

SANTA CLARA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: June 13, 2023

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Brian Burnett
Jillian Coveney
Kristin Tong
Nanki Sekhon

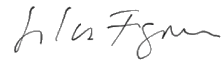
ENTITLED

TwigeEnglish

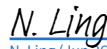
BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING

BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING



Thesis Advisor



N. Ling (Jun 26, 2023 11:45 PDT)

Department Chair

TwigeEnglish

by

Brian Burnett
Jillian Coveney
Kristin Tong
Nanki Sekhon

Submitted in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Computer Science & Engineering
Bachelor of Science in Web Design & Engineering
School of Engineering
Santa Clara University

Santa Clara, California
Jun 13, 2023

TwigeEnglish

Brian Burnett
Jillian Coveney
Kristin Tong
Nanki Sekhon

Department of Computer Science and Engineering
Santa Clara University

Jun 13, 2023

ABSTRACT

TwigeEnglish is an educational platform aimed to teach English in a fun and interactive way. It comprises a mobile application for students and a web application for pre-determined administrators. The mobile application offers three key features which include: a pictictionary-like game, a dictionary flash card game, and a user-friendly interface for uploading photos to be used in both games. The web application will primarily focus on providing administrators with an interface to manage the student-photo database. With moderation powers, administrators can seamlessly review and make decisions to approve or deny the inclusion of photos in the active pictictionary and dictionary pools. This ensures that the content remains relevant, appropriate, and aligned with the platform's educational goals.

Table of Contents

1 Introduction	8
1.1 Motivation	8
1.2 Solution	8
2 Requirements	10
3 Use Cases	12
4 Activity Diagrams	15
4.1 Student Activity Diagram	15
4.2 Administrator Activity Diagram	16
5 Final Design	17
5.1 Mobile Application	17
5.2 Web Application	23
6 Technologies Used	31
6.1 Mobile Application	31
6.2 Administrative Webpage	31
7 Architectural Diagram	32
8 Design Rationale	34
8.1 Technologies Used	34
8.2 Architecture	34
9 Testing	35
9.1 Unit, Widget, and Integration Tests	35
9.2 User Testing	35
10 Risk Analysis	36
11 Societal Issues	37
11.1 Ethical	37
11.2 Social	37
11.3 Political	37
11.4 Economic	37
11.5 Health and Safety	38
11.6 Manufacturability	38
11.7 Usability and Security	38
11.8 Compassion	38

12 Conclusion	39
12.1 Risk Resolution	39
12.2 Obstacles Encountered	40
12.3 Future Work	40
12.4 Lessons Learned	40
12.5 Acknowledgements	41
Appendix A	42
Appendix B	46

List of Figures

3.1 Use Case Diagram	14
4.1 Student User Activity Diagram	15
4.2 Administrative Users Activity Diagram	16
5.1 Mobile Application Student Account Sign In	17
5.2 Mobile Application Home Screen	18
5.3 Mobile Application Upload Screen	19
5.4 Mobile Application Play Pictionary Screen	20
5.5 Mobile Application Play Pictionary Screen	21
5.6 Mobile Application Play Pictionary Screen	22
5.7 Administrative Website: Login	23
5.8 Administrative Website: Register	24
5.9 Administrative Website: Home	25
5.10. Administrative Website: Approved Photos	26
5.11 Administrative Website: Pending Uploads	27
5.12 Administrative Website: List of Users	27
5.13 Administrative Website: Add a User	28
5.14 Administrative Website: View a User	28
7.1 Architectural Diagram	31

List of Tables

2.1 Functional Requirements	10
2.2 Non-functional Requirements	10
2.3 Design Constraints	11
3.1 Upload Picture Use Case	13
3.2 Manage Photos Use Case	13
3.3 Play Pictionary Use Case.....	13
3.4 Play Dictionary Use Case.....	14
10.1 Risk Analysis	34

Chapter 1

Introduction

1.1 Motivation

Rwanda is one of many African countries that has struggled with the lasting impacts of European colonialism. The country's oppressive and violent history has created a cultural divide and set back the development and growth of Rwanda's economic and social institutions. The World Bank reported a significant economic boom in Rwanda in the early 2000s. Sadly, this promising growth was severely stunted by the emergence of the COVID-19 pandemic in 2019. Additionally in 2019, the official language of instruction in primary schools was abruptly changed to English. With ninety-nine percent of the population speaking Kinyarwanda, that leaves less than one percent of French and English speakers. This means rural, Kinyarwanda-speaking communities are struggling to acquire the educational resources they need to learn English.

The Sheer Love Rwanda organization aims to support reintegrating street children with their community and empowering their education. Sheer Love Rwanda estimates that ninety-five percent of the Sheer Love Rwanda parents do not speak English and have not received basic education as a consequence of the 1994 Rwandan Genocide which halted education for many Rwandan communities. An estimated 7,000 Rwandan children are living on the streets because their families and homes have been left in fragile conditions. Consequently, local communities are looking to prioritize proper education for these children to secure their future.

1.2 Solution

There is a language barrier separating Rwanda's current education system and its students. We developed TwigeEnglish as a way to teach English in a more interactive and growing manner. This app gives the power to the student to upload their own pictures to a shared image pool for others to learn from. Pending requests are sent to an online administration website, where a teacher (administrator) has access to either approve or deny any student submissions going into the database. The administrator's responsibilities are

granted to teachers within the Sheer Love Rwanda organization for our case. They verify if the image is of good quality, appropriate, not a duplicate, and if the English word in association is accurate.

From there, each student is distributed points depending on their number of approved dictionary uploads and correct guesses when playing the picture game. The gamification element aims to reinforce continuous student learning by offering a fun incentive. Points in a game are often used to encourage the completion of tasks and inspire motivation. One recommendation from Sheer Love Rwanda is to help keep students engaged with their language learning.

Our solution addresses the problem by providing a unified and easy-to-use platform accessible to all users. The mobile application can be downloaded on both Android and IOS devices and used with different orientations. The main challenge is the uncertainty of internet access that students will have at home or outside the classroom. To mitigate this concern, the plan includes a task to integrate an “offline” version of the app. This allows students to resume their language learning practice in the absence of WiFi connection, but does sacrifice some device storage.

Chapter 2

Requirements

Establishing critical and recommended requirements helped streamline our development process and ensure that we were meeting the needs of our partners at Sheer Love Rwanda. Our functional requirements are in Table 2.1, our non-functional requirements can be seen in Table 2.2, and Table 2.3 shows the constraints limiting our design.

Table 2.1 Functional

Priority	Requirements
Critical	Allow users to upload photos to image database Allow users to add to add new words to the dictionary Allow users to create an account using a phone number as password Allow users to upload profile information Administrator permissions for image/word databases Dictionary mode to study words Pictionary mode to for users to match images to study material
Recommended	Create a points system to incentivize studying Quick search for keywords

For our functional requirements, our application needs to provide student users with the ability to upload both photos and vocabulary terms to the database where administrators oversee submissions. Once approved, the students should be able to review their new terminology through the play pictionary and dictionary modes in the mobile app. Some secondary functional requirements include implementing a points system to gamify the app and encourage greater engagement. We also considered adding a quick search function in the dictionary mode to assist students looking for a particular vocabulary word and image pair.

Table 2.2 Non-Functional

Priority	Requirements
Critical	Ease of user photo upload Fast user photo upload Must be easy to vet and approve photos on administrator side
Recommended	Offline accessibility

The non-functional requirements focus on the quality of our application. Student users should feel a seamless experience when using the app. Upload processes for the images and terminology must be quick and easy so students are not discouraged from frequently interacting with the app. A user-friendly approach can lead to more dedicated users and more photo/word upload submissions makes for a larger dictionary.

On the administration side, we must guarantee that teachers can get through all of the photo/word requests. Our application currently asks the administrators to manually approve any new submissions. This system demands a simple process to help streamline efficiency. Another recommendation is to include an offline accessibility mode for students who only have reliable internet at school. The lack of consistent WiFi connection is a barrier we need to work on minimizing for students. We hope the students can engage with the same functionality when offline, so offline upload requests will be saved and sent to the administrator side during the next time they connect to the internet.

Table 2.3 Design Constraints

Constraints
Users must have offline access on Android devices but administrators must have online access to databases. Users must also be able to connect online to submit images to administration.

The biggest constraint to assess is the variability of our users' internet access. Offline access to the mobile app gives students more accessibility and increases the likelihood of continued usage. However, this prevents images and words from properly being uploaded to the database. Information that is stored on the mobile application will take up more space and require greater storage costs. Additionally, administrators need access to WiFi to allow communication with the student mobile application, to add users, and approve/deny photos.

Chapter 3

Use Cases

Our system delivers functionality to two primary actors through two distinct applications. The first application is designed for students and can be accessed through any mobile device. Students can engage with a flashcards game to continue aiding their English learning journey. This user-friendly application allows students to upload original photos through the process outlined in Table 3.1. Those approved photos are then added to the shared dictionary for all students to interact with.

