The Red Book

Software Design Description

Author: Hanna Jung, Joon Han, Sean Lee

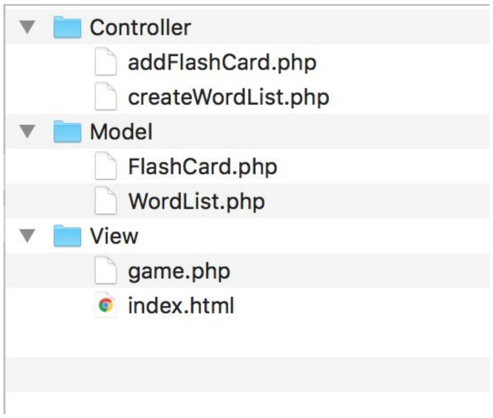
Abstract: This document describes the software design for Red Book, a UML application with other constants.

Based on IEEE Std 1016TM-2009 document format

1.1 Overview

Technology Stacks	Name	Version	Rationale
OS	Ubuntu	12.04.5	Operating system of school server
Web Server	Apache	2.4.28	Familiarity
Database	MySQL	5.5.54	Familiarity
Programming Languages	Python	2.7.3	Compatibility with third party APIs and Back End data processing
	PHP	7.2.2	For Front End Development
	Javascript	1.8.5	For Front End Development
APIs	Bing Image Search API	7	To generate images for new flashcards created by teachers
	Imgur API	3	Efficient image storing
	Google Could Text-to-Speech	1	To provide audio pronunciation of words
Library	requests	2.22.0	Required library for Bing Image Search API
	Pillow	6.2.0	For some image processing / Required library for Bing Image Search API
	matplotlib	3.1.1	Rough sketch of student data visualization / Required library for Bing Image Search API

The decision to use this particular tech stack is due to the team members' familiarity and past experiences of using the technology. The Red Book will be developed using the Model, View, Controller (MVC) framework model. However due to the choice of the tech stack, available popular frameworks such as CakePHP (https://cakephp.org) cannot be used. Instead the team will create our own basic MVC framework. An example structure of the screenshot is provided on the right.



```
▼ Controller
    addFlashCard.php
    createWordList.php
▼ Model
    FlashCard.php
    WordList.php
▼ View
    game.php
    index.html
```

1.2 Components and interfaces

Interface: Main	
Attributes	**Description**
None	None
Operations	**Description**
refresh()	Keeps the window updated for real-time dynamic usage
createUser()	Allows for the creation of a new user to use the application
login()	Already existing users may access the application and their data will be retrieved from the database
playGames()	Users will be able to play games that are available
Class: User	
Attributes	**Description**
Username (String)	Unique username for identification and access
Password (String)	Password which must be required to access the account
UserType (Int)	0 for Student, and 1 for Teacher to differentiate levels of access and options application provides for each UserType
ProfilePic (String)	A string to store HTML image hyperlink of the user
Operations	**Description**
inputTeacherCode()	Action in which a user enters a specific sequence of characters to change their UserType to 1
UploadProfilePic()	Action in which users upload profile pictures associated with their accounts
Class: Teacher (Descendant of User)	
Attributes	**Description**

