Software Design Doc

SmartTalk

January 17, 2021

Sponsor: Dr. Okim Kang

Mentor: Fabio Santos

Version: 2.0

Team members: Joseph Vargovich (Lead), Andrew Munoz, Kehan Cao, Christian Bito-on, Malik Jones

Table of Contents

1.0 Introduction

SmartTalk is a team of 2020-2021 Northern Arizona University Computer Science capstone students working toward the development of a "Gamified Mobile Pronunciation Tutor for Language Learners," which is designed to fill a significant pedagogical gap in existing mobile language-learning technology. While current mobile language learning apps such as Memrise and Duolingo attempt to help users learn a second language, they lack a clear focus on pronunciation practice. Many language learners are thus unable to communicate with others in their target language, and in essence, this results in a lower overall quality of communication. Another issue present in current language-learning technology concerns the ethics of language learning applications; some claim to help you speak 'like a native,' however depending on the region of a country, a native speaker's accent, in other words, their pronunciation, can differ greatly. Our sponsor is Dr. Okim Kang who is the Director of the Applied Linguistics Speech Lab and a Professor at Northern Arizona. Her main research goal for this project is to analyze the nature of the accent of non-native speakers in English.

- ➢ How accent is perceived by listeners
- ➢ How accented speech is characterized linguistically,
- ➢ How the assessment of accented speech is validated through automatic systems,

SmartTalk has designed a mobile software with features specifically designed to address these issues, and ultimately offer our sponsor an innovative technological approach to mobile language learning. In order to best tackle this problem for our sponsor, we have identified key pieces of requirements that this implementation must meet. Below is a list of key pieces of requirements to successfully build what our sponsor has envisioned.

- ➢ ***High-Level Requirements***
  - **Gamification** - Ideally, we will include gamified elements to attract users and increase their interest in learning.
  - **Mobile framework -** In theory, our solution will harvest both iOS and Android applications that will be usable in a cross-platform supporting both mobile and web frameworks.

- **Web framework -** Ideally, we will have administrators be able to modify and update the data in the database on the webpage.
- **Multi-platform shared database -** Having a shared database on the mobile app will allow users to access the content in this database, and having a web application will allow the administrator to modify database content to update knowledge.
- **Automatic Speech Recognition -** ASR technology, which converts spoken words into text, is necessary to ensure the accuracy of speech recognition.

➢ ***Functional Requirements***

- Creation of three main types of tasks: Production, Perception, and Instruction Tasks
- Creation of Courses, Modules, Lessons, and Tasks through the website dashboard
- ASR usage and feedback to be used in linguistics research and analysis of language learning
- Gamification to incentivize and motivate user learning and practice

➢ ***Performance Requirements***

- Usability
- Speed of ASR Feedback
- Battery Life & Phone Resources

➢ ***Environmental Requirements***

- There are no requirements that take place in the project. We are free to use any open-source framework for mobile and web development.

This project will benefit our sponsor by developing a tool for her graduate students to use for their teaching purposes to help speakers with accents that can better communicate with others by sharpening their pronunciation skills. We hope to use these ideas to create a better, more entertaining learning experience for our users through gamification, and we also hope to assist and encourage people to learn a new language in a different way.

2.0 Implementation Overview

In regards to getting a clear understanding of our sponsor issue. SmartTalk has come up with a solution for our client and her graduate students to use our app as a tool for research purposes to analyze the nature of the accent of non-native speakers in English.

Our project involves the mobile terminal and the web terminal, so this is a multi-platform development project. Our idea is to develop mobile and web programs separately, but they are connected to the same database. The function of the web page is to update and maintain this database, and the mobile phone will call the data in the database for users to use. We decided to use Flutter to develop mobile and web pages because Flutter is good multi-platform development software. On the mobile phone, we will also use ASR technology for speech recognition to improve the performance of the mobile phone program. Finally, in the choice of database, we used firebase because of its stability and strong compatibility. Below is a visual diagram of the product we are envisioning for building for our client on how each component will be connected.

