***Bridges***

***Software Design Specifications***

SDS Final Version

April 22, 2018

This Software Design Specification was prepared and provided as a deliverable for CECS 550: Software Engineering at the University of Louisville for the Spring session of 2018. This document is based in part on the IEEE Standard 1016-1998, IEEE Recommended Practice for Software Design Descriptions.

Nikhil Paonikar,

Team leader, language- lessons, co-developer

Brandon Sparrow,

Game designer

Pinhao Guo,

Co-Chief developer

Manorath Dhakal,

Co-Chief developer

Isaiah Thomas,

Chief tester, co-developer

# Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Section/Pages**  **Affected** | **Remarks** |
| Draft 1 | February 12, 2018 | Nikhil Paonikar  Brandon Sparrow  Pinhao Guo  Manorath Dhakal  Isaiah Thomas | All Initial |  |
| Version 3 | April 12, 2018 | Nikhil Paonikar  Brandon Sparrow  Pinhao Guo  Manorath Dhakal  Isaiah Thomas | Interface Description | Edit |
| Final Version | April 22, 2018 | Nikhil Paonikar  Manorath Dhakal  Pinhao Guo  Brandon Sparrow  Isaiah Thomas | List of figures,  Appendix A | Final edit |

# Table of Contents

[Change History](#_3dhjn8m) 3

[Table of Contents 4](#_1smtxgf)

[Preface](#_1fob9te) 6

[Table of Tables](#_3znysh7) 7

[Table of Figures](#_2et92p0) 8

[1. Introduction](#_tyjcwt) 9

[1.1. Purpose](#_tyjcwt) 9

[1.2. Scope](#_3dy6vkm) 9

[1.3. Definitions and Acronyms 1](#_4d34og8)0

[2. References 11](#_2s8eyo1)

[3. Decomposition Description 13](#_17dp8vu)

[3.1 Feature 1: Voice Recognition 13](#_3rdcrjn)

[3.1.1. Use Case Configuration Description 15](#_lnxbz9)

[3.1.2. Configuration Data Flow Diagram 16](#_35nkun2)

[3.1.3. Configuration Process Flows 18](#_2jxsxqh)

[3.1.3. Configuration Process Flows 18](#_4cmhg48)

[3.1.4. Process Analysis of the Purchasing of an aspect of Configuration 20](#_4i7ojhp)

[3.1.5. Configuration Modules Analysis 21](#_1ci93xb)

[3.1.6. Comprehensive Set of Class Diagram 21](#_2bn6wsx)

[3.2. Feature 2: Game Design](#_3as4poj) 13

3.2.1. Purpose of the game 13

3.2.2. Stimulus/Response sequence 13

3.2.3. Associated game requirements 13

3.2.4. Button Press

[3.3. Feature 3: User Interface](#_ihv636) 13

[3.3.1. Stimulus/Response Sequence](#_32hioqz) 13

[3.3.2. Associated User Interface requirements](#_41mghml) 13

[3.3.2.1. Button Press](#_vx1227) 13

[3.3.2](#_1v1yuxt).2 Voice Interface13

[3.4. Feature 4: Language Education](#_19c6y18) 14

3.4.1 Purpose of the Language Education 14

3.4.2 Stimulus/Response Sequence 14

3.4.3 Associated Language Education Requirements 14

3.4.3.1 Accurate Interpretation 14

3.4.3.2 Effective Learning 14

UML diagrams 15

Explanation of class diagram 15

Explanation of sequence diagram 18

[4. Dependency Description](#_3cqmetx) 20

[4.1. Intermodule Dependencies](#_3cqmetx) 20

[4.2 Interprocess Dependencies](#_1rvwp1q) 20

[4.3 Data Dependencies](#_4bvk7pj) 20

[5. Interface Description](#_2r0uhxc) 21

[6. Detailed Design](#_1opuj5n) 23

[Appendix A: Voice Recognition](#_48pi1tg) 24

# Preface

This document is prepared as part of the requirements for the class CECS 550: Software Engineering. The information contained within is based on preliminary information provided in the textbooks, the recommended lecture readings and lecture slides.

