UniiLanguage

IMCAT

Software Design (Draft)

Alina Kim
Isabel Tuason
Thomas Yung
Christina Vu
Mohammed Imran Shilleh

02.13.2022

Table of Contents

Revision History

Version 1.0

- Date: 2/13/22
- Imported existing information from requirements documentation
- Modified Use Case Diagram to better represent existing use cases
- Added User Stories and Use Diagrams
- Added UML user flow diagrams
- Added 3-layer UML class diagram

Version 2.0

- Date: 2/20/22
- Added descriptions for sequence diagrams
- Made UML subclass diagrams for each use case
- Fixed table of contents
- Added class descriptions for the UML class diagram
- General polishing

Introduction

Statement of Work

UniiLanguage is a web-based application designed to reinforce language maintenance and acquisition in students who are learning a new language. Geared towards use in a classroom setting between the grades of K-12, the application will serve as an accessory to teachers who wish to incorporate it as part of a language-learning curriculum.

For the scope of this project, this application will consist of one mini-game that involves having students draw the prompt provided in distinct time intervals of 10, 30, and 60 seconds (the 10/30/1 game). This mini-game is designed to be primarily accessible in an independent manner (pre-programmed with an existing prompt list for each available language), but can include a method by which students can input a course code to access a custom-designed prompt list for use in a curriculum setting.

Within the span of twenty-one weeks, this project will have been designed, developed, and deployed, following the specifications laid out within this document, and with an emphasized focus on both usability and modularity. This application will then be utilized to assist language-learning students (including, and perhaps especially, those who are learning English), as well as to collect information on the benefits of the Olson-Gillingham method in the context of a multi-sensory approach to language maintenance and acquisition.

Assumptions

1. Users will have access to the internet in order to use the application.
2. Students will have access to a Chromebook or some other web-compatible device.
3. The system will be utilized by the intended audience (teachers, students, etc.)
4. The users' devices will have enough storage capacity to download and install the application.
5. The server for the application host will be reliable and persistent.

Functional Requirements

Use Cases

Requirement: Asynchronous operation
Priority: Must Have
Description: Students can use the application to practice drawing prompt exercises independently, on their own time.

Requirement: Web-based framework; the system must be able to be hosted on a server
Priority: Must Have
Description: The application must be web-based and shall be accessed from a browser from any device. The application must be able to be hosted on a server. It shall accommodate for a variety of screen dimensions and follow web design principles and practices.

Requirement: Drawing Board for students to use
Priority: Must Have

Description: This is the main feature of the application. Students shall be able to draw within a designated area using their mouse or touch-screen device.

Requirement: Customized Prompt Sets
Priority: Reach Goal
Description: Teachers should be able to create custom prompts associated with a course code in order to allow their students to access prompts associated with their curriculum.

Requirement: App can generate its own prompts off of basic sight words
Priority: Must Have
Description: The app should be able to generate its own prompts based on pre-programmed word lists so that students are able to practice using the application without the need for a teacher's assignments.

Requirement: App can track previously assigned prompts using browser cookies
Priority: Good to Have
Description: The app should not give users repeated prompts by keeping track of previously assigned prompts using cookies in the user's browser.

