A black and white sign

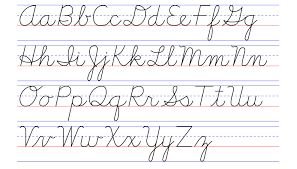
Description automatically generated with low confidence

Capstone Project Phase B

23-1-D-9

Handwriting Application

Making Learning easier using an App



Salman Amer: [Salmana3335@gmail.com](mailto:Salmana3335@gmail.com)

Ghaith Mdah: [Gethe19@gmail.com](mailto:Geth19@gmail.com)

Table of Contents

[1. Introduction 4](#_Toc138622343)

[2. Project Review and Process 4](#_Toc138622344)

[.2.1 HandWriting Theory 4](#_Toc138622345)

[2.2. Traditional Ways to Learn HandWriting 5](#_Toc138622346)

[2.3. Solution 6](#_Toc138622347)

[2.3.1 Base Definitions 6](#_Toc138622348)

[2.3.2 Main algorithm 7](#_Toc138622349)

[2.3.3 Frameworks 8](#_Toc138622350)

[2.4 Engineering Process 9](#_Toc138622351)

[2.4.1 Dataset 9](#_Toc138622352)

[2.4.2 Dataset Creation Process 10](#_Toc138622353)

[2.5 Challenges 13](#_Toc138622354)

[2.6 Testing plan 14](#_Toc138622355)

[2.7 Conclusion 18](#_Toc138622356)

[3. User Documentation 19](#_Toc138622357)

[3.1 User’s guide - Operating instruction 19](#_Toc138622358)

[3.1.1 General description 19](#_Toc138622359)

[3.1.2 Operating instructions 20](#_Toc138622360)

[3.2 Maintenance guide 26](#_Toc138622363)

[3.2.1 How to install our app 26](#_Toc138622364)

[3.2.2 App maintenance guide 26](#_Toc138622365)

[3.2.3 Package Diagram 27](#_Toc138622366)

[3.2.4 Deployment diagram 27](#_Toc138622367)

[3.2.5 Database description 28](#_Toc138622368)

[4. References 31](#_Toc138622369)

**Abstract**

This project book details the development and implementation of a mobile educational application that utilizes deep learning algorithms to teach users how to handwrite. The application utilizes a supervised learning approach by presenting users with a letter to write, and then scanning the screen to process the image using the SIFT (Scale-Invariant Feature Transform) algorithm to determine if the letter is written correctly. In case of errors, the application uses a convolutional neural network (CNN) to classify the type of error and generate appropriate feedback. Additionally, the application includes a quiz system that adapts to the user's level of proficiency. The goal of this application is to provide an effective and user-friendly way for individuals to improve their handwriting skills.

# Introduction

With the increasing popularity of touchscreen devices, the time spent using them takes up a considerable amount of time from our day. TVs, computers, smartphones, tablets, and video game consoles all have screens. Screen-equipped devices can now be found in restaurants, waiting rooms, and even in our pockets. The amount of time spent looking at a screen is referred to as screen time.

According to numerous studies, children's screen time is 4-6 hours per day [1]. Most children waste this time on games and watching videos, which has no positive effect at best, and this is exactly what we would like to change. As adults, we are forced to use these devices a lot in our daily lives, in this age where screens are more common and severe than ever before. It will be harder for us to prevent children from using them because the child's instinct makes him imitate the parents' behavior [2].

Despite the fact that we are unable to resolve this conundrum, we can use it to our advantage by saving children's time and using it for educational purposes, as well as contributing to the child's brain evolution by assigning him educational tasks and challenges. In our project, we will focus on people who spend the majority of their time on these screens (whether children or adults) and want to learn a new language by beginning to write it.

As we all know, learning by completing challenges and solving puzzles while receiving positive feedback can increase the likelihood of perseverance and improve the learning curve [3]. As a result, our solution was to develop an educational interactive application that can process handwritten words and provide appropriate feedback by analyzing every single part of it and allocating the error. To accomplish this, we have employed some Artificial Intelligence (AI) algorithms to analyze images using computer vision.

# Project Review and Process

## .2.1 HandWriting Theory

Handwriting production is a process in which we control how we move our hands to create letters. We do this by changing the frequency or acceleration of back-and-forth movements (oscillations) in different directions. For example, we can control the shape of letters by changing the oscillations in the horizontal and vertical directions. When we combine these oscillations and move our hand to the right at a constant speed, we create letters that are spaced apart from each other [4].

## Traditional Ways to Learn HandWriting

The goal of teaching handwriting in primary schools is to help each child write clearly, smoothly, without discomfort, and quickly enough for everyday use. To teach handwriting effectively, we have to focus on posture, paper, pencil grip, and placement - the four "Ps" of handwriting [5].

**Posture**: Good posture is the correct position of sitting toward the desk while writing. Good posture is essential for good handwriting. The right size and style of chair and desk can greatly impact performance.

**Paper**: Using paper with double lines can help students properly orient their letters, improving their handwriting.

**Pencil grip**: Proper grip is essential for good handwriting. Proper grip is the way the child should hold the pencil to improve handwriting on the desk. It is one of the foundations of handwriting and should be emphasized in handwriting instruction.

**Placement**: Ensuring that the paper is positioned correctly can help children move their writing hand smoothly across the page, improving their handwriting. Paying attention to paper placement is an important part of handwriting instruction.

Once we have mastered the four "Ps" of handwriting - posture, paper, pencil grip, and placement - we can move on to the next stage of handwriting instruction, which is learning the script or alphabet. The first script that we should learn is a simple and practical alphabet that can be written quickly and easily, allowing for some variation without losing a consistent appearance (see figure 1) [6].

