SociaLingo: Social Networking Approach for Computer-Aided Language Learning

Computer Science Honours Project – COMP 4905

Carleton University, Ottawa, ON, Canada

Shaheem Khan (100890500)

Dr. D. Deugo

TABLE OF CONTENTS

2. Introduction

Due to increased connectivity, people now travel and send messages across the globe faster than ever before. However despite being able to congregate at will, people are still having problems communicating with each other, largely due to language barriers. The majority of Canadians agree that learning a new language vastly improves employment prospects and social interactions. However, despite the discernable benefits Canadians and people in general are having trouble mastering a new language, only 17.5% of Canadians reported speaking at least two languages [1], this statistic drops significantly for less diverse countries.

The ability to communicate with each other is crucial for our advancement, currently both adults and children across the World are desperately searching for ways to learn new languages. Ever since computers became affordable people have been using various applications, an entire field of study is dedicated to finding the ideal application, called computer-assisted language learning (CALL) [2]. Countless CALL programs are available in innumerable formats but no particular program has become prominent and an universally accepted tool for language learning. This project is a consequence of extensive research in the field of computer aided language learning and aims to address some fundamental issues with recent CALL programs through social networking.

1.1 Problem

People have been using computers to learn languages since the 1960's, however computers haven't been able to completely replace humans and traditional language learning techniques. Most CALL programs can be separated into three interrelated phases, which have progressed as technology and pedagogical theories progressed [3]. The oldest stage of CALL is referred to as behaviourist CALL, behaviourist CALL focuses on traditional drill-and-practice programs, a method of instruction characterized by systematic repetition of concepts, examples and practice problems [4]. This stage of CALL is often regarded to be monotonous as some students would lose interest quickly. However drill and practise exercises work well for teaching students technique which includes basic vocabulary and sentence structure [5].

Then there is communicative CALL which is antipodal to behaviourist CALL, communicative CALL is more focused on sentence structure in speech and written communication it relies on open ended questions, instead of closed ended structured questions [6]. This sort of CALL is extremely difficult to program, since it is hard for a computer to judge subjective pieces of literature. In order to combat this problem some recent CALL programs incorporate basic

social functionality, where feedback is provided by a human teacher, in such cases the program acts as a mediator to learn languages instead of a standalone tool. These kinds of CALL programs are usually extremely expensive for the average user.

The most recent stage of CALL is integrative CALL, these programs integrate various skills of language learning (listening, speaking, writing, and reading) to provide users with more learning styles to choose from. Although this stage further complicates feedback requirements, it is currently the leading accepted pedagogical theory [7]. No CALL program fits perfectly in one of these three stages, as most call programs incorporate elements from all stages, nonetheless these stages represent several fundamental problems that are present in CALL programs. These problems are:

- Unsatisfactory Feedback System
- Lack of Flexibility
- Financial Barriers

1.2 Motivation

Since its inception, CALL research has focused on traditional drill-and-practice programs, almost all applications have some repetitive tasks that the user is required to complete on a regular basis. Drill-and-practice programs are extremely popular because they provide a simple way to learn the basic vocabulary of a language in a cost effective way. However due to their

inability to provide meaningful feedback for oral communication drill-and-practise programs are not universally accepted [8]. New developments have tried to incorporate web-based distance learning and virtual learning environments (VLE) where participants are organized into groups and share their knowledge and provide feedback [9]. However none of the current VLE solutions provide an ideal teaching experience due to cost factors.

Meanwhile social networking applications have found ingenious ways to incorporate social capital in applications, this entices users to consistently use social networking applications [10]. CALL programs haven't been able to achieve this, although applications like Livemocha [11] have been popular in the past they were primarily a paid service and were forced to shut down. This project is specifically interested in the community aspect of social networking sites. In certain circumstances social networking websites have been able to promote intellectual advancement and become a thriving environment where sharing experiences and ideas becomes rewarding and convenient.

These characteristics of social networking applications can help deviate certain problems with CALL programs. Firstly it totally eliminates the lack of flexibility in certain CALL programs as users will no longer be forced to answer repetitive structured questions, in order to progress. Adapting a social networking approach will help users to pick what aspect of a language they would like to

focus on, for example if a user is more interested in formal speech they can cater their learning to formal speech as there is no program or code stopping them, they are no longer spending hours learning basic vocabulary that they may never use or require. Adapting a social networking approach which is built around a hospitable community will also help alleviate financial barriers and help provide considerable superior feedback.

Being a social networking application it can eventually collect and store large amounts of data about languages, this can eventually employ certain big data and artificial intelligence algorithms that can help classify users data and provide feedback without any users help. A social networking approach to language learning could be able to provide instantaneous feedback without any contributions from the user. All these benefits address the core problems with CALL programs. This project also has considerable personal connotations, as a child I was able to learn three distinct languages, but now as an adult it feels almost impossible to learn new languages.

1.3 Goals

This project provides an alternative solution for people to practise and improve their language skills. People who are interested in learning new languages, are able to learn the basic vocabulary through drill-and-practise programs. However it is nearly impossible to master speech and certain writing

styles of a new language without paying relatively large sums of money for paid networking websites or private tutors. The inability to communicate efficiently is currently detrimental to the planets financial, scientific and social potential. This project will implement certain valuable characteristics of social networking applications in order to create a language learning application for intermediate learners that deviates some of the core issues of CALL programs.

1.4 Objectives

The primary objective of this project is to address the three fundamental problems with most computer aided language learning solutions. These problems are in summary the unsatisfactory feedback system, lack of flexibility and financial barriers. A viable solution to these problems is to adopt a social networking approach, therefore one of the secondary objectives of this project is to build a social networking application using Apache Tomcat [12], Java (JAX-RS) [13], MySQL [14] and Xcode [15], the end result is a social networking application called SociaLingo for the iOS platform that can send and receive, voice and text messages, that can help users practise and improve their language skills, whilst addressing the three identified problems.

1.5 Outline

The primary function of this report is to describe the SociaLingo application and analyze how it improves upon traditional CALL programs. This report starts by providing a brief introduction into computer aided language learning, then it introduces various language learning applications and their drawbacks. It also describes the social features of SociaLingo and how they solve the identified problems. Lastly this report will provide a short conclusion and identify possible future features that can improve the application.

2. Background

In this project I will be frequently referencing computer aided language learning (CALL) concepts. CALL still lacks research methods and a clear theoretical foundation. However the development of CALL can be divided into behaviourist, communicative and integrative as the three distinct phases. The oldest stage of CALL is behaviourist CALL, behaviourist CALL focuses on traditional drill-and-practice programs, a method of instruction characterized by systematic repetition of concepts, examples and practice problems. While communicative CALL was created because some users felt that the drill and practice programs, did not allow enough authentic communication to be of much value.

Communicative CALL basically tries to get rid of drill-and-practise exercises by focusing on using the language rather than analysis of the language, and grammar is taught implicitly rather than explicitly. This sort of CALL is extremely difficult to program, since it is hard for a computer to judge subjective pieces of literature. While integrative CALL was made possible due two important technological developments of the last decade - multimedia computers and the Internet. Integrative CALL programs integrate various aspects of language learning (listening, speaking, writing, and reading) to provide users with more learning styles to choose from. Integrative also includes internet based

distant learning platforms these are often expensive, regardless integrative CALL is currently the leading accepted pedagogical theory. No CALL program fits perfectly in one of these three stages, as most call programs incorporate elements from all stages, nonetheless these stages represent several fundamental problems that are present in CALL programs.

The consumer market for computer aided language learning applications is quite saturated, a quick search will often leave potential learners confused and disappointed. However in recent times two applications have managed to stand out due to their popularity and media exposure. The applications are Duolingo [16] and Rosetta Stone [17], in this project I will be referencing these applications along with Livemocha [18] frequently, as they are excellent case studies to the problems I have identified. Duolingo is a free language-learning platform that includes a language-learning website and mobile applications for various platforms. Duolingo has managed to make language learning a game with progression based objectives and achievements, they provide users a easy way to learn the basic vocabulary and phrases of a language through drill-and-practise exercises, with options to practise and track your progress through several categories of material.