# List of Tables

Table 1: Features

Table 2: Definitions and Acronyms

Table 3: Sequence diagram: explanation

# List of Figures

Figure 1: Class diagram

Figure 2: Sequence diagram

Figure 3: Screenshot of the UI to record the speech in Japanese

Figure 4: Screenshot of the UI to record the speech in English

Figure 5: Option to choose between Japanese and English language to record the speech.

Figure 6: Recorded speech using Voice Recognition API show the text for that speech

FIgure 7: Final version of the game with working Voice recognition API,

FIgure 8: Final version of the game with working Voice recognition API

# 1. Introduction

## *1.1. Purpose*

This document outlines the Software Design Specifications as part of the design plan and specifications for the application program *Bridges.*

This document expands the functionality described by the features in the Software Requirements Specifications (SRS). Each feature discussed will describe the existing functionality of Bridges and describe any additional classes, attributes and methods to be implemented.

## *1.2. Scope*

This document takes the features as outlined in the SRS and expands each of the features to include the design issues of user interface, data flow, process analysis and then module design. The features are described by the Use Case names given in the descriptions and diagrams in the SRS. The features included in this report are listed in Table 1.

|  |  |  |
| --- | --- | --- |
| **Feature number** | **Use case name** | **Reference** |
| 1. | Voice Recognition | Appendix A |
| 2. | Language Education | - |
| 3. | Game Design | - |
| 4. | User Interface | - |

Table 1: Features

This project does not incorporate any existing code, just the services of an API. The design document is written to use the prototyping software model and thus, follows a typical object-oriented design. Each Feature will incorporate aspects of the Object Oriented Design(OOD). The structure, function and interfaces are discussed and described in the section 2.2 in the form of use case UML diagrams in the Software Requirement Specifications(SRS) document. The methods are illustrated in the class diagram. The interfaces are discussed and illustrated using a sequence diagram. The detailed design description is discussed and illustrated by the class diagram, which illustrates the relationships between classes, and outlines the members and methods necessary to implement the features.

## *1.3. Definitions and Acronyms*

|  |  |
| --- | --- |
| **Term** | **Description** |
| SRS | **Software Requirements Specification** is a document that describes the requirements of the software system to be developed. |
| OOD | **Object-Oriented Design** is a concept that forces programmers to plan out their code in order to have a better flowing program. |
| ERD | **Entity-Relationship Diagram** illustrates the logical structure of databases. |
| DFD | **Data Flow Diagram** is a graphical representation of the flow of data in a system. |

Table 2: Definitions and Acronyms

# 2. References

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

[2] Software Engineering: Modern Approaches by Eric Braude and Michael E. Bernstein., Wiley Publishers, Second Edition, 2011. ISBN 978-0-471-69208-9.

# 3. Decomposition Description

This section of this report decomposes each use-case feature into a single sequence diagram. flow processes by examining its data flow diagram and its process flow through the use of sequence diagrams. These assist us in determining the preliminary members and methods of the modules that need to be implemented, or the modifications to existing modules to implement the feature. This document uses the names of the use cases in the Software Requirement Specification document as the names of the features. This section includes the description of the intended design to meet the requirements. When appropriate, the use cases will be expanded to include system requirements. This section also incorporates a number of revised sequence diagrams, updated from the initial prototype and analysis.

The Features discussed in this document are:

## *3.1 Feature 1:* Voice recognition

**3.1.1 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

**3.1.2 Stimulus/Response sequence**

The program will set up different scenes for corresponding levels. Once the user start 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.

**3.1.2.1 Associated functional requirements**

**3.1.2.1.1 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

**3.1.2.1.2 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 Feature 2: Game design

**3.2.1 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.

**3.2.2 Stimulus/Response sequence**

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

**3.2.3 Associated game requirements**

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

**3.2.4 Button Presses**

The button presses will be used to navigate the application.

## 3.3 Feature 3: 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.

**3.3.1 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.

**3.3.2 Associated User Interface requirements**

They are used to navigate the application.