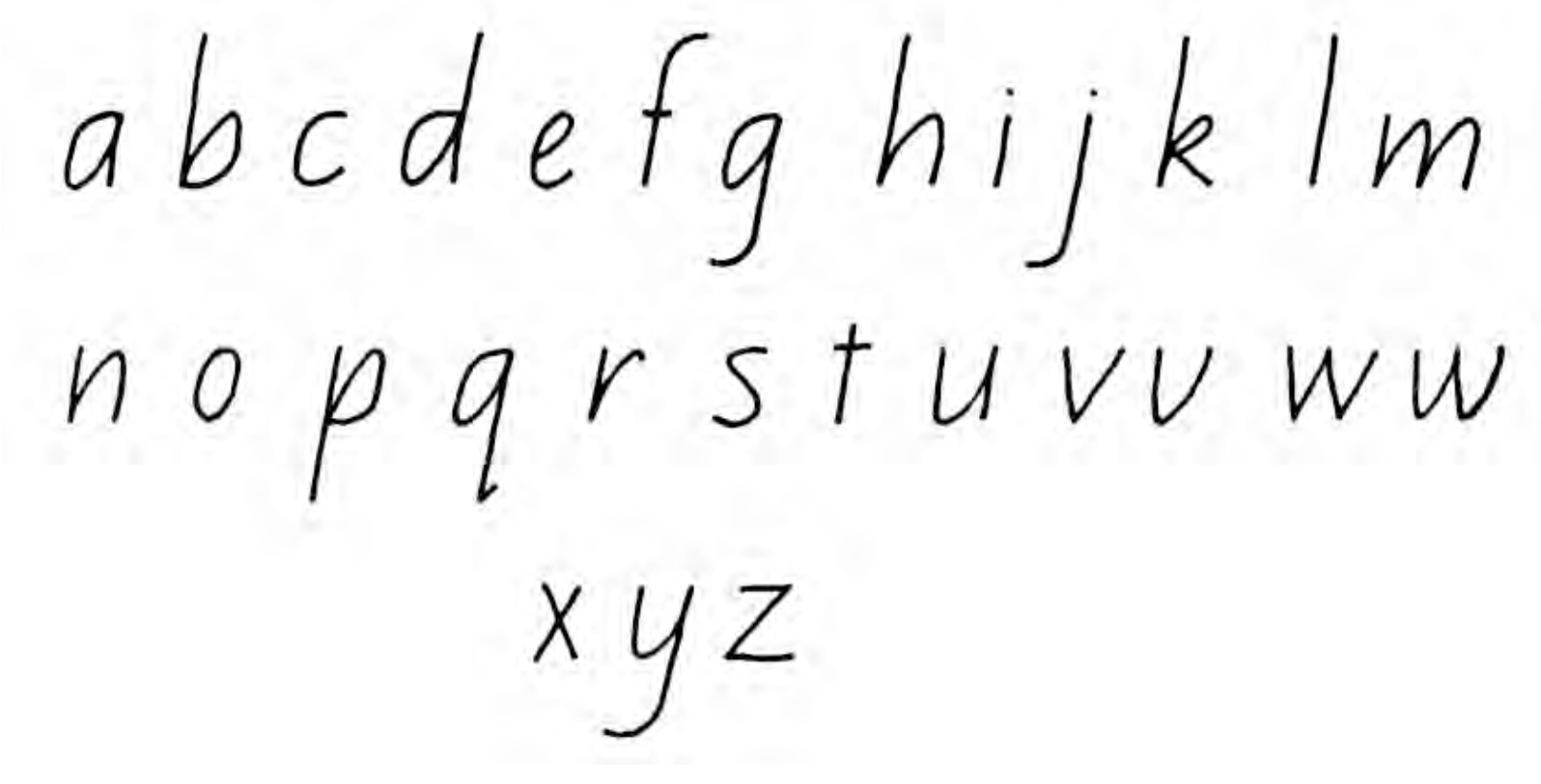


Figure 1: A simple way to the alphabet in English

Because the application cannot regulate the learner's style of sitting, we are unable to integrate all four P's in our application, including posture and pencil grip. However, we may offer some suggestions for the ideal posture. Therefore, to apply the paper, we can provide the learner with a rectangle to write in, making it easier for him to write, and we can assist him in improving his handwriting with positive feedback.

## Solution

Now, we will present our solution to address these issues. Our proposed solution is an application that has the following key features. Firstly, it can categorize errors in handwritten letters accurately, ensuring precise error detection. Secondly, the algorithm works quickly and provides immediate feedback after the user finishes writing a letter. Lastly, the application is compatible with both individual letters and words, making it versatile in its use.

### 2.3.1 Base Definitions

* CNN: A Convolutional Neural Network (CNN) is a specialized type of neural network that is particularly effective in analyzing and processing data with a grid-like structure (see figure 2) [7], such as images or time series. It is designed to automatically learn and extract hierarchical patterns and features from input data through the application of convolutional filters. In our algorithm, we employed two Convolutional Neural Networks (CNNs). The first CNN is responsible for letter classification, while the second CNN is utilized for mistake classification.

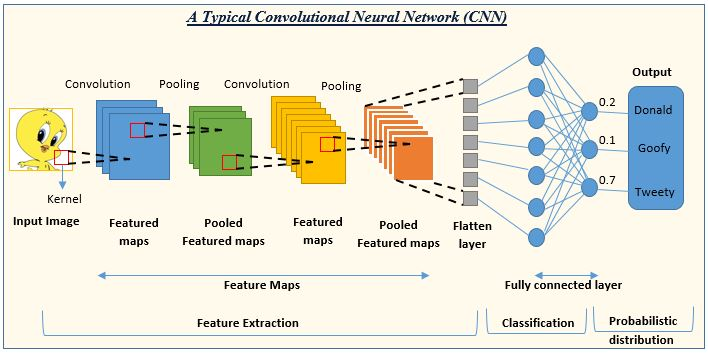


Figure 2: Architecture that describes a CNN for classifying Tweety from Goofy and Donald using convolutional layers: Input -> Convolutional Layers -> Classification Layer.

* OpenCV: OpenCV is a library that offers a collection of programming functions and tools for handling and manipulating visual data. It provides a wide range of functionalities, including image and video input/output, image processing and filtering, feature detection and extraction, object recognition and tracking, camera calibration, and more. These capabilities make OpenCV a versatile and powerful tool for applications in fields such as robotics, augmented reality, surveillance, medical imaging, and more. All image processing operations involved in this Algorithm, such as grayscale conversion, segmentation, union, feature extraction using SIFT, and resizing, are executed utilizing the functionalities provided by the OpenCV Library.

### 2.3.2 Main algorithm

Now that we have established a clear understanding of the Convolutional Neural Network (CNN), we can proceed to describe the architecture of our algorithm. The algorithm utilizes the CNN in multiple stages, thus necessitating an explanation of its workflow.

The algorithm begins with the initial step of image realization and segmentation, where the input image is partitioned into individual letters (in the case of word learning). Additionally, the image undergoes grayscale conversion, and in cases where a letter consists of multiple components (such as dots or other elements in different languages), these parts are unified. Furthermore, the image is resized to match the requirements of the subsequent CNN-based processing. Importantly, throughout this process, the order of each letter within the word is maintained.

