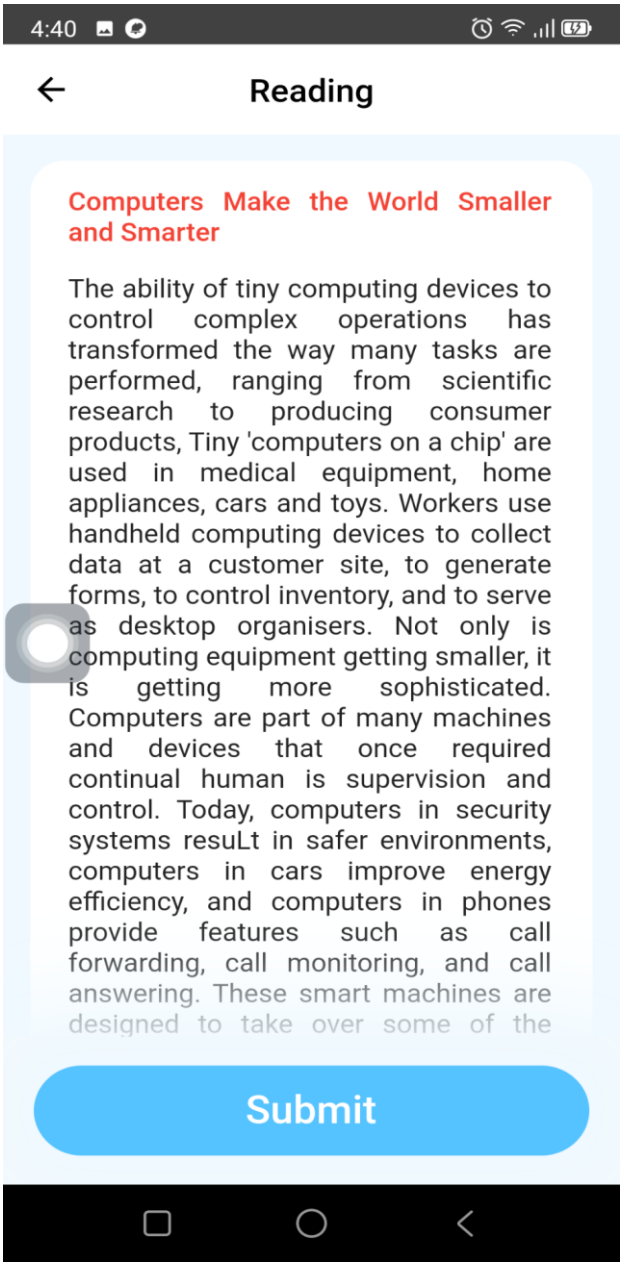

Figure 3.21. Reading Contents Screen

On Click “Submit” button	Display Reading Result Box Screen

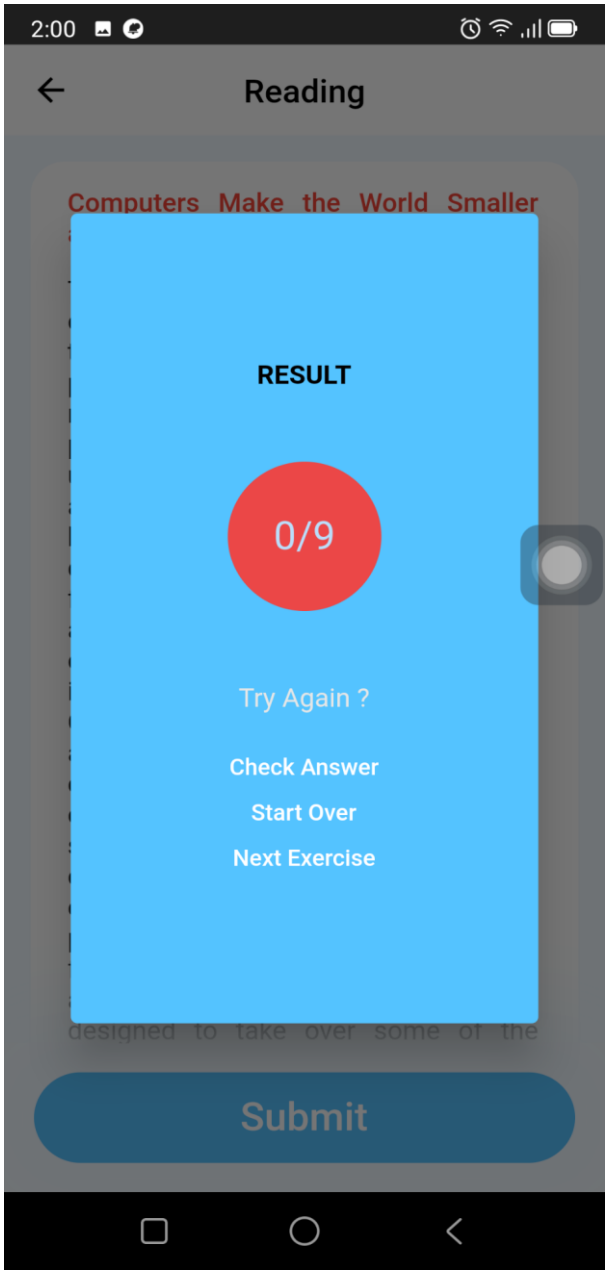

Figure 3.22. Reading Result Box Screen

On click “Check Answer”	Display Reading Check Answer Screen
On Click “Start Over”	Display Reading Contents Screen
On Click “Next Exercise”	Navigate To New Reading Content Screen