If I were to categorize Duolingo it would be a Behaviourist CALL program with Integrative features, although it takes advantage of Integrative features by providing an internet based solution and by combining all aspect of language

(listening, speaking, writing, and reading). Duolingo is still largely based around Behaviourist CALL with their drill-and-practice exercises and systematic repetition of concepts. Although drill-and-practise applications like Duolingo have managed to help people learn the basic vocabulary of a language, research shows that they don't provide intermediate learner the tools to master the language [18]. A personal anecdote that corroborates this research is that when I tried to improve my French skills using Duolingo I was placed on an advanced stage due to my prior experience. However, I quickly found that the exercises I was required to do often seemed irrelevant to my personal needs. I also noticed that I wasn’t able to practise different literary styles and patterns of speech, I attribute this flaw due to the difficulty in programming such a system. This aligns with research in computer aided language learning.

The largest and most popular language learning application is Rosetta Stone Language Learning. The software uses images, text, and sound to teach words and grammar by spaced repetition, without translation. Rosetta Stone calls its approach Dynamic Immersion [19], which uses several images and provides the corresponding sounds and text to learn the words in the native language. For example, the software will show a student four photographs. A native speaker will then make a statement that describes one of the photographs, and then that statement is printed on the screen; the student then chooses the photograph that the speaker described. In another variation, the student completes a textual

description of a photograph. Like Duolingo this system works fairly well for basic vocabulary. Rosetta Stone also provides a virtual learning environment (VLE) [20] based distant learning solution, a VLE is a web-based platform for the digital aspects of courses of study. Rosetta Stone offers learners direct access to native speakers who can set objectives and help them practise and improve their language skill. This can be a great tool to practise pronunciation specially in a social or formal setting this is something I found missing in Duolingo. Although Rosetta Stone is arguably the best possible language learning software, it's pricing structure has limited its outreach. For widely spoken languages Rosetta Stone costs $249 for the complete package, while additional features like the VLE platforms costs an extra $249 for a 24 month subscription [21]. Rosetta Stone is also inflexible with their price structure as each individual software packages cost $124. Each software package is varied by difficulty the complete package contains 5 different levels, this means if you are an intermediate user you are forced to spend more on levels you may not need, this adds to an exorbitant pricing structure.

Rosetta Stone would be classified as an Integrative CALL program, as it tries integrate the teaching of language skills into tasks or projects with the use of multimedia technology. Like most Integrative CALL programs Rosetta Stone is expensive and inflexible, this urged me to look into Livemocha, which was fairly influential in designing SociaLingo. Livemocha was an online language learning

community, providing a platform for speakers to interact with and help each other learn new languages. Livemocha at some point had approximately 11 million monthly users [22], however on April 22, 2016 after being acquired by Rosetta Stone a few years prior, Livemocha closed access to its website [23]. Leaving thousands of daily users without a community to learn languages. Livemocha was constricted by its limited features such as the inability of share audio recordings natively, in Livemocha users had to use third party websites to share their audio recordings, Livemocha also lacked a “game” aspect which is now popular in apps like DuoLingo. SociaLingo is a natural progression of Livemocha.

3. Approach

I started my project by researching various CALL theories and language learning softwares. Through research and personal experience I found that current CALL programs can help you learn the basic vocabulary of a language. However, an intermediate user of a language often have trouble improving and practising their language skill without three major problems. These problems are:

- Financial Barriers
- Unsatisfactory Feedback System
- Lack of Flexibility

Therefore I decided to focus my project to help intermediate users improve their language skills. In order to devise viable solutions for these problems I have designed an Integrative CALL program with social networking features. Before I could start building the software, I had to decide on various software platforms. I chose to design the user interface on a mobile platform because of the availability and portability of mobile phones. Portability is especially important because it allow the user to practise and improve their language skills on the go. For this project I have programed in Swift using Xcode, however a future Android and web based application is also possible thanks to the web server. For the web server I built my own API's using Jersey Java JAX-RS running on an Apache Tomcat server. While the backend database is a MySQL database. In the following sections I will outline my approach to various aspects of the application.

3.1 Mobile Application

The SociaLingo mobile application is based around the distribution and creation of content in different languages. SociaLingo provides native speakers with content both writings and audio recordings produced by students of their language for feedback. SociaLingo also provides language learners a platform to write and record themselves practicing a language. This sort of community based CALL solution allows intermediate users to practise and improve their language skill without paying for expensive tutors or VLE course advisors. The mobile application is built around community and is entirely flexible regarding content category, i.e. the student can choose which aspect of the language he would like to practise. While the feedback system is based around community self moderation, users are able to flag any feedback they deem inappropriate or insufficient, which can then be handled by a moderator any user can apply to become moderate which include perks like increased exp points. The most important aspect of the mobile application is its retrieve function stored in the web server, the retrieve function runs a Hibernate query on its database. My approach for this query was based around engaging new users whilst rewarding experienced and helpful users. The mobile application makes use of Xcode's native localizations features as well as a third party library called LocalizeSwift [24] which allows the user to switch languages without restarting the application.

The iOS application is programmed using the Model-View-Controller (MVC) software pattern. All class are grouped according to their functionality. The three groups are *Menu, DraggableCardsUI* and *Tools.* In Figure 3.0a and 3.0b, you can see the *Menu* group along with their UML information. All classes in *Menu* are UIViewController's and don't inherit any custom classes so unlike conventional UML diagrams the arrows represent their sequence not inheritence. Each class represents a view in the iOS applications and the class names are descriptive of their functionality. Most of the networking functionality in this group is done by overriding key UIViewController methods such viewDidLoad, viewDidAppear and viewDidDisappear. Some classes also require the UITableViewDelegate and UITableViewDataSource these classes are required for UITableView functionality. AVAudioPlayerDelegate and AVAudioRecorderDelegate allows for audio recording and playback. Stories uses a custom cell and uses a custom .xib file and UITableViewCell class.

In Fig 3.0c you can see the *DraggableCardsUI* group*,* it contains legacy Objective C code from a previous project and is reprogramed for SociaLingo. onViewDidLoad both HelpWriting and HelpSpeaking UIViewControler change their view to DraggableViewWritingBackground and DraggableViewSpeakingBackground, theses classes allow for draggable cards.