The two main ways students interact with the dictionary are through playing the pictictionary game or reviewing the dictionary flashcards, as outlined in Tables 3.3 and 3.4, respectively. Both games are easily located as large buttons on the homescreen for the mobile application. The pictictionary game will present a photo from the student dictionary and return a list of random letters. Among those letters, are the correct letters needed to spell the word represented in the image. It is the student's objective to correctly identify the English word and determine the proper spelling. Additionally, the dictionary game offers digital flashcards as a way to provide an easy way to practice learning vocabulary terms. It is important to note that any requested uploads from a student are not included in any games until an administrator approves it.

The second application is specifically designed for administrators and can be accessed through a web browser. The teachers have the power to moderate participating students and handle pending upload requests. A dashboard will appear as the landing page after they sign in using an administrator's email and password. Here, users can efficiently manage the pending requests and view the number of students with accounts. Using the labeled sections on the navigation menu, administrators may view and edit the dictionary database at any time as described in Table 3.2. This ensures that the learning environment for the students remains safe and relevant.

Table 3.1 Upload Picture Use Case

Goal	Allow actors to upload photos to database where they will be used for games
Actor(s)	Student
Precondition(s)	Actor has photographed one or more objects and has submitted them under the proper tab in the phone application
Postcondition(s)	Photo is now viewable from the admin portal. If a photo is appropriate it will be added to the dictionary and pictionary games for all students to see
Exception(s)	Actor is unable to upload photo because of lack of internet

Table 3.2 Manage Photos Use Case

Goal	Maintain an appropriate and accurate dictionary database
Actor(s)	Admin
Precondition(s)	Access to all approved and unapproved student uploads in the dictionary database. Actor controls the addition and deletion of words to the dictionary.
Postcondition(s)	Actor verifies the photo or deems it inappropriate. Photo is discarded or will be visible to all in the verified photo pool.
Exception(s)	Students have not uploaded photos so there are none to process

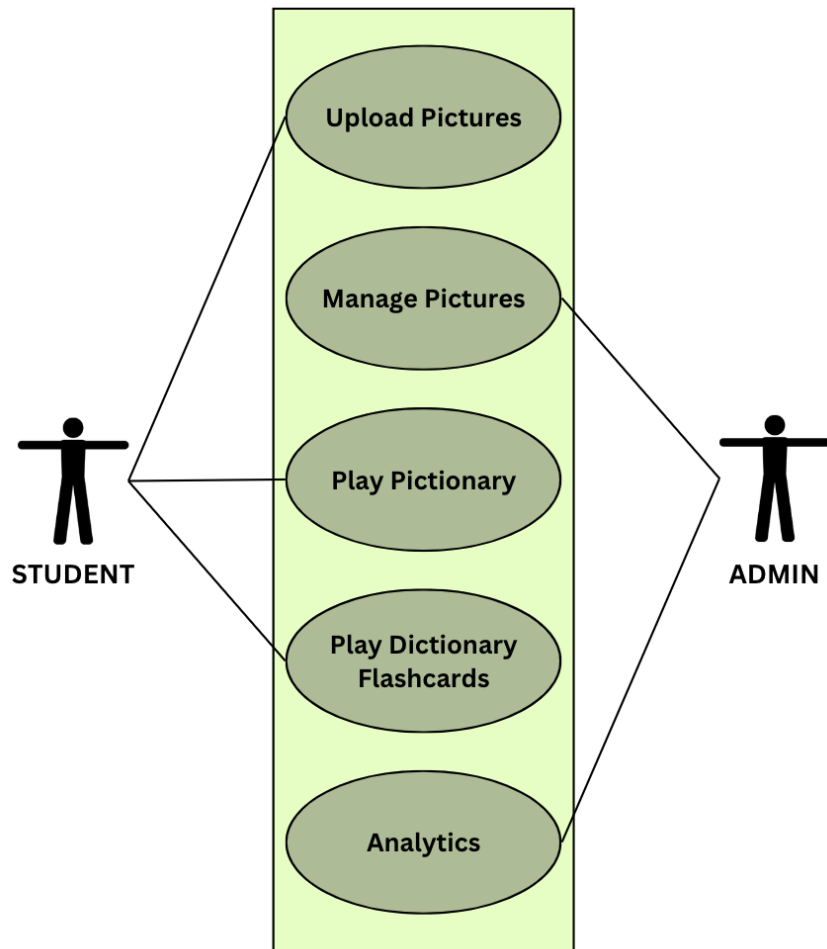
Table 3.3 Play Pictionary Use Case

Goal	Play game with seamless integration of student uploaded photos
Actor(s)	Student
Precondition(s)	Actor has made an account and is able to log into the phone application, has studied words in either school or dictionary game, and wants to test their ability
Postcondition(s)	Actor is shown an image and given a set of letters. Actor must guess the English word represented by the image and use a set of letters to spell it out
Exception(s)	Students have not uploaded photos and/or words so there are none to process

Table 3.4 Play Dictionary Game Use Case

Goal	Study word pairs
Actor(s)	Student
Precondition(s)	Actor has an account and has uploaded words into the database.
Postcondition(s)	Actor is shown Kinyarwanda - English word pairs to study
Exception(s)	Student has not uploaded words into the database

Figure 3.1 Use Case Diagram



Chapter 4

Activity Diagrams

The two user groups consist of students and teacher administrators. Students will be using the mobile app to learn English and the administrators can make alterations to the Firebase databases using the admin website.

4.1 Student Activity Diagram

As shown in figure 4.1, Students will be able to login, upload words and pictures to their mobile Pictionary, and view their profile information. They will have two study options, a game to match letters of a word when presented a picture and a pool of letters, and another option to practice their understanding with flashcards. The Pictionary game will repeat the same vocabulary term until the student guesses correctly or explicitly chooses to skip to the next word. In contrast, the dictionary game enables the student to scroll freely throughout the dictionary list to spend more time focusing on less familiar words.

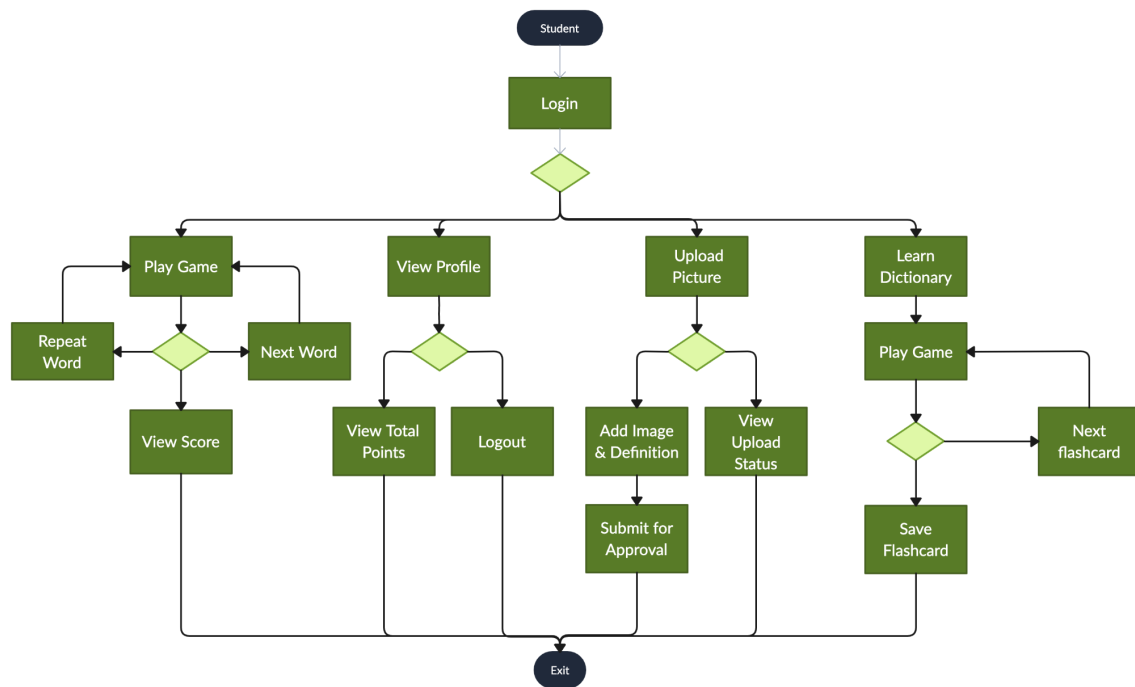


Figure 4.1 Student User Activity Diagram

4.2 Administrator Activity Diagram

As shown in figure 4.2 Administrators will launch a web browser and login with their verified credentials. The dashboard presents the admin with the total user count for the app and a preview of pending requests from students. In the Pending approvals section, Administrators are responsible for determining the appropriateness and accuracy of a submission. If the dictionary or student databases need to be modified after creation, edits can be done in their respective sections. Administrators will have permission to make changes to both databases at any time.