Once individual images for each letter in the word are obtained and preprocessed accordingly, they are fed into a basic CNN for letter recognition. This particular CNN operates on 28x28-pixel images and consists of multiple layers, including convolutional layers with activation functions and fully connected layers. By leveraging these layers, the CNN accurately determines the specific letter present in each image.

Subsequently, the algorithm performs a validation check to verify if the written letter matches the intended one. If an incorrect letter is detected, appropriate feedback is provided. Conversely, if the written letter is correct, the algorithm proceeds to the next stage.

In the next stage, the image of the letter is passed through a function that extracts its features. Specifically, the SIFT (Scale-Invariant Feature Transform) function from OpenCV is employed to extract and compare features against the desired features for each letter. This process ensures the evaluation of the letter's quality. If all the required features are successfully extracted, it indicates a well-written letter. However, if any features are missing, the algorithm proceeds to the next step.

In the subsequent section, the algorithm focuses on classifying specific mistakes within the image. Notably, it should be mentioned that a significant portion of the images may not reach this stage due to their satisfactory quality. For those that require further analysis, the image of the inadequately written letter is processed by another CNN. This CNN has been trained on three distinct types of datasets, each containing numerous images with the same type of mistake. The three types of mistakes are slant, incomplete, and incompatible. The architecture of this CNN includes convolutional layers, activation functions, and fully connected layers, enabling accurate classification of the mistake within the image. Following the classification stage, the result from the CNN is passed to a simple feedback function. The feedback generated is then displayed on the screen and stored in a database, allowing the user to track their progress over time.

In summary, our algorithm leverages a combination of CNN-based letter recognition, feature extraction using SIFT, and a specialized CNN for mistake classification. This approach ensures accurate evaluation, feedback, and progress monitoring in handwriting learning applications.

### 2.3.3 Frameworks

### 

In the development of our application, we relied on a set of essential frameworks. These frameworks were instrumental in shaping our application, enabling us to build a robust and user-friendly solution. In the following section, we will introduce the key frameworks that we utilized, outlining their functionalities and highlighting their impact on our application's development process.

* TensorFlow: TensorFlow is an open-source machine learning framework that facilitates the development and deployment of deep learning models. It provides a comprehensive set of tools, libraries, and resources for building and training neural networks, allowing developers to tackle a wide range of machine-learning tasks. At its core, TensorFlow enables the creation of computational graphs, where nodes represent mathematical operations, and edges represent the flow of data between these operations. This graph-based approach offers flexibility and scalability, enabling efficient execution of computations on CPUs, GPUs, or even specialized hardware. With its extensive ecosystem and community support, TensorFlow has become a go-to framework for researchers and practitioners in the field of artificial intelligence, empowering them to explore complex data, extract meaningful insights, and build powerful machine learning models.
* Kivy: Kivy is an open-source Python framework designed for the development of multi-touch applications. With its cross-platform capabilities, Kivy allows developers to write code once and deploy it across various platforms, including Windows, macOS, Linux, Android, and iOS. By providing a consistent API across platforms, Kivy simplifies the development process and eliminates the need for writing platform-specific code. Its Pythonic syntax and extensive widget toolkit enable developers to create rich user interfaces and implement complex functionalities with ease. Kivy's hardware-accelerated graphics ensure smooth performance, even in applications with demanding graphical elements.
* KivyMD (Material Design for Kivy Applications): KivyMD is an extension of Kivy that brings the principles of Material Design to Kivy-based applications. Material Design, developed by Google, offers a set of guidelines for creating visually appealing and intuitive user interfaces. By integrating KivyMD into Kivy projects, developers gain access to a vast collection of pre-built Material Design components, including buttons, cards, dialogs, and navigation drawers. These components adhere to Material Design standards, ensuring consistency and familiarity for users. With KivyMD, developers can easily create modern and visually stunning user interfaces that enhance the overall user experience.
* SQLite3: SQLite3 is a software library that provides a relational database management system (RDBMS) implemented in a small, self-contained, and serverless manner. It is widely used as a database engine due to its simplicity, efficiency, and ease of integration into various applications. SQLite3 stores data in a single file, making it a lightweight and portable solution for managing structured data.

## 2.4 Engineering Process

Our development process began with an in-depth study of deep neural network structures and the TensorFlow framework, which we intended to utilize for algorithm development. Subsequently, we focused on acquiring knowledge about designing mobile applications using Kivy and employing the Sqlite3 database. During the initial three weeks, our efforts were primarily dedicated to thorough research. Afterward, we proceeded with constructing the algorithm and assembling the dataset. Lastly, we designed the application, transitioning from Kivy to KivyMD for enhanced functionality and aesthetics

### 2.4.1 Dataset

The task of developing a CNN capable of letter detection was relatively straightforward. However, the lack of available datasets significantly complicated our efforts to complete this task. In order to ensure the functionality of the algorithm, it is necessary to train our Convolutional Neural Networks (CNNs) using datasets that contain images and corresponding labels for each image. The primary objective of the first CNN is to recognize and classify the 26 letters of the English alphabet. To achieve this, we decided to utilize the EMNIST dataset, an extended version of the MNIST dataset. The EMNIST dataset comprises approximately 145,600 characters, evenly distributed across the 26 letter classes. This dataset aligns perfectly with our requirements.

After training the first CNN on the EMNIST dataset, we achieved an accuracy of approximately 95%. However, when it came to training the second CNN, we encountered a challenge. We needed a dataset that consisted of incorrect letter representations along with corresponding labels indicating the specific mistake. Unfortunately, we were unable to find an existing dataset that fulfilled all our requirements. Consequently, we undertook the task of creating a new dataset from scratch, which proved to be the most significant challenge during the development process.

To construct this dataset, we identified the most common mistakes typically made when writing letters, such as slant, incompleteness, and incompatibility. Subsequently, we generated a separate dataset for each type of mistake, employing methodologies that will be discussed in detail in the next section of this paper. Through this creation process, we produced approximately 100,000 labeled images for each mistake, totaling 300,000 images. Following the creation of this custom dataset, we trained the second CNN and achieved an accuracy of around 93%. This level of accuracy compares favorably to other existing methods.

### 2.4.2 Dataset Creation Process

As previously stated, we undertook the creation of a new dataset from scratch. Our initial focus was on addressing the simplest mistake to detect, which is the "incomplete" mistake. To accomplish this, we employed the EMNIST dataset, which comprises both well-written and poorly-written letters. Utilizing the previously trained first CNN, we identified the subset of well-written letters that achieved an accuracy exceeding 97%. Once we obtained this set of accurately recognized letters, we proceeded to introduce the "incomplete" aspect. This involved selectively covering specific parts of each image that contained critical details about the letter. The resulting modified images were then saved to a designated folder.