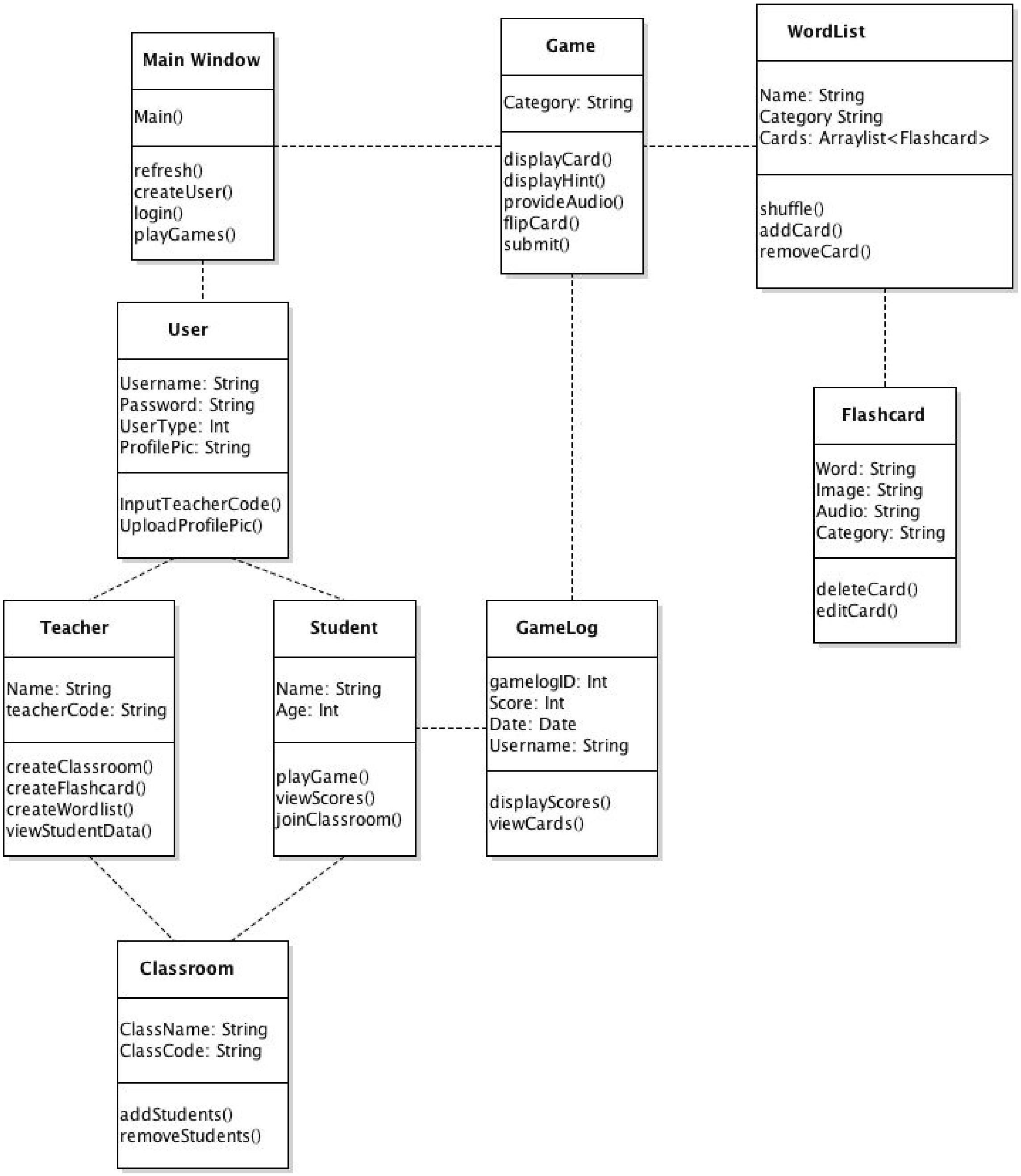
Name (String)	Personal name of the teacher
teacherCode (String)	The specific sequence of characters associated with the teacher
Operations	**Description**
createClassroom()	Allows the teacher to create a class **Classroom**
createFlashCard()	Allows the teacher to create a class **Flashcard**
createWordList()	Allows the teacher to create a class **WordList**
viewStudentData()	The teacher is able to view an organized dataset of their classrooms' students' scores and activities.
Class: Student (Descendant of User)	
Attributes	**Description**
Name (String)	Personal name of the student
Age (Int)	Age of the Student
Operations	**Description**
playGames()	Users will be able to play games that are available
viewScores()	Teachers will be able to view scores of their students
joinClassroom()	Allows a student to join a classroom that a teacher has created
Class: Gamelog	
Attributes	**Description**
gamelogID (Int)	Unique identification of the game session log for easy access
Score (Int)	The score of the particular game session
Date (Date)	The date of game session
Username (String)	Username of the user who played the game session

