Appendix

[1] - Interview 1 with client: The scenario

(Me): Hi, I'm N. In my computer science course, I was asked to do an interview where I ask a client for information about their day-to-day problems to do a project where I can solve the problem. Let's start with the first question. What are some time consuming or inefficient issues that you encounter?

(Client): I think that I struggle to learn new key words in my English class. It's hard for me to keep consistency if I don't remember the previous key words. My studying methods are ineffective and it's time consuming not being able to move on to new words without remembering old ones.

(Me): What are the consequences/difficulties that arise from not solving the issue?

(Client): The consequences are having to revise old notes every time which is ineffective and time consuming for my learning. I can't move on to new content without learning the old words. Also, I'm not engaged enough with the content I'm learning.

(Me): How is the issue currently solved?

(Client): I currently rely on online translators which sometimes have errors. They don't necessarily help me learn either. I try to make notes on paper but I lose them or can't keep track of old and new content.

[2] - Website comparing Java and C++

Meltzer, Rachel. "C++ vs Java: A Guide for Beginners." *Course Report*, 21 December 2021, https://www.coursereport.com/blog/c-vs-java-a-guide-for-beginners. Accessed 3 September 2022.

[3] - Interview 2 with client: Success Criteria

(Me): The purpose of this program is to help you learn and remember new words in an engaging way, let you personalize and interact with the program, and help you organize the content you learn. This second interview is to discuss the program's

requirements in order for it to be successful. Let's start with the first question. What would this program need in order to be successful?

(Client): I think that a program that works on my school Mac laptop and my home WIndows desktop that lets me add, remove and save new words, has a section for me to write down the pronunciation, and things that I connect the words to would help me to easily remember the words. I would also like to be able to view the definitions and copy information between my devices.

(Me): How would the program be organized?

(Client): I think that alphabetical order would be the best way to organize the words. And maybe record the date where I added the new words.

(Me): What inputs would the program need?

(Client): I think that the date I add the new words, the new words, their pronunciation, and things I can connect/relate them to.

(Me): What do you think about an interactive game that tests your knowledge?

(Client): I think that an interactive game that lets me test my understanding or correct use of the vocabulary words would help me engage with the words and learn faster.

(Me) What would you not want in this program?

(Client): I want the program to be easy to understand for me, so no complicated or too advanced words for now would help me control the program better.

[4] M. "English Student - Evaluation." Personal interview. 16 Jun 2023.

(Me): What do you think about the login page?

(Client): I think it was a good idea to add a login page as a security measure and that it even tells you if it's empty.

(Me): And from the vocabulary list, what improvements do you think could be made?

(Client): All the textboxes work correctly and I can easily write text, but maybe adding images would be helpful. I think that would be better than writing things I associate the word with. And for the input for a new vocabulary word, I think it would be very helpful for the program to check spelling mistakes and notify me if any are made.

Additionally, it would be helpful to get the translation alongside the definition of a word.

(Me): Did you understand the labels and each button's name?

(Client): Yes, I think the labels and buttons are clear and simple for me to know what they mean and do.

(Me): Were you able to run the program on your own device?

(Client): Yes, I was able to use the program on my laptop and everything worked correctly, including the game.

[5] - Program Code

StartAplication.java

```
package com.example.tutorial;

import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class StartApplication extends Application {
                public     static     ObservableList<Vocab>     vocab     =
FXCollections.observableArrayList();

   @Override
   public void start(Stage stage) throws IOException {
                                      FXMLLoader     fxmlLoader     =     new
FXMLLoader(StartApplication.class.getResource("login-view.fxml"));
       scene = new Scene(fxmlLoader.load());
       stage.setTitle("Vocab.Me"); //name of the program
       stage.setScene(scene);
       stage.show();
   }

   private static Scene scene;
   static void setRoot(String fxml) throws IOException {
                                      FXMLLoader     fxmlLoader     =     new
FXMLLoader(StartApplication.class.getResource(fxml + ".fxml"));
```

```
        scene.setRoot(fxmlLoader.load()); //change view to the selected fxml
file
    }

    public static void main(String[] args) {
        launch();
    }
}
```