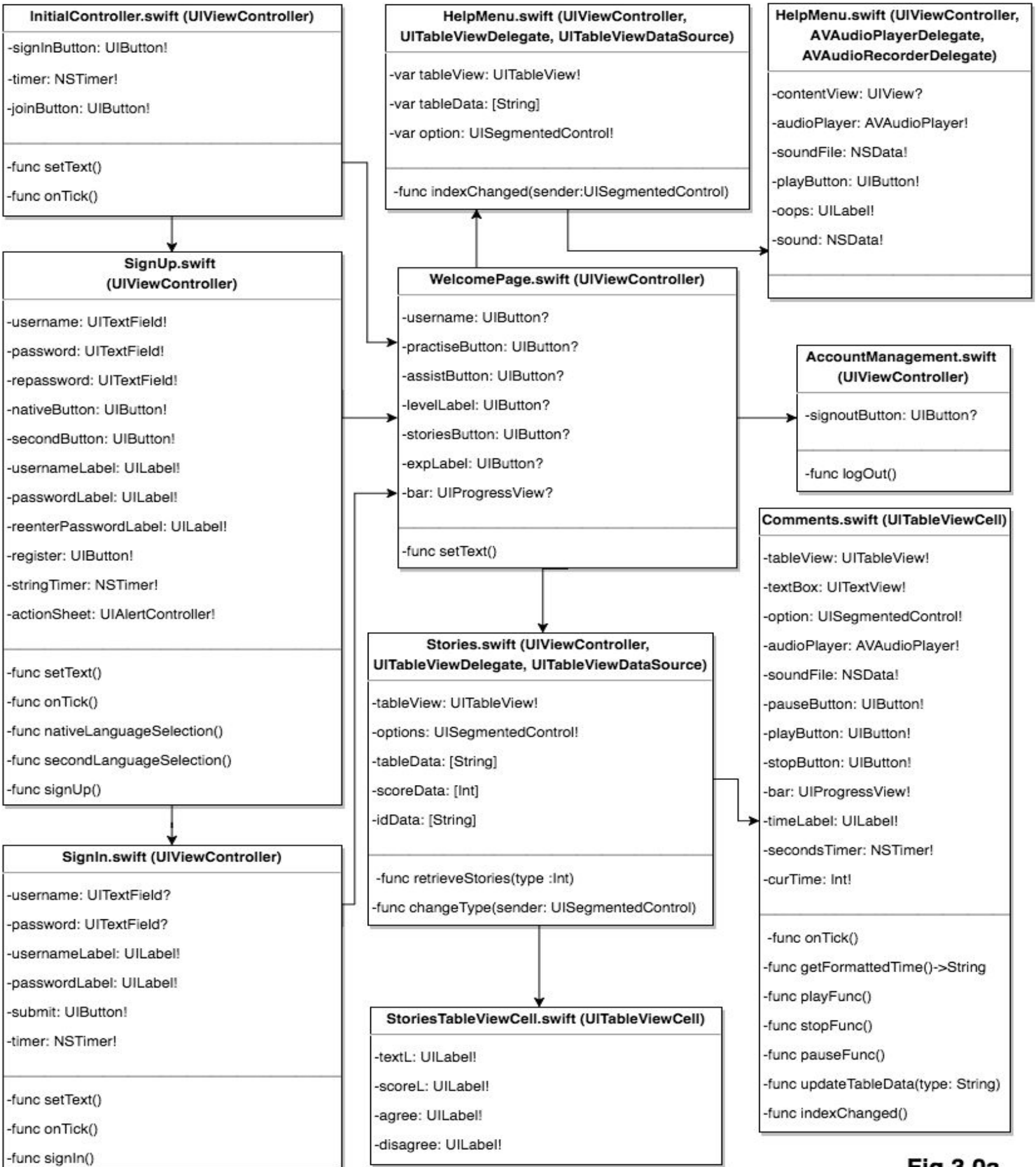
InitialController.swift (UIViewController)
-signInButton: UIButton!
-timer: NSTimer!
-joinButton: UIButton!
-func setText()
-func onTick()
HelpMenu.swift (UIViewController, UITableViewDelegate, UITableViewDataSource)
-var tableView: UITableView!
-var tableData: [String]
-var option: UISegmentedControl!
-func indexChanged(sender:UISegmentedControl)
HelpMenu.swift (UIViewController, AVAudioPlayerDelegate, AVAudioRecorderDelegate)
-contentView: UIView?
-audioPlayer: AVAudioPlayer!
-soundFile: NSData!
-playButton: UIButton!
-oops: UILabel!
-sound: NSData!
SignUp.swift (UIViewController)
-username: UITextField!
-password: UITextField!
-repassword: UITextField!
-nativeButton: UIButton!
-secondButton: UIButton!
-usernameLabel: UILabel!
-passwordLabel: UILabel!
-reenterPasswordLabel: UILabel!
-register: UIButton!
-stringTimer: NSTimer!
-actionSheet: UIAlertController!
-func setText()
-func onTick()
-func nativeLanguageSelection()
-func secondLanguageSelection()
-func signUp()
WelcomePage.swift (UIViewController)
-username: UIButton?
-practiseButton: UIButton?
-assistButton: UIButton?
-levelLabel: UIButton?
-storiesButton: UIButton?
-expLabel: UIButton?
-bar: UIProgressView?
-func setText()
AccountManagement.swift (UIViewController)
-signoutButton: UIButton?
-func logOut()
Comments.swift (UITableViewCell)
-tableView: UITableView!
-textBox: UITextView!
-option: UISegmentedControl!
-audioPlayer: AVAudioPlayer!
-soundFile: NSData!
-pauseButton: UIButton!
-playButton: UIButton!
-stopButton: UIButton!
-bar: UIProgressView!
-timeLabel: UILabel!
-secondsTimer: NSTimer!
-curTime: Int!
-func onTick()
-func getFormattedTime()->String
-func playFunc()
-func stopFunc()
-func pauseFunc()
-func updateTableData(type: String)
-func indexChanged()
Stories.swift (UIViewController, UITableViewDelegate, UITableViewDataSource)
-tableView: UITableView!
-options: UISegmentedControl!
-tableData: [String]
-scoreData: [Int]
-idData: [String]
-func retrieveStories(type :Int)
-func changeType(sender: UISegmentedControl)
SignIn.swift (UIViewController)
-username: UITextField?
-password: UITextField?
-usernameLabel: UILabel!
-passwordLabel: UILabel!
-submit: UIButton!
-timer: NSTimer!
-func setText()
-func onTick()
-func signIn()
StoriesTableViewCell.swift (UITableViewCell)
-textL: UILabel!
-scoreL: UILabel!
-agree: UILabel!
-disagree: UILabel!

Fig 3.0a

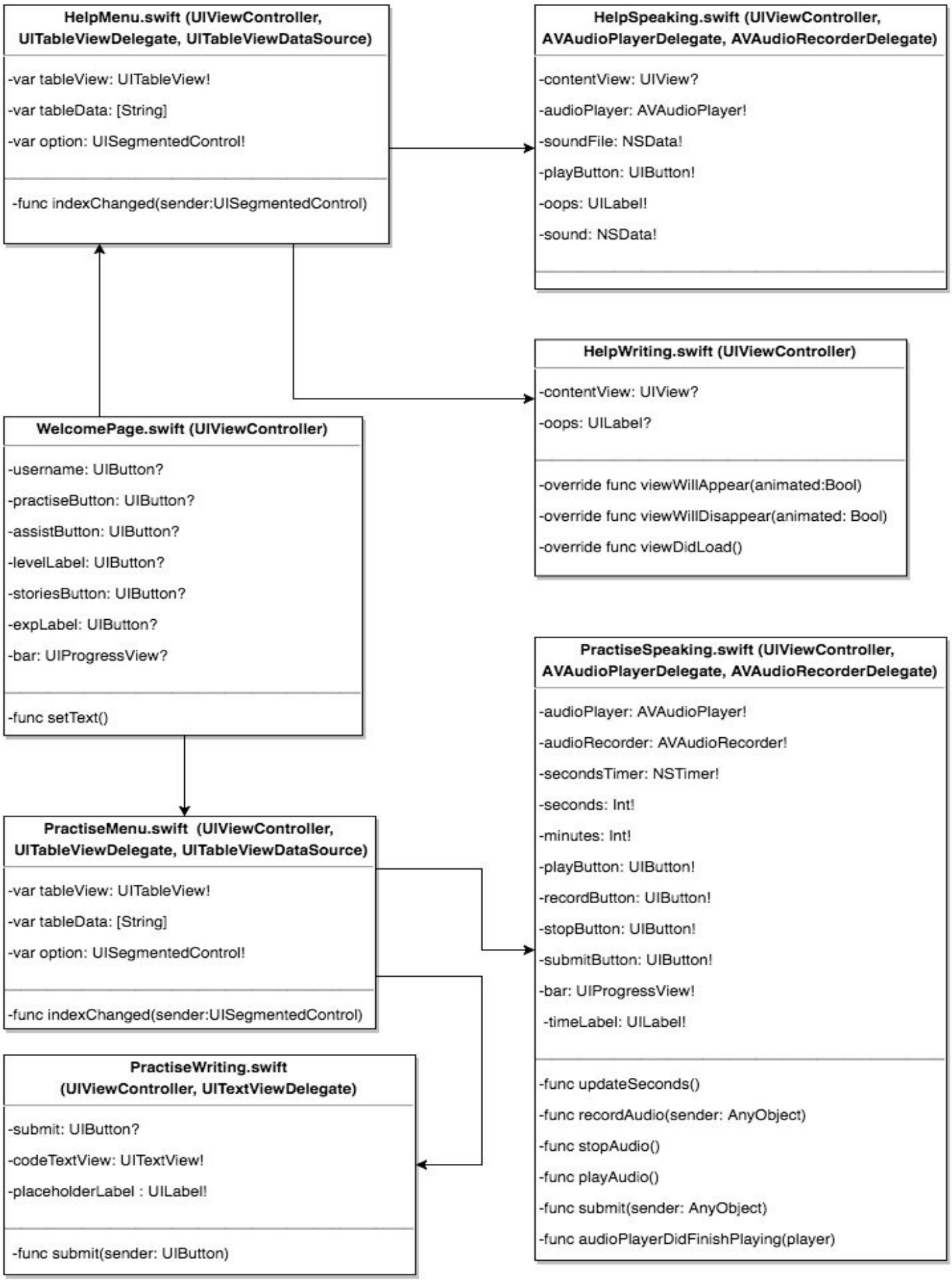
HelpMenu.swift (UIViewController, UITableViewDelegate, UITableViewDataSource)
-var tableView: UITableView!
-var tableData: [String]
-var option: UISegmentedControl!
-func indexChanged(sender:UISegmentedControl)
HelpSpeaking.swift (UIViewController, AVAudioPlayerDelegate, AVAudioRecorderDelegate)
-contentView: UIView?
-audioPlayer: AVAudioPlayer!
-soundFile: NSData!
-playButton: UIButton!
-oops: UILabel!
-sound: NSData!
HelpWriting.swift (UIViewController)
-contentView: UIView?
-oops: UILabel?
-override func viewWillAppear(animated:Bool)
-override func viewWillDisappear(animated: Bool)
-override func viewDidLoad()
WelcomePage.swift (UIViewController)
-username: UIButton?
-practiseButton: UIButton?
-assistButton: UIButton?
-levelLabel: UIButton?
-storiesButton: UIButton?
-expLabel: UIButton?
-bar: UIProgressView?
-func setText()
PractiseSpeaking.swift (UIViewController, AVAudioPlayerDelegate, AVAudioRecorderDelegate)
-audioPlayer: AVAudioPlayer!
-audioRecorder: AVAudioRecorder!
-secondsTimer: NSTimer!
-seconds: Int!
-minutes: Int!
-playButton: UIButton!
-recordButton: UIButton!
-stopButton: UIButton!
-submitButton: UIButton!
-bar: UIProgressView!
-timeLabel: UILabel!
-func updateSeconds()
-func recordAudio(sender: AnyObject)
-func stopAudio()
-func playAudio()
-func submit(sender: AnyObject)
-func audioPlayerDidFinishPlaying(player)
PractiseMenu.swift (UIViewController, UITableViewDelegate, UITableViewDataSource)
-var tableView: UITableView!
-var tableData: [String]
-var option: UISegmentedControl!
-func indexChanged(sender:UISegmentedControl)
PractiseWriting.swift (UIViewController, UITextViewDelegate)
-submit: UIButton?
-codeTextView: UITextView!
-placeholderLabel : UILabel!
-func submit(sender: UIButton)