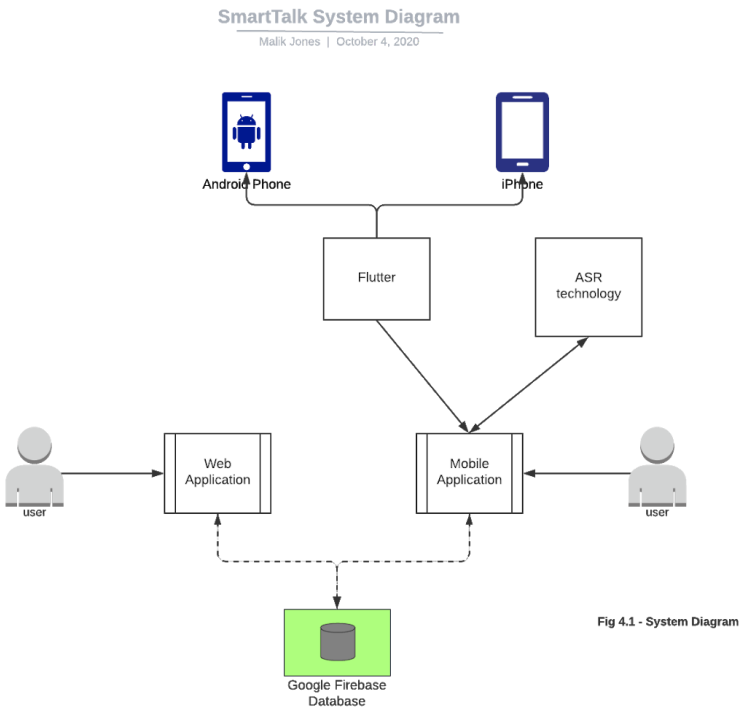
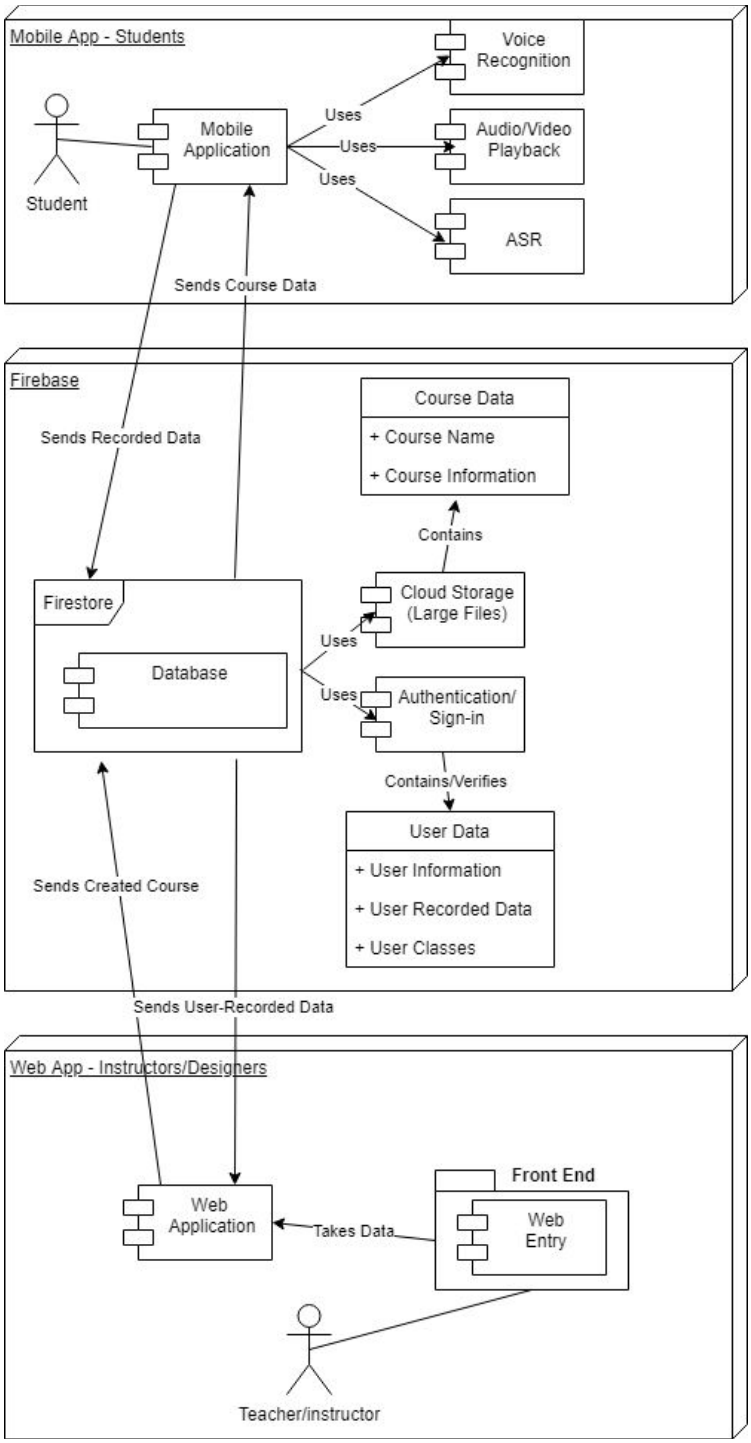


