**CECS 550**

**Spring 2018**

**Software Requirements Specification**

**BRIDGES**

Nikhil Paonikar

Brandon Sparrow

Pinhao Guo

Manorath Dhakal

Isaiah Thomas

**Table of Contents**

### Introduction

* 1. Purpose
  2. Scope
  3. Definitions, Acronyms, and Abbreviations
  4. References
  5. Overview

### The Overall Description

* 1. Product Perspective
     1. System Interfaces
     2. Interfaces
     3. Hardware Interfaces
     4. Software Interfaces
     5. Communications Interfaces
     6. Memory Constraints
     7. Operations
     8. Site Adaptation Requirements
  2. Product Functions
  3. User Characteristics
  4. Constraints
  5. Assumptions and Dependencies
  6. Apportioning of Requirements

### Specific Requirements

* 1. External interfaces
  2. Features
  3. Performance Requirements
  4. Logical Database Requirements
  5. Design Constraints
     1. Standards Compliance
  6. Software System Attributes
     1. Reliability
     2. Availability
     3. Security
     4. Maintainability
     5. Portability
  7. Organizing the Specific Requirements
     1. System Mode
     2. User Class
     3. Objects
     4. Feature
     5. Stimulus
     6. Response
     7. Functional Hierarchy

### Change-Management Process

1. **Document Approvals**
2. **INTRODUCTION**

There are some applications on mobile markets that help you learn a new language but most of them are quite impractical. This project will focus on teaching users a new language that can be used in everyday conversations while maintaining correct grammar usage.

There are several things to consider that need to be addressed while working on this project. Firstly, the user’s learning process will be gamified and the application will be first developed for desktop OSs, and then mobile OSs. Next, we will first make lessons available for two languages. Finally, the application will have a progression of 30 levels to learn the basics of the two languages.

## Purpose

This is a software requirements specification document about Bridges; it’s an app to gamify learning languages”, which provides a complete description of all the properties that will be implemented and a complete definition of system attributes. Specialties of the system and functionalities of the project will be defined, explained and demonstrated by using some models.

The main purpose of the project is to create a unique learning experience that will help users learn a new spoken language. There are 30 levels in each language that the user shall progress by speaking into the microphone of their device and answering questions presented in levels. This document’s audience are developers, testers and end-users.

## Scope

The project is named “Bridges”. The scope of this project is playing a unique game that consists of multiple levels, each teaching the user 20-30 new words in the form. The aim of this project is to create an uniquely entertaining game-application that’ is supposed to teach users to learn a new language. This project consists of two learning methodologies; viz. vocabulary and conversation. This will be a single-player game with all files stored locally on the device.

## Definitions, Acronyms, and Abbreviations.

|  |  |
| --- | --- |
| **Term** | **Description** |
| API (Application Programming Interface) | An application program interface (**API**) is a set of routines, protocols, and tools that specify how software components interact. |
| Class diagram | A diagram in UML that describes the system-structure by showing the system’s classes, their attributes, operations or methods and the relationship among the classes. |
| Game engine | Software framework designed for the creation and development of video games. |
| IDE | Software application that provides some good and useful facilities to programmers for software development. |
| IEEE | The Institute of Electrical and Electronics Engineers |
| ms | Millisecond |
| MB | Megabyte |
| Scene | In Unity, each screen is referred to as a scene. |
| Sequence diagram | It shows how processes operate with one another and in which order. |
| SRS | SRS Software Requirements Specification. |
| Unity 2D | A cross-platform game engine that is used to develop games and simulations for computers-devices. |
| Use case diagram | UML diagram that represents the user’s interaction with the system. |
| User | The end-user who will use the finished product |
| VS (Visual Studio) | A Microsoft IDE used to develop programs, web sites, web apps, web services and mobile apps. |

* 1. **References**

The resources listed below are the references that has been used during the requirements analysis; IEEE Standard Documents:

[1] IEEE Guide to Software Requirements Specifications (Std 830-1993)

[2] Duolingo as a Bilingual Learning App: a Case Study.” Arab World English Journal, vol. 7, no. 2, 2016, pp. 255–267

[3] 20 Years of EUROCALL: Learning from the Past, Looking to the Future.