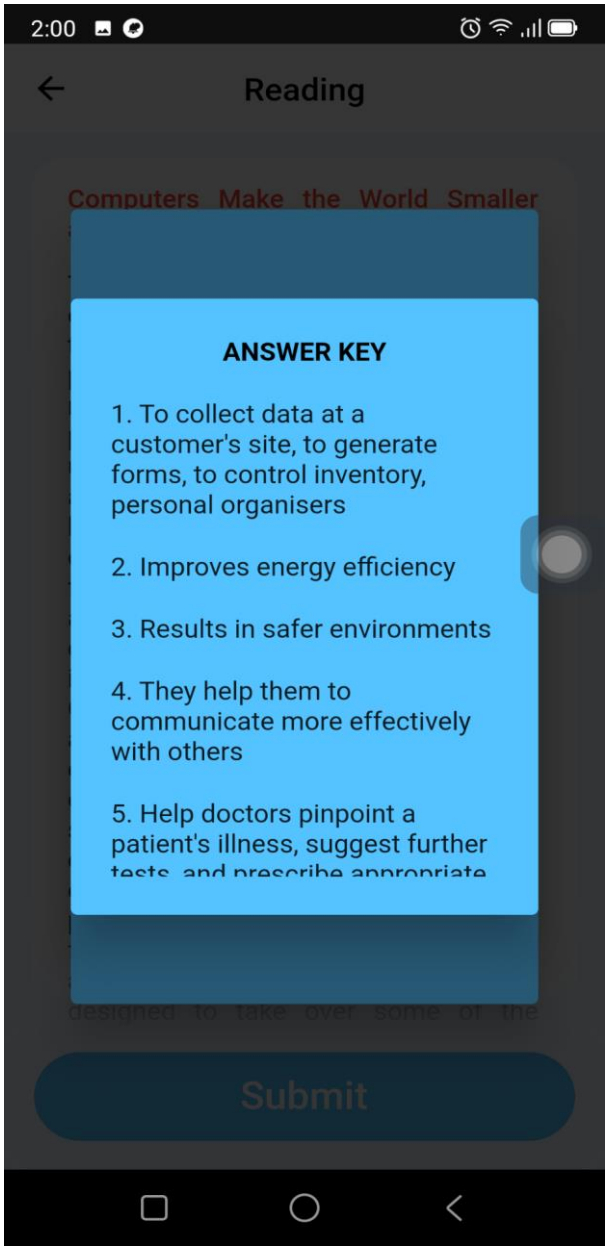

Figure 3.23. Reading Answer Key Screen

On Click	Navigate Back To Reading Result Box Screen

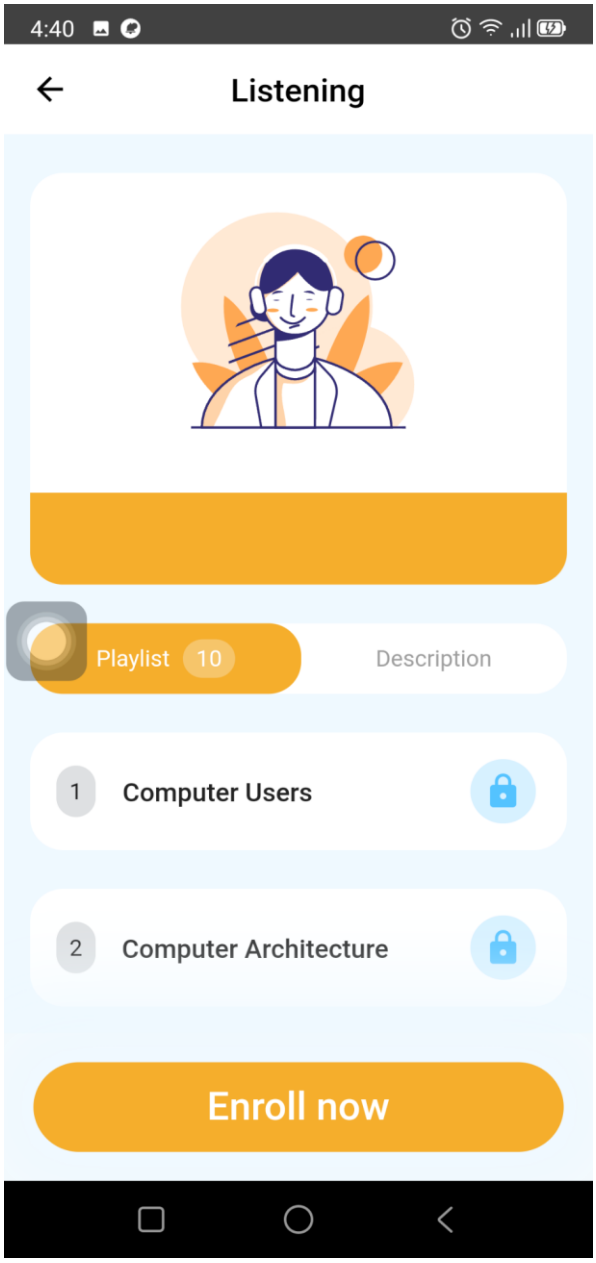

Figure 3.24. Listening Enrollment Screen

On Click “Enroll now” button	Display Unlocked Listening Lesson Screen

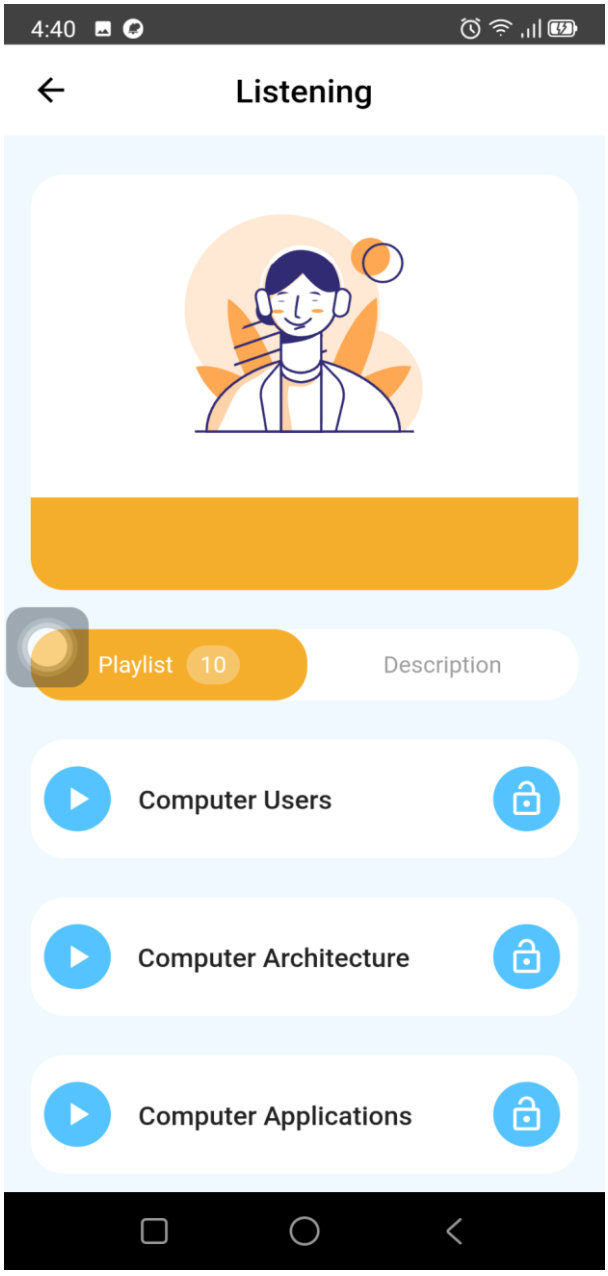

Figure 3.25. Unlocked Listening Lesson Screen

On Click Each Lesson	Display Listening Contents Screen

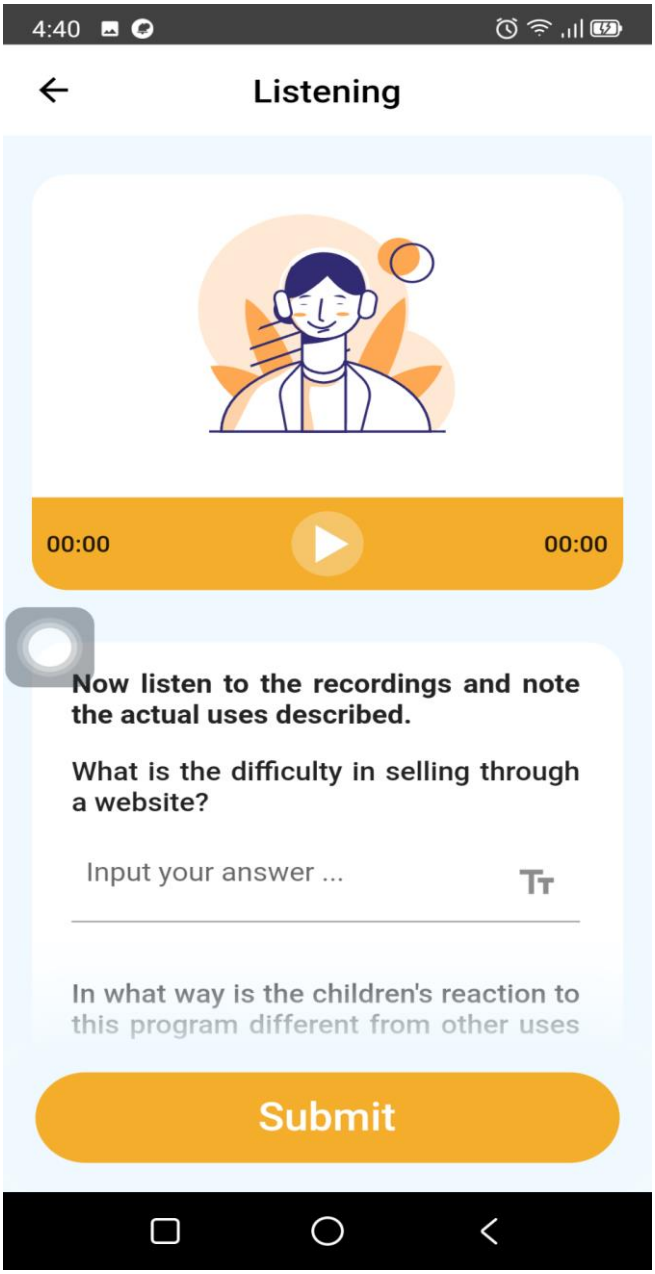

Figure 3.26. Listening Contents Screen