**3.3.2.1 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.

**3.3.2.2 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.

## 3.4 Feature 4: Language education

**3.4.1 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.

**3.4.2 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. Google 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.

**3.4.3 Associated Language Education requirements**

**3.4.3.1 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.

**3.4.3.2 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.

**UML diagrams:**

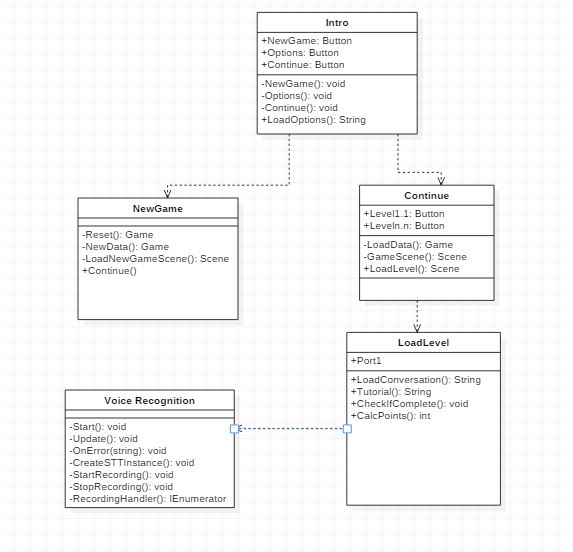


Figure 1: Class diagram

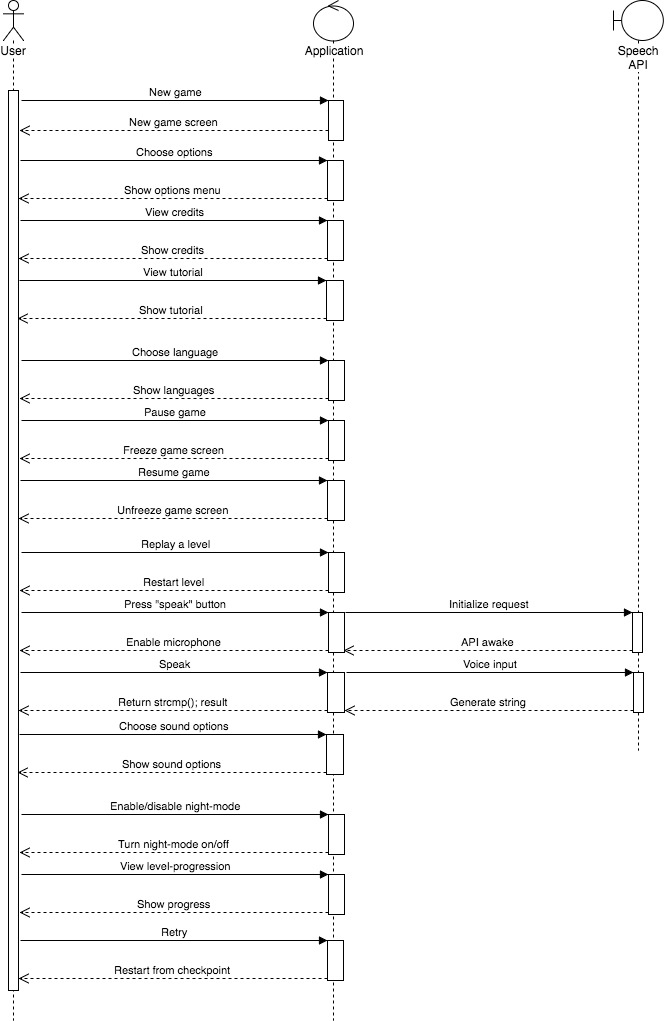
**Explanation of class diagram:**

The above class diagram illustrates the class design to be used for the project. The primary class is the *NewGame* class which will load the NewData method and the scenes for the game. It will also contain the reset method which deletes all the data from the game as configured.

We have another class called *Continue*, which loads the existing data and the scenes for the game. This class also contains the methods for the levels, where it will have buttons which when enabled, will load the new levels.

The level contains the string of conversations and it will work simultaneously with the voice recognition where it will have functions like accepting speech from users and checking if the word exists in the particular (spoken) language.