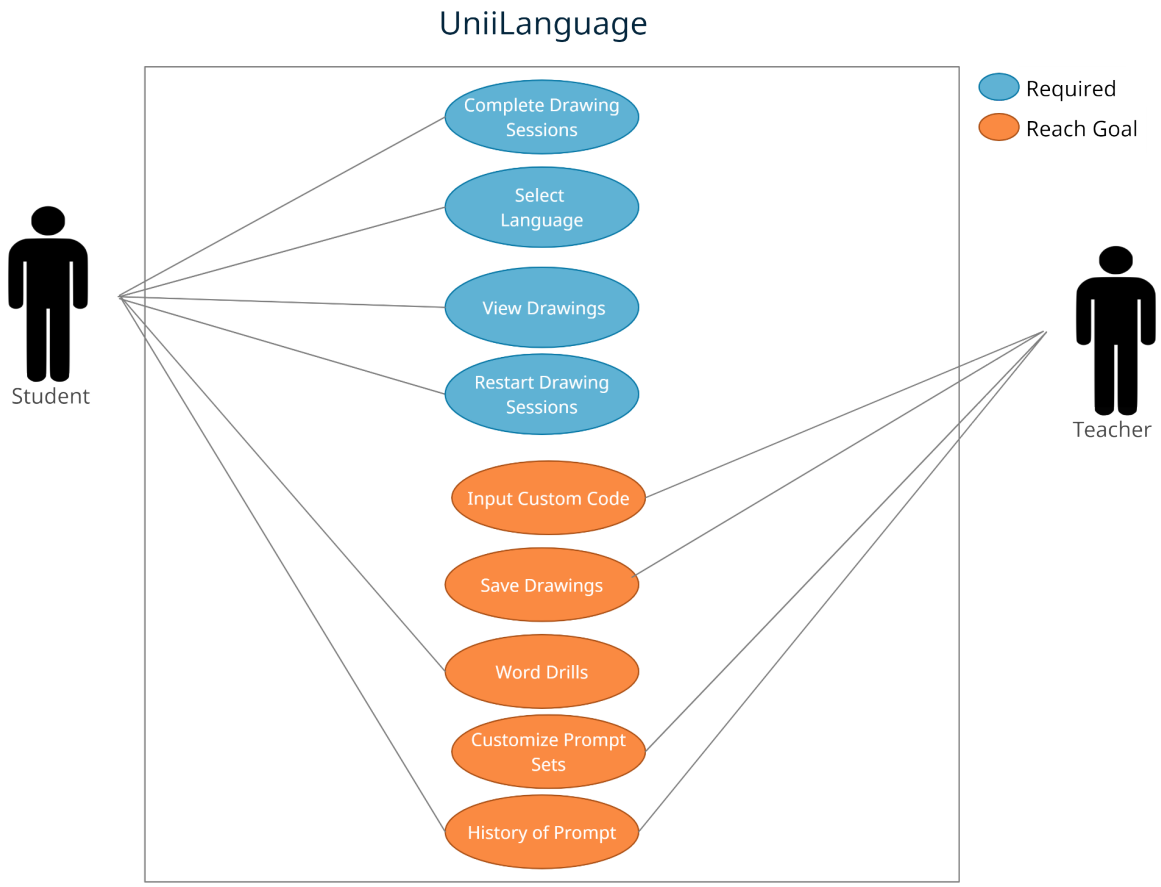
Requirement: The app can generate its own prompts off of teacher-assigned word lists.
Priority: Reach Goal
Description: It would be nice for the app to be able to generate its own prompts based on word lists that are assigned by teachers.

Requirement: There shall be a home page where students can select the language they want to practice.
Priority: Must Have
Description: Students should be able to select a language they want to practice when they are on the website.

Requirement: Students should have the option to save their drawings.
Priority: Good to Have
Description: After finishing each prompt from their chosen language set, the system shall prompt the user if they would like to save their drawings for future reference.

Requirement: Students shall have the option to begin another drawing prompt after completing one.
Priority: Must have
Description: After finishing the drawing session, the system shall prompt the user if they would like to start a new drawing session with a different set of prompts.

Use Case Diagram



User Stories

User: Student

Priority: Required/Must Have

Use Case: Student uses Uniilanguage to complete a randomized drawing session

User Story:

Matthew is a 3rd grader and native English speaker. He is enrolled in special education at his elementary school, and loves spending his free time on coolmathgames.com or playing Minecraft with his friends. He has Spanish class every other day, and today his Spanish teacher assigned him word drills and Unilanguage practice as homework. After

he gets home and eats a snack, Matthew logs on to his family desktop computer and completes his Spanish word drills. He then navigates to the Unilanguage website and selects “Spanish” when asked what language he wants to study. Unilanguage prompts Matthew to draw a “Perro con un Sombrero'' (Dog with a Hat) in 10 second, 30 second, and 1 minute intervals. Unilanguage shows Matthew all his drawings after he completes them, and Matthew runs to the next room to get his dad and show him his silly drawings. Then, Matthew presses the “Start New Drawing Session'' button and completes 2 more drawing sessions to complete his homework.