On Click “Submit” button	Display Listening Result Box Screen

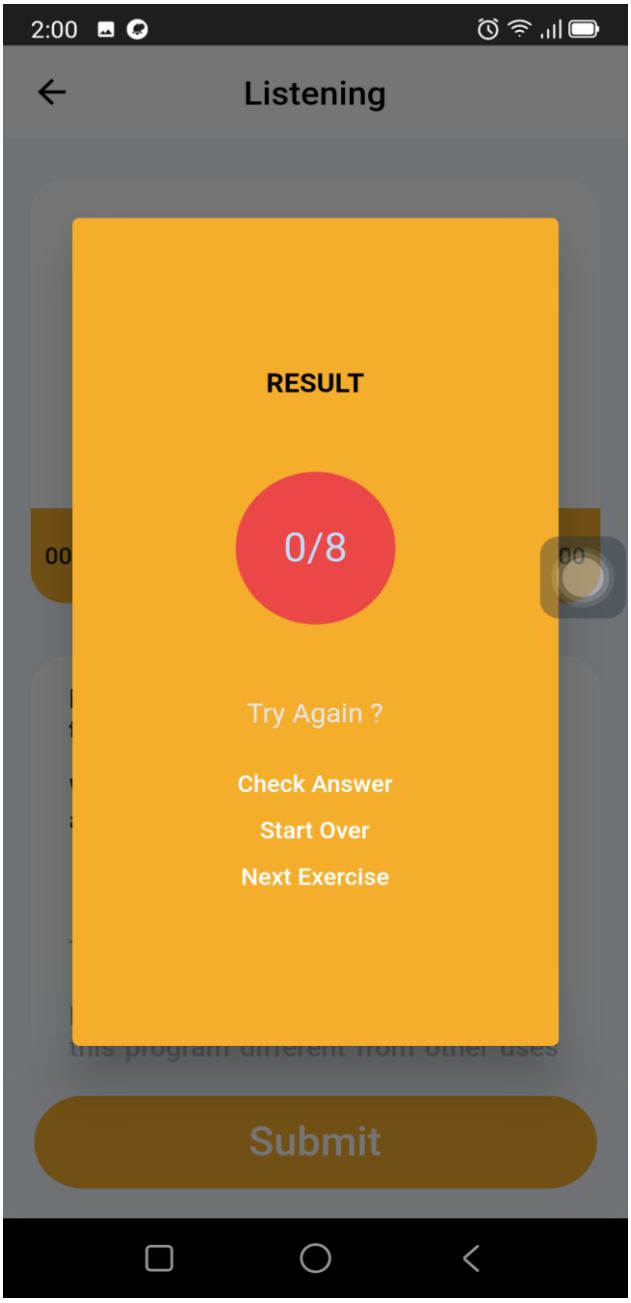

Figure 3.27. Listening Result Box Screen

On Click “Check Answer”	Display Listening Answer Key Screen

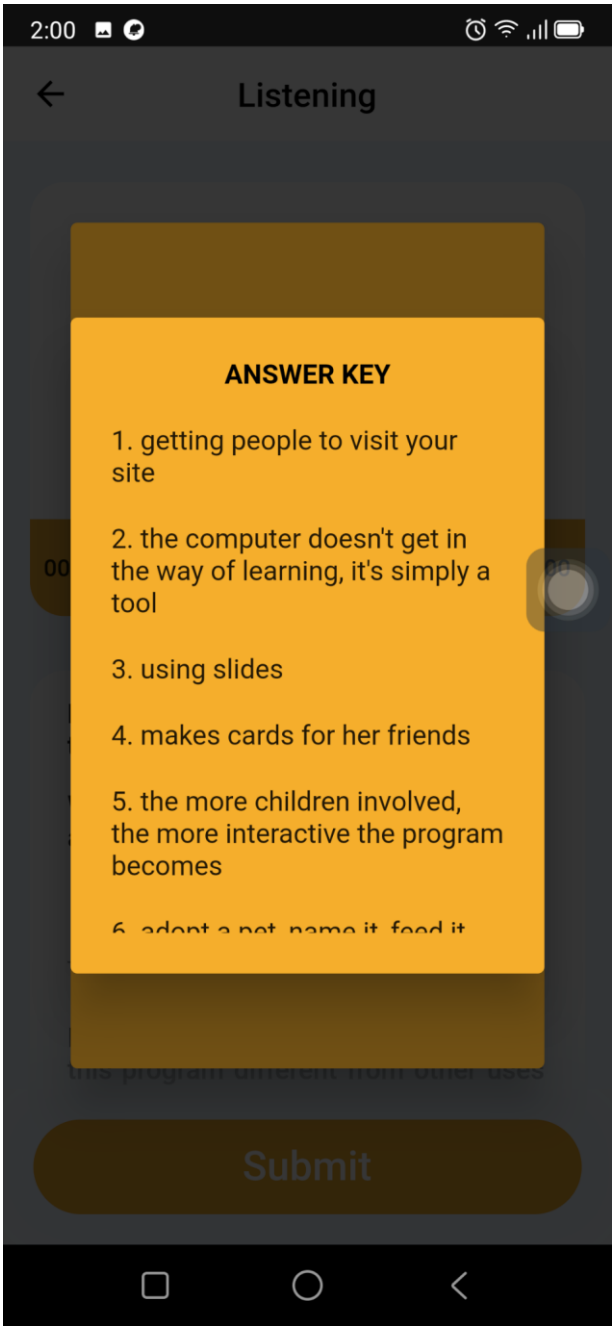

Figure 3.28. Listening Answer Key Screen

On Click	Navigate Back To Listening Result Box Screen

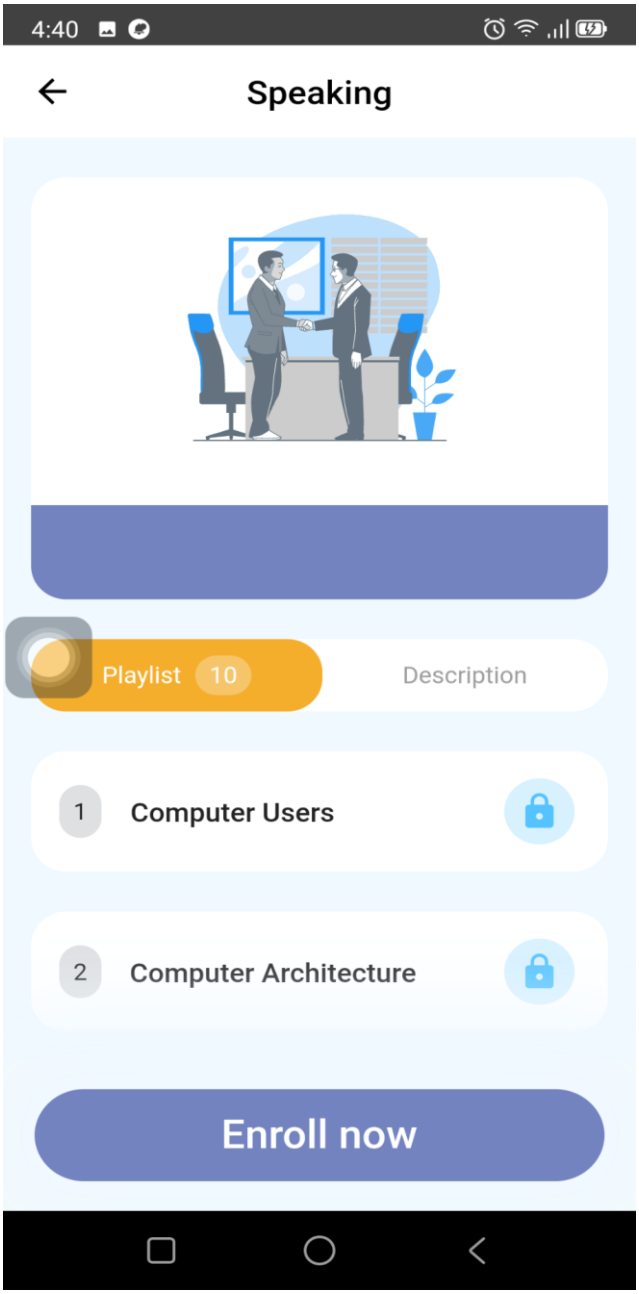

Figure 3.29. Speaking Enrollment Screen

On Click “Enroll now” button	Display Unlocked Speaking Lesson Screen