LoadLevel class also contains the method to check if the game is completed or not, and then it calculates the score the user received after each level.

Figure 2: Sequence diagram

**Explanation of sequence diagram:**

The sequence diagram covers all 4 main features of the application. This is done using 3 primary actors. The user is an external actor who interacts with the main system. The main system is formed by the application program and the speech API.

The application manages the flow of interaction of the scenario and therefore is a control actor. On the other hand, the speech API is a boundary actor because it lies on the periphery of the system.

As the features were explained earlier, their corresponding operations made by the user are illustrated in the above sequence diagram.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature(s)** | **Action** | **Initiator** | **Response** | **Responder** |
| Game design, User interface | New game | User | New game screen is displayed | App |
| User interface | Choose options | User | Options menu is displayed | App |
| User interface | View credits | User | Credits are shown | App |
| Game design, User interface | View tutorial | User | Tutorial screen is displayed | App |
| Game design, User interface | Choose language | User | Language selection menu is shown | App |
| Game design, User interface | Pause game | User | Game screen is freezed | App |
| Game design, User interface | Resume game | User | Game screen is unfreezed | App |
| Game design, User interface | Replay a level | User | Selected level is restarted | App |
| Game design, User interface, Voice recognition | Press “speak” button | User | The | App |
| Game design, User interface, Voice recognition | Speak | User |  | App |
| Game design, User interface | Choose sound options | User | The sound-options are shown. | App |
| Game design, User interface | Enable/disable night-mode | User | The night-mode is turned on/off | App |
| Game design, User interface | View level-progression | User | Shows the user’s progress in a level | App |
| Game design, User interface | Retry | User | Game restarts from checkpoint | App |
| Game design, User interface, Voice recognition | Initialize speech request | App | API is initialized to provide service | Speech API |
| Game design, User interface, Voice recognition | Provide voice as input | App | Convert sound into string | Speech API |

Table 3: Sequence diagram: explanation

# 4. Dependency Description

## *4.1. Intermodule Dependencies*

The intermodule dependencies can be seen with the class diagram. The whole entire game will be dependant from the Intro class. This class will ultimately be the class to load the game data or to start the creation of it. After the data is loaded or created, the game can be played and the Load Level class will take over and run the level requested by the user. The content of the game will be dependant of the game data. The game data will be the determinant of the available level. Each level will be dependant of the speech API, as the game cannot be played until the data is received from Watson.

## *4.2 Interprocess Dependencies*

The process of movement for the non-player characters with the selection from the user will be a interprocess dependency. The movement will not move when clicked on and will have to halt with the selection due to the conversation that will occur.

## *4.3 Data Dependencies*

The data dependencies are very limited in this project. The only data that is dependant is the game data with the user inputs and the Speech API. The game cannot proceed without the user clicking on sprites. The game also has to take the data from the API before it can continue. The whole game resides on the data from the API. The string received from the API is very important because the game needs the string to compare the answer from the user with the actual answer.

# 5. Interface Description

The following are the interface screenshots of prototype 1.