Fig 3.0b

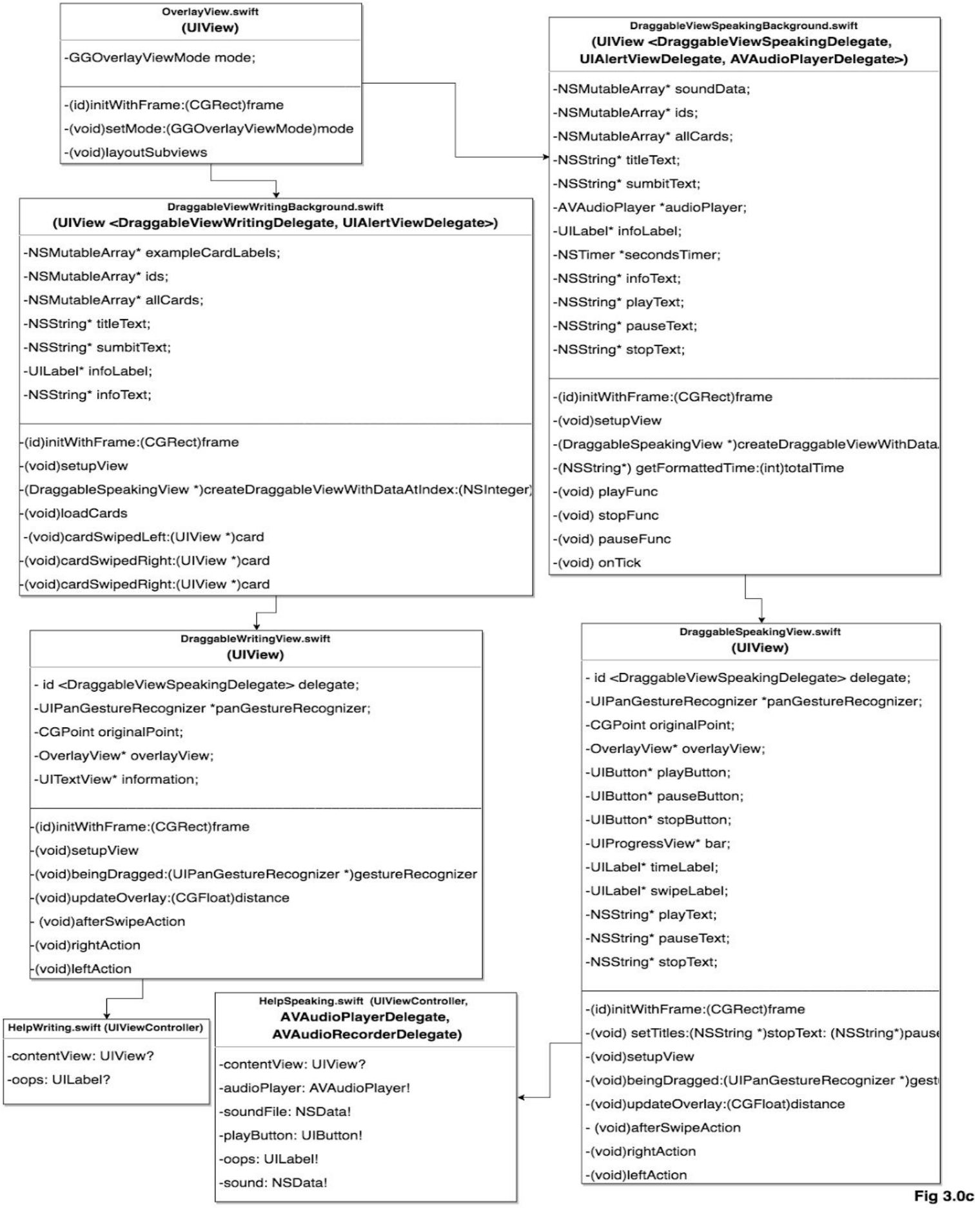
OverlayView.swift
(UIView)
-GGOverlayViewMode mode;
-(id)initWithFrame:(CGRect)frame
-(void)setMode:(GGOverlayViewMode)mode
-(void)layoutSubviews
DraggableViewSpeakingBackground.swift
(UIView <DraggableViewSpeakingDelegate, UIAlertViewDelegate, AVAudioPlayerDelegate>)
-NSMutableArray* soundData;
-NSMutableArray* ids;
-NSMutableArray* allCards;
-NSString* titleText;
-NSString* sumbitText;
-AVAudioPlayer *audioPlayer;
-UILabel* infoLabel;
-NSTimer *secondsTimer;
-NSString* infoText;
-NSString* playText;
-NSString* pauseText;
-NSString* stopText;
-(id)initWithFrame:(CGRect)frame
-(void)setupView
-(DraggableSpeakingView *)createDraggableViewWithData
-(NSString*) getFormattedTime:(int)totalTime
-(void) playFunc
-(void) stopFunc
-(void) pauseFunc
-(void) onTick
DraggableViewWritingBackground.swift
(UIView <DraggableViewWritingDelegate, UIAlertViewDelegate>)
-NSMutableArray* exampleCardLabels;
-NSMutableArray* ids;
-NSMutableArray* allCards;
-NSString* titleText;
-NSString* sumbitText;
-UILabel* infoLabel;
-NSString* infoText;
-(id)initWithFrame:(CGRect)frame
-(void)setupView
-(DraggableSpeakingView *)createDraggableViewWithDataAtIndex:(NSInteger)
-(void)loadCards
-(void)cardSwipedLeft:(UIView *)card
-(void)cardSwipedRight:(UIView *)card
-(void)cardSwipedRight:(UIView *)card
DraggableWritingView.swift
(UIView)
- id <DraggableViewSpeakingDelegate> delegate;
-UIPanGestureRecognizer *panGestureRecognizer;
-CGPoint originalPoint;
-OverlayView* overlayView;
-UITextView* information;
-(id)initWithFrame:(CGRect)frame
-(void)setupView
-(void)beingDragged:(UIPanGestureRecognizer *)gestureRecognizer
-(void)updateOverlay:(CGFloat)distance
- (void)afterSwipeAction
-(void)rightAction
-(void)leftAction
DraggableSpeakingView.swift
(UIView)
- id <DraggableViewSpeakingDelegate> delegate;
-UIPanGestureRecognizer *panGestureRecognizer;
-CGPoint originalPoint;
-OverlayView* overlayView;
-UIButton* playButton;
-UIButton* pauseButton;
-UIButton* stopButton;
-UIProgressView* bar;
-UILabel* timeLabel;
-UILabel* swipeLabel;
-NSString* playText;
-NSString* pauseText;
-NSString* stopText;
-(id)initWithFrame:(CGRect)frame
-(void) setTitles:(NSString *)stopText: (NSString*)pause
-(void)setupView
-(void)beingDragged:(UIPanGestureRecognizer *)gest
-(void)updateOverlay:(CGFloat)distance
- (void)afterSwipeAction
-(void)rightAction
-(void)leftAction
HelpWriting.swift (UIViewController)
-contentView: UIView?
-oops: UILabel?
HelpSpeaking.swift (UIViewController,
AVAudioPlayerDelegate,
AVAudioRecorderDelegate)
-contentView: UIView?
-audioPlayer: AVAudioPlayer!
-soundFile: NSData!
-playButton: UIButton!
-oops: UILabel!
-sound: NSData!