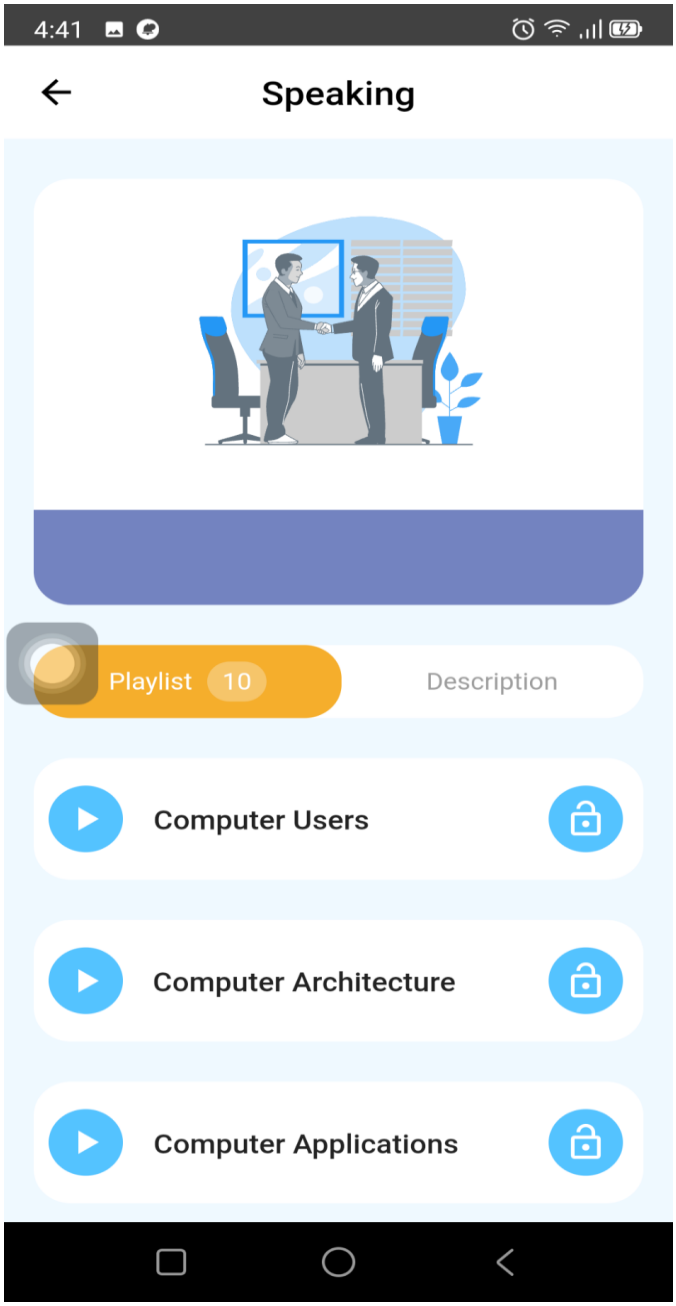

Figure 3.30. Unlocked Speaking Lesson Screen

On Click Each Lesson	Display Speaking Content Screen

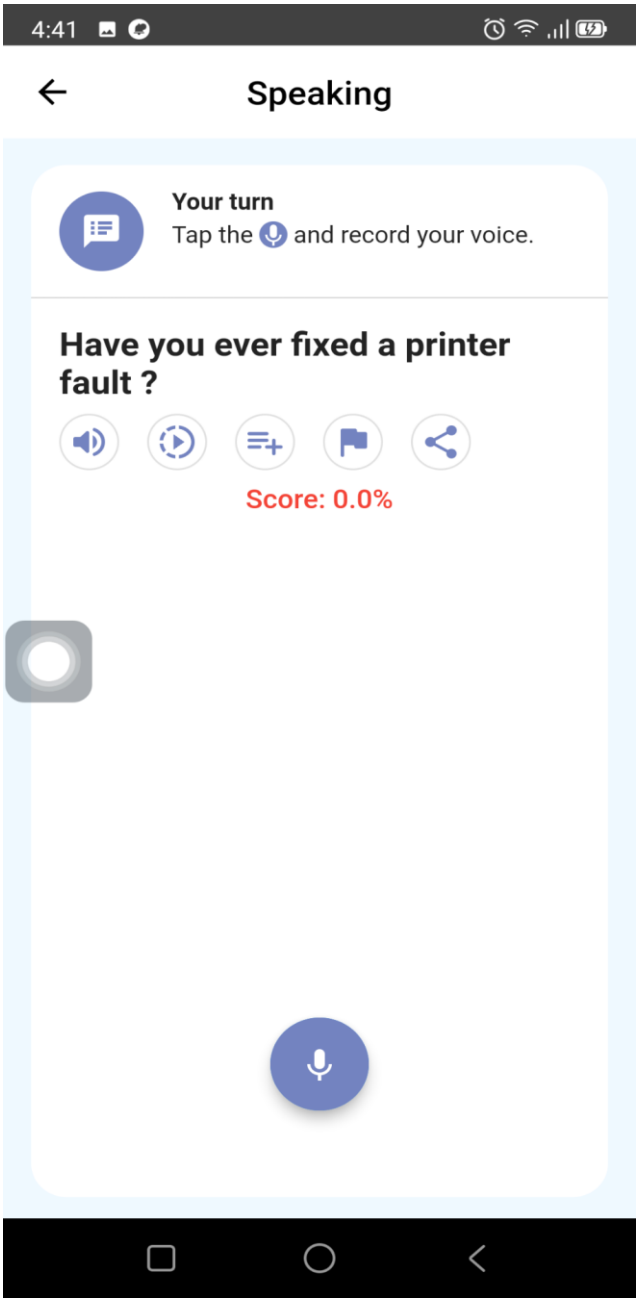

Figure 3.31. Speaking Contents Screen

On Click Icon Button	The system record audio and the score is changed

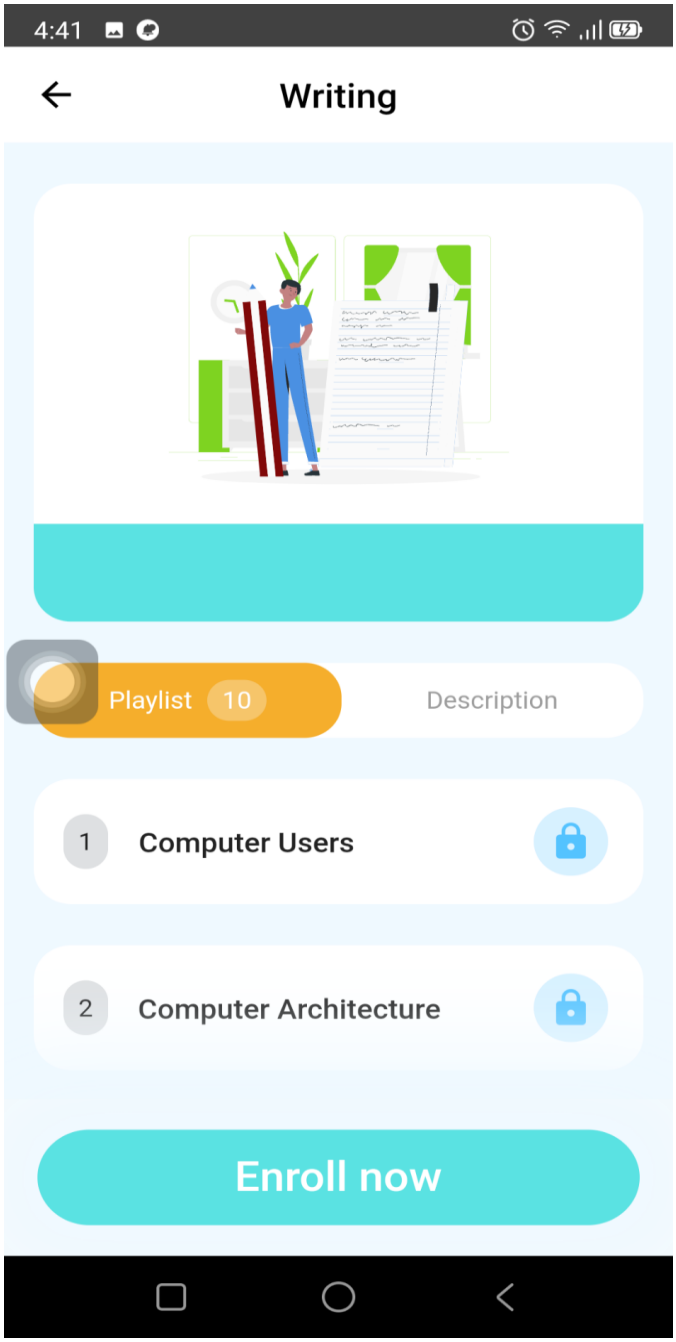

Figure 3.32. Writing Enrollment Screen

On Click “Enroll now” button	Display Unlocked Writing Lesson Screen

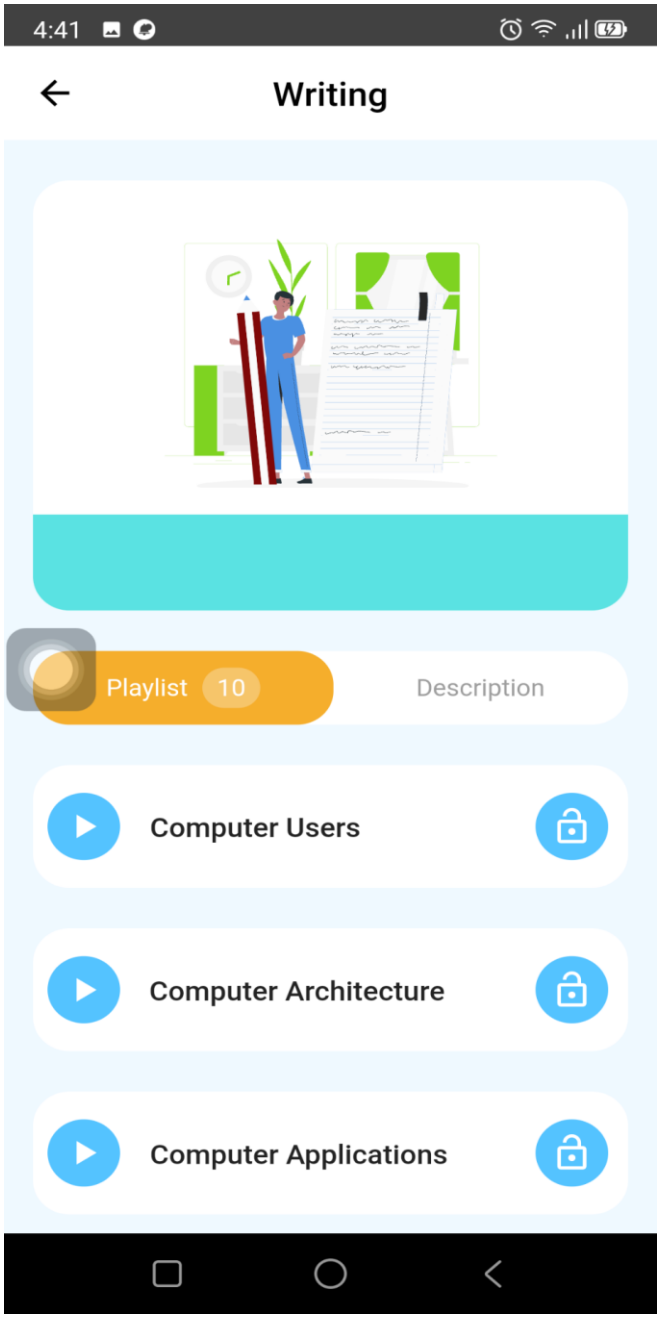