## Overview

The SRS document is divided into 5 chapters. The first chapter gives an introduction to the app; Bridges. This chapter is intended for everyone ranging from naive users to developers. The second one gives a high-level description, which is pertinent to the end-users. Chapter 3 is for the understanding of developers and is thusly written to explain specific requirements for the project development. Chapter 4 explains in brief, how changes will be documented. The final chapter simply acknowledges the document approvals made by the team members.

# THE OVERALL DESCRIPTION

## Product Perspective

Several apps on the marketplace claim to teach you a language. But the parts they teach aren’t readily usable. BRIDGES is a unique language-learning experience that is gamified. The game will primarily focus on two aspects of language; vocabulary and conversing. The user will be guided from the basics of a preferred language and will be taught to converse from the very first lesson. Emphasis will be given on the more relevant, most-used parts of languages.

### System Interfaces

The only external system that the game will connect to will be Watson Speech API. It will be used to check the accuracy of the user’s pronunciation.

### Interfaces

The user will navigate the game using on-screen buttons and touch. The user will first listen to a conversation and then learn the words comprising it (vocabulary). Then, the user’s vocabulary and conversational skills will be tested using their on-device microphones.

### Hardware Interfaces

The game will be developed for mobile and computer operating systems. The hardware interfaces for the computer OSs; Windows, Linux and MacOS would be keyboard, mouse and a microphone.

### Software Interfaces

2.1.4.1 **Watson Speech API**

This game uses the IBM’s Watson Speech API to receive input from the user in form of speech. The API will communicate with the app using Watson’s platform.

### Communications Interfaces

Unity’s built-in library methods are used to interface communication with Watson Speech API.

### Memory Constraints

Primary memory required: 512 MB of RAM.

Secondary memory required: 60 MB of on-device storage.

### Operations

The operation conditions will be normal. A detailed look at the operations are provided in section 2.2.

### Site Adaptation Requirements

Since the cloud and speech services are provided by IBM Watson, we don’t need to state these requirements. Furthermore, on the customer-end, their device operating systems will provide for the smooth functioning of our application.

### Product Functions

This section provides the list of higher-level specification of our game.

* + 1. User selects **new game**



* + 1. User views **options**



* + 1. User views **credits**



* + 1. User views **tutorial**



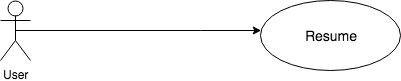
* + 1. User **selects language**



* + 1. User chooses to **pause** game



* + 1. User chooses to **resume** game



* + 1. User chooses to **replay** a level



* + 1. User chooses to **press the “Speak” button**



### User chooses sound options



* + 1. User **enables/disables night-mode**



* + 1. User **views level-progression**



* + 1. User chooses to **retry**



## User Characteristics

This game project is meant to be for users who want to learn to speak a language by playing a game. The primary language to teach the other languages will be English. No prior experience is needed with the languages or with playing a game.

## Constraints

This is a multi-platform game, so it is supported on Windows 10 PC, MacOS and Linux. This game uses speech recognition system. Thus, it uses device’s microphone and speaker. Other than that, it does not have a huge graphics that uses hardware constraints. We are using some Unity’s built-in libraries, and some third-party libraries for the speech recognition system. These libraries have restrictions on some older versions of the desktop OSs.

## Assumptions and Dependencies

Unity game engine, and Operating systems are our dependencies.

## Apportioning of Requirements.

The requirements will remain same for the entire duration of development cycle.

# SPECIFIC REQUIREMENTS

## External Interfaces

### User interfaces

The User Interface will incorporate multiple platforms with multiple methods to interact with the game. The primary design of the game will be Graphical User Interface (GUI) that will be interacted through mouse and keyboard, or by touchscreen.