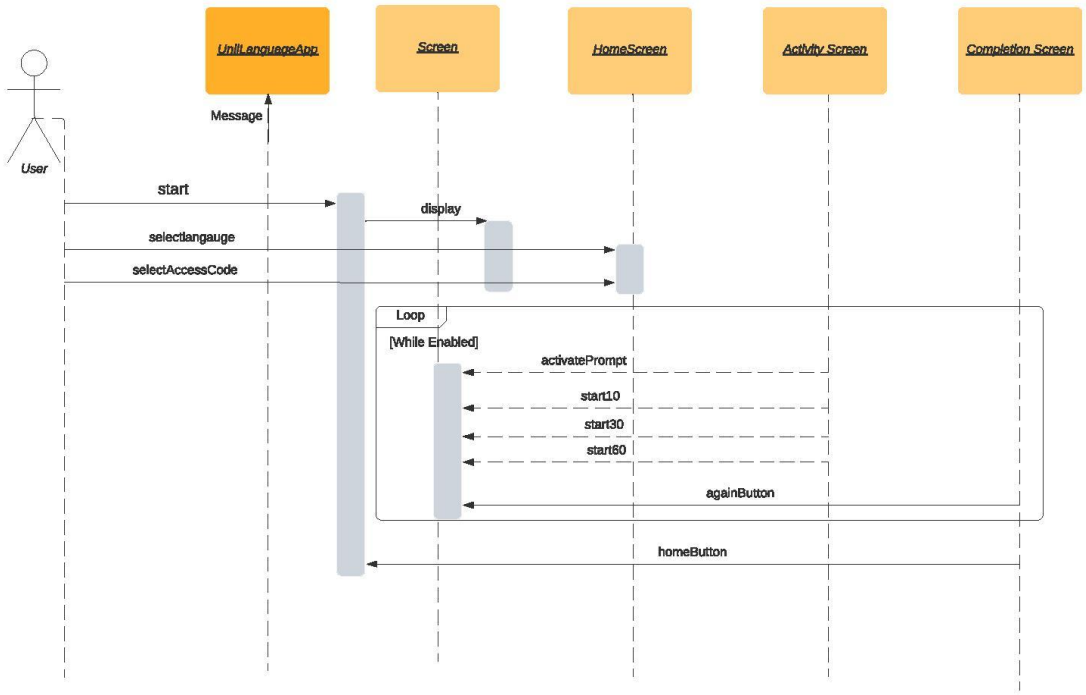
User: Student
Priority: Reach Goal
Use Case: Student uses Uniilanguage to complete a custom, teacher-created drawing session and saves drawings
User Story:
Isabella is a 4th grader and native speaking Spanish speaker. She struggles with dyslexia and ADHD, and is working on better pronouncing her English “j”, “h”, “ch'' letter sounds. In class, Isabella’s teacher introduces a new list of animal words for the class to study. She gives the class a custom Uniilanguage code to help with some of these new words and letter sounds. When Isabella gets home, she opens up her Chromebook — lended out to her by her school — finishes her assigned word drills, and navigates to the Uniilanguage website. She enters the custom code provided by her teacher written down in her notebook. The site accepts her code, then prompts her to draw a “Chicken Wearing a Jacket''. She giggles at the prompt, then proceeds to draw a chicken wearing a jacket three times in increasing intervals of 10 seconds, 30 seconds, and 1 minute. The website displays all three of her drawing attempts alongside the prompt, and asks if Isabella would like to save the images to her school computer. Isabella decides to save the images, then asks her teacher at school the next day if she can print out the pictures she saved to show to her mom. Isabella’s mom is delighted to see her daughter’s pictures and learn more about her daughter’s learning. She hangs the printed photos on the family fridge for everyone to see.