Figure 3.33. Unlocked Writing Lesson Screen

On Click Each Lesson	Display Writing Contents Screen

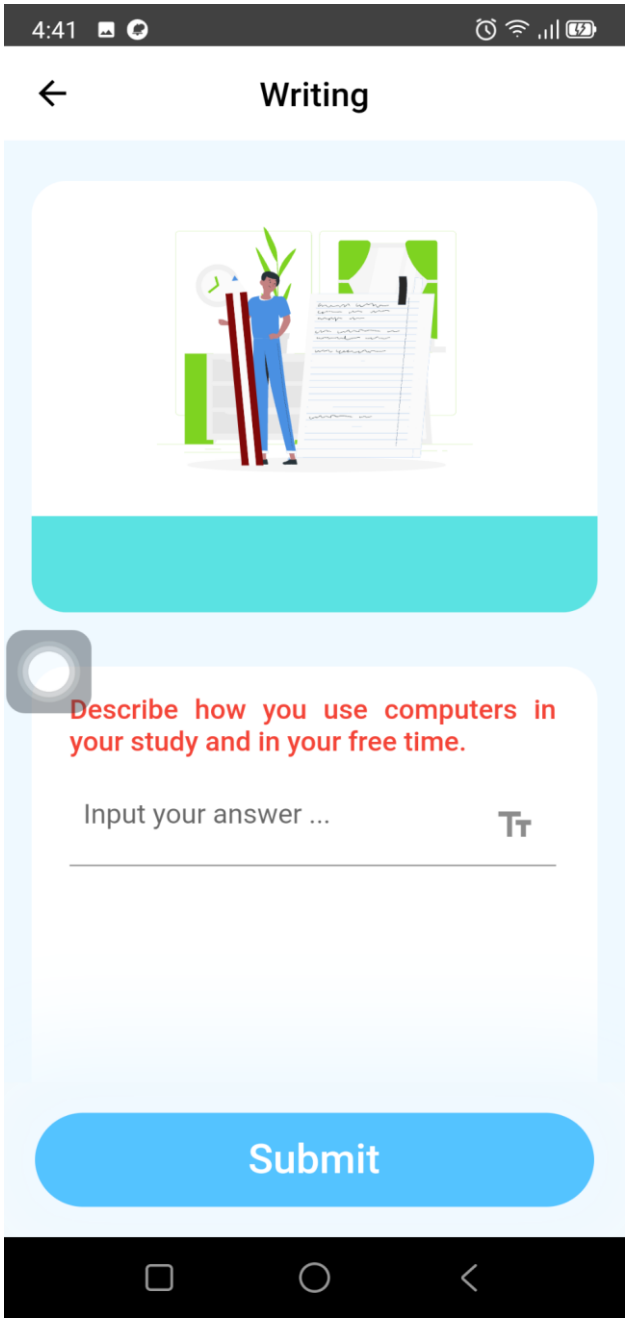

Figure 3.34. Writing Contents Screen

On Click “Submit” button	Display Writing Result Box Screen

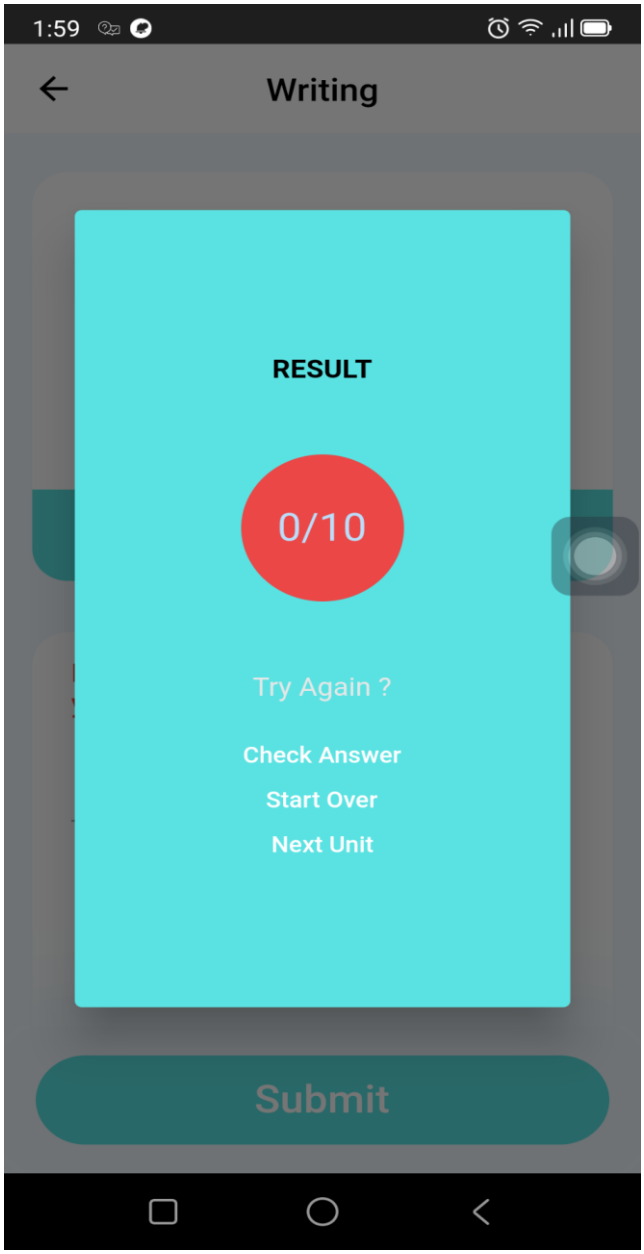

Figure 3.35. Writing Result Box Screen

Users' answer is less than 5 words	
On Click "Check Answer"	Display Writing Answer Key Screen 1
Users' answer is more than 5 words	
On Click "Check Answer"	Display Writing Answer Key Screen 2

