WriteRight

by

NC Ahmed Bilal Siddiqui

NC Maryam Kamal

NC Muhammad Nabeel

NC Mishal Zahra

Supervised by

Supervisor. Prof. Dr. Nauman Ali Khan

In Partial Fulfillment

Of the Requirements for the degree

Bachelors of Engineering in Software Engineering (BESE)

Military College of Signals

National University of Sciences and Technology

Rawalpindi, Pakistan

May 2024

In the name of Allah, the Most Benevolent, the Most Courteous.

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report “WrightRight” is carried out by Ahmed Bilal Siddiqui, Maryam Kamal, Muhammad Nabeel, Mishal Zahra under my supervision and that in my judgment, it is fully ample, in scope and excellence, for the degree of Bachelor of Software Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Asst Prof Dr. Nauman Ali Khan

Date: May 12, 2024

DECLARATION OF ORIGINALITY

DEDICATION

I dedicate this thesis to all those who believed in me and stood by me during this challenging yet rewarding journey. To my family, friends, and mentors, your support and encouragement have been invaluable. Special thanks to my project supervisor, Dr Nauman Ali Khan, whose expertise and guidance were essential to the completion of this work.

Dedicated to our beloved families and our country Pakistan.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.
Our Parents, colleagues and most of all our supervisor, Dr. Nauman Ali Khan,
without your support and guidance it would have never been
possible for us to complete our Final Year Project.

Plagiarism Certificate (Turnitin Report)

We hereby declare that this project report entitled “WrightRight” submitted to the “DEPARTMENT OF COMPUTER SOFTWARE ENGINEERING”, is a record of an original work done by us under the guidance of Supervisor “ASST. PROF. DR. Nauman Ali Khan” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science. This thesis has an **10%** similarity index. Turnitin report endorsed by Supervisor is attached at the end.

Team Members	**Signature**
NC Ahmed Bilal Siddiqui	______________________
NC Maryam Kamal	______________________
NC Muhammad Nabeel	______________________
NC Mishal Zahra	______________________

Supervisor	**Signature**
Asst Prof Dr. Nauman Ali Khan	______________________

Date:
May 12, 2023
Place:
Military College of Signals, NUST,
Rawalpindi Pakistan

ABSTRACT

Handwriting is an important skill that can affect one’s academic and professional performance, as well as self-esteem and confidence. However, many people struggle with poor handwriting due to various reasons, such as lack of practice, improper guidance, or physical or mental challenges. Improving one’s handwriting can be a tedious task, especially for children and young adults who are used to digital devices and online communication. The need for an app that improves handwriting skills has never been more apparent, especially in educational contexts where handwritten assignments, exams, and note-taking continue to be vital.

WriteRight arises from the recognition that technology can be harnessed to make the process of improving one's handwriting engaging and accessible to users. We have developed a gamified version of a handwriting improvement app that makes learning and practicing handwriting fun and rewarding. The app uses the latest technologies, such as Flutter, Firebase, and Django, to create an interactive and user-friendly interface, a secure and scalable database, and a smart and accurate AI engine. The app allows users to register and log in, see a map of different levels, see their profile and progress history, and rate words on each level. The app aims to help users achieve better handwriting quality, and consistency and boost their self-confidence and satisfaction.

Table of Contents

List of Figures

Chapter 1. SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

1.1.1 Purpose

This is a thesis document for mobile-based handwriting improvement application. WrightRight makes handwriting practice fun, engaging, and ultimately, successful. It joins people, devices, and applications and students, teachers and schools providing primary education to help students improve their handwriting and act as an assistant to teachers improving overall ecosystem time efficiently. The system will provide gamified levels, feedback to the answers, live statistics and a Teacher-Student ecosystem.

1.1.2 Document Conventions

- Headings are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, per their level.
- All the main headings are titled as a single-digit number followed by a dot and the name of the section (Time New Roman, size 18).
- All second-level subheadings for every sub-section have the same number as their respective main heading, followed by one dot and subsequent subheading number followed by the name of the sub-section (Time New Roman, size 16).
- All third-level subheadings follow the same rules as above for numbering and naming, but different for font (Time New Roman, size 14).
- Further level subheadings i.e. level four have font (Time New Roman, size 12).
- This document is typewritten using a font size of 12pt in Time New Roman, with margins of 2 cm and line spacing of 1.5 lines.