Subsequently, we shifted our attention to the next mistake, namely the "slant" mistake, which posed greater complexity compared to the previous one. In this stage, we utilized the aforementioned "good letters" dataset and applied a specific transformation to each image. To accomplish this, computer graphics techniques such as shearing and rotating were employed at specific angles.

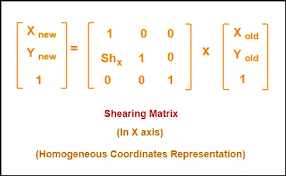


Figure 3: Representing the Shearing matrix used in Computer Graphics

The shearing matrix, illustrated in the figure 3 [8], played a crucial role in this process. The parameter "" represents the shearing factor, Positive and Negative values of SHx in the shearing matrix represent shearing in opposite directions. A positive SHx value indicates shearing towards the right side, while a negative SHx value indicates shearing towards the left side. determining the angle of shearing in relation to the original object.

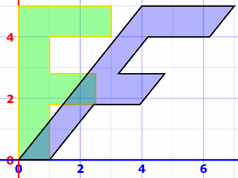


Figure 4: Represents the object before and after applying the shear effect

The second figure (figure 4) [9] depicts the impact of shearing on the image content. On the left side, the original letter 'F' is displayed, while on the right side, the sheared 'F' demonstrates the presence of the slant mistake. Subsequently, these modified images were saved to the designated folder for further usage and analysis.

Upon completing the first two mistakes, we encountered the final and most challenging mistake, known as "incompatible". This particular mistake proved to be the most intricate due to the requirement of distinct parts within each letter being different from the rest of the letter. Furthermore, these differing parts varied from letter to letter. To overcome this complexity, we employed Generative Adversarial Networks (GANs) as a solution.

GAN: A machine learning framework consisting of two neural networks: a generator and a discriminator. The generator learns to create synthetic data that resembles a target dataset by mapping random noise or input data to the desired distribution. The discriminator, acting as a binary classifier, distinguishes between real and synthetic data(see figure 5)[10], striving to correctly classify each sample. Through an adversarial process, the generator and discriminator compete and improve iteratively. The goal of GANs is to train the generator to produce realistic data that is indistinguishable from real data, while the discriminator becomes increasingly challenged in differentiating between real and synthetic samples. GANs have found application in diverse domains such as image and text generation, style transfer, and data augmentation.

A diagram of a structure

Description automatically generated with low confidence

Utilizing GANs, we were able to generate images of letters with a high level of accuracy. However, our objective was to generate images with incompatible characteristics, which posed a challenge. We trained the GAN specifically for this task, stopping the training process at a point where the GAN could generate letter images. However, at this stage, the generated letters still exhibited imperfections such as slight variations in size, readability, and overall appearance. Despite these limitations, the GAN's utilization played a crucial role in our pursuit of achieving the desired outcome for this complex mistake, taking into account the unique variations required for each letter. Finally, we saved the generated data into a designated folder for future use and analysis.

After completing the entire process, the next step involved combining all three datasets: the dataset for the "incomplete" mistake, the dataset for the "slant" mistake, and the dataset for the "incompatible" mistake. These three distinct datasets were merged into a single comprehensive dataset.

## 2.5 Challenges

Throughout the semester, we encountered numerous challenges. In the following account, we will delineate these challenges and elaborate on the strategies we employed to address them.

**Datasets**, the primary and foremost challenge we faced during the semester as we previously elucidated in section 2.4.1.

**Kivy**, as we previously explained, is a Python-based platform used for designing mobile applications. We chose Kivy primarily for two reasons. Firstly, Python serves as the coding language for our machine learning algorithm, which forms the core of our application. Secondly, Kivy offers cross-platform compatibility, allowing us to develop apps for both Android and iOS devices.

However, we encountered several disadvantages while using Kivy. The first drawback was its limited design capabilities. Kivy lacked extensive customization options, which restricted our ability to create visually appealing interfaces. Additionally, we found a scarcity of references and explanations for Kivy online. For instance, commonly used components such as spinners (also known as drop-down menus) had only one predefined style available, with no options for alternative designs or variations.

To overcome this challenge, we addressed it by transitioning our app from Kivy to KivyMD. KivyMD, an abbreviation for Kivy Material Design, provided us with a solution that offered access to a wider range of tools and a greater variety of tool types. By making this switch, we were able to enhance the design capabilities of our app and utilize a more extensive set of components to create a visually appealing and user-friendly interface.

**Time** posed another significant challenge for us. The substantial time required for developing the main algorithm and preparing the datasets left us with limited time to work on the app itself. Additionally, the decision to remake the app using KivyMD instead of Kivy further consumed our time resources. Consequently, we faced the challenge of managing our time effectively to ensure the completion of essential features within the given timeframe.

To overcome this challenge, we devised a schedule that prioritized the implementation of crucial features for our app. By focusing on the most important functionalities, we ensured that they were developed and integrated within the designated time constraints. Subsequently, we allocated the remaining time to enhance the user interface, aiming to improve the overall user experience and visual appeal of the app. This strategic time management approach helped us make efficient use of the available time and accomplish both the essential features and UI enhancements.

**APK**, in order to make our app compatible with Android devices, we needed to create an APK (Android Application Package) file. We discovered a tool called Buildozer that could convert Kivy apps into APK files. However, we encountered a few challenges during this process.

One significant challenge was that not all directories within our app were compatible with Android devices because they were not written in Python. To address this, we thoroughly examined the directories used in our app and identified two directories that posed compatibility issues: "opencv-python" and "TensorFlow." Fortunately, there was a solution for "opencv-python" in the form of a recipe directory developed by Kivy users, enabling it to run on Android devices. However, no such recipe directory existed for "TensorFlow."

To overcome this challenge, we decided to switch from using TensorFlow to TensorFlow Lite. This involved converting our neural network models from the "h5" format to "tflite" files. While this conversion process may have slightly reduced the accuracy of the models, we anticipated that the impact would be minimal. Fortunately, we succeeded in this endeavor, and the TensorFlow Lite models performed well. Unfortunately, despite these efforts, we encountered difficulties in generating the APK file. It is possible that further troubleshooting or exploration of alternative tools and methods would be necessary to resolve this issue.

## 2.6 Testing plan

To ensure the proper functioning of the application, we conducted validation checks on both the algorithm and the database. This was done to verify that both components are working correctly and producing accurate results.