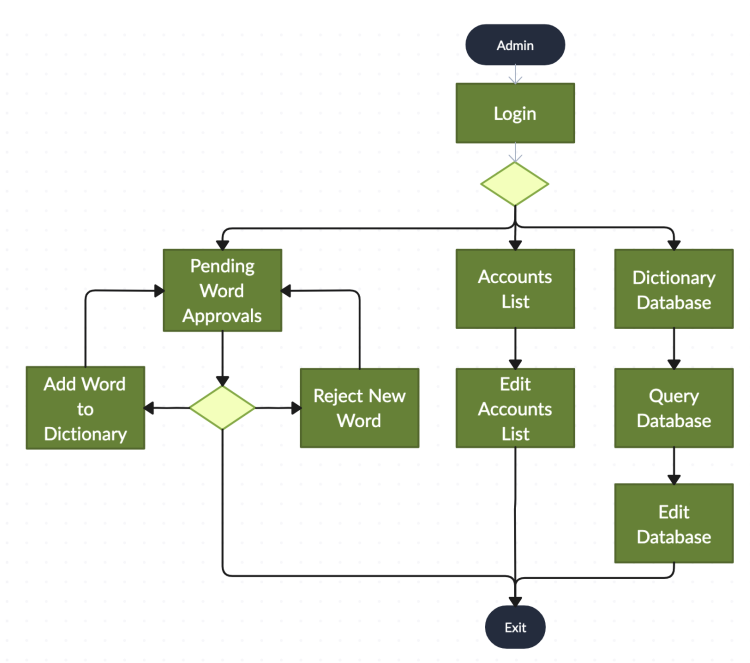


Figure 4.2 Administrative Users Activity Diagram

Chapter 5

Final Design

Users interact with TwigeEnglish in two ways: students upload and review English words and their accompanying pictures through a mobile application, while administrators accept or deny submissions through an administrative web portal. The figures in Section 5.1 represent the final user interface of the mobile application, and in Section 5.2 they represent the final user interface for the administration website.

5.1 Mobile Application

Upon launching the application, a user is prompted to sign in to their account using their full name and a phone number, which serves as their password. The account must already be created by an administrator.

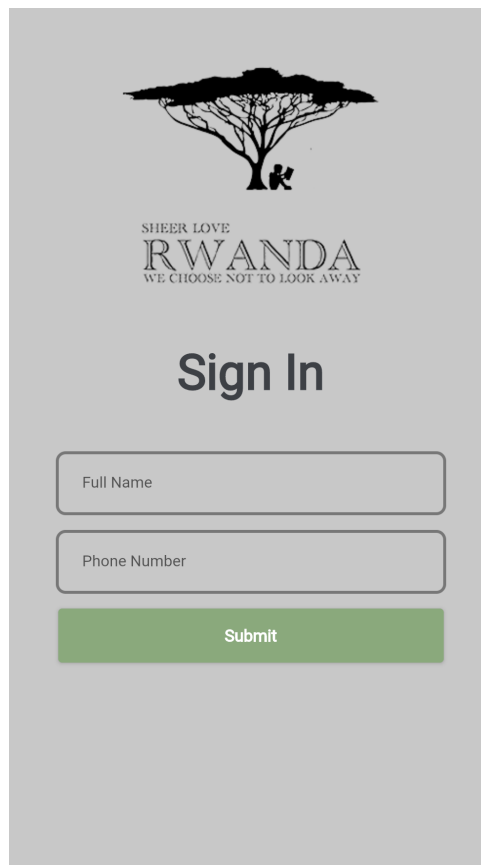


Figure 5.1 Mobile Application Student Account Sign In

Students are presented with the option to play the primary pictictionary game or review the comprehensive dictionary. The bottom navigation bar can redirect students to the upload, profile or home screen based on need.

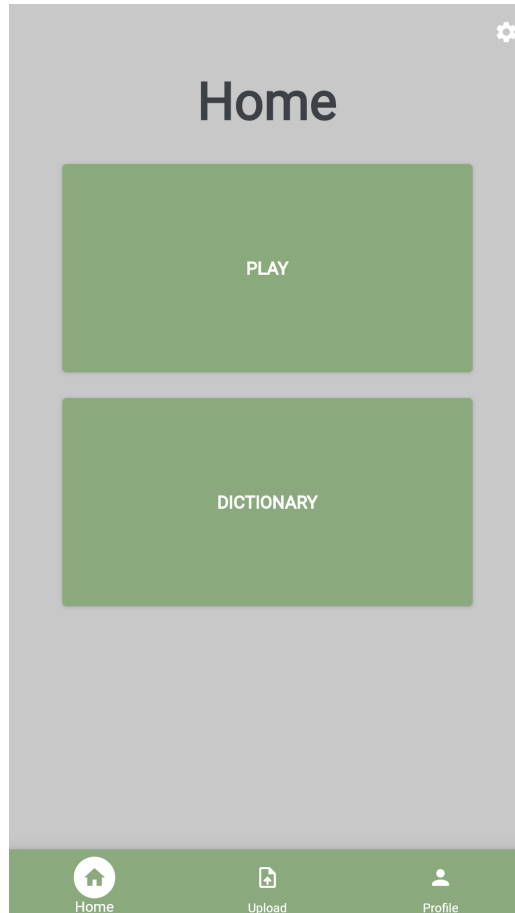


Figure 5.2 Mobile Application Home Screen

Students may select an image from the photo library on their mobile device to upload for review. They must enter the correct Kinyarwandan and English spelling of the word represented in the selected image.

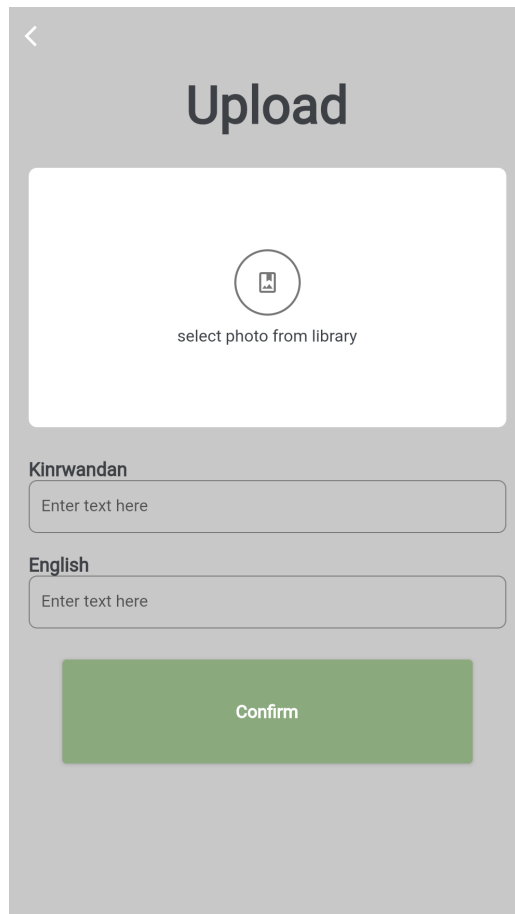


Figure 5.3 Mobile Application Upload Screen

Students must tap on the letters in the correct order to spell out the word that is best represented in the image above (e.g., "C-A-T"). If a student enters the word incorrectly, the letters reset and they may either guess again or go to a new word.

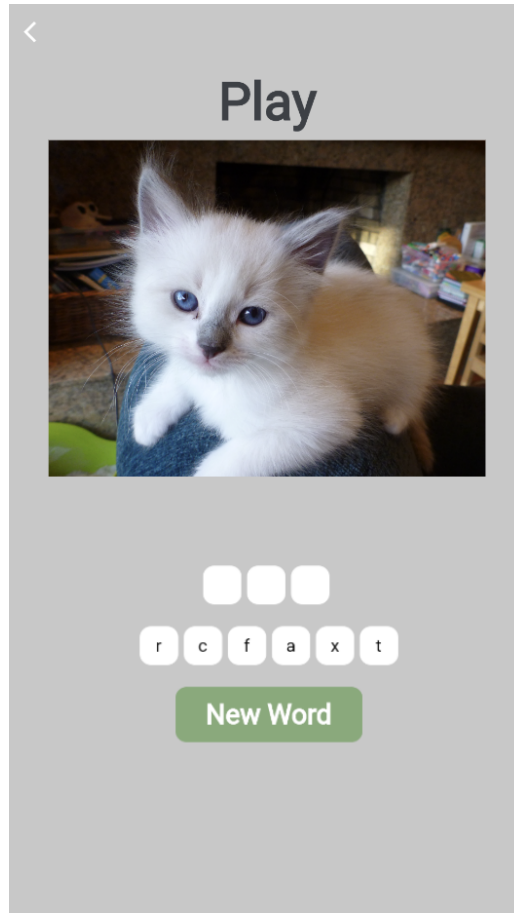


Figure 5.4 Mobile Application Play Pictionary Screen

Students can review the vocabulary terms through the digital flashcards. They are first presented with the Kinyarwandan word (left image) and the English word is revealed on the back when tapped (right image). Other dictionary words are accessed using the arrows underneath the cards.