### Hardware interfaces

The hardware that the game is designed for will be any laptop/desktop. As long as the device is compatible with the outlined software interfaces, the game will be compatible.

### Software interfaces

The platforms that the game will reside on will be Windows 7 and later. The main software interface will be Windows 10, but will be portable for other software interfaces through the engine that is being utilized to create the game.

### Communications interfaces

The communication interfaces will be the ones between the on-device application and Watson API services which will be implemented using object oriented principles.

## Features

### User Interface

* + - 1. **Purpose of User Interface**

The User Interface is the part of the game that will give provide the user the ability to feed input into the game. The game will then interpret the input and provide appropriate options for the user.

### Stimulus/Response sequence

The game will provide some sort of information such as a menu screen with buttons. The user will make a decision from the screen and click onto the buttons either by mouse or touchscreen. The decision will take the user to a desired screen that will display specific contents.

### Associated User Interface requirements

They are used to navigate the application.

### Button Presses

Each button will have a designated box that if pressed within the box, the action will execute. Each press will have a debounced effect so that there are no multiple presses being read from the input. When is button is pressed, it will not do anything until the button is released.

### Voice Interface

The game will require input from the user to progress through the game. The game will interpret the sounds from the microphone. The sounds will be passed to a speech API which will then return some type of data indicating that the speech was correct or incorrect and the game will give appropriate options for the user.

### Voice Recognition

For getting the input from users.

### Purpose of Voice Recognition

The Voice Recognition functionality is used for users to interact with the program, based on the voice input by the user, the program will search reasonable responds in the build-in database

### Stimulus/Response sequence

The program will set up different scenes for corresponding levels. Once the user starts one scene, the program will play the canned sentences and user should response the sentence by speaking to the microphone or any voice record devices in their machine, and the program will run this very voice recognition function trying to understand what did the user just respond and send the result as the argument to other function in the program to continue further procedure.

### Associated functional requirements

* + - * 1. **Voice record interface**

We need voice recording interface to detect and record the voice input from the user and send it to the voice recognition function

### User Interface

We need the user interface to prompt text generating by the voice recognition function back to the user interface, so the user has some idea what he/she is doing wrong.

### 3.2.3 Game Design

It is going to be implemented using the Unity engine and follow the prototyping model.

### Purpose of the game

The application program is actually a gamified learning experience that is supposed to teach the user to learn to speak a spoken language.

### Stimulus/Response sequence

The stimulus will be a question that the user is supposed to answer vocally, which will make the response.

### Associated game requirements

There won’t be any pre-requisite skill required on the user’s behalf except navigating a mobile app.

### 3.2.3.3.1 Button Presses

The button presses will be used to navigate the application.

### Language Education

* + - 1. **Purpose of the Language Education**

The purpose of the game is to teach the user different languages. The game is intended to be interactive with the user through speech. The first method that people learn a language is by speaking the language itself, so the game is designed to have the user say the most common words and phrases.

### Stimulus/Response sequence

The method of how the user will interact with the game is through speaking through the microphone and the speaker will play dialogue. Watson speech API will be utilized to listen and speak with the user. The game will play specific dialogue during specific parts of the game and the user will speak to the game. The speech API will then interpret the speech and send that data to the game. The game will then compare that speech with the correct response/s and provide feedback and progression dependent on the answer.

### Associated Language Education requirements

* + - * 1. **Accurate Interpretation**

The game has to be able to take in information and parse it accurately to provide correct feedback to the user. The users education will only hurt if the game provides incorrect feedback. For this requirement, the speech API needs to be very reliable.

### Effective Learning

Education of a language has many aspects to it for a person to learn it. For a game to be effective as a teacher, the game needs to follow certain guidelines. To be effective, we need immersion of the language with the user. The more the user interacts with the language, the more comfortable the user will be. Repetition is an effective method of learning and having a game that immerses the user and forces the user to have to use the same words and phrases causes the users brain to make patterns and provides information retention.