After completing the training phase of the two CNNs used in the main algorithm, we performed an evaluation process on them. This evaluation involved testing the CNNs on a separate testing set, which is a subset of the complete dataset. Once the evaluation process was complete, we obtained all the necessary scores, including accuracy and error percentage, to assess the performance of the CNNs.

In the database component, we conducted unit testing to ensure the accurate insertion and retrieval of data. We summarized these tests into the following cases, which covered various scenarios:

| Case | Purpose | Headline | Results |
| --- | --- | --- | --- |
| Registration and Login | To guarantee that the inputted information is accurately processed and securely stored. | Registration | The new account was added to the database successfully |
| Log in | The account username, password, Email and ID was retrieved from the database |
| Failed Log in | The account with the same username and password was not found in the database, returning the proper feedback |
| Learning Progress | Ensure that the progress is error-free and the user learns the letter well | Progress creation | The progress was created successfully  and added to the database |
| Progresses retrieval | All the Progresses with all their components were retrieved from the database |
| Progress retrieval(by name) | The Progress with the same name was retrieved from the database |
| Action creation | The new action with the feedback and image was successfully saved to the database |
| Actions retrieval(by progress) | All the Actions in the same progress with all their components were retrieved from the database |
| Image saving | The image of the Writing area was successfully scanned, converted to BLOB and inserted to the database |
| Image retrieval | The BLOB was successfully retrieved from the database and converted to regular image |
| Letters saving | The splitted letters were successfully converted to BLOBs and inserted to the database |
| Letters retrieval | The BLOB was successfully retrieved from the database and converted to regular image for every letter |
| Letters Deletion | All the splitted letters were removed from the database |
| Quiz | Ensure correct display and functionality of quiz based on selected level and language. | Quiz creation | The Quiz was created successfully  and added to the database without a grade or a date |
| Quiz finish | The grade and the current date were added to the quiz in the database |
| Action addition to the quiz | The new action with the feedback and image was successfully added to the to its appropriate  quiz in the database |
| Actions retrieval (Quiz) | All the Actions in the same quiz with all their components were retrieved from the database |
| Graph creation | The graph of the quizzes’ grades was successfully converted to BLOB and inserted to the database |
| Graph retrieval | The BLOB was successfully retrieved from the database and converted to regular image and displayed on the screen |

## 2.7 Conclusion

Our ultimate objective was to create a mobile application aimed at assisting users, particularly learners, in improving their handwriting skills. To achieve this goal, we developed an algorithm capable of detecting three key mistakes commonly encountered in handwriting: slant, incompleteness, and incompatibility.

The application functions by providing interactive lessons on writing letters and words. It achieves this by detecting each individual letter as the user writes and offering real-time feedback tailored to the specific errors made. This feedback aims to guide users towards correcting their mistakes and ultimately improving their handwriting proficiency.

In addition to the core features mentioned earlier, we incorporated quizzes into the app to enhance the learning experience and make it more engaging for users. These quizzes allow users to test their skills and assess their progress in a fun and interactive manner.

By including quizzes, users have the opportunity to compare their results over time, providing a tangible measure of their learning progress. This feature not only adds an element of entertainment but also serves as a motivational tool, encouraging users to continue practicing and striving for better performance.

During the development we faced many challenges (for examples see section 2.5), in order to solve these challenges we met for brainstorming sessions, where we analyzed the issue, researched possible solutions, and chose the best fit for our problem. When choosing a solution, we considered a few things: easy or fast solution, engineering practices, available tools and knowledge.

Unfortunately, due to the challenges encountered in generating the APK file, we were unable to deploy the app on Android devices. As a result, we were unable to obtain user evaluations and feedback on the application's performance and user experience. This limitation prevented us from gathering valuable insights and making further improvements based on user input.

However, despite not being able to obtain user evaluations, we remain committed to addressing the technical issues and exploring alternative methods for generating the APK file. By resolving these challenges, we hope to eventually make the app available on Android devices and gather valuable feedback from users to further enhance and refine the app's functionality and usability

# 3. User Documentation

## 3.1 User’s guide - Operating instruction

### 3.1.1 General description

The app is designed to facilitate the learning and improvement of handwriting skills for beginners. Users have the option to select their desired level, language, and whether they want to practice individual letters or words.

If the user chooses to practice letters, they will start by learning the letters in alphabetical order, based on the selected language. They will progress to the next letter only after writing the current letter perfectly. As they learn more letters, they will have the opportunity to take quizzes where they will be presented with words that primarily consist of the letters they have recently learned. This allows them to practice the newly acquired letters in a practical context.

Alternatively, if the user opts for word practice, the app will provide them with random words from a collection appropriate for their chosen level. This helps users expand their vocabulary while honing their handwriting skills.

To track their progress, users can access a history log that displays their previous actions and scores. This information can be viewed in different formats such as graphs or lists, providing a comprehensive overview of their improvement over time.

By offering personalized learning paths and meaningful practice exercises, the app aims to assist beginners in enhancing their handwriting skills in an engaging and effective manner.

### 3.1.2 Operating instructions

**General pages for all users:**

# 

Log-in page - enter username and password if you are already signed in, you can sign-in by clicking create account button

# 

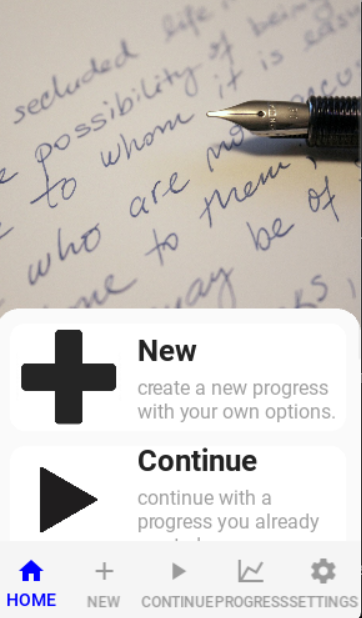
# 

# 

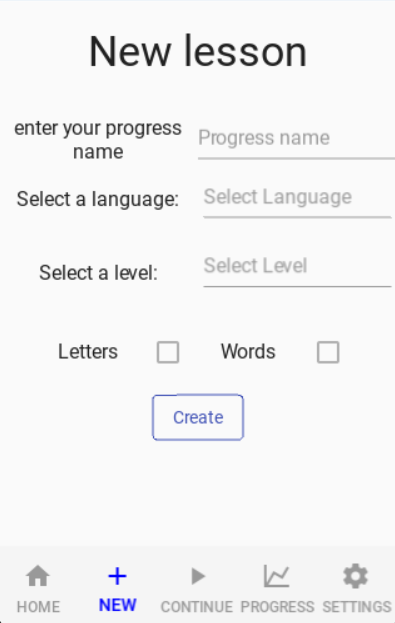
# 

# 

Create account page- fill the form with the requested details. After the registration is complete, you will be logged in automatically.