1.1.3 Intended Audience and Reading Suggestions

Anyone with some basic knowledge of programming can understand this document. The document is intended for Users, Developers, Software architects, Testers, Project managers and Documentation Writers. But anyone with a programming background and some experience with UML can understand this document.

It is divided into Several phases with sections being intended for developers and software managers, but other sections can be understood by anyone having little knowledge about software.

To design meaningful test cases and provide useful feedback to developers, testers must first grasp the system's functionality. The developers must understand the specifications of the software product they are creating.

This paper is intended for broad conversations about WrightRight Application implementation decisions. The product's user should understand the fundamentals of App Development, Firebase, Digital Image Processing,interfaces, and classes.

1.1.4 Product Scope

"WriteRight" is a comprehensive handwriting improvement tool. It provides a gamified, interactive learning environment for all ages, with real-time feedback and progress tracking. It includes a "Teacher Dashboard" for educators to monitor student progress and manage assignments. The app uses Flutter, Firebase, Django, and an image comparison algorithm, and is compatible with Android and iOS devices. It aims to create an inclusive, effective, and enjoyable platform for handwriting improvement.

1.2 Overall Description

1.2.1 Product Perspective

Our project aims to develop "WriteRight," a comprehensive handwriting improvement tool that revolutionizes the way users of all ages enhance their writing skills.

1.2.2 Product Functions

The WriteRight application will be able to do following:

1. Gamified Learning
2. Real-time Feedback
3. Teacher Dashboard

1.2.3 User Classes and Characters

There are two main user classes.

1. **Students**

   These users will be allowed to play the gamified levels and climb up the points table depending upon their score on each level. They will have to login to access the system. They can also get registered with a teacher by sending a request.

2. **Teachers**

   The teachers are required to first login into the system. They can see the requests students have made to register with them which can either be accepted or declined. Teachers can create customized exercises which are visible to all of their registered students.

1.2.4 Operating Environment

Operating environment for the WrightRight is as listed below.

- Cloud database
- Client/server system
- Operating system: Android/IOS
- Database: Firebase database
- Platform: Flutter/Django/Python

1.2.5 Design and implementation constraints

- The information of all users, events must be stored in a database that is accessible by the application.
- The results are to be considered as mere suggestions from an assistant and not a final decision.
- The server must be up and running 24/7.
- The results depend upon the angle and surrounding conditions in which the image is captured.

1.2.6 Assumptions and Dependencies

The product needs following third party products

- Tesseract OCR.
- Firebase ecosystem to store data.

- Flutter/Python to develop the product.

1.3 External Interface Requirements

1.3.1 User Interfaces

User Interface will have following components:

1.3.1.1 Authentication Screens

The homepage will have a form for username and password to authenticate teachers and students. On click of the ‘LOGIN’ button, the user will be authenticated and taken to the dashboard, where students and teachers will have their respective screens.

1.3.1.2 Dashboard

The student module consists of 5 Tab screens i.e. Map, Profile, History, Leadersboard, and Register Teacher. Students can navigate among all these screens using the tab bar.

1. Map
2. Profile
3. History
4. LeadersBoard
5. Register Teacher.

Teacher module consists of

1. Teachers Dashboard
2. Enrolled Students
3. Request Screen

1.3.1.3 Map

Students will be displayed a map of all the available levels. He/she can select a level from the map. The student will be presented with a reference image and will have the option to either copy the given image on paper and upload a photo of their handwritten work, or write the alphabet directly on the screen using a stylus. After clicking the ‘submit’ button the user will be shown a result popup with the user's score on it.

1.3.1.4 Leadersboard

The Leaderboard will show a list of our top students. This will help in creating a competitive environment for students.

1.3.1.5 Profile

Profile screen will display the user’s data like username etc. It will display the user's current total score.