## Performance Requirements

The performance of this game has to be very accurate as it is a learning device. Having parts of the game being text based will give a 100% accuracy for learning. The requirements for the speech will not be able to be 100% accurate due to the speech to text API is not 100%. The accuracy of the Watson speech API is 95.1%. For our purposes, this will give approximately 9 to 10 out of 10 phrases/words correct interpretations. This will be sufficient for our needs and since the accuracy is not 100%, a confidence level will be integrated to ensure that there is no misteaching. The performance of the game itself should be very quick and take a minimal amount of energy to perform. The games brightness should be the same brightness as the phones current brightness and the game should not continue if the game is not the primary program. With each transition of the game, there should be a wait of maximum 1 second between each screen. Buffering and loading large amounts of data should not adhere to this but should stay within a 1 minute timeframe. The frames that the game should aim to have per second should be no lower than 15 and no higher than 25 due to the drain on the battery.

## Logical Database Requirements

Game data will be stored on-device. There won’t be a need for a discrete DBMS.

## Design Constraints

The constraints that could impair this project would be the usage limits of Watson Speech API. A license may be required to have unlimited access, or each installation of the game could need its own license. If this is the case, then another API or a Unity developed speech API may be required.

### 3.5.1 Standards Compliance

All data collected from end users will be stored for the life of the users account plus one additional year. Data will be retrievable upon request within a year of a closed account. All audits requesting data post one year of a closed account will not be available.

Data stored will include:

* + - User name
    - User email
    - Game stats: languages mastered, completion percentages, duration of play, etc.
    - Age of account
    - Device used

## Software System Attributes

### Reliability

The application will have a MTBF of more than 1,000 hours. Application crashed will only be caused by operating system errors.

### Availability

The application will provide continuous availability. In the instance of a crash, users will lose only data of the current level being played. All other data will be recoverable.

### Security

No sensitive user data will be stored in this application. All users will login with a unique username and password. A history of user logins will be stored along with basic user information.

### Maintainability

The application will contain well documented and versioned code. Comments will provide understanding of each step in the code. Modularity will be maintained for future requirement changes and modifications to the application.

### Portability

This application will be available to users in various environments including:

* + - Windows 10 (desktop version only)
    - MacOS
    - Linux
    - ~~Android 7+~~
    - ~~iOS 10+~~

All data will be seamlessly transferable between listed environments.

## Organizing the Specific Requirements

### System Mode

The default mode will be single-player mode and since the software development model will be a prototyping model, new functionalities will be added with every iteration. These mainly comprise of new levels. The addition of extra features cannot be perceived based on the requirements of current prototype.

### User Class

The only class of users will be the end-user; the player who wants to learn to speak a language using the app.

### Objects

The objects will be created while programming and this shall be elaborated upon in detail, in the design document.

### Feature

The game will have two major features: vocabulary-learning and

conversation-learning. A feature is an externally desired service offered by the system that may require a sequence of inputs to affect the desired result. Each feature is generally described in as sequence of stimulus-response pairs, which will be the questions given by the app (stimulus) and the user’s vocal answer (response).

### Stimulus

The stimulus will be a question asked in each level. These may be vocabulary-related, or questions asked in a conversation.

### Response

The response will be the vocal input given by the user in response to the stimulus provided by the game.

### Functional Hierarchy

Section 2.2 has a detailed information about the functional requirements by explaining use cases in detailed manner, so there won’t be any detailed information in here.

# CHANGE-MANAGEMENT PROCESS

All requirement changes will be handled through development meetings. When a change is presented, the entire development team will weigh in before a change is officially issued and implemented. All changes will be properly documented and available for reference.

# DOCUMENT APPROVALS

This document will approved by the development team including:

Nikhil Paonikar - NRP - 04/22/2018

Manorath Dhakal - MD - 04/22/2018

Pinhao Guo - PG - 04/22/2018

Brandon Sparrow -BS- 04/22/2018

Isaiah Thomas - IT - 04/22/2018