UML Design Diagrams

Use Case Sequence Diagrams

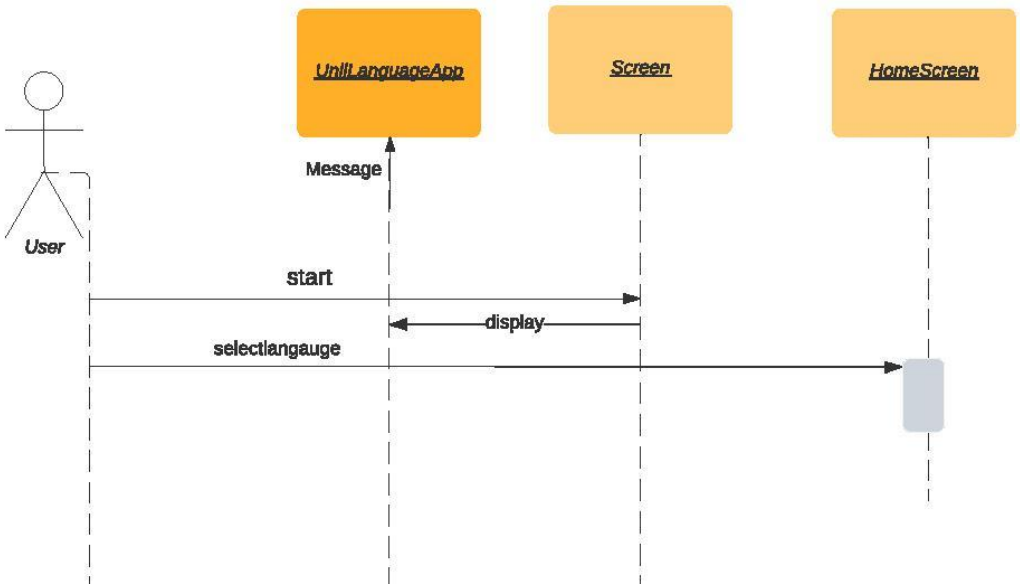
Requirements

- Asynchronous operation
- Web-based framework; the system must be able to be hosted on a server



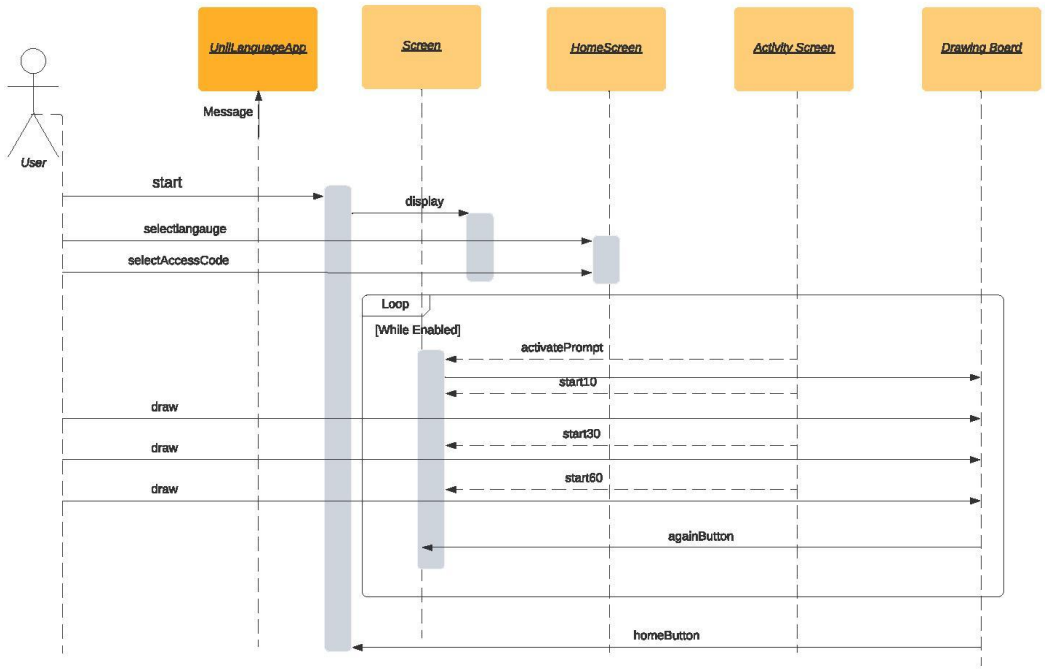
Requirements

- There shall be a home page where students can select the language they want to practice.