Operations	**Description**
displayScores()	Displays individual scores of each answer of the game session
viewCards()	Displays each individual flashcard that was in the particular game session
Class: Game	
Attributes	**Description**
Category (String)	Category of the wordlist
Operations	**Description**
displayCard()	Displays the image of the flashcard
displayHint()	Displays hint of the word of the flashcard
provideAudio()	Provides an audio .wav file of the word
flipCard()	Once answer has been submitted, the flashcard is flipped to show the valid word
submit()	The user submits their input.
Class: Wordlist (Descendant of User)	
Attributes	**Description**
Name (String)	Name of the wordlist
Category (String)	Category of the wordlist
Cards (Arraylist <FlashCard>)	List of the flashcards that are associated with the wordlist
Operations	**Description**
shuffle()	Randomizes the order of the flashcards to be displayed in-game
addCard()	Adds a flashcard to the Cards arraylist
removeCard()	Removes a flashcard from the Cards arraylist

Class: Flashcard	
Attributes	**Description**
Word (String)	Word label of the image in the flashcard
Image (String)	Image URL of the flashcard image
Audio (String)	Path to the audio file of the word in the flashcard
Category (String)	Category of the flashcard
Operations	**Description**
deleteCard()	Deletes the card from the application
editCard()	Edits the card attributes
API: Imgur API	
Operation	**Description**
Upload	Accepts a base64 image file then uploads it into the imgur server, and returns the imgur image hyperlink. This saves vast amount of database space. Specific details of the API can be accessed: https://apidocs.imgur.com/?version=latest
API: Bing Image Search API	
Operation	**Description**
Request	Accepts a string and returns a set of most relevant images. This is used for creating new flashcards to provide teachers with images of their input. Specific details of the API can be accessed: https://dev.cognitive.microsoft.com/docs/services/8336afba49a84475ba401758c0dbf749/operations/56b4433fcf5ff8098cef380c
API: Google Cloud Text-to-Speech API	
Operation	**Description**
Synthesize	Accepts a string and returns audio file of the string. This is used for providing audio cues of the flashcard words for