Fig 4.1 - System Diagram

3.0 Architectural Overview

By developing a solution for our sponsor in this section we will be diving deeper into the systems and how all the components will communicate with each other.

Here is a high-level overview of our system:



Overall, the system is divided into three different main components: the mobile application, Firebase, and the web application. The largest of these components, the mobile application, is where most user activity will happen. Its main function is to provide users with a platform to accomplish structured learning courses. Key functions include: signing in the user, usually a student, and getting their assigned courses; displaying the course; and recording and evaluating the user's voiced responses alongside providing feedback, either manually or automatically. There are also other functions for supplementing the user experience and preparing data analysis, chiefly a gamification element that implements achievements to track user progress, and a connection to the database component to transfer user responses.

The next component, the web application, is mainly reserved for teachers and course designers. Its core feature is to allow these users to design courses for students and also provide a platform for them to analyze student responses. Here, a front-end web portal is used to allow users to sign in, create or modify courses, assign the courses to the students and analyze for a specific course a student's recorded responses. For the last aspect, feedback can be created and sent back to the user for their review.

The last component, the Firebase database, serves as a connection between the mobile and web applications as well as storage for transferred data. For mobile application users, it sends course information created by teachers and manual feedback from them as well. It also receives user responses. For web application users, the reverse happens: the database handles the reception of created course information and the sending of user responses to the web portal. To store this information, cloud storage is used to keep the larger forms of data such as the above courses, responses, and feedback. Furthermore, the database also has functions for the authentication and sign-in of users of both applications. User information not related to responses or answers like user IDs are kept here.

Between each of the three components, most of the communication will be through the internet, meaning users will have to be connected through means such as wifi or LTE. The information types communicated through this method are course information and user response/feedback. This is accomplished through integrated firebase plugins for our platform. For the information

flow of our system, it works similar to a two-way street. There is a direct flow from the web application to the mobile application for course data, while the other directional flow is used for recorded answers. These streams of data are generally independent of each other. An exception to this is that the mobile application must have the course data to then acquire and send user responses.

For our system, we draw from a few architectural styles for our system. With the need to communicate between a web portal and a mobile application, we adopted data-centric implementation with the use of the Firebase database as a middleman, handling the transfer of data back and forth between the two. We also implemented the use of plug-ins to accomplish a number of features related to the recording and evaluation of voices. We also took some inspiration from a client-server application (in terms of concept, rather than implementation). This mainly stems from the idea that the web application and its users provide the course information needed by mobile application users to learn. Lastly, the overall system leans toward a service-oriented architecture. It provides a service for both language learners and teachers: For learners, it is a method of learning through a mobile course, while for teachers it provides them with a way to review their students' progress and work.