Home page - you can read an explanation about the different pages in the app in order for the user to be able to use the app more easily.

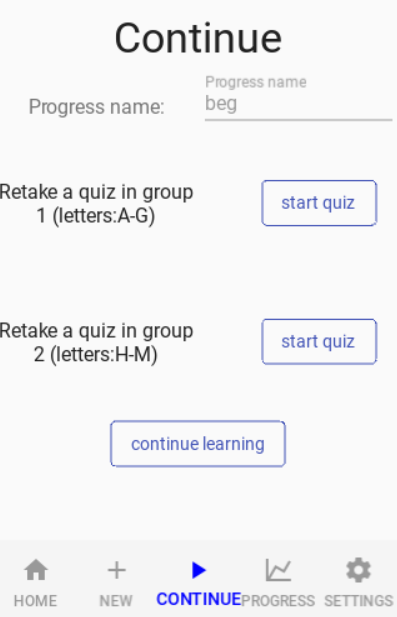


New page - start new progress with your own level, language and you can choose letters or words, progress

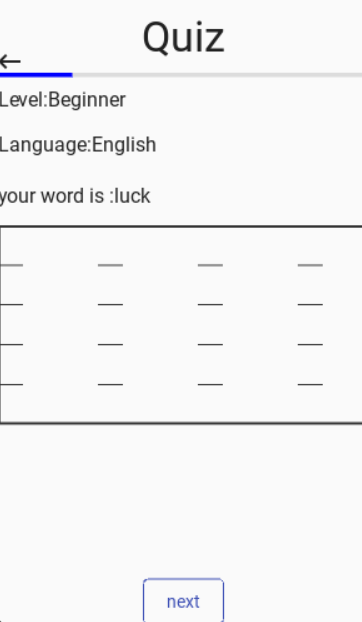
name is required in order to be able to continue learning and view results in future.



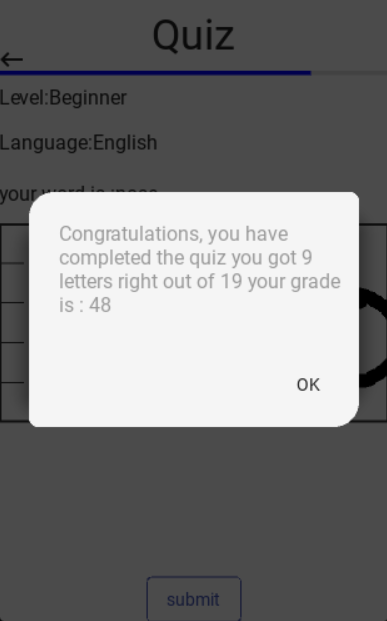
Continue page - you can select from the list of your progresses a progress to continue the learning.



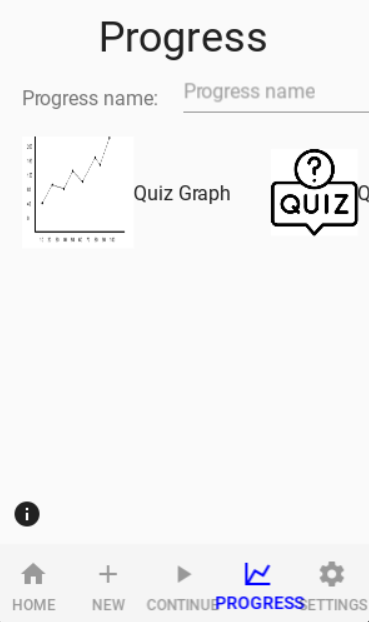
After selecting progress from the list you will have to choose either to retake a quiz in a group you already unlocked or to continue learning .



After pressing start quiz from any group you will go to the Quiz page, you will have to write five words and you will see a progress bar that will show you your way on the quiz.



After completing the five words the button will be named “submit” instead of next and when clicking the button you will get a popup with the result of the quiz.



Progress page- you can choose between three ways to view your progress, graphs, quizzes, and actions, there is an advice button that can explain what to expect when choosing.

A picture containing text, screenshot, diagram, plot

Description automatically generated

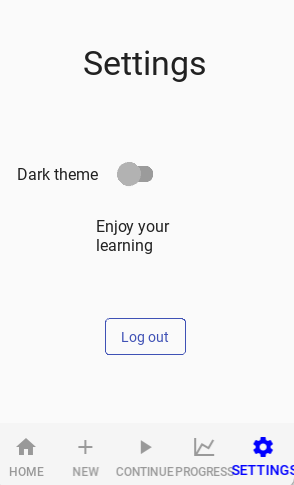
After choosing a progress and choosing “Graph” you will be shown a graph of your scores in the quizzes you did in this progress, in X axis is the quiz number in the order you take it and in the Y axis are the quizzes grades.

A screenshot of a phone

Description automatically generated with medium confidenceAfter choosing a progress and choosing “Quizzes” you will be shown a list of the words you wrote while doing the quizzes of this progress you chose and the feedback you got , pressing on the row will show your writing of the word with the full feedback.

A screenshot of a phone

Description automatically generated with medium confidenceChoosing the third option “Learning progress” is almost the same as the second option “Quizzes” with only one difference which is the list of words/letters are the words/letters you wrote while learning and not doing the quiz.



Setting page - you can change from light mode to dark mode, and log out from your account.

## 

## 3.2 Maintenance guide

### 3.2.1 How to install our app

Currently, we still haven't converted the app to an APK file that can be executed at Android devices so you have to run the app on your computer and to do that you have to follow the instructions below, in the future an APK file will be available.

### 3.2.2 App maintenance guide

In order to maintain the code, you first need to install a few tools and working environments on your computer.