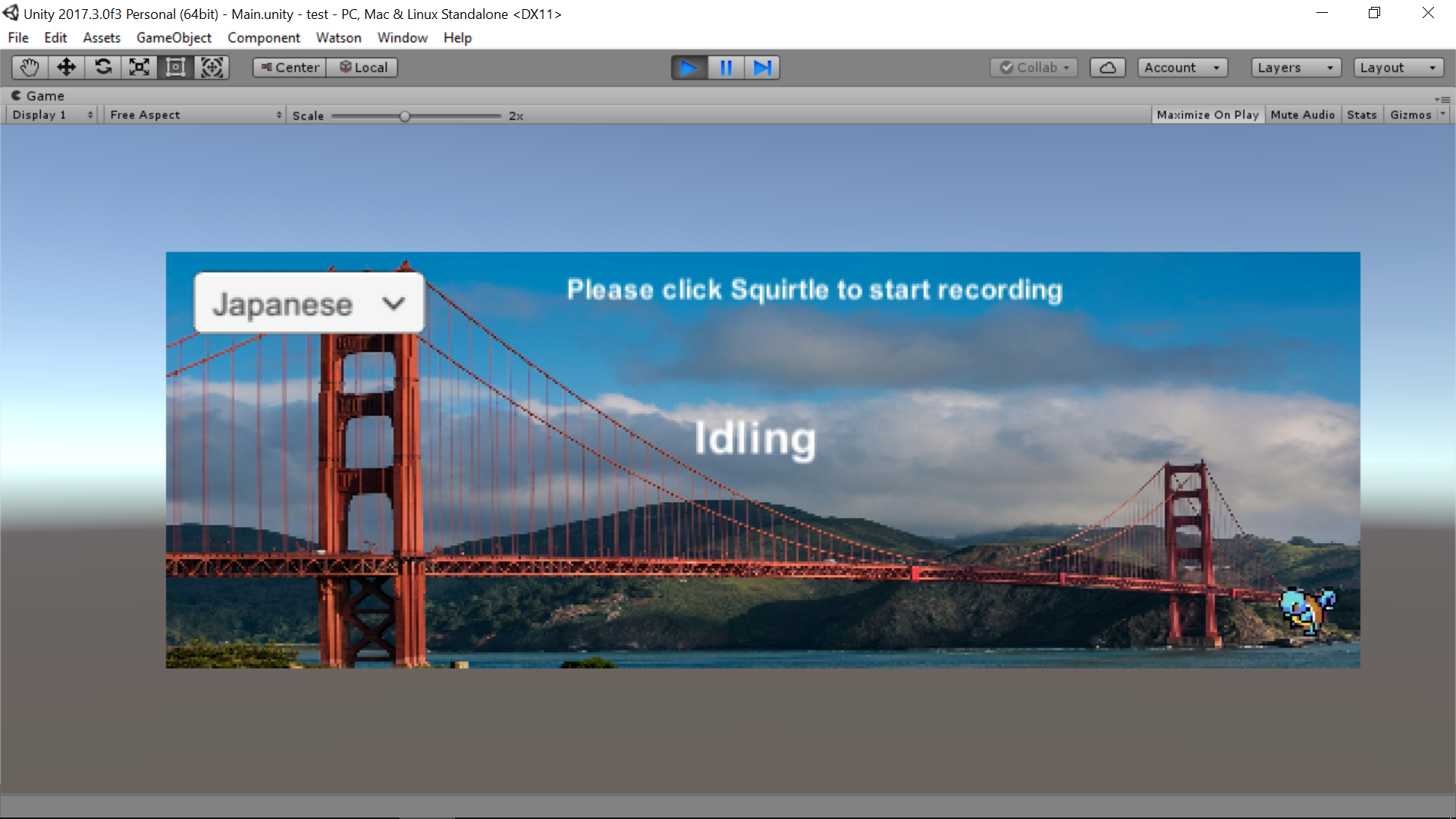


Figure 3: Screenshot of the UI to record the speech in Japanese