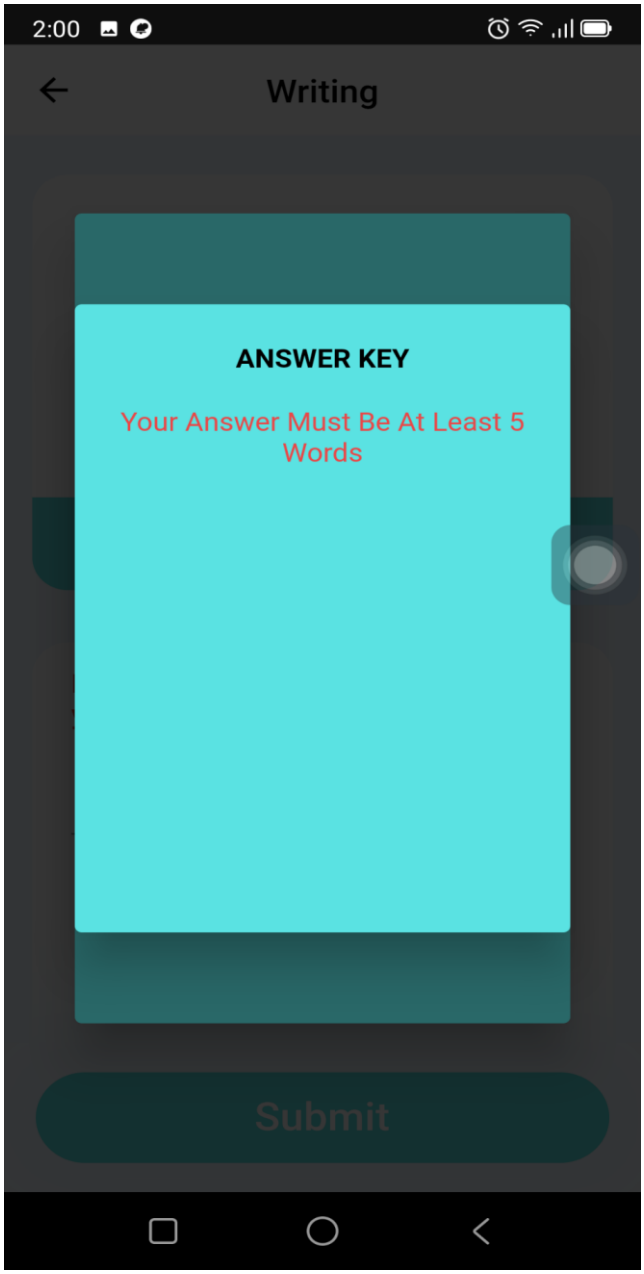

Figure 3.36. Writing Answer Key Screen 1

On Click	Navigate Back To Writing Result Box Screen

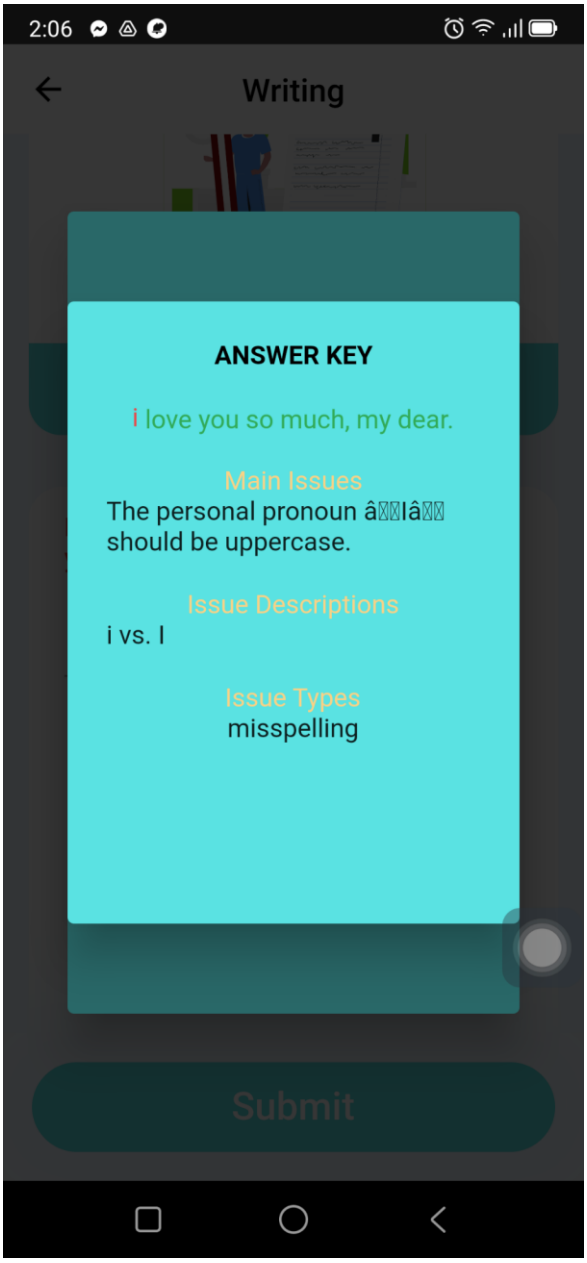

Figure 3.37. Writing Answer Key Screen 2

On Click	Navigate Back To Writing Result Box Screen

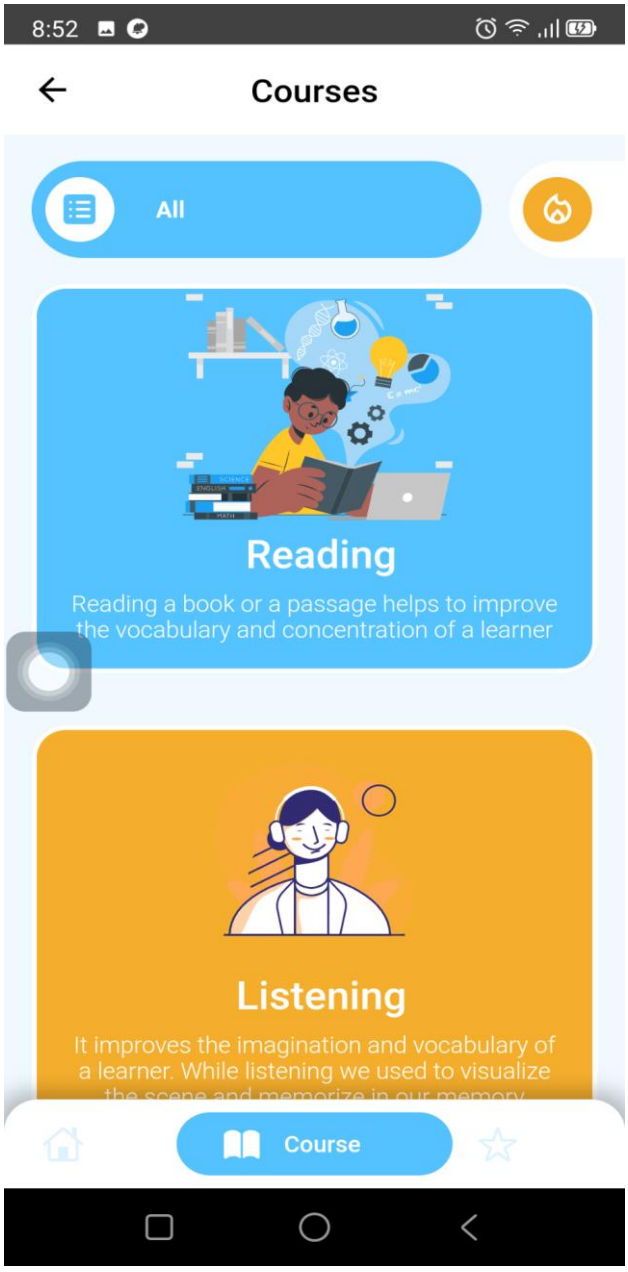

Figure 3.38. Course Screen

On Click “Reading”	Display Unlocked Reading Lesson Screen
On Click “Listening”	Display Unlocked Listening Lesson Screen
On Click “Speaking”	Display Unlocked Speaking Lesson Screen
On Click “Writing”	Display Unlocked Writing Lesson Screen