Starting with:

1. VSCode - [Download Visual Studio Code - Mac, Linux, Windows](https://code.visualstudio.com/download) (Optional, can be any other environment that runs python files)
2. Python - [Download Python | Python.org](https://www.python.org/downloads/)
3. You have to install a couple of directories to your environment; you can do that by copying these to your terminal:

pip install tensorflow

pip install opencv-python

pip install kivy

pip install kivymd

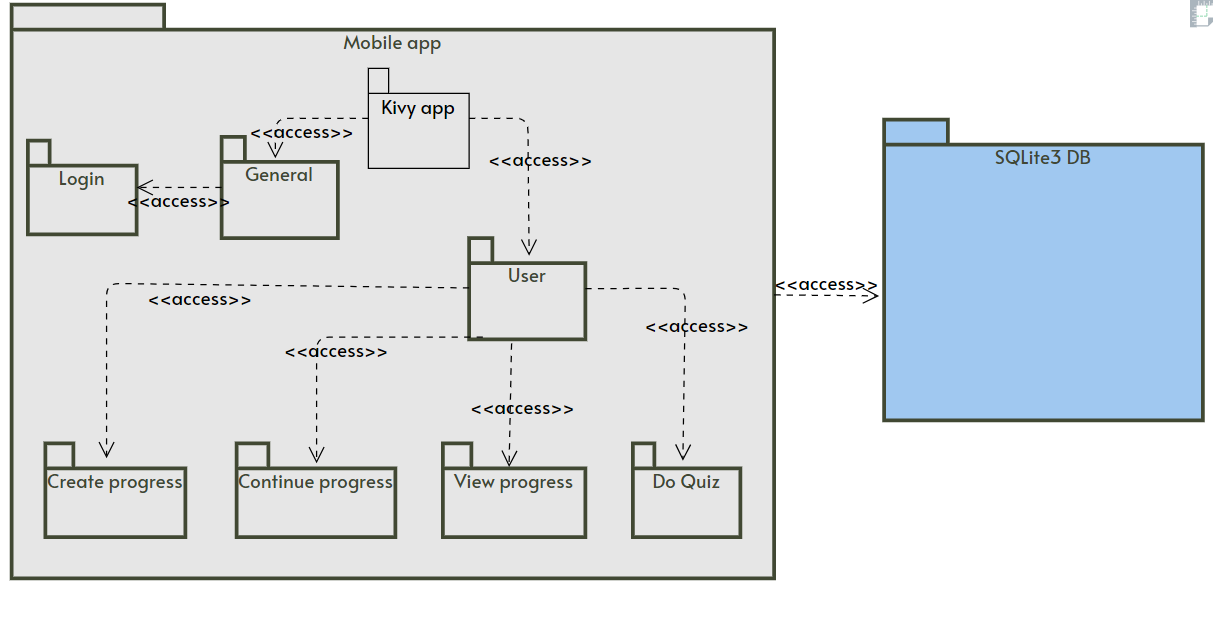
pip install ssl

pip install secure-smtplib

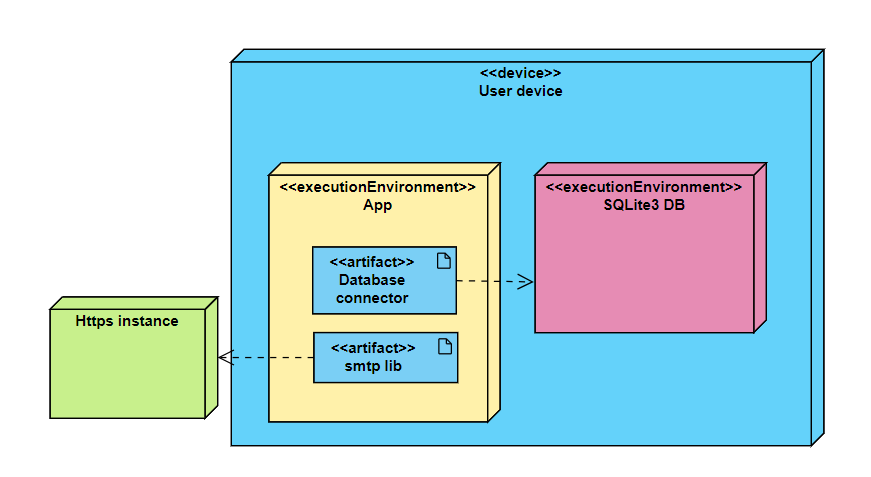
pip install imutils

After installing all these directories you have to clone the project from this repository in Github- <https://github.com/SalmanAmer/Capstone-Project-/tree/main> , and then you can run the code on your computer.