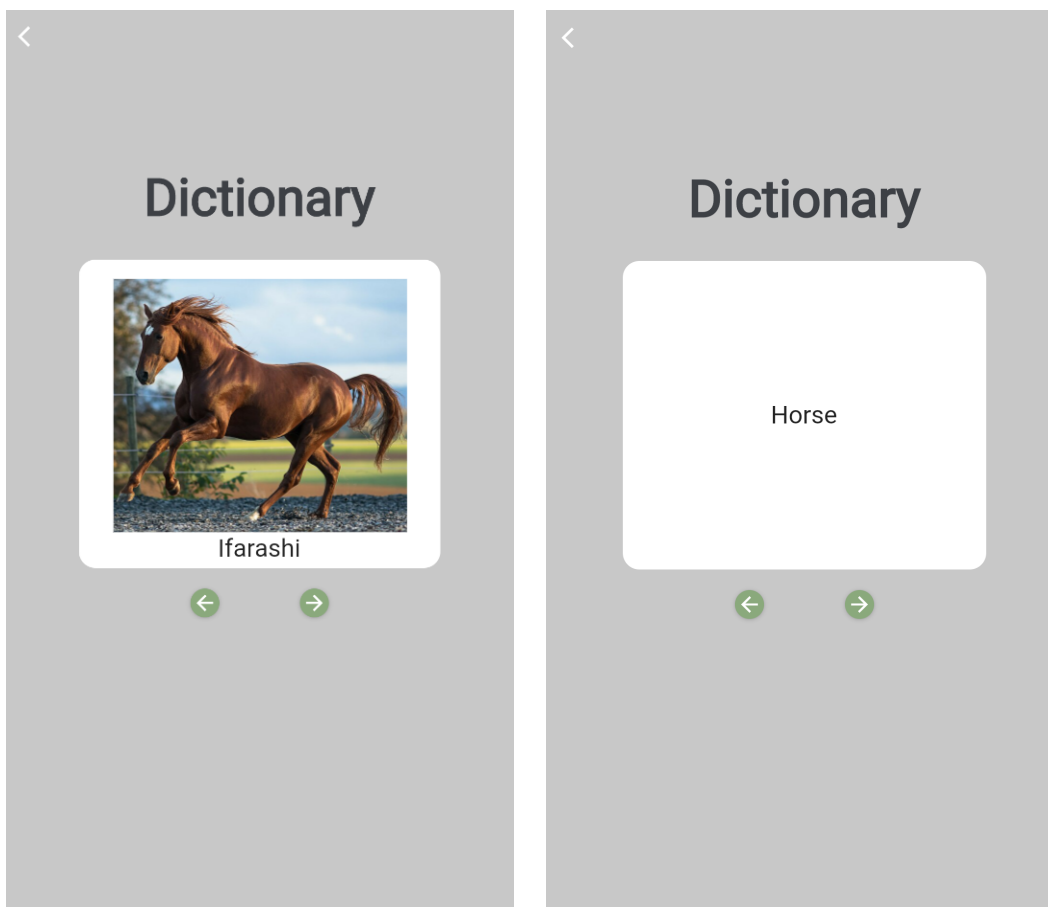


Figure 5.5 Mobile Application Play Pictionary Screen

Students may view their profile information including name, phone number, score, and a brief personal introduction. The introduction can be modified by tapping on the pencil icon. Users will logout of their accounts using the “Logout” button on this screen.

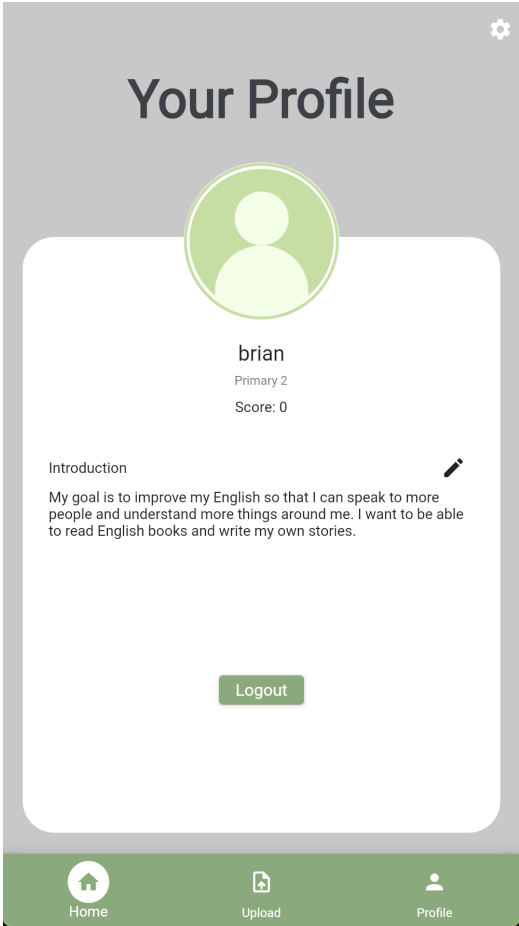


Figure 5.6 Mobile Application Play Pictionary Screen

5.2 Web Application

The login page authenticates a registered user to have access to the administrative web application and all of its functionality.

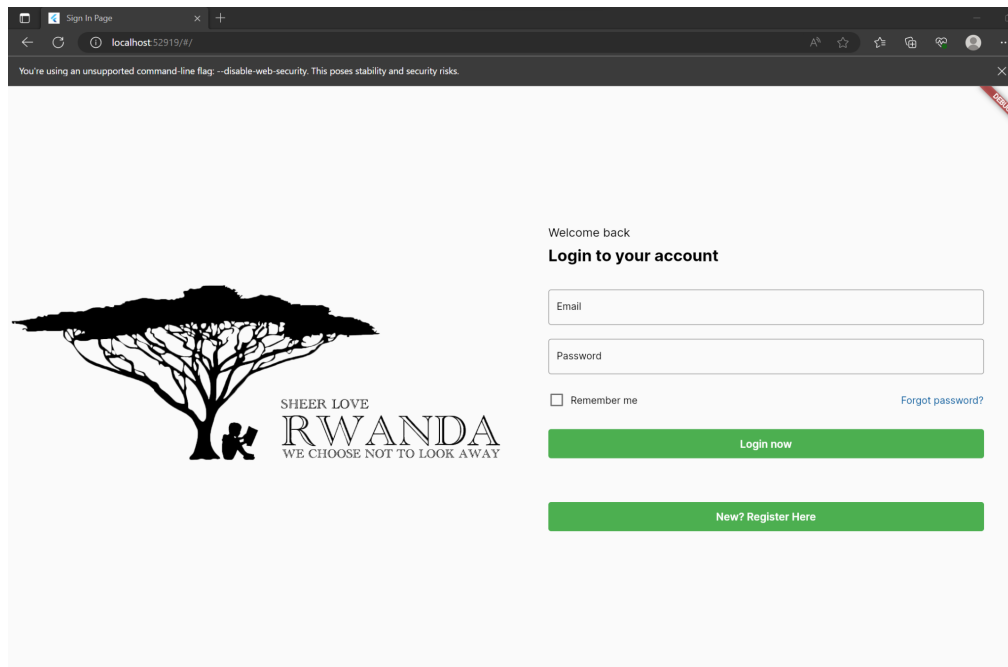


Figure 5.7 Administrative Website: Login

The registration page serves to create a new administrator to have the powers to add or delete students and approve uploaded photos.

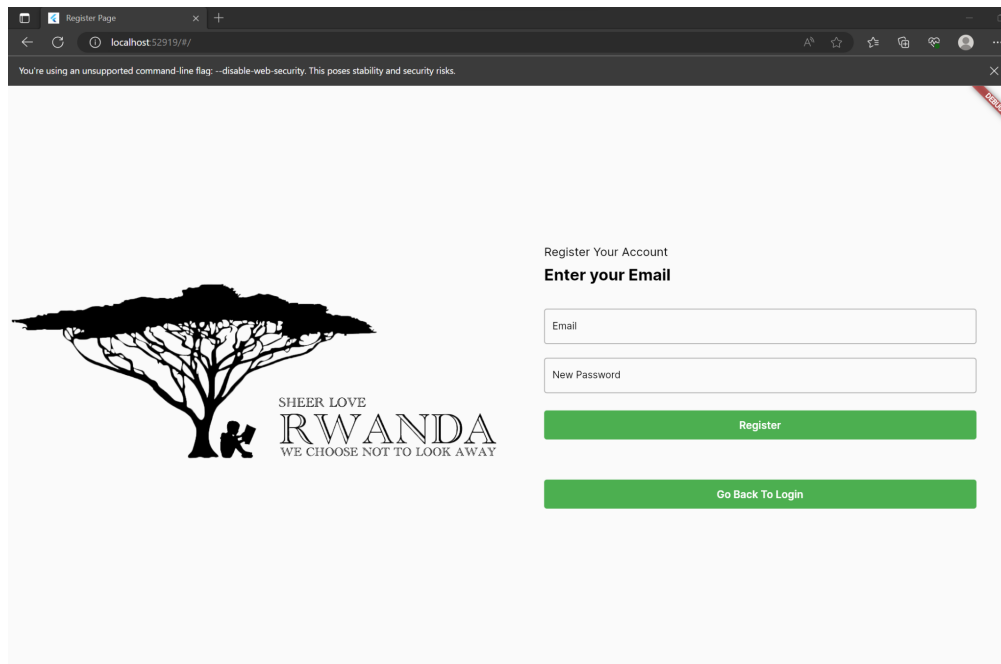


Figure 5.8 Administrative Website: Register

The home page is the main page of the website. It provides accurate information for the active users in the classroom and pending uploading. A list view on the left side makes it easy to go to the different pages involving user management, pending uploads, and already approved photos.