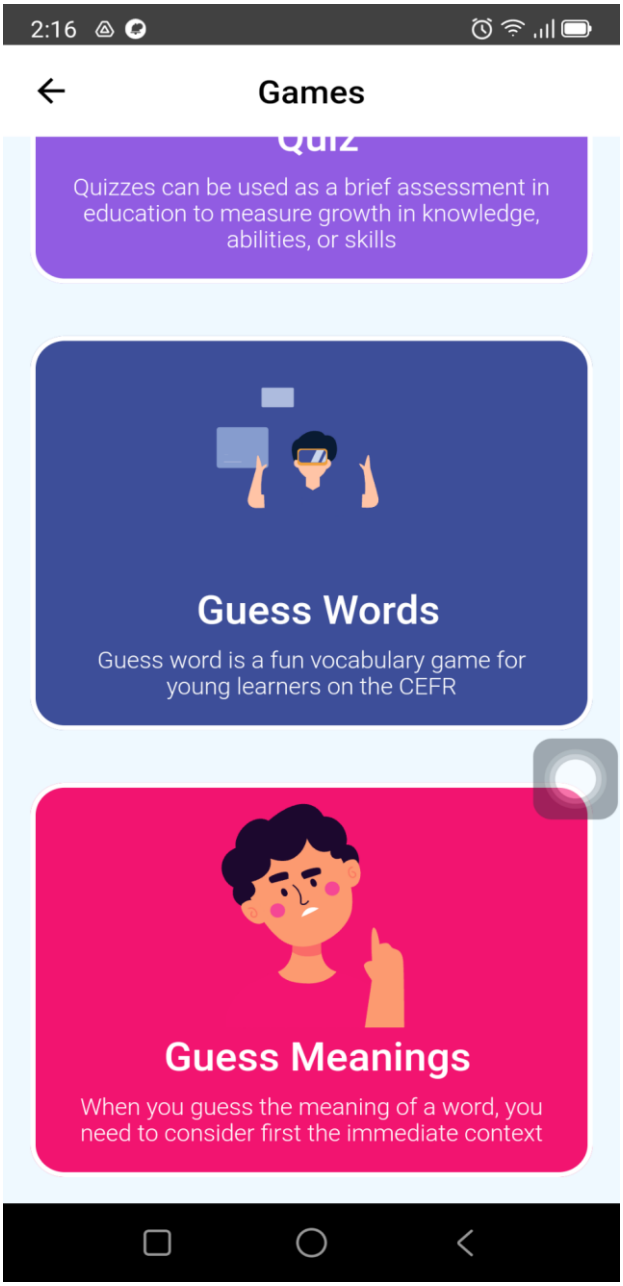

Figure 3.39. Games Screen

On Click “Quiz”	Display Quiz Mode Screen

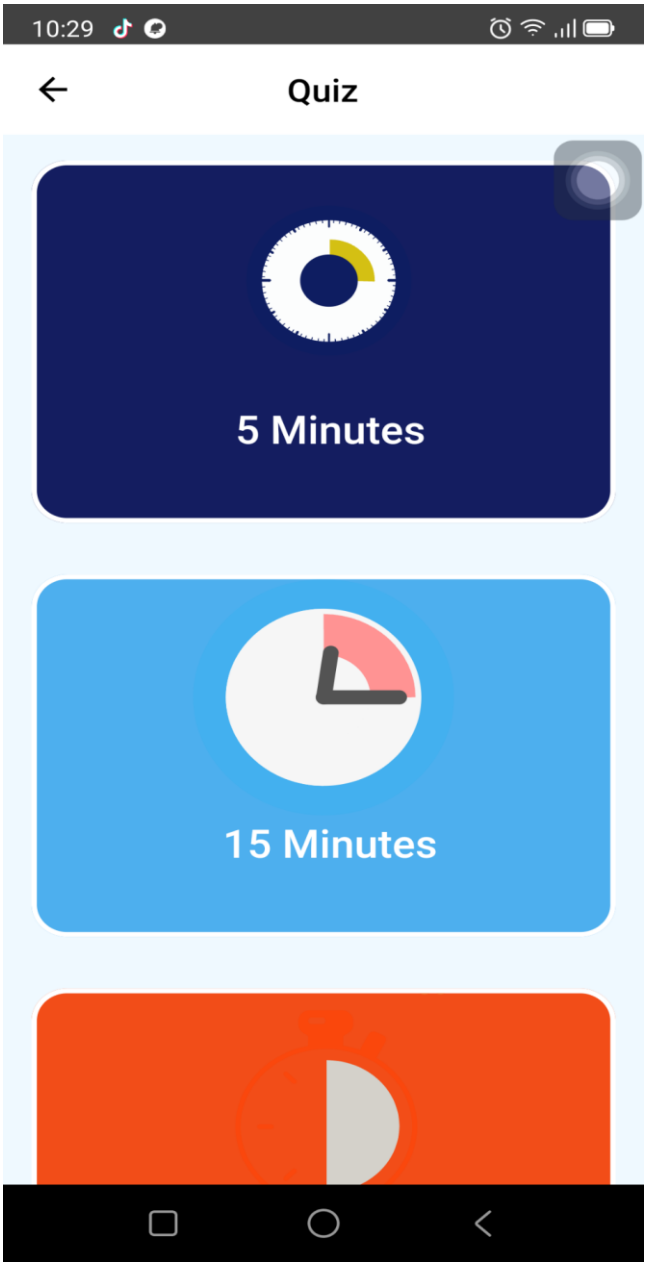

Figure 3.40. Quiz Mode Screen

On Click Each Mode	Display Quiz Content Screen

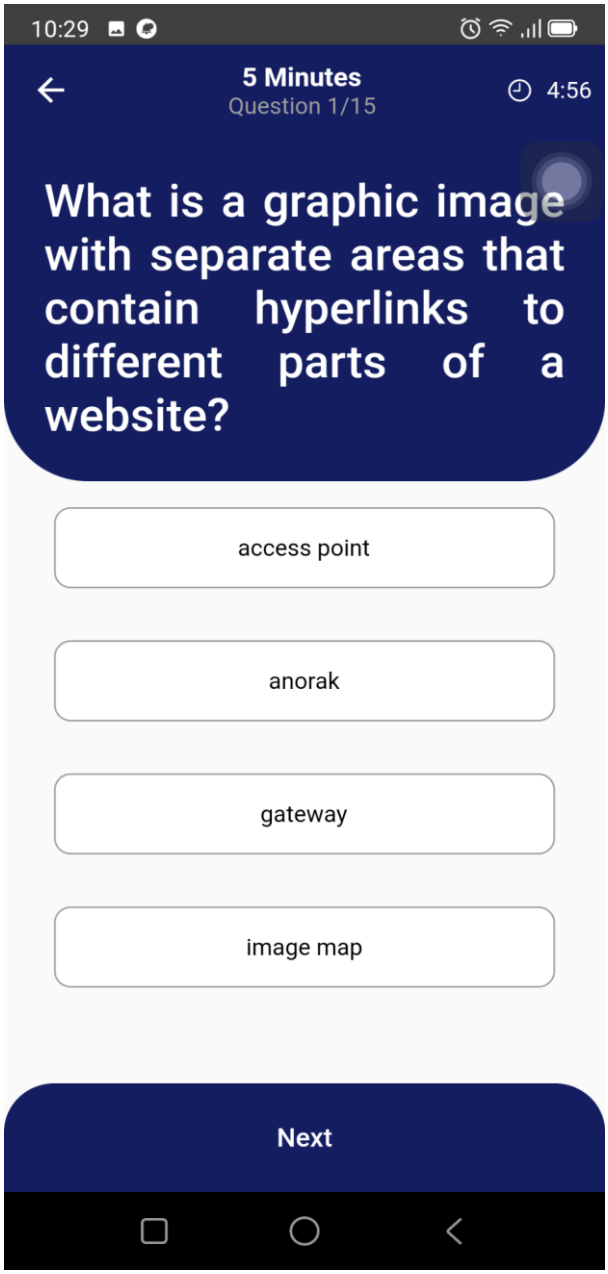

Figure 3.41. Quiz Content Screen

On Click Each Option	Display Quiz Check Answer Screen
On Click “Next” button	Show Toast “Please Select Any Option”

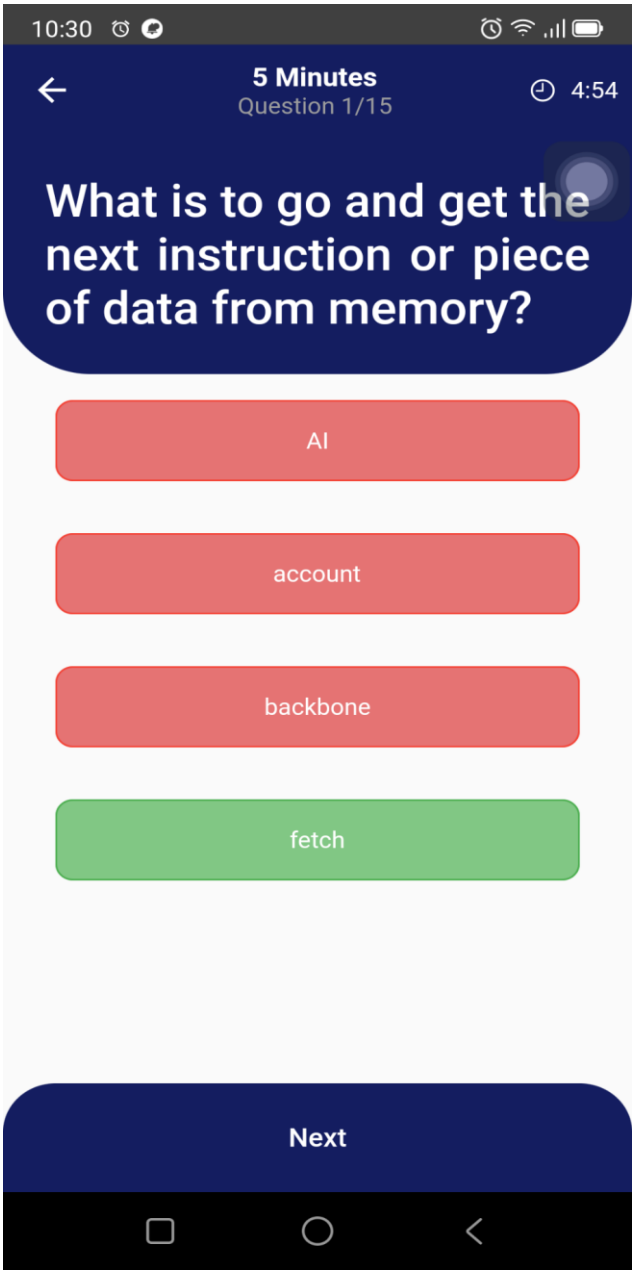

Figure 3.42. Quiz Check Answer Screen

On Click “Next” button	Display New Quiz Content Screen
Time Over/ Users Answer Until The Last Question	Display Quiz Result Box Screen