Login.java

LoginController.java

```
package com.example.tutorial;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.*;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;

public class LoginController {
    public TextField cUsernameTxt;

    @FXML

       public            ObservableList<Login>            username            =
FXCollections.observableArrayList();
    public Label wrongLogin;

public void initialize() throws IOException {
    loadUsername();
}
    @FXML
    private void loadUsername() throws FileNotFoundException {
        //load username from saved file.
        Gson gson = new Gson();
        try (Reader reader = new FileReader("username.json")) {
                        ArrayList<Login>  imports  =  gson.fromJson(reader,  new
TypeToken<ArrayList<Login>>() {
```

```
            }.getType());
            username = FXCollections.observableArrayList(imports);
    } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void checkLogin() throws IOException { //check username

        if(cUsernameTxt.getText().equals(username.get(0).getUsername())) {
            wrongLogin.setText("Welcome!");
            StartApplication.setRoot("vocab-view");
        } else if(cUsernameTxt.getText().isEmpty()) {
            wrongLogin.setText("Please enter a username."); //deny
        } else {
            wrongLogin.setText("Wrong username"); //deny
        }
    }
}
```

Vocab.java

```
package com.example.tutorial;

public class Vocab {

    String newWord;
    String definition;
    String translation;
    String pronunciation;
    String links;


    public Vocab(String newWord, String definition, String translation, String
pronunciation, String links) {
        this.newWord = newWord;
        this.definition = definition;
        this.translation = translation;
        this.pronunciation = pronunciation;
        this.links = links;

    }

    public String getNewWord() {
        return newWord;
    }

    public void setNewWord(String newWord) {this.newWord = newWord;}

    public String getDefinition() {return definition;}
```

```
    public void setDefinition(String definition) {
        this.definition = definition;
    }

    public String getTranslation() {
        return translation;
    }

    public void setTranslation(String translation) {
        this.translation = translation;
    }

    public String getPronunciation() {
        return pronunciation;
    }

    public void setPronunciation(String pronunciation) {
        this.pronunciation = pronunciation;
    }

    public String getLinks() {
        return links;
    }

    public void setLinks(String links) {
        this.links = links;
    }

}
```