4.0 Module and Interface Description

Module 1 - The Firebase Backend System

This module describes the backend architecture that is encapsulated in our Cloud Firebase instance. We use three distinct modules within Firebase: Cloud Firestore, Cloud Storage, and Authentication.

Module 1.1 - Cloud Firestore

Cloud Firestore serves as the NoSQL database to store information needed within both the mobile application (module 2) and web application (module 3) and allows them to communicate with each other via the Flutter Firebase plugins. This serves as the primary backend database system for storing courses, modules, lessons, and tasks as well as the necessary feedback information for completed tasks and instructor feedback. Figure 4.1 shows the overall design of the NoSQL collections and document fields. Documents house each instance of each object in a collection. For example, the Designer collection houses multiple Designer documents, which each contain the unique info pertaining to a certain Designer user.

Each type of task will not have a separate document specified, but rather will be sorted into its necessary type of Task in the mobile application based on the taskType flag provided.

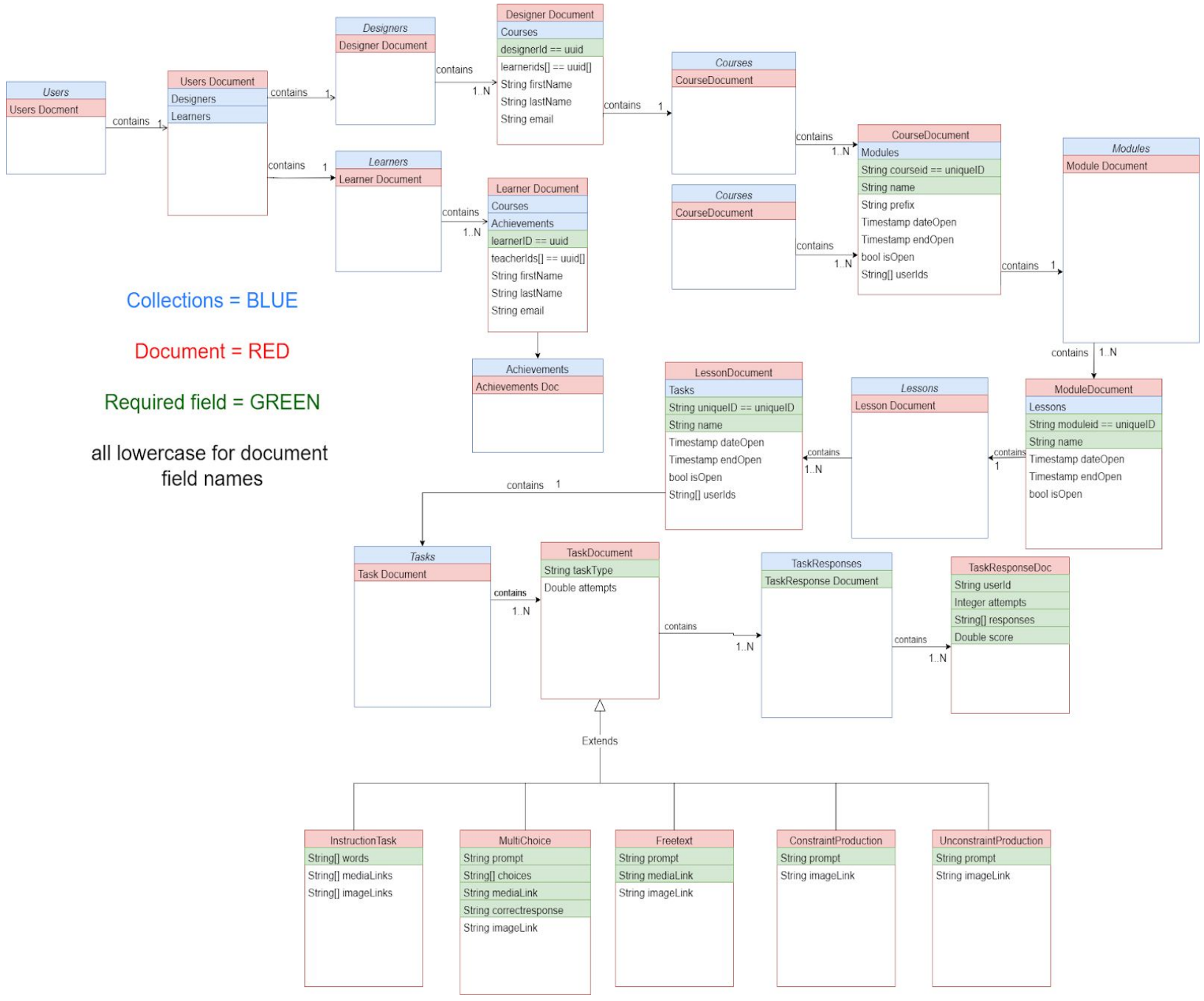


Figure 4.1 - Diagram of the noSQL Cloud Firestore database

Module 1.2 - Cloud Storage

Firebase Cloud Storage is another module in the Firebase suite. This is the dedicated storage for larger files, such as audio, video, or text files. These files will be uploaded to cloud storage and linked to created and completed tasks in the Firestore collection. These tasks will have these links stored as the medialink field. This allows the Flutter plugins to access this medialink to find the file stored in Cloud Storage to load remote images and audio. Videos are stored as youtube video links, but those can be stored as the medialink for a task and will be loaded correctly as well. This allows the specific youtube video player plugin for Flutter to find and load videos in the youtube format for perception tasks.

Module 1.3 - Authentication

Our sign in and signup of Designer and Learner accounts is handled through the Firebase Authentication service. This service allows users to sign up with an email and password that is unique to their account in order to create a unique record of their information in the backend database. This authentication adds a user to the list of registered users, as well as putting an entry with a document id that is unique to them in the Designer or Learner collection depending on which type of account they created. We are able to search for these unique ids to determine if a user is a Designer or Learner for example.

Module 2 - The Mobile Application

The mobile application, programmed completely using Flutter and platform specific code (for iOS and Android ASR integration) is the main platform used by learners to complete and access their learning materials. This component is largely responsible for pulling information from the Cloud Firebase backend, described in Module 1.