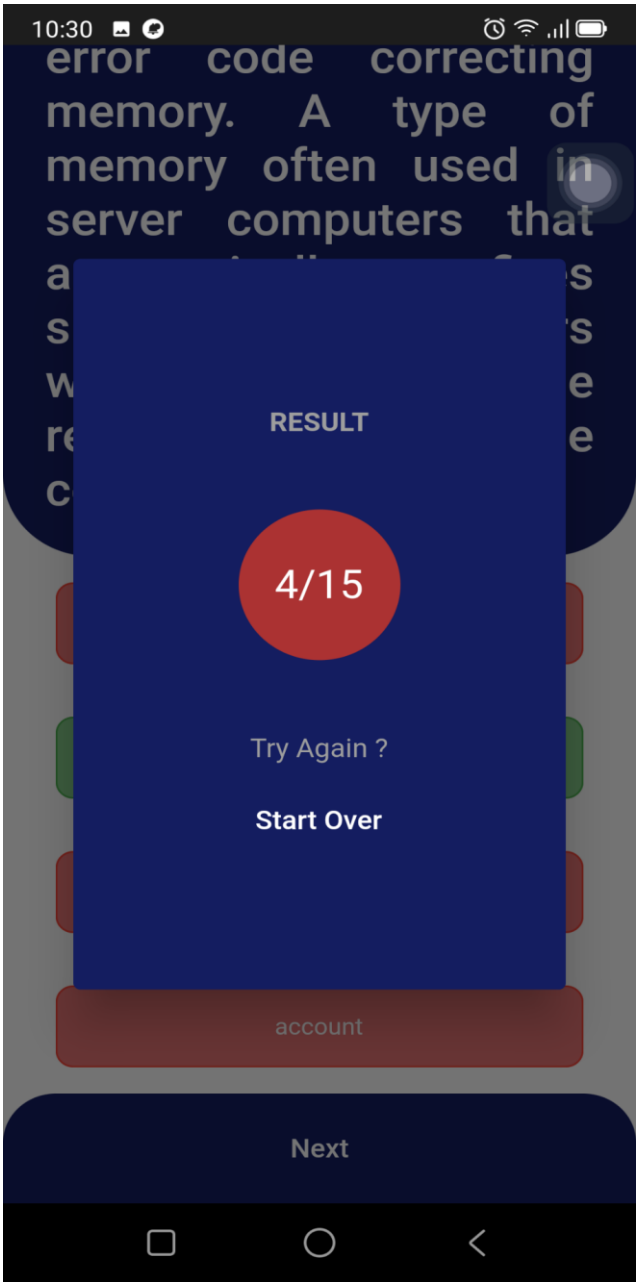

Figure 3.43. Quiz Result Box Screen

On Click “Start Over”	Display Quiz Content Screen

CHAPTER 4: IMPLEMENTATION AND RESULT

4.1. ACCESSIBILITY

Feature	Detail
Get Writing Data	Get Writing data by accessing to Writing Button in the Home Screen
Get Reading Data	Get Reading data by accessing to Writing Button in the Home Screen
Get Listening Data	Get Listening data by accessing to Writing Button in the Home Screen
Get Speaking Data	Get Speaking data by accessing to Writing Button in the Home Screen
Get Quiz Data	Get Quiz data by accessing to the Quiz Screen
Get User Data	Get User Data by accessing to Profile Screen

Table 4.1. Test Report

4.2. USER MANUAL

4.3. TEST PLAN

4.3.1. Test Report

Test Report helps formally summarize the testing activities. It should contain:

- Testing the application name and overview
- Testing the hardware and software environment
- The number of test cases executed/passed/failed

For each issue confronted, provide the following information:

- Bug description
- Bug status (open, fixed, etc)
- Bug location
- Steps to reproduce an issue

Test Suite	Passed	Failed	Blocked	Never Run	Active	Runable Percentage
Identification Module						
English for IT Test Case	31	5	0	0	0	86.11%
Web Server						
GET Requests	4	1	0	0	0	80.00%
POST Requests	4	1	0	0	0	80.00%
Application						
Listening	5	0	0	0	0	100.00%
Writing	5	0	0	0	0	100.00%
Speaking	5	0	0	0	0	100.00%
Reading	5	0	0	0	0	100.00%
Profile	1	1	0	0	0	50.00%
Interupt	3	1	0	0	0	75.00%
Deploy	1	1	0	0	0	50.00%
Security	2	1	0	0	0	66.67%
Regression	4	1	0	0	0	80.00%

Table 4.2. Test Report

4.3.2. Testing Timeline

Period	Tasks
01/06/2022 - 28/06/2022	Performance Validation
30/06/2022 - 17/07/2022	Test Case Writing
20/07/2022 - 30/07/2022	Unit Testing
01/08/2022 - 05/08/2022	System Testing
09/08/2022 - 15/08/2022	Final Testing

Table 4.3. Testing Timeline

4.4. MAINTENANCE PLAN

Priority	Initial Response Time	Contact Frequency	Description
1	2 Hours	Daily	Customers is having problems recovering mission critical data from backup and desires immediate assistance
2	3 Hours	Daily	Customers is having problems with backup or restore processing. A

			potential for loss of data exists
3	Next Business Day	Weekly	Customers is having problems with current product use that is not related to backup or restore processing.
4	Three Business Days	Weekly	Customers is having problems using a previously unused feature of the product and desires technical assistance.
5	Four Business Days	Monthly	Customers is requesting a new product feature that is not currently available

Table 4.4. Maintenance plan

4.5. SETUP AND INSTALLATION

Client	● Clone project from Github: https://github.com/alinh99/eng_for_it.git ● Install Dart and Flutter (Installation instructions located in Appendixes) ● Install Android Virtual Machine ● Run the Android Machine. ● Access project's folder ● Open command line ● Run 'flutter run' command.

Table 4.5. Setup and installation

CONCLUSION AND DISCUSSION

To be honest, I would not have gotten this far without the assistance of my instructors. So, first and foremost, please accept my heartfelt gratitude for the unfailing assistance of Mrs. Dang Thao and Dr. Vu Tran.

I struggled to develop my idea into a final product. Fortunately, I was able to solve the difficulties by reading documents and watching video lectures. Indeed, I doubt I could answer those challenges if I hadn't practiced research skills in a university setting.

The core elements of English for IT have been completed up to this moment. The system has collected enough data for analysis, and the user interface is well-designed and simple to use. Although there are still certain sections that cannot be optimized, as I mentioned in the previous chapter, I plan to enhance and maintain this website in the future.

Being able to turn my idea into a genuine product has allowed me to develop a variety of abilities. In addition to acquiring technical skills such as Flutter, Dart and so on, I gained expertise in many other areas such as planning, time management, and deployment.

REFERENCES

[1]https://www.tutorialspoint.com/sdlc/sdlc_overview.html

[2]https://datarob.com/essentials-software-development-life-cycle/

[3]https://www.linkedin.com/pulse/what-sdlc-understand-software-development-life-cycle-balajee-seo/

[4]https://medium.com/@niitwork0921/what-is-a-software-life-cycle-7ce1b214ebfc

[5]https://zenkit.com/en/blog/agile-methodology-an-overview/

[6]https://www.experienceux.co.uk/faqs/what-is-wireframing/

[7]https://careerfoundry.com/en/blog/ux-design/how-to-create-your-first-wireframe/

[8]https://www.goodcore.co.uk/blog/software-prototyping/

[9]https://sceweb.sce.uhcl.edu/helm/SWEN5432-SDLC/myfiles/TableContents/Module-9/module9page.html#:~:text=The%20prototype%20does%20not%20always,try%20them%20out%20before%20implementation.

[10]https://www.theme-junkie.com/what-is-figma/

[11]https://www.freecodecamp.org/news/git-and-github-overview/

[12]https://agiletech.medium.com/flutter-advantages-10-reasons-why-using-flutter-for-your-next-project-d8bdd065ada4

[13]https://kodytechnolab.com/blog/why-flutter-uses-dart/

[14]https://blog.back4app.com/firebase-vs/

BIOGRAPHY

Nguyễn Hoàng Lĩnh