VocabController.java

```
package com.example.tutorial;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.reflect.TypeToken;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.MouseButton;

import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Reader;
import java.util.ArrayList;
```

```
import java.util.Collections;
import java.util.Comparator;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

public class VocabController {
   public TextField cNewWordTxt;
   public TextArea cDefinitionTxt;
   public TextField cTranslationTxt;
   public TextField cPronunciationTxt;
   public TextField cLinksTxt;

   public TableView contactsTable;

   @FXML
   private Label welcomeText;

    public TableColumn<Vocab, String> vocabWord = new TableColumn<>("Word");
//name/label of column

   @FXML
   protected void onHelloButtonClick() throws IOException {
       StartApplication.setRoot("vocab-view");
   }

   public void initialize() {

       loadVocab(); //load words from the vocabulary list

               vocabWord.setCellValueFactory(new  PropertyValueFactory<Vocab,
String>("newWord"));

       contactsTable.getColumns().add(vocabWord);
       contactsTable.setItems(StartApplication.vocab);

       contactsTable.setRowFactory(rowClick -> {
           TableRow<Vocab> row = new TableRow<>();
           row.setOnMouseClicked(event -> {
                           if  (!row.isEmpty()  &&  event.getButton()  ==
MouseButton.PRIMARY && event.getClickCount() == 2) {
                   Vocab clickedRow = row.getItem();
                   cNewWordTxt.setText(clickedRow.getNewWord());
                   cDefinitionTxt.setText(clickedRow.getDefinition());
                   cTranslationTxt.setText(clickedRow.getTranslation());
```

```
                    cPronunciationTxt.setText(clickedRow.getPronunciation());
                    cLinksTxt.setText(clickedRow.getLinks());
                }
            });
            return row;
        });
    }

    private void loadVocab() {
        //load vocab from saved file.
        //Open and read Json for any previously saved data.

        Gson gson = new Gson();
        try (Reader reader = new FileReader("vocab.json")) {
            //convert JSON file to Java Object
                      ArrayList<Vocab> imports = gson.fromJson(reader, new
TypeToken<ArrayList<Vocab>>() {
            }.getType());

            Collections.sort(imports, new Comparator<Vocab>() {
                @Override
                 public int compare(Vocab vocab1, Vocab vocab2) { //to sort in
alphabetical order, compare first letter of words
                    return vocab1.getNewWord().compareTo(vocab2.getNewWord());
                }
            });

                                                StartApplication.vocab    =
FXCollections.observableArrayList(imports);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

        public void saveBtn(ActionEvent actionEvent) {

        boolean exists = false;
        for (Vocab c: StartApplication.vocab) {
            if(c.getNewWord().equals(cNewWordTxt.getText())) {
                exists = true;
                System.out.println("Already exists"); //to not save duplicates
            }
        }

        if(exists == false) {
                  StartApplication.vocab.add(new Vocab(cNewWordTxt.getText(),
cDefinitionTxt.getText(),                          cTranslationTxt.getText(),
cPronunciationTxt.getText(), cLinksTxt.getText()));
            } //save new word and all its aspects (definition, translation,
pronunciation, links)
```

```
            Gson gson = new GsonBuilder().setPrettyPrinting().create();
            try(FileWriter writer = new FileWriter("vocab.json")) {
                gson.toJson(StartApplication.vocab, writer);
                System.out.println("Saved. "); //save to GSON file
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        public void delBtn(ActionEvent actionEVent){
                                        ObservableList<Vocab>   tempVocab   =
FXCollections.observableArrayList();

            for (Vocab v: StartApplication.vocab) {
               if(!v.getNewWord().equals(cNewWordTxt.getText())) {
                  tempVocab.add(v);
               }
            }
               StartApplication.vocab.removeAll(); //remove word from list if
delete button is clicked
            StartApplication.vocab=tempVocab;
            contactsTable.setItems(StartApplication.vocab);
        }

    public void searchBtn(ActionEvent actionEvent) throws Exception {
        String word = cNewWordTxt.getText();
         String url = ("https://api.dictionaryapi.dev/api/v2/entries/en_US/" +
word); //to search definition in dictionary API
        System.out.println(url);
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();

        //optional default is GET
        con.setRequestMethod("GET");

                            BufferedReader    in    =    new    BufferedReader(new
InputStreamReader(con.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();

        JSONParser parser = new JSONParser();
        JSONArray jsonArray = (JSONArray) parser.parse(response.toString());
        JSONObject jsonObject = (JSONObject) jsonArray.get(0);

        JSONArray meaningsArray = (JSONArray) jsonObject.get("meanings");
        JSONObject meaningsObject = (JSONObject) meaningsArray.get(0);
```

```
                            JSONArray    definitionsArray    =    (JSONArray)
meaningsObject.get("definitions");
        JSONObject definitionsObject = (JSONObject) definitionsArray.get(0);

        String definition = (String) definitionsObject.get("definition");
        cDefinitionTxt.setText(definition); //set definition in definition box
    }

    public void signOutBtn(ActionEvent actionEvent) throws IOException {
        StartApplication.setRoot("login-view");
    } //change view to login page if sing out button is clicked

    public void startBtn(ActionEvent actionEvent) throws IOException {
        StartApplication.setRoot("play-view");
    } //change view to game page if game button is clicked
}
```