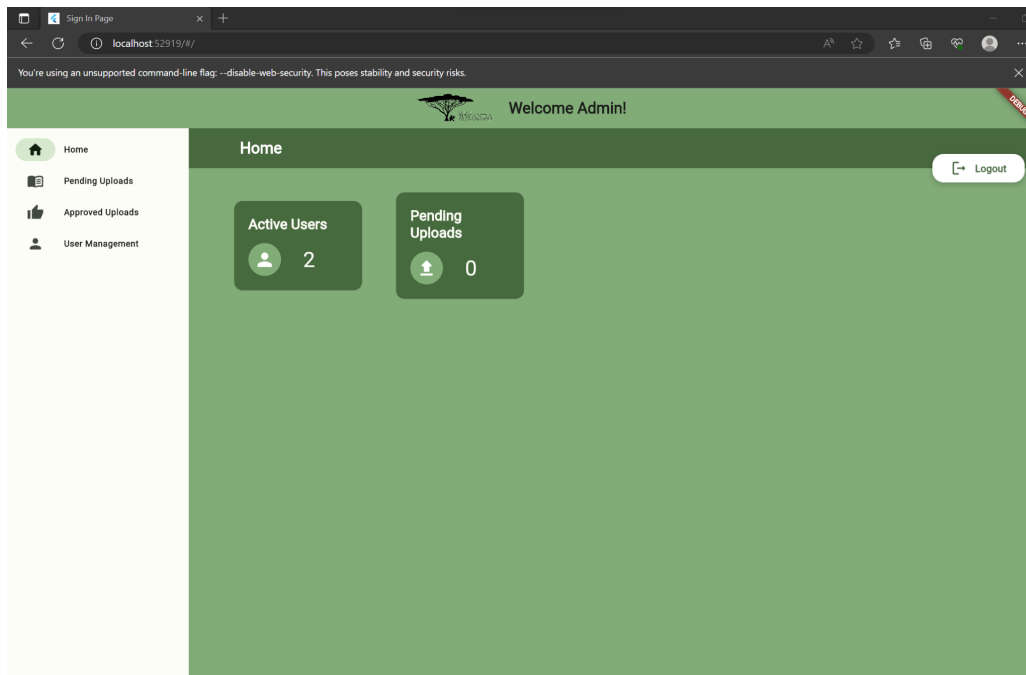


Figure 5.9 Administrative Website: Home

The approving uploads page is a block list view of the photos students have uploaded. Every photo also has an accompanying correct English and Kinyarwanda word.

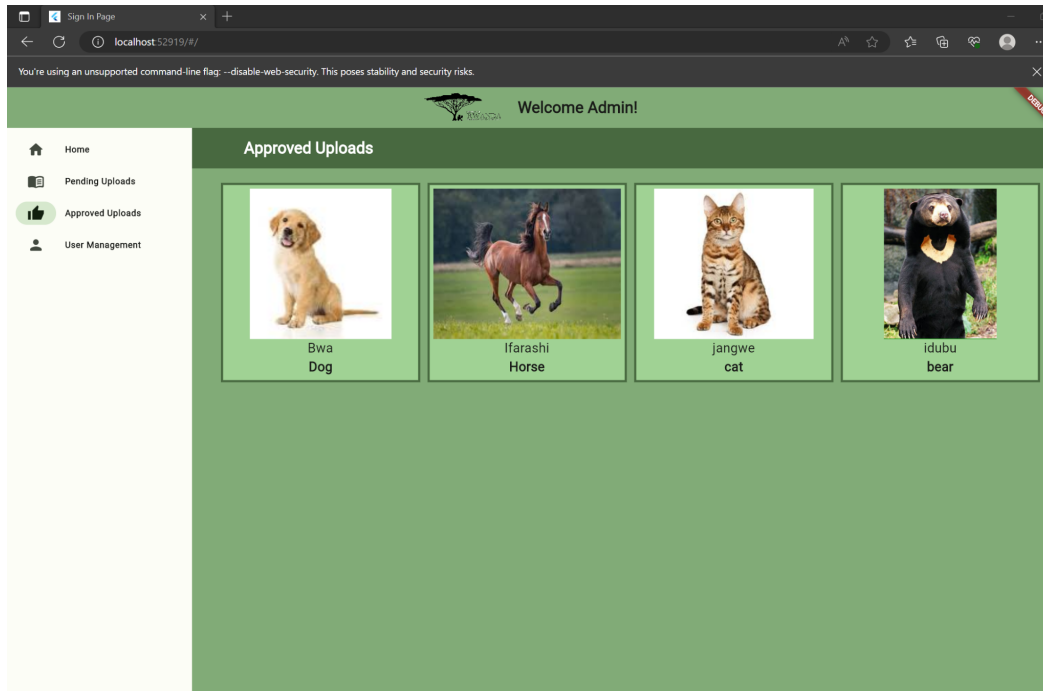


Figure 5.10 Administrative Website: Approved Photos

The pending uploads page displays the guesses students make for different pictures and the correlated English word. Currently, there are no requests so it appears blank.

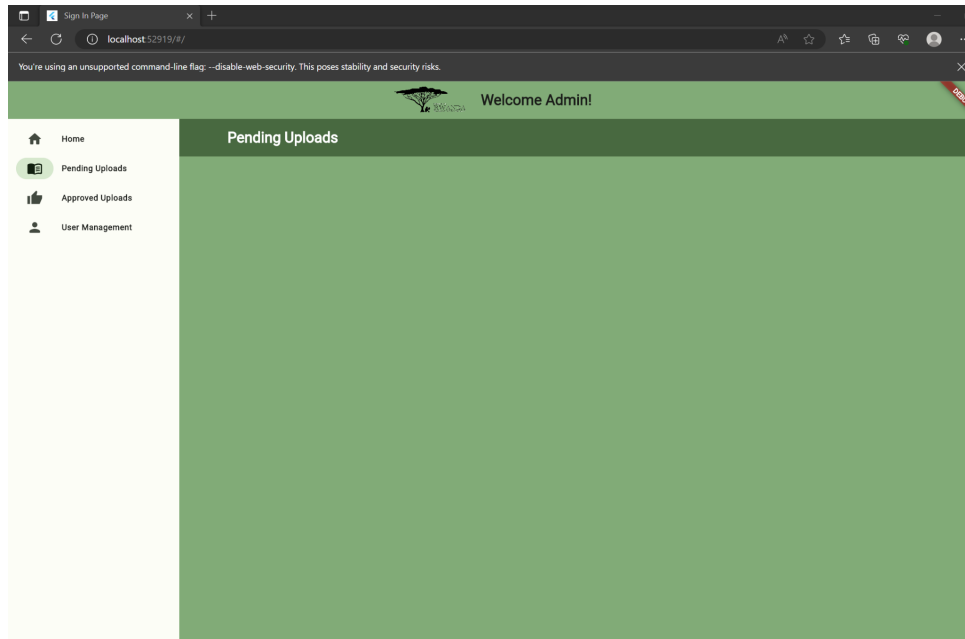


Figure 5.11 Administrative Website: Pending Uploads

The user management page lists out all the students who have accounts and access to the mobile application.

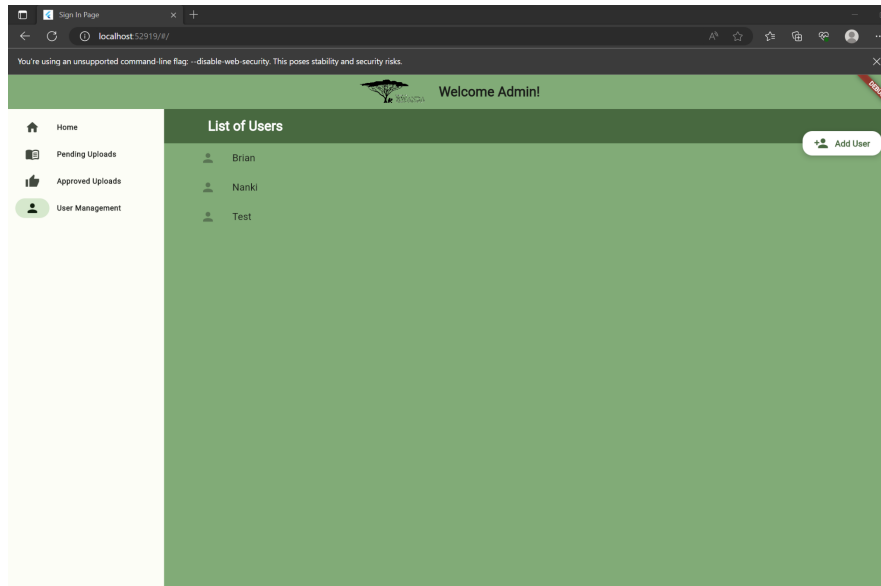


Figure 5.12 Administrative Website: List of Users

By clicking the 'Add User' button on the upper-righthand corner, the administrator can enter a student name and phone number that will then have access to the mobile application.