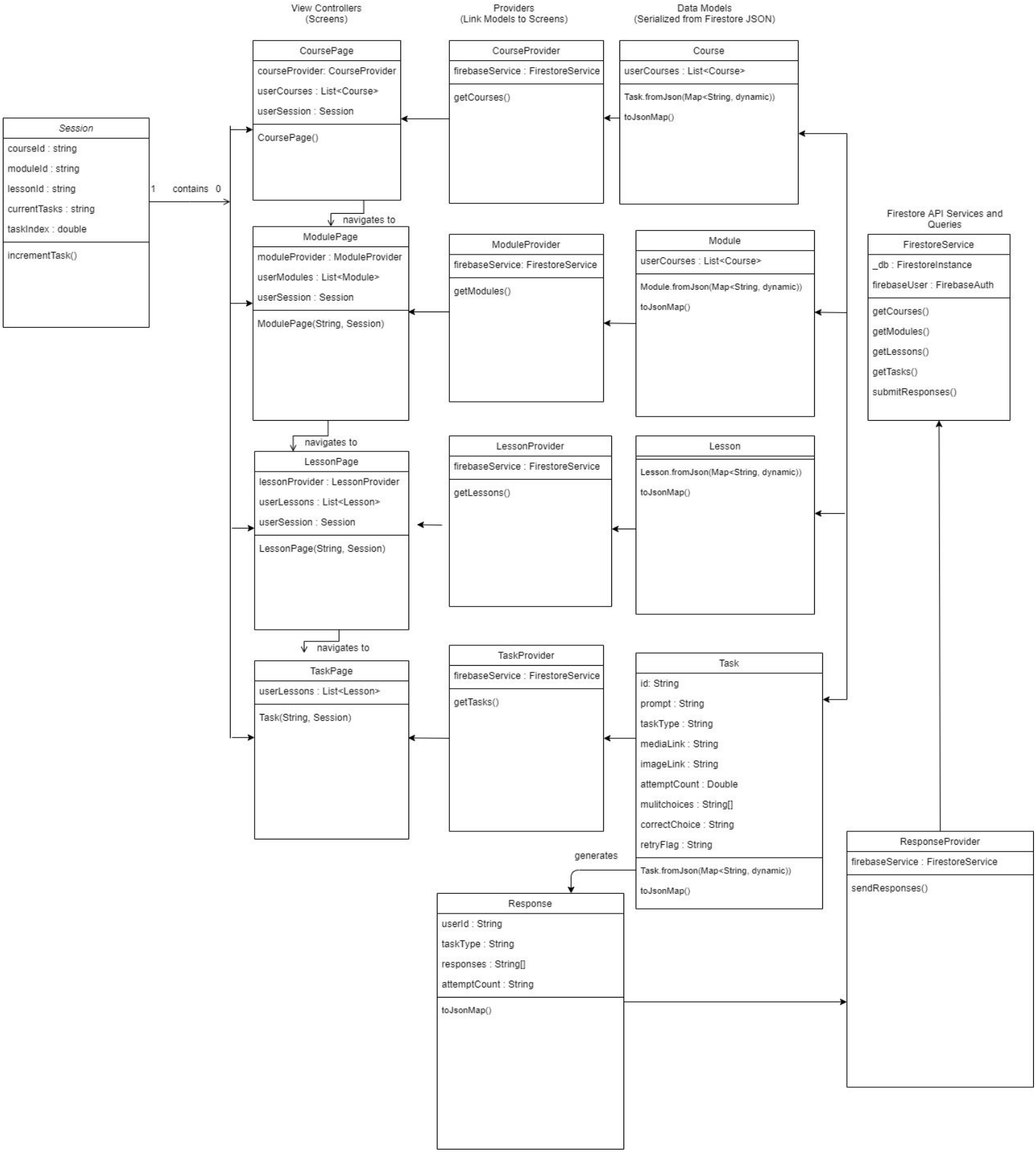


Figure 4.2 - UML Diagram of Mobile App Components

In order to load courses, Learner users will need to reference their stored Designer ids in order to find the collection holding all the courses assigned to them. Upon loading into the course page, the Learner will be presented with all the courses for the designers they subscribed to that list their LearnerId in the list of learnerIds for the course. Designer users can also access the mobile application and will have access to the courses they created on the website dashboard along with any other Designers they have subscribed to.

Figure 4.2 shows a UML diagram of the main components of the mobile application.

The main viewcontrollers (widgets in Flutter) that represent pages that navigate between each other are the CoursePage, ModulePage, LessonPage, and various types of TaskPages. Each of these pages maintains a Session object that provides information about the current Course, Module, Lesson, and Task the user has navigated to in order to read from and push to the correct Firestore collections to load each course stored.

This takes us to the providers, which are an intermediary between the viewcontrollers and the data models retrieved and serialized from the Firestore API calls and queries. These providers are useful to easily access Course, Module, Lesson, Task, and Response models that are pulled from and/or pushed to Cloud Firestore within the mobile application. Specifically, fields in the Course, Module, Lesson, Task, and Response documents need to be updated to reflect user action, such as completing a Lesson. Thus, the provider is invoked and uses an instance of the FirestoreServices class which houses the necessary FireStore queries and CRUD operations in the backend database.

Module 3 - The Web Application

The web application is being considered in Flutter through the use of their web application development platform. The web application is only accessible to users who are considered “designers”. If a “learner” attempts to access the web application they will be hit with a prompt that will direct them to make a designer account or warn them that they need to have a designer account in order to access the web application. Once entered into the web application the user who is logged in has the ability to via their current courses or create a new course. When viewing the set course list the designer will be able to edit their courses. This includes things such as

answers for their tasks and even the users who are signed up for the course. Edits take place and are stored in the firebase system. The best way to accomplish this is to use a firebase instance that will allow us to identify which user is currently signed into the web application. This will ensure we don't allow the users to edit other courses that they aren't supposed to.