### 3.2.3 Package Diagram

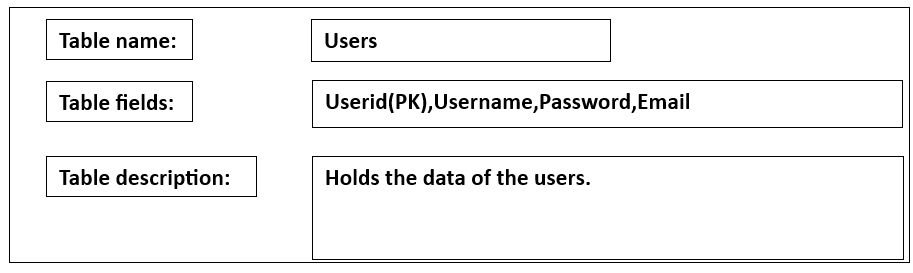


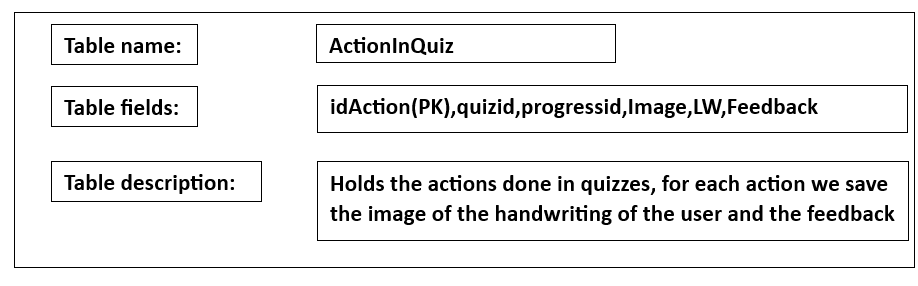
### 3.2.4 Deployment Diagram

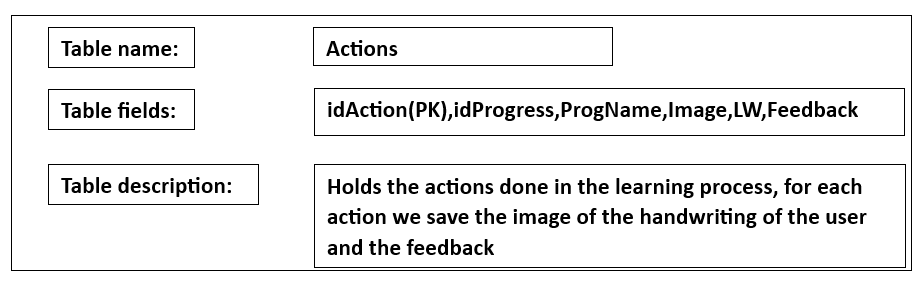


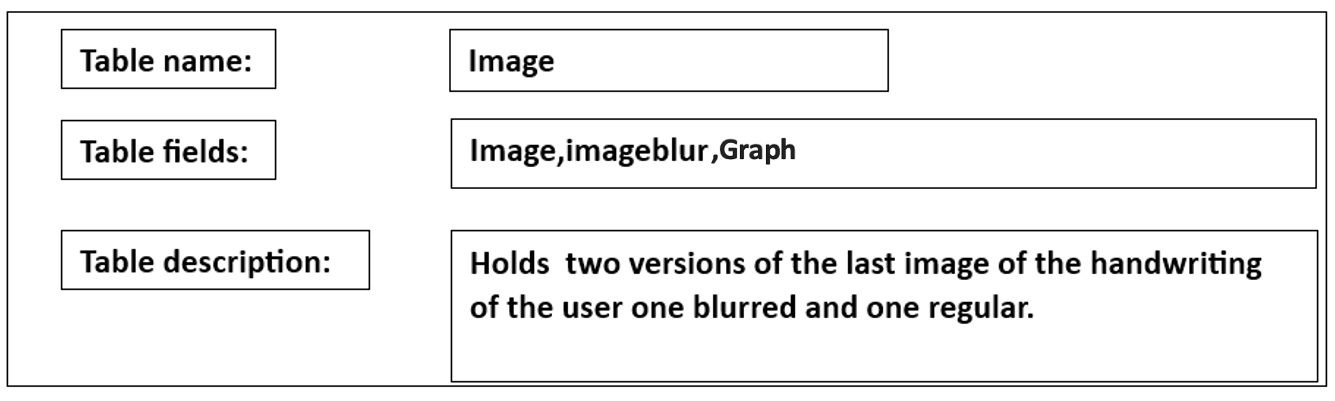
### A picture containing diagram, line, receipt, plan Description automatically generated3.2.5 Activity Diagram

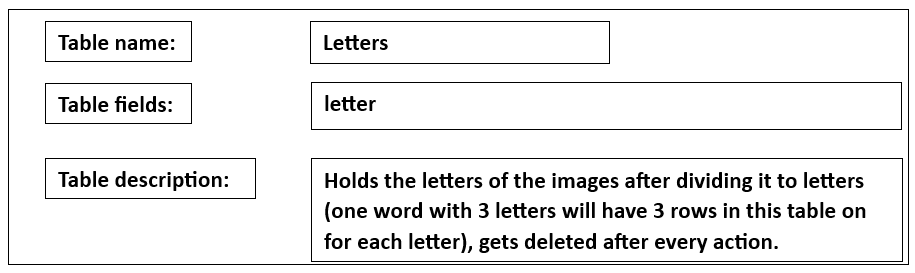
### 3.2.5 Database description

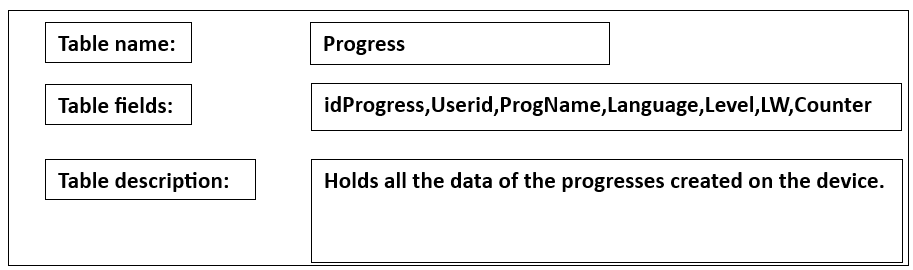


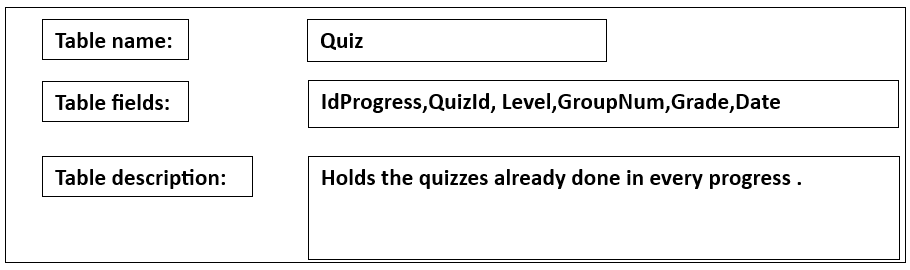


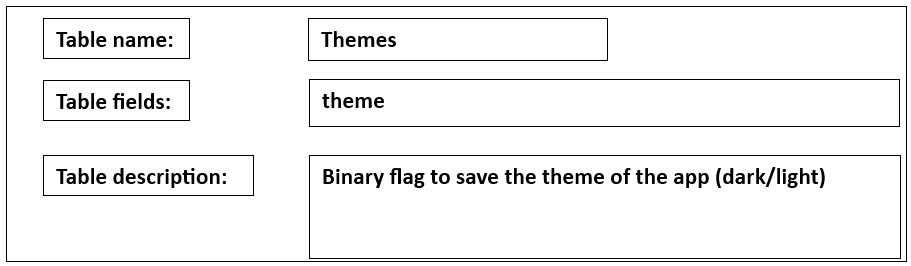












# 4. References

1. <https://www.aacap.org/AACAP/Families_and_Youth/Facts_for_Families/FFF-Guide/Children-And-Watching-TV-054.aspx>
2. <https://psychcentral.com/news/2018/05/27/modeling-behavior-for-children-has-long-lasting-effects>
3. Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of educational research*, *77*(1), 81-112.
4. Hollerbach, J. M. (1981). An oscillation theory of handwriting. *Biological cybernetics*, *39*(2), 139-156.
5. <https://www.lwtears.com/blog/teaching-tips-4-ps-handwriting-instruction>
6. <https://katzr.net/713e67>
7. <https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/>
8. <https://www.gatevidyalay.com/2d-shearing-in-computer-graphics-definition-examples/>
9. <https://www.mathsisfun.com/algebra/matrix-transform.html>
10. <https://www.analyticsvidhya.com/blog/2021/10/an-end-to-end-introduction-to-generative-adversarial-networksgans/>