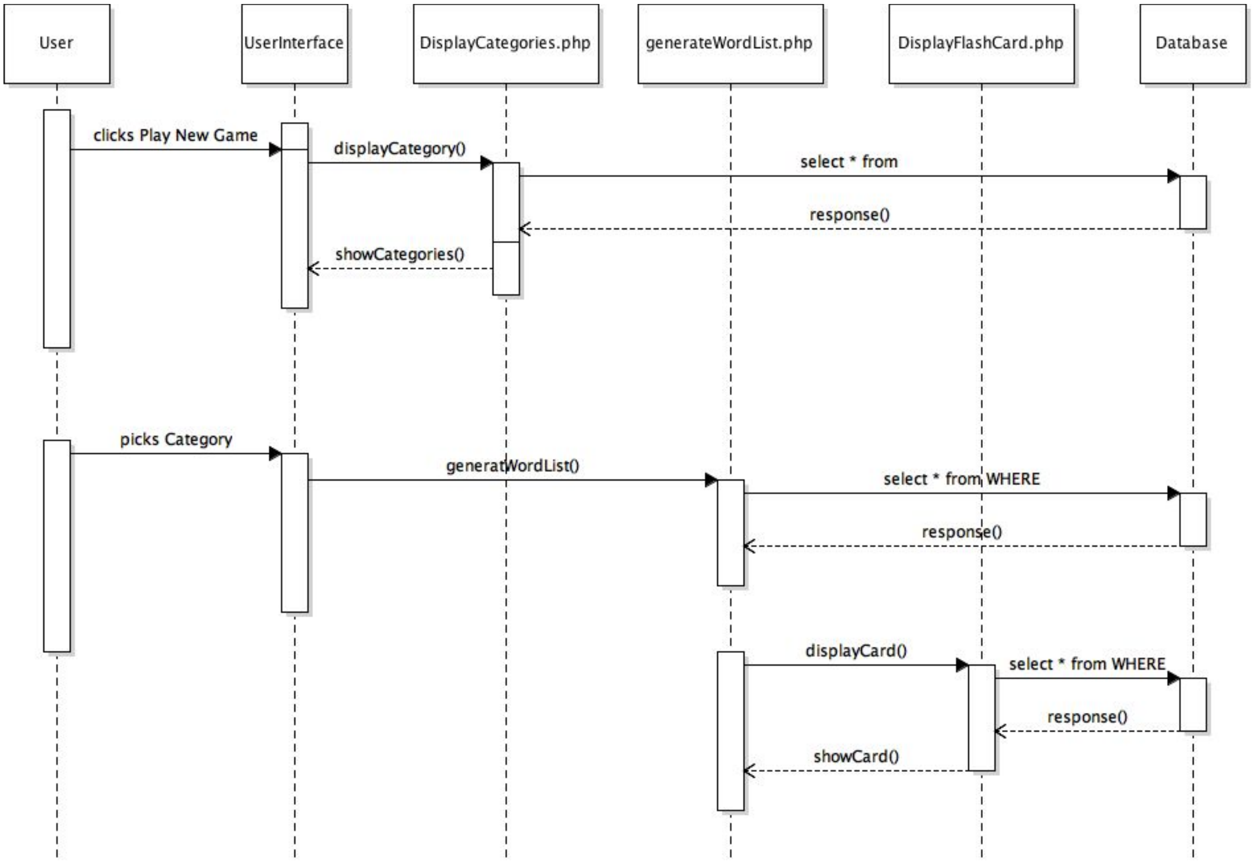
	the students. Specific details of the API can be accessed: https://cloud.google.com/text-to-speech/docs/