Fig 3.0c

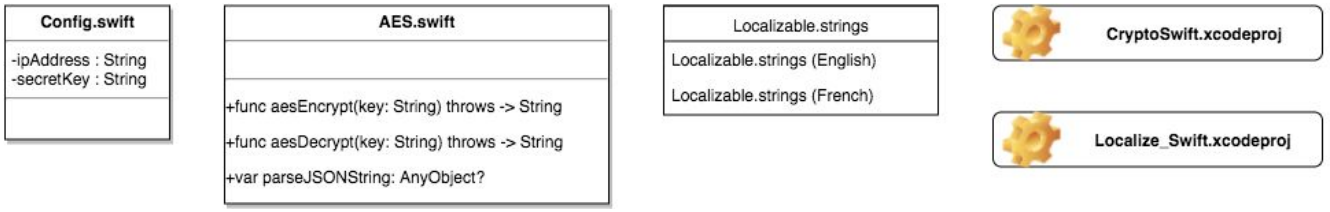

Fig 3.0d

In Fig 3.0d you can see the *Tools* group and third party libraries. The Config.swift class contains the IP Address for networking and the secretKey for AES encryption and decryption, while AES.swift class contains the actual functions to encrypt and decrypt passwords these functions require the CryptoSwift [25] a third party library. While Localizable.strings allows for seamless language support thanks to LocalizeSwift, adding new languages in SociaLingo just requires adding new Localizable.strings file for the language.

3.2 Database

The SociaLingo web server connects to a MySQL database and requires five tables for all its functionality. In figure 3.1 you can see the database schema.

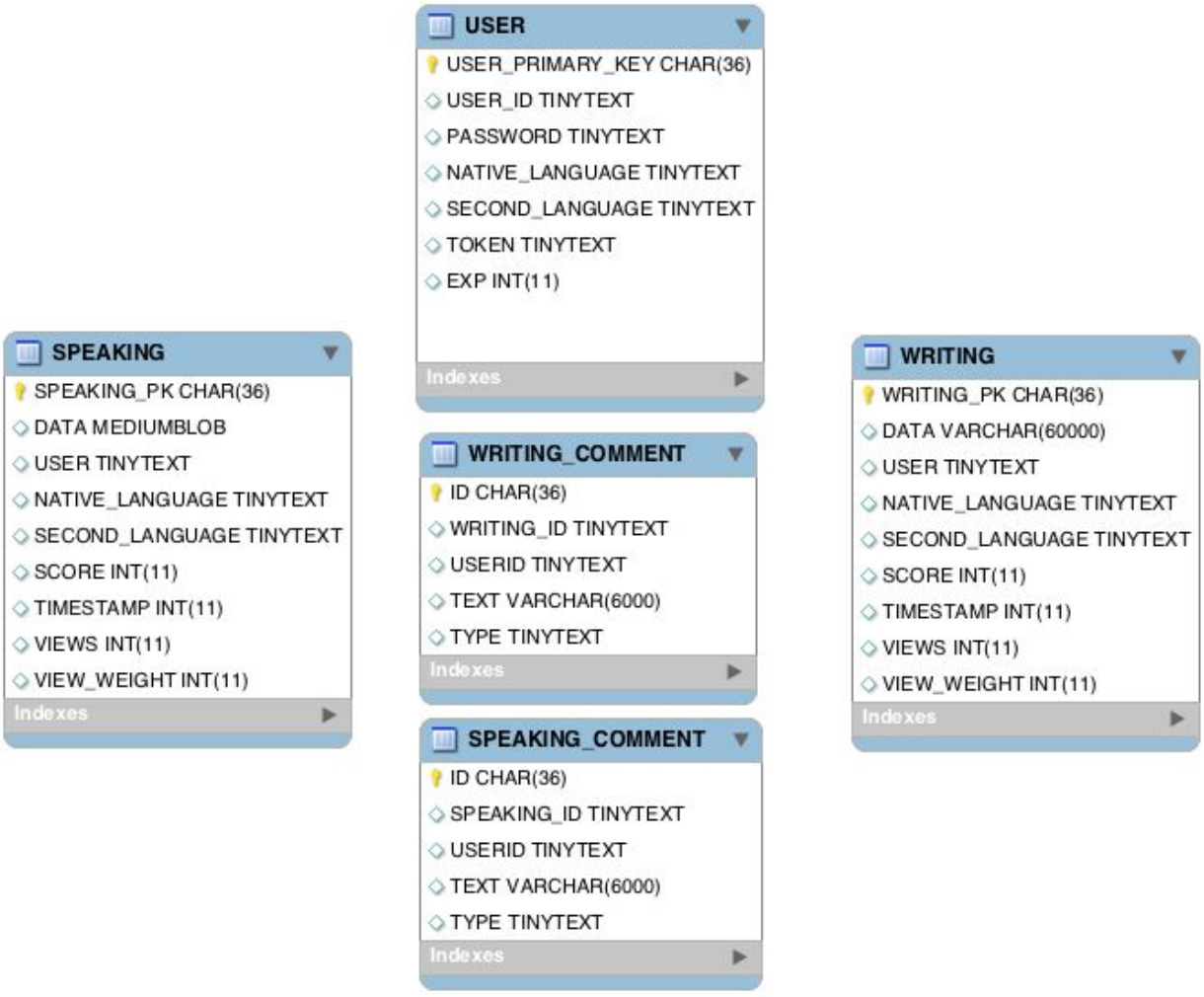

Fig 3.1

The five tables are USER, SPEAKING, WRITING, SPEAKING_COMMENT and WRITING_COMMENT. Most of the database architecture is fairly trivial however there are a few key relationships and data fields essential for SociaLingo. In the USER table, EXP field is used to facilitate the “game” and retrieve aspect of SocaiLingo which I will talk about in the next chapter. While the PASSWORD field is stored using AES 128 Encryption and the TOKEN key is also encrypted and used for validation. The SPEAKING and WRITING tables are very important for SociaLingo. They store both writings and speeches, the writings data is stored using a VARCHAR(60000) which allows for approximately 1000 word, while the speeches data is stored using a MEDIUMBLOB which allows for 16MB of data or 1:30 minutes of audio recordings. The MEDIUMBLOB is converted from and to NSData in Swift using base64 encoding and decoding [26]. While both tables contain a SCORE field which is upvotes-downvotes, the tables also contain VIEWS and VIEW_WEIGHT these fields are essential for the retrieve algorithm in the web server. The SPEAKING and WRITING tables also store the USER foreign key the connects to the USER table. Lastly the SPEAKING_COMMENT and WRITING_COMMENT tables are used to store feedback, data is stored in the TEXT field and is limited by 100 words, these table also contain SPEAKING_ID and WRITING_ID which foreign keys that connect to the SPEAKING and WRITING table respectively.