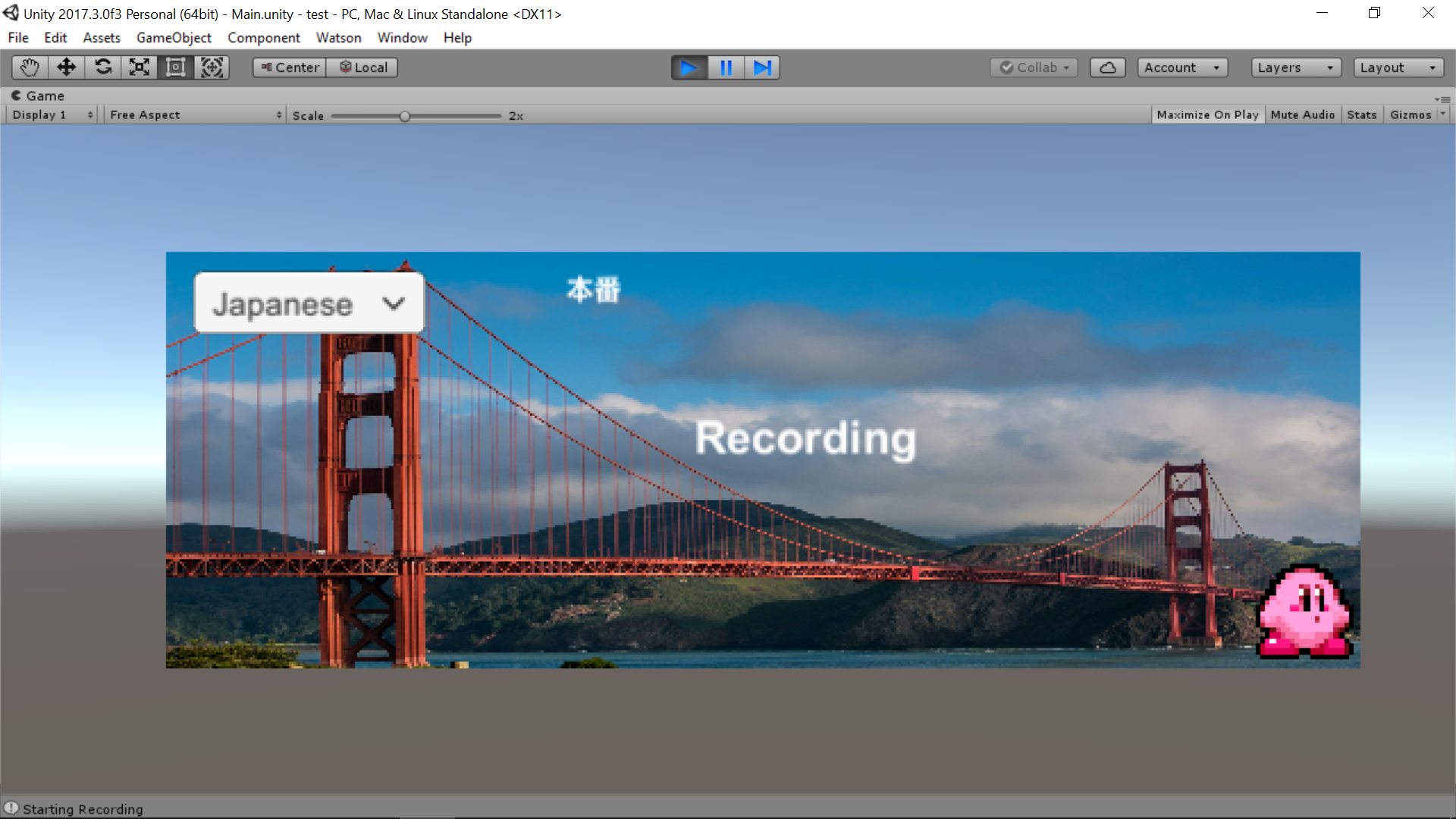


Figure 4: Screenshot of the UI to record the speech in English