GameController.java

```
import javafx.scene.control.TextField;
import java.io.IOException;
import java.util.Random;


public class GameController {

    //variables for the GUI
    @FXML
    TextField let1;
    @FXML
    TextField let2;
    @FXML
    TextField let3;
    @FXML
    TextField let4;
    @FXML
    TextField let5;
    @FXML
    TextField let6;
    @FXML
    TextField let7;
    @FXML
    TextField let8;
    @FXML
    TextField input;
    @FXML
    Label letter_count;
    @FXML
    Label points;
    @FXML
```

```
   Label wrongRight;

      TextField[]  letters  =  {let1,let2,let3,let4,let5,let6,let7,let8};  //a
letter for each text box
   public void initialize(){
   }

   public static String getRandom() { //get random word from vocabulary list
       Random random = new Random();
                                                  String      randomWord      =
StartApplication.vocab.get(random.nextInt(StartApplication.vocab.size())).ge
tNewWord();
       return randomWord;
   }

   public void GiveNumLetters(){ //display the number of letters in the word
randomly selected
       String numLetters = String.valueOf(randomWord.length());
       letter_count.setText(numLetters);
   }

   int guessed = 0; //int for correct guesses
   int score = 0; //int for user's score

   public void CheckInput(){
       String str = input.getText(); //set the user's input as a string
        if (randomWord.contains(str)) { //check if the word selected contains
the user's input (letter guessed)
           int index = 0;
            for(int i=0; i<randomWord.length(); i++) { //check every index of
the word selected
               char c = randomWord.charAt(i);

                 if (String.valueOf(c).equals(str)) { //if the current index
(letter it's looking at) matches the user's input
                        setLetter(index, Character.toString(c)); //if true,
display the letter in the text box
                     wrongRight.setText("Right!"); //display "Right!" for
affirmation
                  score += 100; //add 100 points to user's score
                   String scoreText = String.valueOf(score); //turn int into
string
                  points.setText(scoreText);
                  guessed++; //add 1 to correct guesses
                              System.out.println(guessed  +  "  out  of  "  +
randomWord.length()); //keep track of correct guesses

                  if(guessed == randomWord.length()) {
                        wrongRight.setText("Done!"); //Display "Done!" once
all the whole word has been guessed correctly
                  }
```

```
                }
                index++;
            }
        }
        else if (isInteger(str)) {
            wrongRight.setText("No integers!"); //notify to not use integers
        }
        else {
                wrongRight.setText("Wrong"); //display "Wrong" is the letter
guessed is not in the word
            score -= 10; //subtract 10 points from score
            String scoreText = String.valueOf(score);
            points.setText(scoreText); //update score
        }
    }

    private boolean isInteger(String str) {
        try {
               Integer.parseInt(str); //try to convert the input string to an
integer
            return true; //input string can be parsed as an integer
        } catch (NumberFormatException e) {
            return false; //input string cannot be parsed as an integer
        }
    }

    public void setLetter(int index,String str){ //set one letter of the word
for each box in the game
        if(index==0)
            let1.setText(str);
        else if(index==1)
            let2.setText(str);
        else if(index==2)
            let3.setText(str);
        else if(index==3)
            let4.setText(str);
        else if(index==4)
            let5.setText(str);
        else if(index==5)
            let6.setText(str);
        else if(index==6)
            let7.setText(str);
        else if(index==7)
            let8.setText(str);
    }

    public void signOutBtn(ActionEvent actionEvent) throws IOException {
        StartApplication.setRoot("login-view"); //go to Log In page
    }

    public void myVocabBtn(ActionEvent actionEvent) throws IOException {
```

```
        StartApplication.setRoot("vocab-view"); //go to vocabulary list page
    }

    String randomWord;
     public void changeWordBtn(ActionEvent actionEvent) throws IOException {
//restart the game and select a new random word from the list
        randomWord = getRandom();
        GiveNumLetters();
        let1.setText(""); //make text boxes blank again
        let2.setText("");
        let3.setText("");
        let4.setText("");
        let5.setText("");
        let6.setText("");
        let7.setText("");
        let8.setText("");
        input.setText("");
    }
}
```