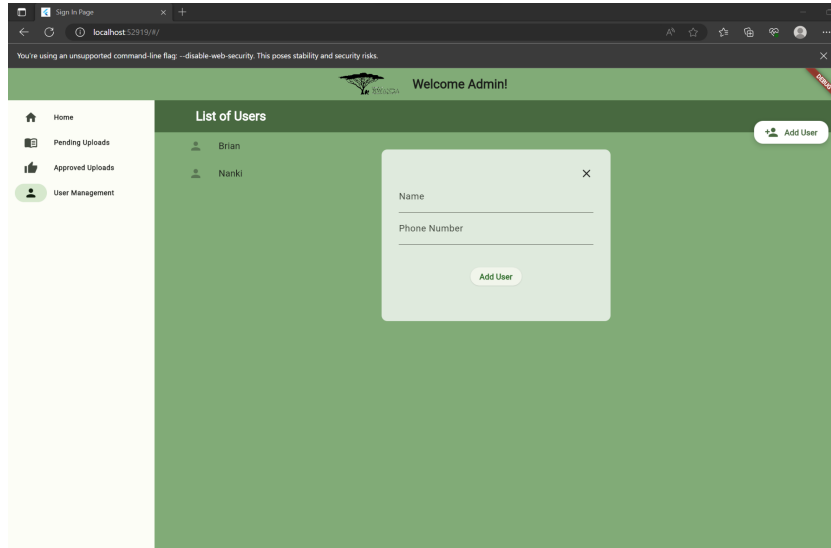


Figure 5.13 Administrative Website: Add a User

Clicking on a student's name under the list of users will open an overlay that displays their username, phone number, and any accumulated points they may have.

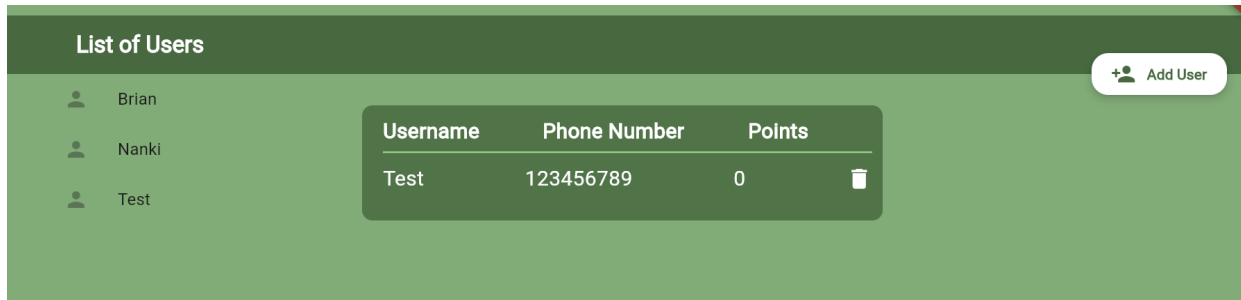


Figure 5.14 Administrative Website: View a User

Chapter 6

Technologies Used

6.1 Mobile Application

The mobile application is an interface for students available for both Android and iOS devices. It is the result of cross-platform development utilizing the Flutter framework.

6.2 Administrative Webpage

The webpage was constructed with the help of several useful technologies:

- Framework: Flutter
- Database: Firebase
- Other:
 - Dart

Chapter 7

Architectural Diagram

The application shown in figure 8.1 is based on a data-centric architecture. A central server/database will house the data, and the web-app and mobile application will be able to connect as clients and receive updates and push new data. *Note that both code for the front end and back end are stored on the same code-base.*

On the mobile application, students must sign in and match pre-populated account information on Firestore to gain access to the components. On uploading a photo from a personal library (C1), photos are pushed to Firebase's Cloud Storage solution where they are tagged and stored as *'unapproved_photos'*. Playing the Dictionary or Pictionary Games (C2 & C3) pulls photos tagged and stored as *'approved_photos'*, not before checking if those photos have already been downloaded which then could be pulled from the application's cache.

Data from the 3 utilized Firebase services: Cloud Storage, Authentication, and Firestore is aggregated onto the Webapp. Verified administrators on the web-app can directly manipulate information on firebase, including but not limited to, verifying and rejecting photos, adding student accounts, and deleting student accounts.

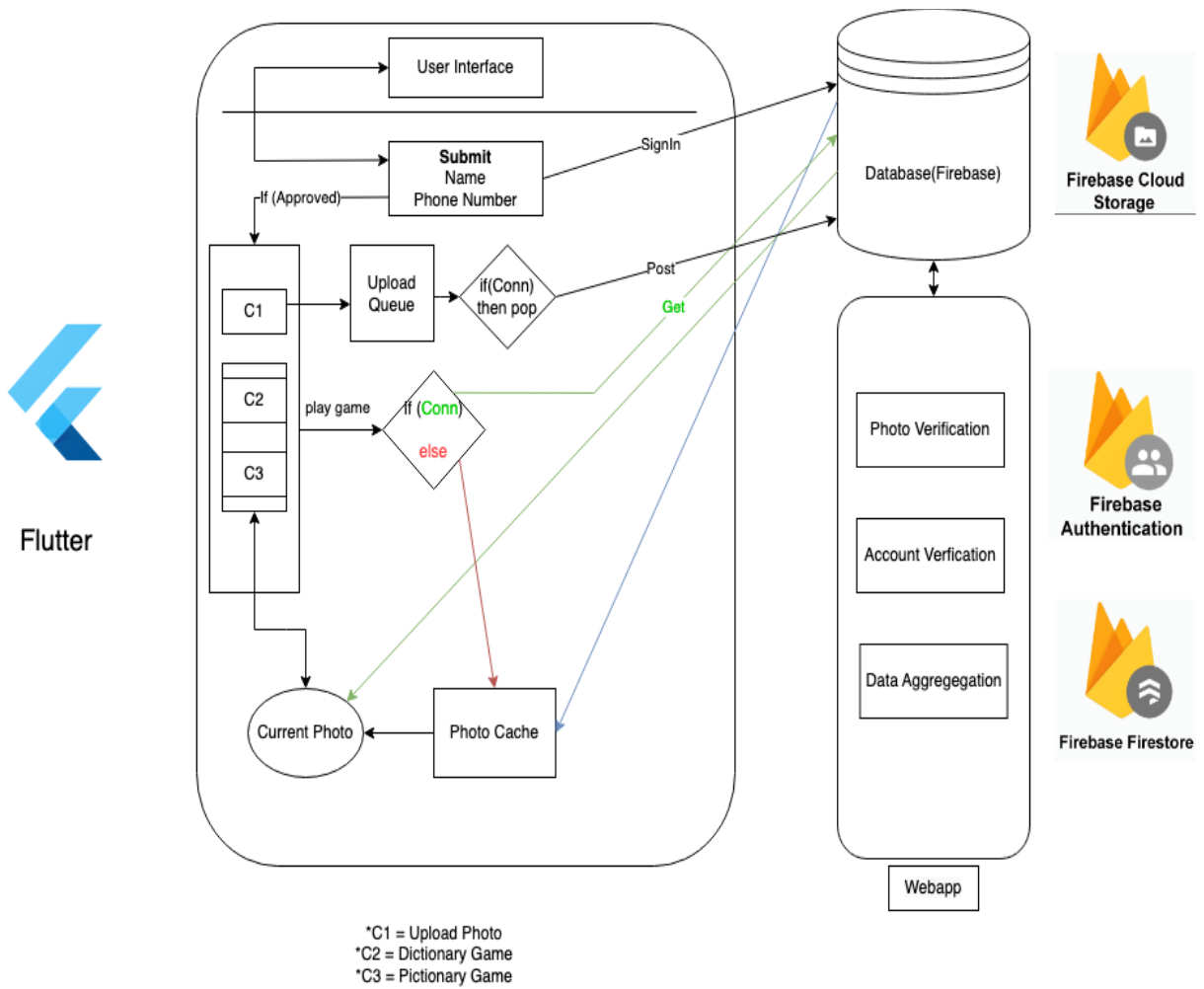


Figure 7.1 Architectural Diagram

Chapter 8

Design Rationale

8.1 Technology Used

The Flutter framework was primarily used for developing the front-end layout of our mobile application and the administrative webpage. There are many advantages that come with Flutter including its single cross-platform development for iOS and Android applications. Interfaces designed with Flutter take advantage of the native components on a device and use basic element trees to structure the code. Flutter UI elements are known as widgets and can be reused infinitely. Flutter is a trusted software development kit built by Google and has three forms of testing: unit, widget, and integration. All three forms were thoroughly used throughout the coding process to ensure the app would be fully functional for launch.