3.3 Web Server

The web server makes use of RESTful Webservices using JAX-RS and is programmed in Java. The web server contains three resources, UserResource, WritingResource and SpeakingResources. These resources connect to the MySQL database using Hibernate for Java. The server is an Apache Tomcat server and has cross-platform support, it can easily support an Android or Web based client. I chose Tomcat due to its Java support, this allowed for innumerable library support. The server is very important for my application and handles all data requests from the client. It also contains all of the important Hibernate algorithms/queries the most important queries are the SELECT_WRITING_ALGO and SELECT_SPEAKING_ALGO these queries are stored in persistence.HibernateDatabaseWritingManager and persistence.HibernateDatabaseSpeakingManager respectively. See Fig 3.2 and Fig 3.3 for the queries.

```
private static String SELECT_SPEAKING_ALGO = "from Speaking as speaking where
speaking.user != ? and speaking.secondLanguage = ? and speaking.speakingPK not in
(select speakingID from SpeakingComment as sb where sb.userID = ?) ORDER BY
speaking.viewWeight ASC";
```

Fig 3.2

```
private static String SELECT_WRITINGS_ALGO = "from Writing as writing where
writing.user != ? and writing.secondLanguage = ? and writing.writingPK not in (select
writingID from WritingComment as wc where wc.userID = ?) ORDER BY
writing.viewWeight ASC";
```

Fig 3.3

These queries give the users other students content in their native language whilst avoiding the user's own content and any content that has already given feedback. Each query is limited by 10 and are based on field called viewWeight which is calculated and updated using the following equations

$$userWeight = | (totalNumberOfUser/2) - (playerRankAccordingToExp) |$$

$$viewWeight = userWeight - views$$

Fig 3.4

These equations ensure that new users and more experienced users get the most exposure while older content gets pushed aside for newer content. The content.viewWeight and user.EXP fields are recalculated after each feedback. Most other queries are quite simple, they are mostly select queries to ensure logon/logoff validation and content/feedback submission. Validation is done using an encrypted token which is stores a timestamp and some user data. The token is required for all web server requests a copy of the token along with some additional data is also stored client-side in NSUserDefaults tables

3.4 Architecture

The final architecture of SocaiLingo is summarized in Fig 3.5

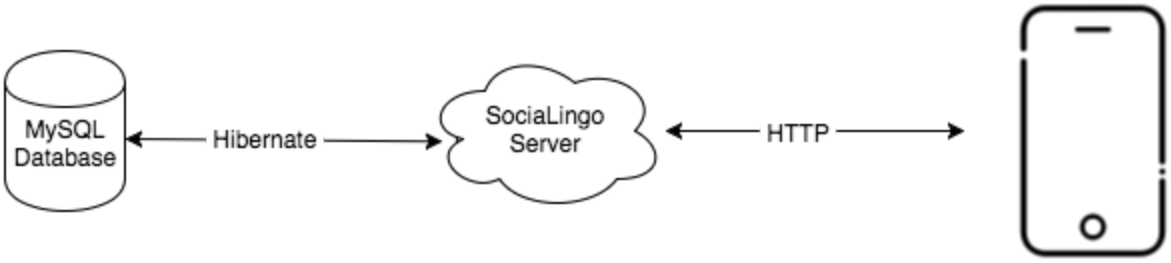

Fig 3.5

The iOS application sends and receives all its content from the SociaLingo server. The Server acts as mediator between the iOS application and the MySQL database which stores all the data. Requests between the application and the server is handled with HTTP. While requests between the SociaLingo Server and MySQL database is handled by Hibernate.

4. Results/Validation

The end product of my project is a social networking application for the iOS platform. When the user first opens the application he will notice that the main menus of my application rotate between the applications languages every 3 seconds, currently the system supports English and French (See Figure 4.1).

Fig 4.1

This kind of animation is also present in the sign in and sign out page. In order to sign up a user needs to provide a username, a password, a native language and a language they want to practise, after signing up all menus appear in the native language they had selected. Once the user has signed in, they are greeted by a Welcome page see Fig 4.2.

Fig 4.2

In Fig 4.2 you can also see the user's experience points, these points are crucial for the retrieve algorithm and provide a "game" and achievement aspect which was missing in Livemocha. One of the problems I have identified with current CALL solutions are their lack of flexibility. Most current CALL programs are extremely rigid, which leaves intermediate users stuck and unable to practise

specific aspects of a language. In Fig 4.3 you can see the practise menus for SociaLingo which are free from restrictions and allows the user to practice any aspect of the language.

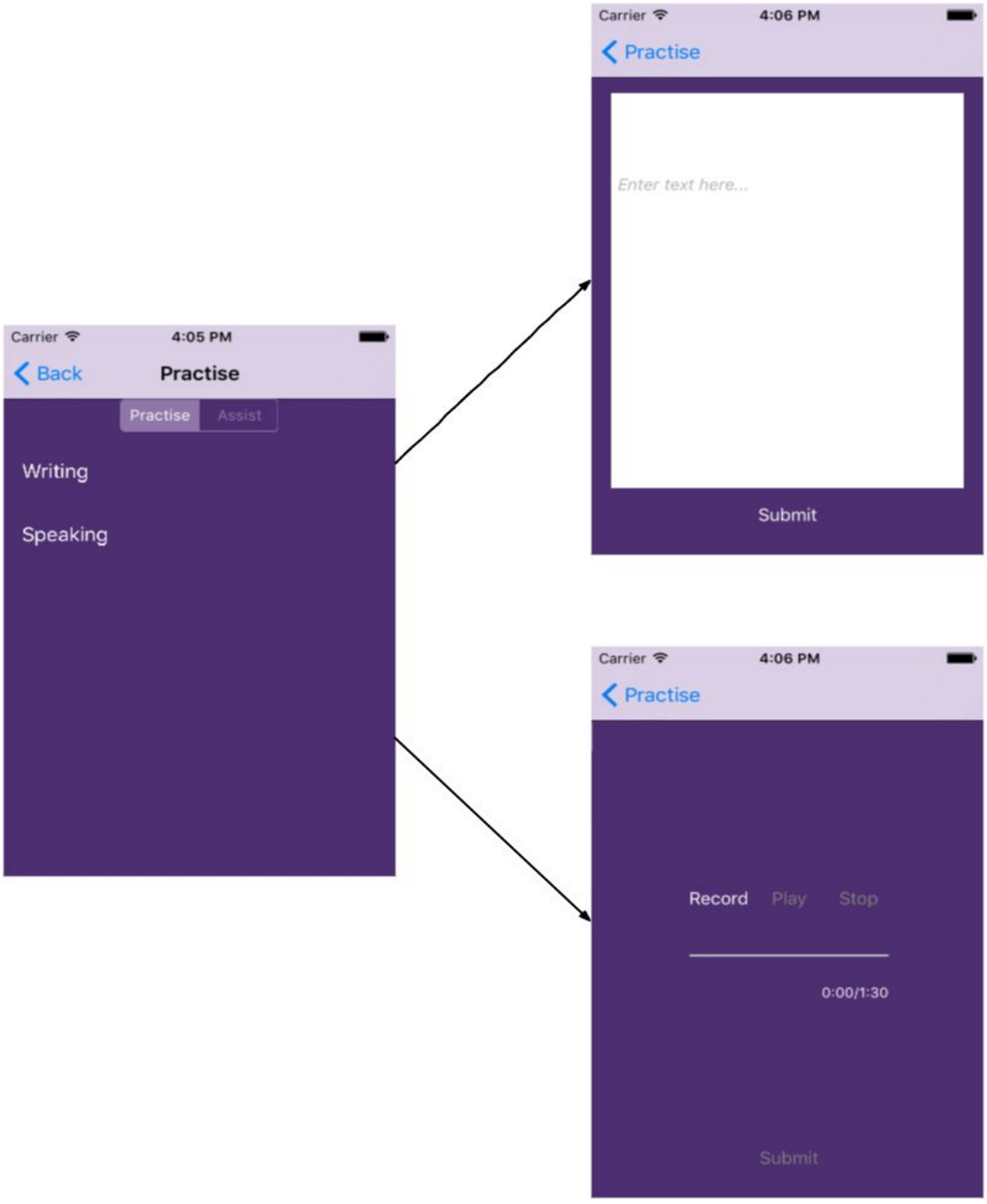

Fig 4.3

Users are able to practise both writing and speaking, writings are limited to 1000 words and speeches are limited to 1:30 mins. The current system allows for varied literary styles and speech patterns, so intermediate users can practise their language skills according to their requirements. Unlike other CALL programs SociaLingo doesn't require the user to perform any repetitive task and exercises that he doesn't need. The other major issue with current CALL programs is their unsatisfactory feedback system. Most CALL programs rely on an automated question and answer feedback system, where the correct answer to an exercise is stored in the system and the program just has to retrieve the answer. SociaLingo's feedback system is unstructured and based on social contribution by other users who happen to be native speakers of the language. In Fig 4.4 you can see the feedback menus of the