On the other hand, they can also create new courses. Courses are designed to be identified by their unique ID and will be searchable through the email of the designer who designed the course. Once created and added to the database the information will then be stored in a collection that will house all of the information needed for the course. The Hierarchy of creation is Course→ Lesson → Module → Task. Various fields will be available for users to create and edit. These fields will be used to distinguish between the various courses the designer has created.

Overall the web application will be relatively simple and lightweight, allowing for even non-technological users to be able to use it to create courses. We only plan on having about 8 screens. The list is as follows, registerScreen, forgotPasswordScreen, loginScreen, homePage, createCourse, editCourse, viewCourse, viewUsers, Similar to the mobile application this web application will be created through the use of ViewControllers. This web application will also be using a series of classes designed to hold all of our functions for interacting with the database and verifying the user accounts.

Module 4 - ASR Analysis

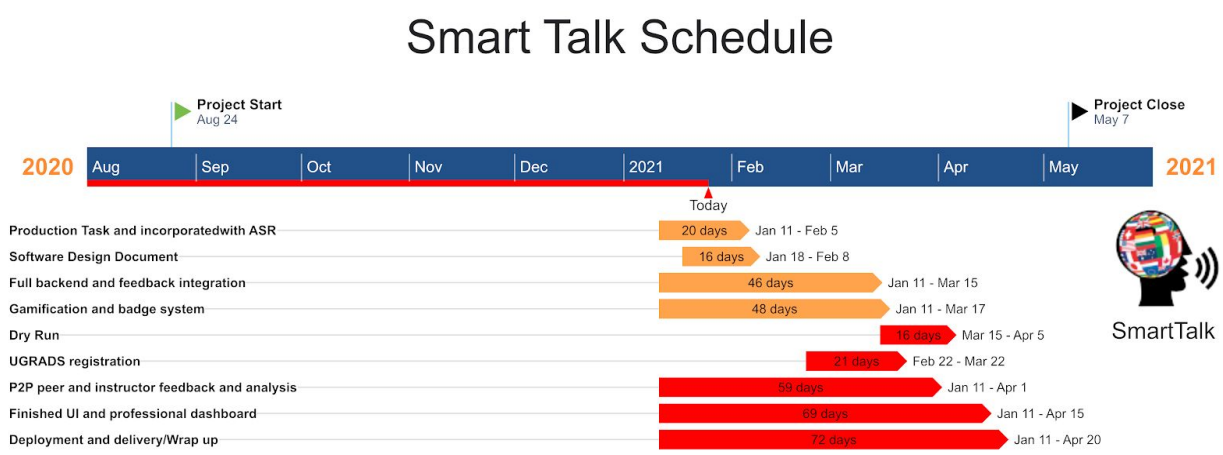
Our ASR incorporation into the application will be done through the pocketSphinx library. One main advantage of using a library such as PocketSphinx is we are able to analyze more than just the words said. Similarly to how we speak to a google assistant we take in the word said and are able to analyze it for what we think was said. However, we further examine this input by taking a look at things such as the structure of the speech. There are three models to consider while analyzing input with pockSphynx. We first analyze the acoustics, which will cover how what was said sounds. We then consider phonetics in order to map our phones to specific words. Next, we consider the context of what was said. This is a logical process that will restrict results based

on previous sounds. In total this will make up our recognition software and allow us to take in input and output it for learners and designers.

When it comes to implementation we are taking an individual approach per OS used by the phone. This is due to the library and the functionality it needs based on whichever operating system the phone is using. The process in order to access permissions to things such as the microphone and collect data from it is vastly different. We make use of flutter's ability to create and modify individualized code for different platforms in order to develop the code based on the platform. One big part of ASR development on the IOS platform is being able to use Xcode on a MAC OS system. We also have reviewed the platform restrictions in order to make sure our code is compliant with the IOS terms of service for developers. On the other hand, android is a lot less restrictive and is more flexible with allowing for development across multiple platforms.