1.3 UML Class Diagram

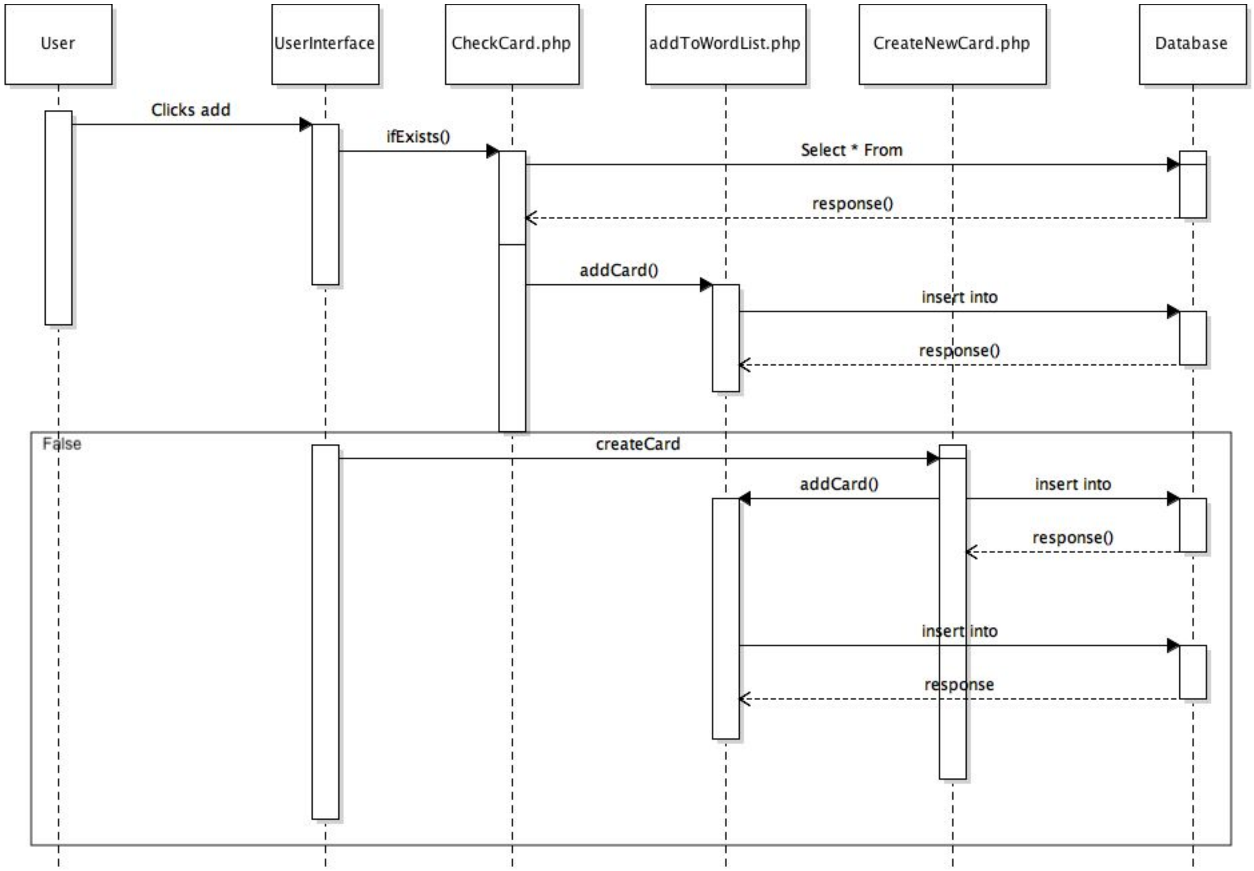


1.4 UML Sequence Diagrams

1.4.1 Play New Game

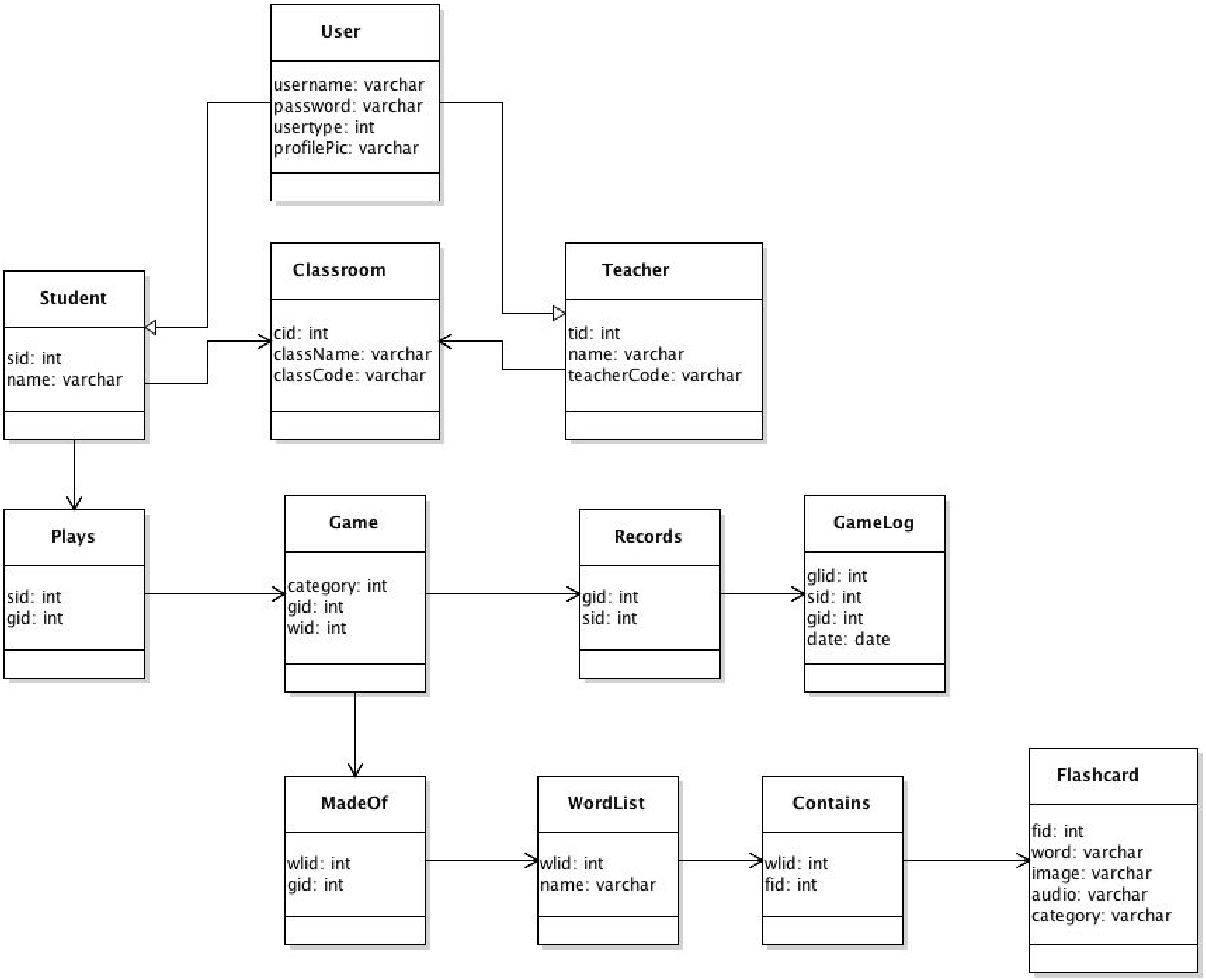


1.4.2 Add Flashcard to Wordlist



1.5 Data Storage

The Red Book will store date using MySQL Version 7.2.2. All of the data stored in the MySQL tables will mostly be ‘varchar’ or ‘int’ to save space. Images will be uploaded to an external host using the Imgur API and the hosted links of the images will be stored in the MySQL tables. Audio files will be stored in the SUNY Korea provided server and the directory path will be stored in the MySQL tables. A high-level database schema is shown below:



1.6 Deployment

The web application will be deployed using a SUNY Korea provided web server. The services to host the website has already been set up with Ubuntu version 12.04.5 along with Apache version 2.4.28 and MySQL Version 7.2.2.