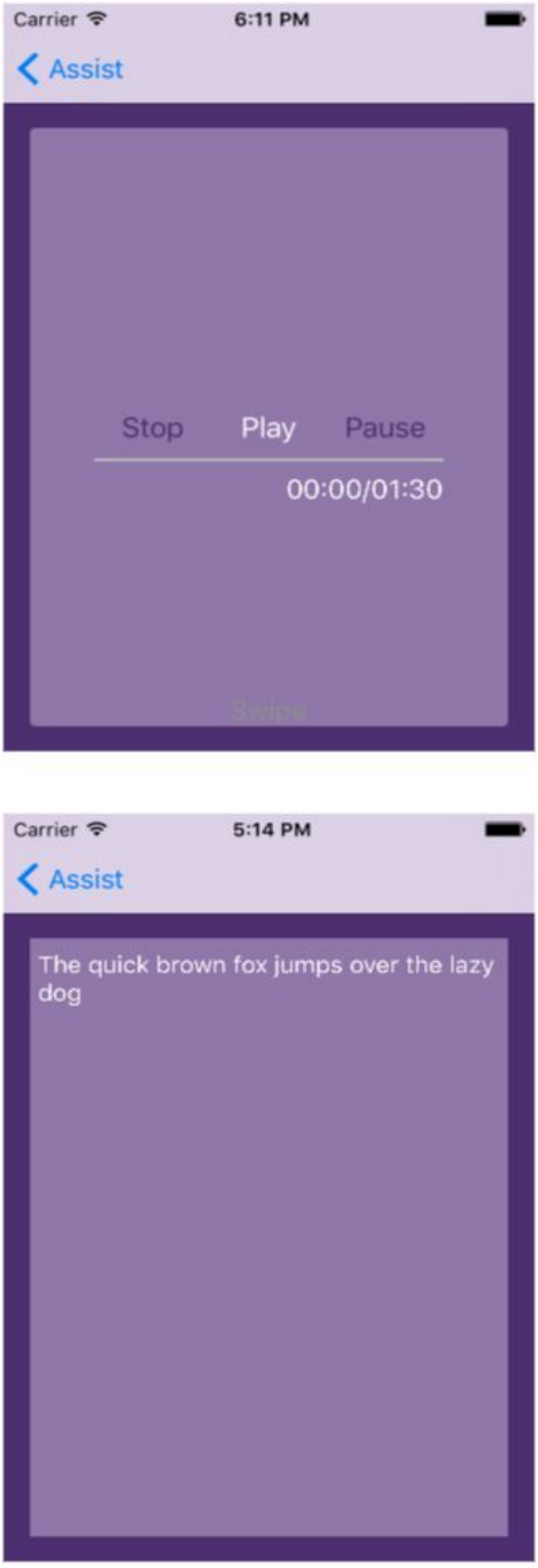

Fig 4.4

application. The feedback menus are based on cards which you can swipe left or right (see Fig 4.4). A swipe right counts as upvote and a swipe left as a downvote. After the user has voted he can leave comments on the content that the user has just read or heard (see Fig 4.5). The user can only swipe once he has heard the complete audio recording in order limited spam and incomplete feedback. Unlike conventional CALL systems, this system allows for constructive feedback the user can expect for more comprehensive feedback on things like grammar, vocabulary and pronunciation. The feedback system is built with the ability to add moderators who can monitor comments and block users for extra experience points. Any users can currently apply to become a moderator in the Account Management page, once

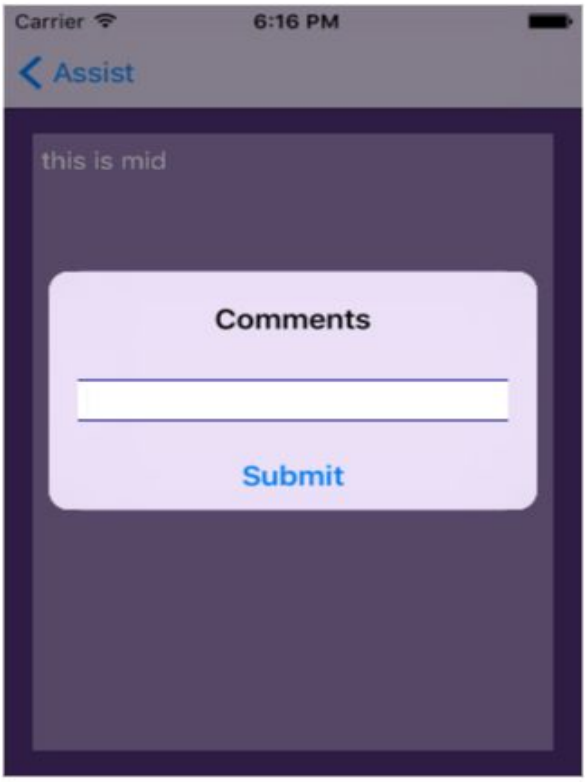

Fig 4.5

the user is verified he is given restricted access to the SpeakingComment and WritingComment tables where he can view the comments and flag them for removal. Regular users can check their feedback in the “Your Stories” page, seen in Fig 4.6. Here the user can review his past submissions and see other users comments which can be sorted by upvotes and downvotes.