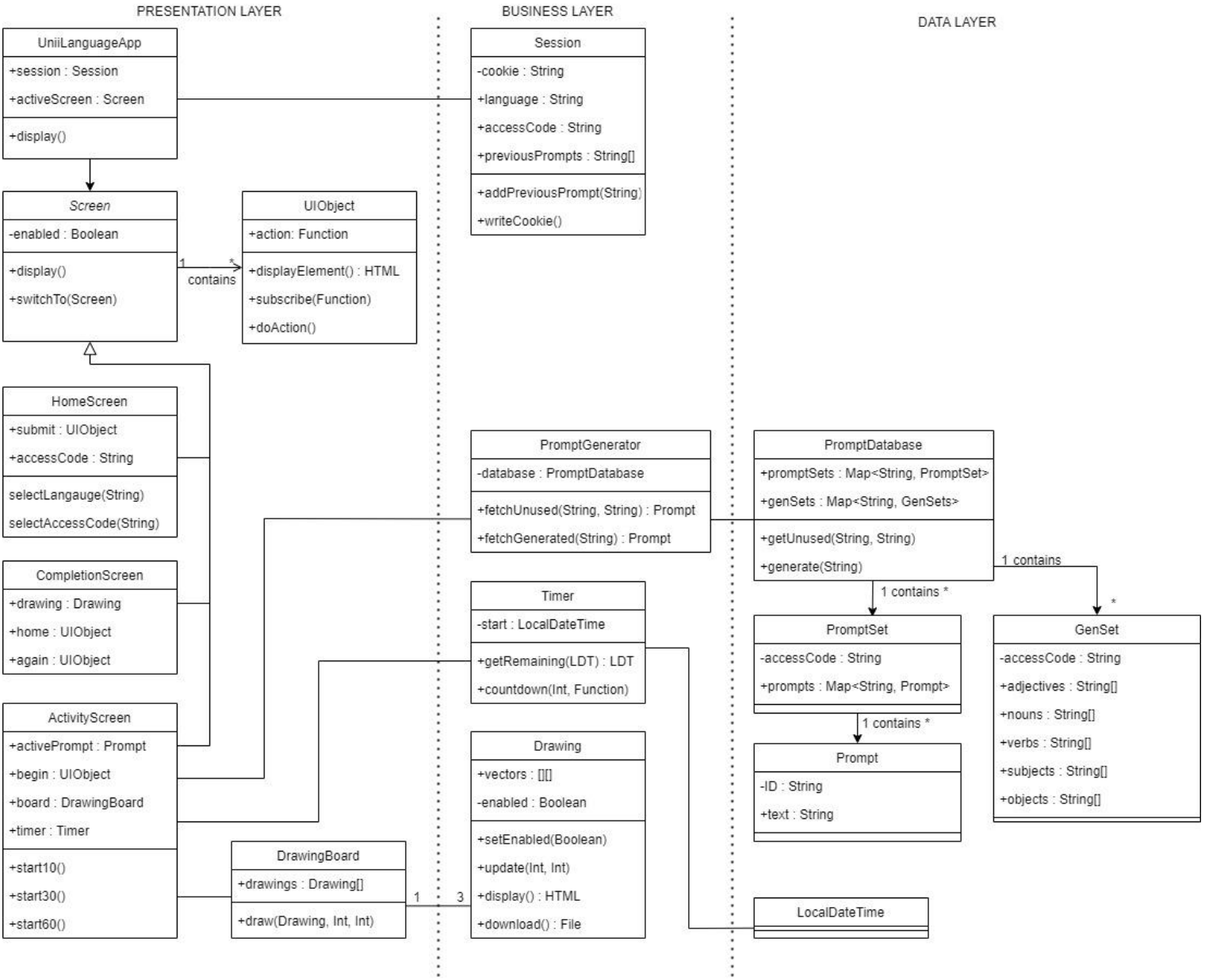


Requirements

- Drawing Board for students to use
- Students shall have the option to begin another drawing prompt after completing one.



3-Tier UML Class Diagram



Classes

UniiLanguageApp

> Primary application framework. Tracks static information, including current session information, along with the active screen.

Screen

> Abstraction of code behind front-end webpage. display() function is an abstraction of the HTML build (depending on development framework), and switchTo(Screen) is an abstraction of a link redirect.

HomeScreen

: Screen with Language and Access Code selection prompts. Both serve functionally identical purposes, with Access Code overriding Language selection. Both are used as keys to access prompt lists in the database.

CompletionScreen

: Screen displayed after activity is completed. Displays created drawing, and offers options for download and retrying.

ActivityScreen

: Screen displayed during activity. Uses timer hooks to activate / disable drawing areas. Contains functionality for executing the activity based on the Session information and prompt list selected.

UIObject

: Abstraction of an interactive HTML element. Constructed with an even subscription and triggered execution function.

DrawingBoard

: Interactive element that can be drawn on by having the mouse button down on its surface. Tracks drawn history using x and y coordinates.

Drawing

: Saves drawn information and converts to a downloadable file.

Timer

: Countdown timer that hooks into functions and triggers them asynchronously once the allotted time has passed.

PromptGenerator

: Generates prompts based on database and Session information. Fetches unused prompts and generates new ones if such lists exist. Serves as the interface between the system as a whole and the database which contains prompts by enforcing access rules.

PromptDatabase

: Stores maps of Language/Access Code to sets of prompts.

PromptSet

: List of prompts associated with a given access code.

Prompt

: String text with an identifier to check for previous assignments.

GenSet

: Lists of prompt pieces that can be combined to form unique prompts.

Session

: Interface between browser cookie and system utilization. Converts between the two formats much like a JSONObject does in other languages. Contains utility functions to extract partial information from the cookie.