Website Link: http://momgoose.cs.sunykorea.ac.kr/iauth/test/docs/CSE308FINAL/

1.7 Alternatives

1.7.1 Using Machine Learning API to Allow Student-created Flashcards

An initial design choice that was explored was to allow students to upload their own images and a third party API would return the English label.

Pros: Additional user-interaction and personalization could engage students use the application more frequently. Another advantage of this design is that it can be expanded to allow students to share their flashcards with other students creating an additional layer of social interaction.

Cons: The design relied heavily on the accuracy and reliability of the third party API. Additionally, a system would need to be designed to filter out inappropriate images, which would more complexity in to building the application.

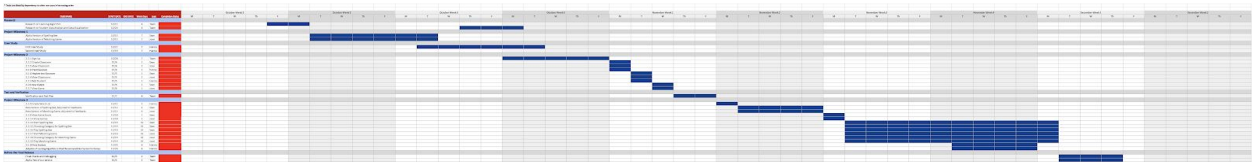
1.7.2 Creating Educational Tools for Older Demographic