All of the backend is facilitated through Firebase. Firebase is scalable and accessible for mobile app development. Additionally, it comes with its own built-in authentication, databases, and user models which can easily be integrated into our application. As TwigeEnglish continues to develop, a strong backend with Firebase grants flexibility of storing and accessing larger and more complex data.

8.2 Architecture

The design is based on a hybridized data-centric and layered architecture. Our requirements specify the need to store and manipulate data related to each student phone number. Using a data-centric architecture should help ensure that all of this data is stored and easily accessible in a central place. Meanwhile, our mobile application was layered so it can handle caching of the data when the internet is unreliable. This includes a presentation, middleware, services, repository, and final database layer.

Chapter 9

Testing

To ensure that our system works properly and conforms to our requirements, we applied standard industry testing practices to our project. Flutter has an automated system built-in that works to perform unit, widget, and integration tests which we used throughout the development stage to debug for edge cases. Over this summer, TwigeEnglish will be deployed to be used in the Share Love Rwanda classroom by SCU graduate students for our final user testing as well as a beta release for our project.

9.1 Unit, Widget, and Integration Tests

Flutter's testing package helped simplify testing during the development process. As functions and additional features were built out, automatic cases were written and run in Dart. The unit testing consisted of checks made to ensure that all functions worked as expected and errors were caught depending on the different data inputs. Widget testing was a step up from unit testing and expanded its verification with bigger widgets. Widgets in Flutter is a graphical user interface that shows information or provides interaction with information to the user; widget can either be stateless which means they don't change (i.e., Icon) or stateful which means it is dynamic (i.e., Slider). Lastly, integration tests target a screen's functionality and communication with other pages in the project. For example, how can a user navigate from the Home page to the Upload screen and what checks have to be in place. This process ensured project requirements had properly been implemented and met end-user approval in the final product.

9.3 User Testing

Our final stage of testing will be a beta release to a subset of the users in the Sheer Love Rwanda Program. These are 39 students from varying ages enrolled in the Sheer Love Rwanda empowerment program. The purpose of doing so will be to observe how students may interact with TwigeEnglish, where their intuitiveness may lead them, and additional functionalities to add in the future.

Chapter 10

Risk Analysis

This table shows an overview of probable issues that may have arisen during development, along with details about their consequences, likelihood of occurrence, severity, and potential solutions. These probabilities are general estimates of how likely we think these issues were to arise rather than being computed in a particular way. The probability and severity numbers are multiplied to create the impact values, which produce a normalized impact rating on a scale from 1 to 10.

Table 10.1 Risk Analysis

Risk	Consequences	Probability	Severity	Impact	Mitigation
Insufficient Time	The desired application is missing some features.	0.5	9	4.5	Maintain a strict timeline.
Missing Requirements	The final application does not satisfy the customer needs and leads to a redesign.	0.3	8	2.4	Establish clear and consistent communication with the client.
Unfamiliar Technologies	Team members must learn new technologies prior to implementation which may affect the final product quality.	0.9	6	5.4	Actively learn and research technologies.
Bugs	The application does not perform as expected. Delays the launch.	0.9	4	3.6	Test features as they are developed with clear expectations.

Chapter 11

Societal Issues

11.1 Ethical

There are a few ethical dilemmas we faced when building TwigeEnglish. There are concerns of accuracy and fairness of the admin powers towards uploaded photos from the students. Admin has the power to approve or disapprove any uploaded image. An approved image should be a clear image with a correct English word. But, there is no system for monitoring if both factors are true. There is also no safety net giving the admin the correct English word in situations where they do not know. These situations may lead to unfair evaluation of accepted photos.

11.2 Social

A potential concern of our project was that by building an application for Share Love Rwanda, we are taking an opportunity from entrepreneurs in Rwanda to build their own solution. In the beginning stages of planning and designing our product, however, we identified inefficiencies and stagnations that could prevent local people from building a prior solution. Learning English in Rwanda is being pushed to better prepare their people for more opportunities. Thus, TwigeEnglish's positive impacts outweigh the possibility of taking an opportunity away from young people in Rwanda.

11.3 Political

As a mobile and web application, our product has no direct political concerns.

11.4 Economic

The first long term costs of maintaining our solution is extended code-base maintenance which will be provided by SCU Graduate Students. The other cost will be upkeep on our cloud services/databases which we have verified will be more than affordable for our clients.

11.5 Health and Safety

As a mobile and web application, our product has no direct health concerns.

11.6 Manufacturability

Our software has already been manufactured and is ready for a beta level deployment.

11.7 Usability and Security

Since our project will be used by real world clients, usability has been a big goal of ours. The student can sign in by phone number for ease of access. This was setup to help eliminate barriers of using TwigeEnglish. But in doing so, a number of different security concerns rise. Different classmates have the ability to “hack” into other classmate profiles and upload obscene images from there. Although, we implemented a system to text the user when they login and have them be prompted to enter a security code. There are ways to get around this by taking a classmate's phone and entering the information from there. In our future product, we hope to make changes to this current system.

11.8 Compassion

TwigeEnglish was built with compassion for students in the Share Love Rwanda program to help them learn English. This has the potential to open more doors for them in their educational and work-related future as it is practiced worldwide. The CEO, Alexis Amani Simbayobewe, founded Share Love Rwanda with the mission to “give back and help other children like himself realize their dreams” and have the support to follow through with them. We hope that TwigeEnglish better makes that idea into a reality.

Chapter 12

Conclusion

12.1 Risk Resolution

When assessing our potential risks in the early stages of this project we identified four main concerns with varying likelihoods and projected impacts on our work. The first was insufficient time which we were able to overcome with regular meetings and check-ins to ensure that progress was being made at all times. When we encountered problems that halted our work we were able to offer one another's perspectives and support.

The next concern we noted was the chance of missing required features in our app. We did not predict that it would be a very likely event and we did a good job of focusing on the main functionalities that our partners at Sheer Love Rwanda requested. We were clear with them about the limitations of our efforts and were able to deliver an app that featured all of the core components we discussed with them.

The third risk we absolved was learning how to use unfamiliar technologies. We knew that this was unavoidable given that none of our team members had ever used Flutter or Dart in a project before. This was a high priority issue in the beginning of our work but the learning curve was steep and we were able to offer help and support to each other during our weekly meetings. Our advisor also provided us with invaluable guidance by connecting us with third party Flutter developers who were able to teach us more about the new languages and frameworks we were introduced to.

Our last analysis was the likelihood of bugs which was projected very high. We knew that our initial lack of knowledge and experience with Flutter and Dart would lead to more bugs and performance issues down the line but we agreed to be thorough from the beginning and we were able to resolve bugs during development which greatly decreased the amount of work that we had to do at the very end of our experience.

12.2 Obstacles Encountered

This project was the first time any of our team members had worked with Dart or Flutter so we had to familiarize ourselves with the language and framework quickly. The overall structure of these new tools took some time to understand but once we were comfortable progress took off. The front end developers struggled with the initial introduction to some of the built-in features like layouts, state management, and response functions.

Our back end developers initially chose to work with AWS Amplify at the beginning of our project development. This was difficult for them because AWS Amplify lacks robust documentation when compared to other Backend-As-a-Service platforms. We made the decision to switch to Firebase ultimately to minimize overhead and also to increase compatibility with Dart and Flutter which are both provided by Google as well.

12.3 Future Work

We hope and anticipate that the Frugal Innovation Hub will continue to support and carry on this project even further after we wrap up our contributions. Listed below are some of the ideas and features we had hoped to incorporate into this application before deploying it for user testing:

- Reduce Firebase overhead
- Add points system to track app use/progress
- Audio to hear English pronunciation
- Local download functionality
- Offline app access
- User password recovery
- Admin contact page on mobile app

12.4 Lessons Learned

We learned many valuable lessons during the development of this project. One important takeaway is that the simplest solution is not always the fastest or most efficient. We gained a lot of experience working as a team of developers and learning not only new hard skills but also collaborative and communicative skills. We utilized agile software development techniques and held regular team meetings to maintain a productive and progressive direction throughout the project. These meetings were essential to our success because we were able to touch base with new questions or obstacles as

well as offer and receive help and advice from one another.

Working on a year-long software project has allowed us to sharpen our approach to the design process and stay current with new methods. We familiarized ourselves with Git to control code versions and keep track of changes made by each team member. This process involved the most communication between our group members because it allowed us to respect each other's workflow and development style while still constantly developing.