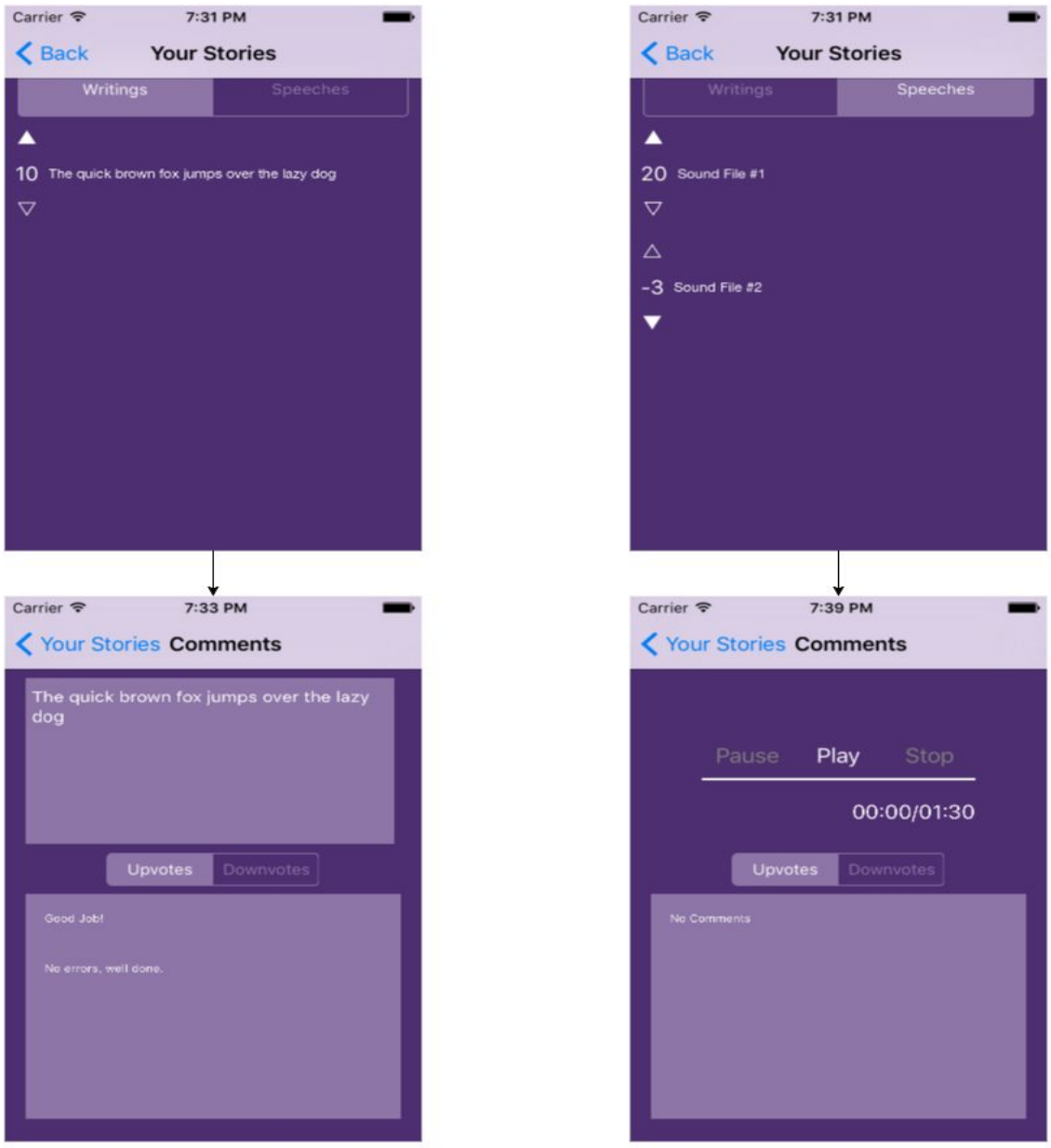

Fig 4.6

SociaLingo will be published as a free application on the app store this will get rid of the financial constraints most application impose.

5. Conclusion

SociaLingo is the ideal application for intermediate language learners, people who have mastered basic vocabulary and grammar can now easily practise and improve their language skills. The primary objectives of this project have been meet as unlike current CALL options, SociaLingo provides language learner an affordable and flexible solution that can provide dynamic feedback. By adopting a social networking approach I have managed to build an application that achieves the primary objectives. The feedback system of SociaLingo is extremely flexible and adaptable to different kinds of content both speaking and writing is natively supported. SociaLingo also provide flexibility in terms of pricing structure and type of contents, users are able to cater their material according to their needs. SociaLingo will be uploaded to App Store as a free application and will not rely on tutors or language instructors thereby ensuring a low cost solution that can be provided for free. Overall SociaLingo shows that adapting a social networking approach to language learning may be beneficial.

5.1 Future Work

SociaLingo can benefit from certain features and attachments both at its present state and once it becomes a thriving community. At its present state SociaLingo can benefit from more platforms specifically an Android and a Web

based application. SociaLingo can also benefit from a targeted marketing plan and more language support. The SociaLingo web server also needs to be hosted and the iOS application uploaded to the app store. Once the social network is a thriving environment, I can also use the feedback information to in theory create an AI that can provided dynamic feedback without any users involvement.

6. References

[1] Canada. Statistics Canada. Social and Aboriginal Statistics Division.Linguistic Characteristics of Canadians. By Jean-Pierre Corbeil. Statistics Canada, n.d. Web. <http://www12.statcan.gc.ca/census-recensement/2011/as-sa/98-314-x/98-314-x2011001-eng.cfm#a2>.

[2] Levy, M. (1997) CALL: Context and conceptualization. Oxford: Oxford University Press.

[3], [4] Torat, Bamrung. "COMPUTER-ASSISTED LANGUAGE LEARNING: AN OVERVIEW." COMPUTER-ASSISTED LANGUAGE LEARNING: AN OVERVIEW. University of Warwick, 2001. Web. 13 Aug. 2016. <http://web.warwick.ac.uk/CELTE/tr/ovCALL/booklet1.htm>.

[5] Mohammed, Zakieh, and Demand Media. "The Advantages & Disadvantages of Practice & Drills in Teaching." The Advantages & Disadvantages of Practice & Drills in Teaching. Seattlepi, n.d. Web. 14 Aug. 2016. <http://education.seattlepi.com/advantages-disadvantages-practice-drills-teaching-3685.html>.

[6], [7], [8] Warschauer, Mark. "Computer Assisted Language Learning: An Introduction." Multimedia Language Teaching. Tokyo: Logos International, n.d. 3-20. Print. Fotos S

[9] Weller, Martin. Virtual Learning Environments: Using, Choosing and Developing Your VLE. London: Routledge, 2007. 4-5. Print.

[10] Ellison, Nicole B., Charles Steinfield, and Cliff Lampe. "Journal of Computer-Mediated Communication." The Benefits of Facebook “Friends:” Social Capital and College Students’ Use of Online Social Network Sites. Wiley, n.d. Web. 14 Aug. 2016. <http://onlinelibrary.wiley.com/doi/10.1111/j.1083-6101.2007.00367.x/pdf>.

[11], [18] Livemocha Info. Livemocha Info. Livemocha, n.d. Web. 14 Aug. 2016. <http://www.lincc.org/pub/livemocha-info.pdf>.

[12] "Apache Tomcat®." - Apache Tomcat 7 Software Downloads. N.p., n.d. Web. 14 Aug. 2016. <https://tomcat.apache.org/download-70.cgi>.

[13] "Building RESTful Web Services with JAX-RS - The Java EE 6 Tutorial." Building RESTful Web Services with JAX-RS - The Java EE 6 Tutorial. N.p., n.d. Web. 14 Aug. 2016. <http://docs.oracle.com/javaee/6/tutorial/doc/giepu.html>.

[14] "MySQL." MySQL. N.p., n.d. Web. 14 Aug. 2016. <https://www.mysql.com/>.

[15] "Xcode 8." Xcode 8. Apple Inc., n.d. Web. 14 Aug. 2016. <https://developer.apple.com/xcode/>.

[16] "Learn a Language for Free." Duolingo. N.p., n.d. Web. 14 Aug. 2016. <https://www.duolingo.com/>.

[17] "Official Rosetta Stone® - Learn a Language Online - Learn Spanish." Official Rosetta Stone® - Learn a Language Online - Learn Spanish. N.p., n.d. Web. 14 Aug. 2016. <http://www.rosettastone.eu/>.

[18] Vesselinov, Roumen, and John Grego. Duolingo Effectiveness Study. Rep. N.p.: n.p., n.d. Web. <https://s3.amazonaws.com/duolingo-papers/other/vesselinov-grego.duolingo12.pdf>.

[19], [20] "Ready to Dive In?" How Rosetta Stone Works. N.p., n.d. Web. 14 Aug. 2016. <http://www.rosettastone.eu/how-it-works>.

[21] "Rosetta Stone® Language Learning Pricing Overview." How Much Does Rosetta Stone® Cost? N.p., n.d. Web. 14 Aug. 2016. <http://www.rosettastone.com/pricing>.

[22] "Livemocha.com: Alexa Rank and Traffic History for 7 Years." Livemocha.com. Alexa, n.d. Web. 14 Aug. 2016. <http://www.rank2traffic.com/livemocha.com>.

[23] "Livemocha: Rosetta Stone." Livemocha: Rosetta Stone. N.p., n.d. Web. 14 Aug. 2016. <http://product.rosettastone.com/portfolio_page/livemocha/>.

[24] Marmelstein, Roy. "Marmelroy/Localize-Swift." GitHub. N.p., n.d. Web. 14 Aug. 2016. <https://github.com/marmelroy/Localize-Swift>.

[25] Krzyzanowski, Marcin. "Krzyzanowskim/CryptoSwift." GitHub. Marcin Krzyzanowski, n.d. Web. 15 Aug. 2016. <https://github.com/krzyzanowskim/CryptoSwift>.

[26] Yang, Herong. "Base64 Encoding Algorithm." Base64 Encoding Algorithm. N.p., n.d. Web. 14 Aug. 2016. <http://www.herongyang.com/Encoding/Base64-Encoding-Algorithm.html>.