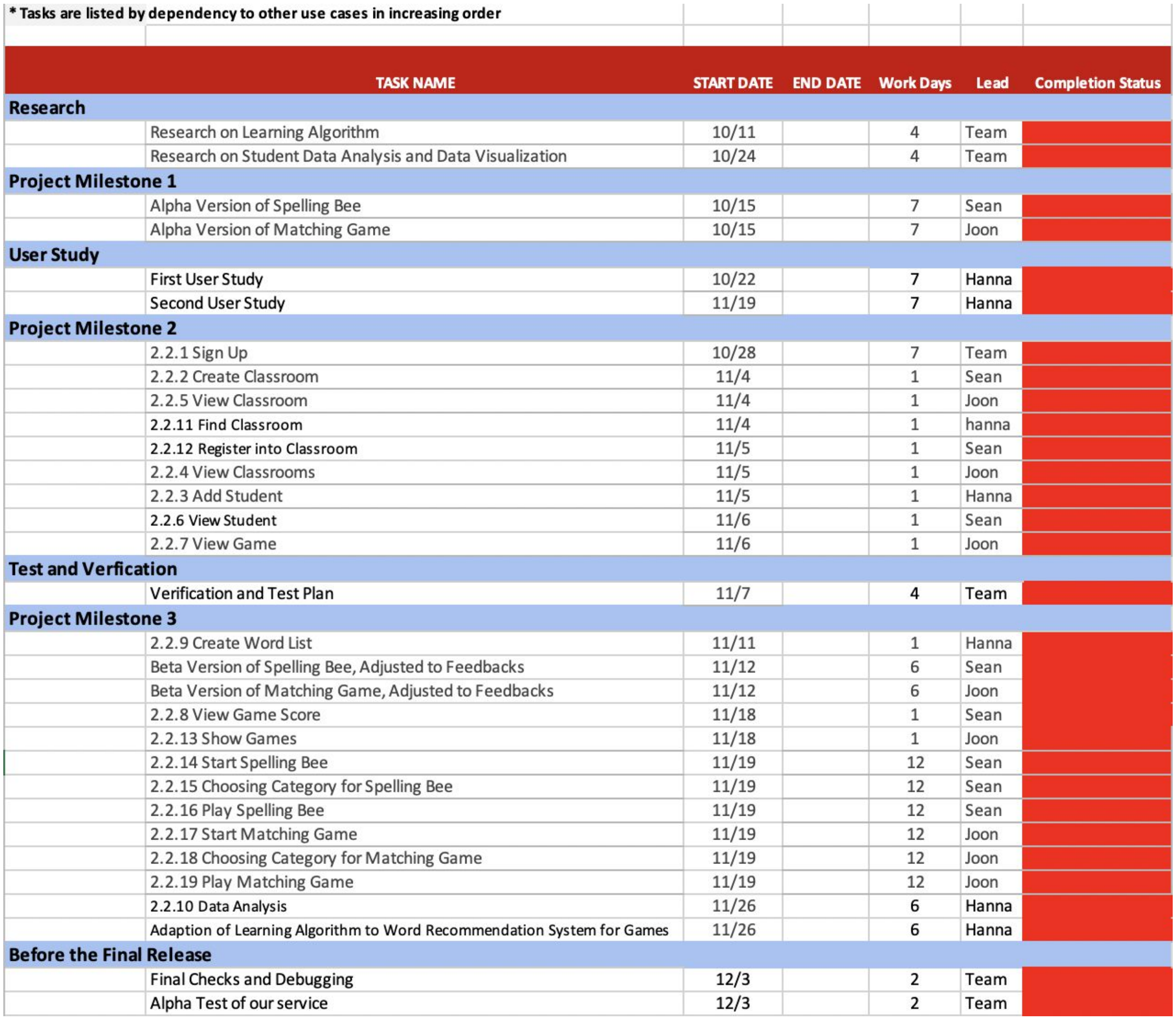
Another design choice that was explored was to expand the target audience of The Red Book to all age groups by creating more difficult and different types of learning games.

Pros: By creating additional levels of difficulty and different games, The Red Book would be usable for all age groups. Not only would the increased difficulty and complexity attract more age groups, it could also increase the amount of time a student uses the application, as it will constantly offer more difficult learning material.