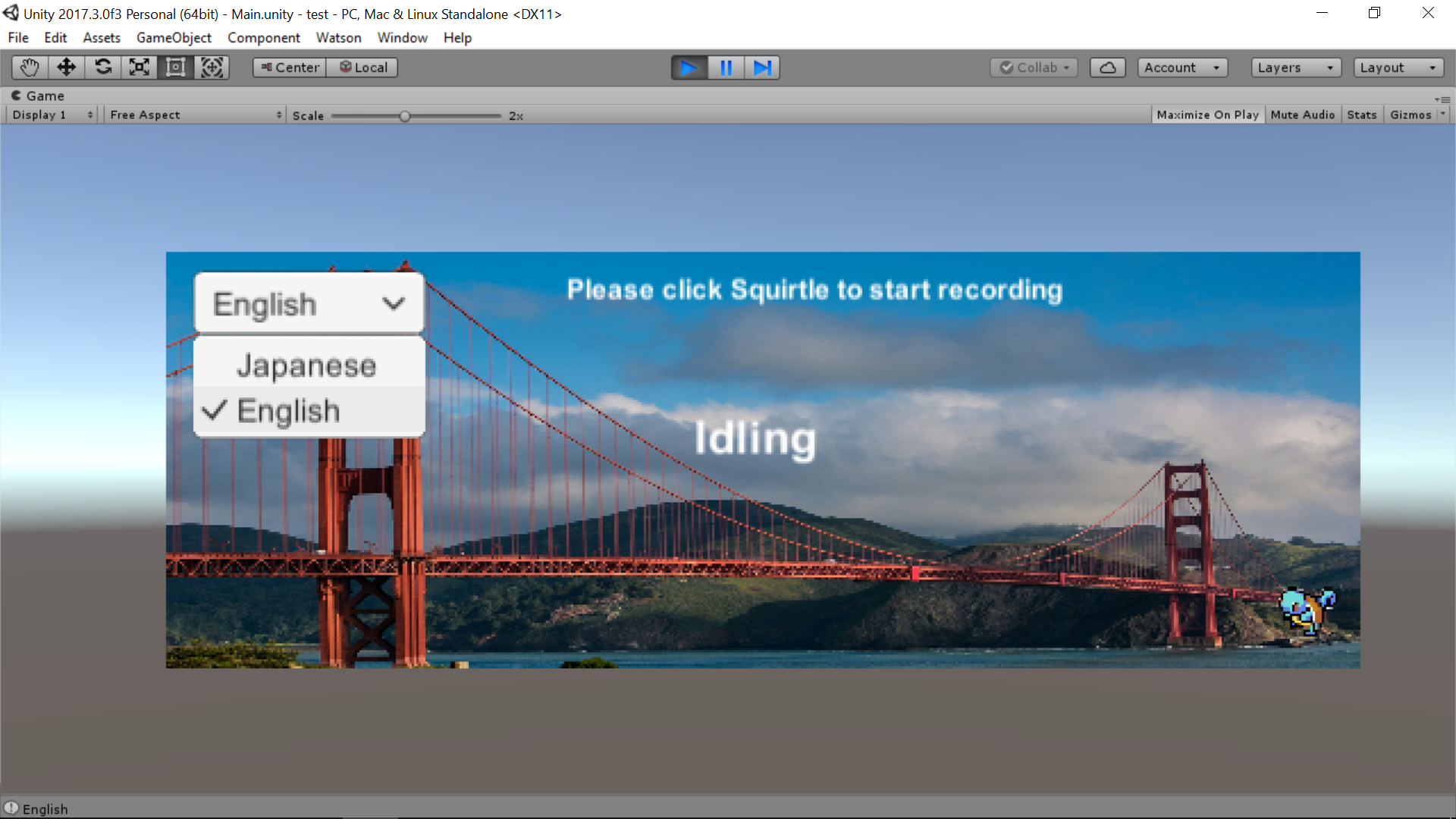


Figure 5: Option to choose between Japanese and English language to record the speech