12.5 Acknowledgements

We would first like to thank the Frugal Innovation Hub. Through them we were put in contact with Sheer Love Rwanda and its founder Alexis Amani Simbayobewe. Alexis and his foundation provided us with the opportunity to work on this project. His feedback in the early stages of our project was invaluable for our growth and direction. Prof. Silvia Figueira, our Advisor, served a crucial part at every step of our project. She gave us guidance and support when we needed it and made sure we were on track to complete our work in a timely manner. Her help can not be understated. Frugal Innovation Hub also connected us with Graduate student Giulliano Silva Zanotti Siviero. He helped us in numerous ways with learning Flutter and Firebase for both our front-end and back-end development

Appendix A

User Manual

Web (Administrators)

Login

1. Open your preferred web browser
2. Navigate to administrator website
3. Enter your approved email address and password in the designated fields
4. Click “Login now” button to redirect to administrator dashboard with access to controls and features

Basic Use

1. Launch administrator website
2. Login using verified email and password
3. View pending word approvals, student account database, and dictionary database
4. Perform CRUD operations on the dictionary and student account databases
5. Use the navigation menu on the left column to explore other subpages
 - a. Subpages:
 - i. Pending Uploads
 - ii. Approved Uploads
 - iii. User Management

Add New Student User

1. Launch administrator website
2. Login using verified email and password
3. Navigate to “User Management” in the menu
4. Click on the add person icon in the top right corner
5. Enter the student’s name and phone number
6. Verify you have the correct information and click “Add User”

Evaluate Pending Uploads

1. Launch administrator website
2. Login using verified email and password
3. Pending word requests will appear on dashboard or navigate to pending uploads page
4. Click on the check mark to approve or the “X” to reject the word
5. If approved
 - a. Photo and word are added to approved uploads in dictionary database
6. If rejected
 - a. No edits are made to dictionary and student must resubmit

Modify Vocabulary Database

1. Launch administrator website
2. Login using verified email and password
3. Navigate to “Approved Uploads” in the menu
4. Select photo or word to edit
5. Save the necessary changes or delete word

Modify Student Database.

1. Launch administrator website
2. Login using verified email and password
3. Navigate to “User Management” in the menu
4. Select an existing user
5. Click on the trash icon to delete selected user
6. Approve changes by hitting save

Mobile

Installation

iOS

1. Download TwigeEnglish from the Apple App Store

Android

1. Download TwigeEnglish from the Google Play Store

Use

1. Student Login
 - a. Launch the application on a mobile device
 - b. Enter your approved phone number as a login credential
 - i. Teacher is responsible for administrative duties such as creating new accounts with a phone number
 - c. Complete sign-in using the “Submit” button
 - d. Failed logins:
 - i. If you forgot your credentials, contact your teacher (administrator)
 - ii. If your internet connection failed, reconnect to wifi and please try again
2. Play Pictionary game
 - a. Launch mobile application
 - b. Select “Pictionary” button to start game
 - c. Use the randomized letters to guess the English word for the object in the photo presented
 - i. If correct, new word is displayed with a new set of letters to guess from
 - ii. If incorrect,
 1. letters will reset and user tries again
 2. Or skip word
3. Play Dictionary game
 - a. Launch mobile application
 - b. Select “Dictionary” button to start game

- c. Guess the English word for a photo and Kinrwandan word presented on the front of a flashcard.
 - d. Tap to flip over the flashcard and view word in English
 - e. Use the arrow buttons for moving to the next and previous vocabulary terms
 - f. Scroll vertically to view a dynamic side-by-side list of all vocabulary terms in the database
4. View Profile
- a. Select profile icon on bottom right of navbar.
 - b. Guess the English word for a photo and Kinrwandan word presented on the front of a flashcard.
 - c. Tap to flip over the flashcard and view word in English
 - d. Use the arrow buttons for moving to the next and previous vocabulary terms
 - e. Scroll vertically to view a dynamic list of all vocabulary words side-by-side

Appendix B

Development Manual

The entire code base for our project is hosted on GitHub and was passed off to the Frugal Innovation Hub at Santa Clara University for future development in coordination Sheer Love Rwanda.

Web

Dependencies

Dependencies are located in pubspec.yaml

you can run *flutter pub get* to install all dependencies from the root folder `.../TwigeEnglishAdmin`

Code Structure

The codebase is stored in `.../TwigeEnglishAdmin/lib`

- models
 - firebase_file.dart
 - Defines a class called `FirebaseFile` with properties for referencing, naming, URLs, and language translations of a file stored in Firebase Storage, along with the user who uploaded it.
 - -user.dart
 - Defines a class called `AdminUser` with properties for the user's UID, email, and approval status, while the second code snippet defines a class called `StudentUser` with properties for the user's UID, name, phone number, and points.
- screens
 - authenticate
 - authenticate.dart
 - Implements a `StatefulWidget` called `Authenticate` that dynamically switches between a sign-in screen and a registration screen based on

the value of a boolean variable.

- register.dart
 - Defines two classes, Reg and Register, where Reg is a StatefulWidget that displays a Register widget containing a registration form with email and password fields, and enables user registration using Firebase Authentication.
- sign_in.dart
 - Includes two classes, SignIn and LoginDesktop, where SignIn is a StatefulWidget that builds a Scaffold containing a LoginDesktop widget with a login form that includes email and password fields, a "Remember me" checkbox, and options for signing in or registering a new account.
- home
 - approved.dart
 - Defines a StatefulWidget called ApprovedPage that displays a grid view of approved uploads fetched from Firebase storage, including metadata for each upload, such as English and Kinyarwanda descriptions, and the source URL of the file.
 - home.dart
 - Defines a Flutter app with multiple screens for an admin dashboard, including features such as user management, pending uploads, and approved uploads, along with a main home screen displaying various statistics and actions.
 - uploads.dart
 - Defines a Flutter widget called UploadsPage, which displays a list of pending uploads fetched from Firebase Storage. It allows the user to approve or reject each upload, showing a popup overlay with details and options when an upload is tapped.
 - users.dart
 - Defines a Flutter widget called UserManagement that displays a list of users fetched from Firestore. It allows adding and deleting users, as well as showing user details in an overlay when a user is tapped. The user information includes the username, phone number, and points.
- wrapper.dart

- `userTile.dart`
 - Defines two classes, `UserTile` and `UserListTile`, which represent user tiles and provide functionality for displaying user information and deleting users.

Installation and Configuration

This installation assumes you have already done the following:

- Cloned the project to your development machine
- Installed the Firebase CLI
- Install the Flutter SDK

Please note that instructions for setting up the tools and project may be different if you are running on Windows.

#Navigate into Web Folder

```
cd TwigeEnglishAdmin/lib
```

Pull the latest code from the github

```
git pull
```

Go back to root folder

```
cd ..
```

Run

Use Flutter Doctor to check if all correct dependencies and libraries are installed correctly

```
flutter doctor
```

You are now ready to run the flutter project through firebase

```
flutter run
```

Mobile

Libraries and Dependencies

Dependencies are located in `pubspec.yaml`

you can run `flutter pub get` to install all dependencies from the root folder `.../TwigeEnglish`

Code Structure

Models

- Model for database objects
- Includes fields for reference, name, image url, English word, Kinyarwandan definition

Router

- Contains the navigation for the application as defined in `routes.dart`

Screens

- All of the code for the user interface for each different frame
- Widgets
 - Related and reused widgets/functionalities for each respective screen

Services

- Database management and models

Widgets

- Files for widgets that are heavily reused throughout development
- Navigation bar, menu, custom buttons etc.

Install and Run

Pull up the most recent version of the code.

```
git pull
```

Download all the dependencies.

```
flutter pub get
```

Run the app.

```
flutter run
```











TwigeEnglish Report

Final Audit Report

2023-06-26

Created:	2023-06-26
By:	Valerie Woitte (vwoitte@scu.edu)
Status:	Signed
Transaction ID:	CBJCHBCAABAA3L3aVy9GUzldH4CfJC7qkN66fglC5ihY

"TwigeEnglish Report" History

-  Document created by Valerie Woitte (vwoitte@scu.edu)
2023-06-26 - 5:50:08 PM GMT
-  Document emailed to sfigueira@scu.edu for signature
2023-06-26 - 5:50:40 PM GMT
-  Email viewed by sfigueira@scu.edu
2023-06-26 - 5:50:54 PM GMT
-  Signer sfigueira@scu.edu entered name at signing as Silvia Figueira
2023-06-26 - 5:52:35 PM GMT
-  Document e-signed by Silvia Figueira (sfigueira@scu.edu)
Signature Date: 2023-06-26 - 5:52:37 PM GMT - Time Source: server
-  Document emailed to nling@scu.edu for signature
2023-06-26 - 5:52:38 PM GMT
-  Email viewed by nling@scu.edu
2023-06-26 - 6:19:10 PM GMT
-  Signer nling@scu.edu entered name at signing as N. Ling
2023-06-26 - 6:45:41 PM GMT
-  Document e-signed by N. Ling (nling@scu.edu)
Signature Date: 2023-06-26 - 6:45:43 PM GMT - Time Source: server
-  Agreement completed.
2023-06-26 - 6:45:43 PM GMT