Cons: Although the added difficulty may allow The Red Book to be more usable to a larger audience, it’s added functionalities and options may make The Red Book not be suitable for elementary students.

2. Schedule





* Tasks are listed by dependency to other use cases in increasing order

	TASK NAME	START DATE	END DATE	Work Days	Lead	Completion Status
Research						
	Research on Learning Algorithm	10/11		4	Team	🟥
	Research on Student Data Analysis and Data Visualization	10/24		4	Team	🟥
Project Milestone 1						
	Alpha Version of Spelling Bee	10/15		7	Sean	🟥
	Alpha Version of Matching Game	10/15		7	Joon	🟥
User Study						
	First User Study	10/22		7	Hanna	🟥
	Second User Study	11/19		7	Hanna	🟥
Project Milestone 2						
	2.2.1 Sign Up	10/28		7	Team	🟥
	2.2.2 Create Classroom	11/4		1	Sean	🟥
	2.2.5 View Classroom	11/4		1	Joon	🟥
	2.2.11 Find Classroom	11/4		1	hanna	🟥
	2.2.12 Register into Classroom	11/5		1	Sean	🟥
	2.2.4 View Classrooms	11/5		1	Joon	🟥
	2.2.3 Add Student	11/5		1	Hanna	🟥
	2.2.6 View Student	11/6		1	Sean	🟥
	2.2.7 View Game	11/6		1	Joon	🟥
Test and Verfication						
	Verification and Test Plan	11/7		4	Team	🟥
Project Milestone 3						
	2.2.9 Create Word List	11/11		1	Hanna	🟥
	Beta Version of Spelling Bee, Adjusted to Feedbacks	11/12		6	Sean	🟥
	Beta Version of Matching Game, Adjusted to Feedbacks	11/12		6	Joon	🟥
	2.2.8 View Game Score	11/18		1	Sean	🟥
	2.2.13 Show Games	11/18		1	Joon	🟥
	2.2.14 Start Spelling Bee	11/19		12	Sean	🟥
	2.2.15 Choosing Category for Spelling Bee	11/19		12	Sean	🟥
	2.2.16 Play Spelling Bee	11/19		12	Sean	🟥
	2.2.17 Start Matching Game	11/19		12	Joon	🟥
	2.2.18 Choosing Category for Matching Game	11/19		12	Joon	🟥
	2.2.19 Play Matching Game	11/19		12	Joon	🟥
	2.2.10 Data Analysis	11/26		6	Hanna	🟥
	Adaption of Learning Algorithm to Word Recommendation System for Games	11/26		6	Hanna	🟥
Before the Final Release						
	Final Checks and Debugging	12/3		2	Team	🟥
	Alpha Test of our service	12/3		2	Team	🟥

October Week 2					October Week 3				
M	T	W	Th	F	M	T	W	Th	F
				🟦	🟦				
						🟦	🟦	🟦	🟦
						🟦	🟦	🟦	🟦

October Week 4					October Week 5				
M	T	W	Th	F	M	T	W	Th	F
			■	■	■				
■	■								
■	■								
	■	■	■	■	■	■			
					■	■	■	■	■

November Week 1					November Week 2				
M	T	W	Th	F	M	T	W	Th	F
■									
■									
■									
	■								
	■								
	■								
		■							
		■							
			■	■					
					■				
						■	■	■	■
						■	■	■	■

November Week 3					November Week 4				
M	T	W	Th	F	M	T	W	Th	F
■									
■									
	■	■	■	■	■	■	■	■	■
	■	■	■	■	■	■	■	■	■
	■	■	■	■	■	■	■	■	■
	■	■	■	■	■	■	■	■	■
	■	■	■	■	■	■	■	■	■
	■	■	■	■	■	■	■	■	■
						■	■	■	■
						■	■	■	■

December Week 1				
M	T	W	Th	F
🟦				
🟦				
🟦				
🟦				
🟦				
🟦				
	🟦	🟦	🟦	
	🟦	🟦	🟦	