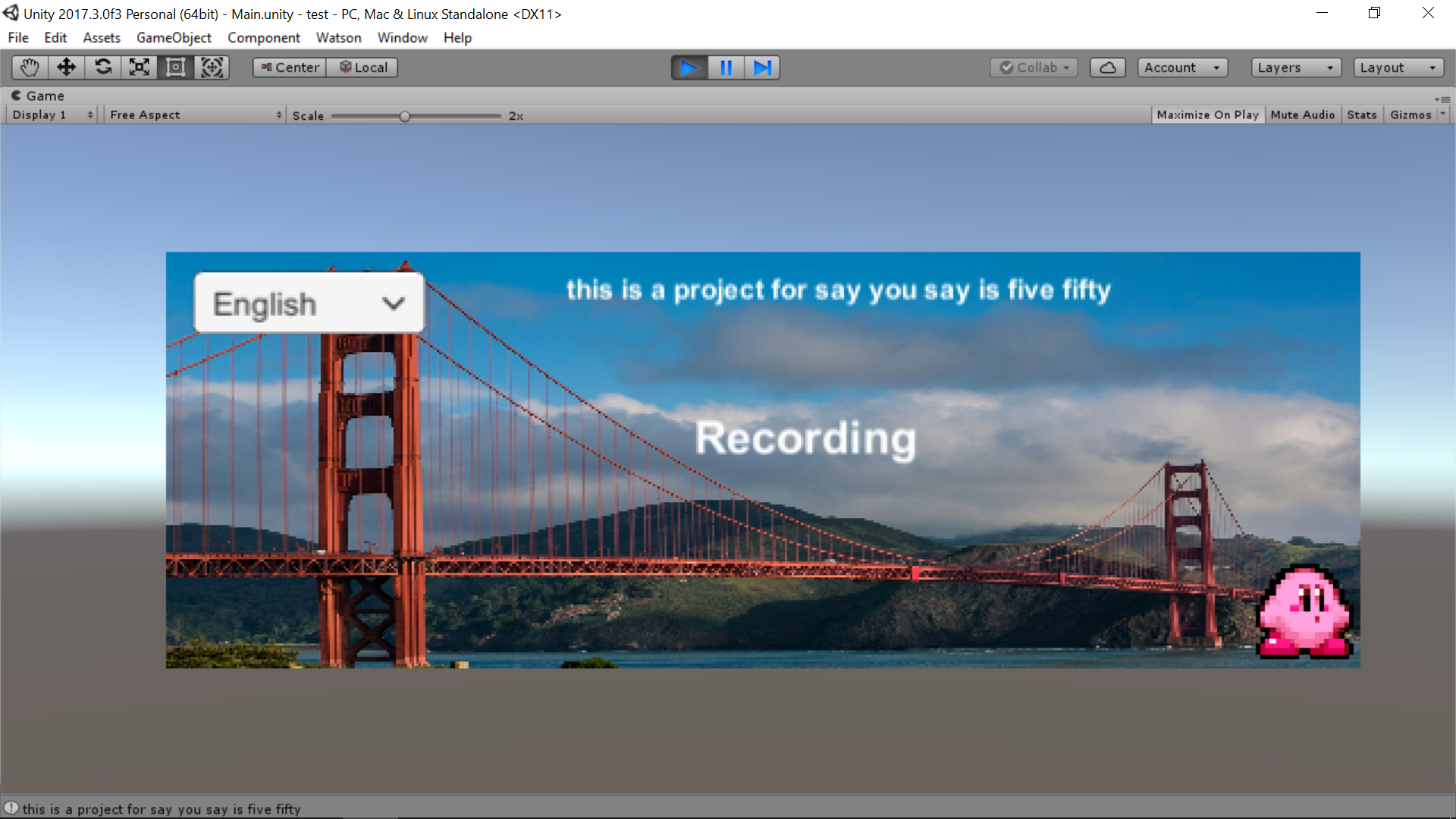


FIgure 6: Recorded speech using Voice Recognition API show the text for that speech

# 

FIgure 7: Final version of the game with working Voice recognition API

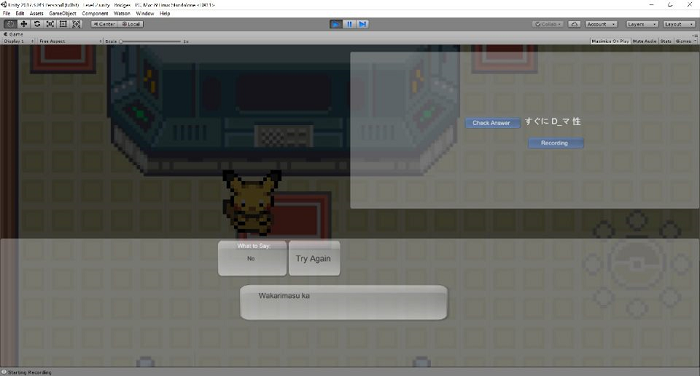


FIgure 8: Final version of the game with working Voice recognition API

# 

# 6. Detailed Design

**Note**: The software model being used for this project is the Software Prototyping model. Prototype 1 was a partial delivery which consisted the integration of API into the application.

This section shall be expanded before and during the design, implementation of the next prototypes.

# Appendix A: Bridge Package Class Source Listing and Documentation

These can be found in the **Source code** folder.