Using this ASR will be a major part of our gamification process. We want to be able to create lessons, which are available for designers to model their courses after, that will be both fun and engaging for students. A bonus to creating this platform is we can take in and receive data that can be informative on the user's ability to learn and pick up the language they are learning. We analyze the audio samples that the user records and push it as a response to our database. Here we can use the ASR technology to analyze how they are saying the words they are practicing. We analyze things like which parts of speech they are pronouncing right and where the problem of pronunciation lies. This allows for us to create a way for designers to monitor their students and figure out if their designed courses are working. It will also allow them to adapt their teaching to find out what is working and what isn't.

5.0 Implementation Plan



There are multiple modules we need to complete for this module, which are:

- ➢ Cross-Platform database set
- ➢ Course Structure Design
- ➢ Link Mobile App to the database
- ➢ Production Task and incorporated with ASR
- ➢ Full backend and feedback integration
- ➢ IOS development debug
- ➢ Gamification and badge system
- ➢ P2P peer and instructor feedback and analysis
- ➢ Finished UI and professional dashboard
- ➢ Deployment and delivery/Wrap Up

Among these, the database task has been completed in the last semester; this semester we will optimize the use of the database. In this Gantt chart, we can see that our team is working with multiple tasks at the same time. Each module here corresponds to a different team member. The person in charge of **ASR technology** is Christian, the **full backend** is being handled by Joe and Andrew together, and Malik and Kehan are in charge of, respectively, **gamification and the IOS development and integration**. After these basic modules are completed, all members of our

team will be involved in program development work. The current stage we are in is the development and interaction of basic modules. During the preparation work last semester we discussed in detail the development direction of the project and the technology to be used with the customer and mentor. These involve ASR (speech recognition), multi-platform development, cross-platform database. Many of these technologies have never been involved before, we need corresponding members to understand and learn these technologies, and finally give feedback to the entire group on how to integrate them. Because many technologies need to be used, it is unrealistic to arrange for all team members to learn the same technology together, and therefore this is not allowed in the development cycle of the project. Therefore, different members of us are now solving the problems of different basic modules, such as the compatibility of ASR, the compatibility of multi-platform development, the allocation of database calls, and the adaptability of game modules. At present, we have made great progress in different modules, and different basic modules will soon enter the interactive testing phase. When the interactive test is completed, we will perfect the code work and complete the project development.

6.0 Conclusion

SmartTalk aims to solve the core challenges of language learning addressed by Dr. Okim Kang and her team. We seek to provide a system that will conveniently allow English language learners to practice pronunciation and perception of English. Additionally, our system will provide Dr. Kang and her team with valuable linguistics data in order to analyze and study English language learning patterns.

The overall significance of this document is to discuss the architectural design and the design for each module that we are going to be implementing into the project. This architectural design will be used as a blueprint of our final product. We will be able to see problems that might occur during development and be able to think these issues through to overcome them, rather than jumping starlight into development. Having weekly meetings with our sponsor and as a group, help to get a clearer understanding of the problems; however, the meetings are not a substitute for having valuable feedback being sent back to the user, especially for something like stressing on the correct syllable of a word. Also, another issue is the lack of gamified elements being implemented into language learning apps. These would motivate the user to keep practicing their

second languages, enjoyably as well as helping with focus on the pronunciation rather than trying to memorize a small phrase segment in that specific language. With practice, they will be able to speak fluently in their foreign languages so that a native will be able to comprehend what they are saying.

By using this blueprint as a guide for developing this product for our sponsor as a solution to her problem, we will be able to create an enjoyable way for users to sharpen their pronunciation skills. We will do this by applying gamified elements correctly to keep the user motivated. In addition, our sponsored graduate students can use the design as a teaching tool to help people with heavy accents to enhance their communication skills. SmartTalk being able to work with Dr.Kang on developing this product helps students improve their pronunciation and perception of a foreign language in an engaging, entertaining way. That way we can effectively motivate English language learners to continue on their journey to language mastery, while also providing linguistics researchers with valuable data to support their studies.