1.3.1.6 History

The history screen will show the progress of the student with help of graphs and charts. This will help students in keeping track of their progress.

1.3.1.7 Teachers Dashboard

The teacher's dashboard will provide an overview of their profile, including the number of students enrolled, exercises assigned, and other relevant details.

1.3.1.8 Enrolled Screen

The enrolled screen will display a list of all students currently enrolled with the teacher. Clicking on a student's name will show a list of all submissions made by that student.

1.3.1.9 Requests Screen

The Request screen will display all the student registration requests received by the teacher. Teachers have the option to either accept or decline these requests.

1.3.2 Hardware Interface

This system needs the following hardware requirements.

1.3.2.1 Camera Interface

The application would require access to the device's camera for capturing images of handwritten content.

1.3.2.2 Storage Interface

Access to storage interfaces is necessary for storing and managing the images captured by the camera. This includes both internal and external storage options (e.g. SD cards) for saving and retrieving images.

1.3.2.3 Operating System APIs

Access to the device's operating system APIs is essential for interacting with hardware components and managing image files, along with other system-level functions.

1.3.2.4 Networking Interface

The application involves uploading images to a server for analysis or comparison, a network interface would be necessary for internet access via Wi-Fi or cellular data.

1.3.3 Software Interfaces

The software interface should follow the Model-View-Template (MVT) model for rendering and modeling data objects. The interface must connect to a database to fetch user information, retrieve previous performance, display provided sample images, and display the final result. Sample images and images uploaded can have a format of JPEG, PNG, and JPG.

1.3.4 Communication Interfaces

The communication architecture must follow the client-server model. Communication between the client and server should utilize a REST-compliant web service and must be served over HTTP Secure (HTTPS). The client-server communication must be stateless. A uniform interface must separate the client roles from the server roles.

1.4 System Features

In this subsection, we will examine the features of the system in detail by categorizing them according to their functionality. For each of the features, we will give an introduction, purpose, and a stimulus/response sequence. Introduction part will give basic background information about the feature. Alternative flow of events will be given in stimulus/response subsection

1.4.1 User Registration

1.4.1.1 Description

The registration feature of our app serves as the initial step for users to create their accounts and access the app.

1.4.1.2 Stimulus/Response Sequence

Stimulus: User opens the app for the first time.

Response: Display a registration or login screen.

Stimulus: User clicks on the "Sign Up" button.

Response: Display a registration form to user

Stimulus: User clicks on the "Login" button.

Response: Display a login form with fields for email and password. After successful login, redirect the user to the homepage.

1.4.1.3 Functional Requirements

REQ-1: Users can create an account by providing a username, email, and password.

REQ-2: Users can log in with their credentials.

REQ-3: Users can reset their password if forgotten.

1.4.2 Tracking History

1.4.2.1 Description

Analytics and progress tracking allow users to see their improvement trends and identify areas where students need more practice.

1.4.2.2 Stimulus/Response Sequence

Stimulus: User visits History screen.

Response: Display historical performance data and track the progress and learning trends of each student over time.

1.4.2.3 Functional Requirements

REQ-1: Track and display a user's historical performance and improvement over time.

REQ-2: Provide detailed statistics, such as similarity scores and learning curve etc.

1.4.3 Gamification and Level System

1.4.3.1 Description

Gamification adds engagement and motivation to the app. Users are motivated to improve their performance to reach higher levels.

1.4.3.2 Stimulus/Response Sequence

Stimulus: User achieves a certain similarity score.

Response: Update the user's level, reward them, and display their progress.

Stimulus: User levels up and enters a new level.

Response: Display a celebratory message and similarity score of student’s submission.

1.4.3.3 Functional Requirements

REQ-3.1: Define multiple levels of achievement.

REQ-3.2: Set a threshold score for each level.

REQ-3.3: Reward users upon reaching a new level (e.g., stars)

REQ-3.4: Track and display the user's current level and progress.

1.4.4 Image Upload

1.4.4.1 Description

This is the core feature of this app, users can submit handwritten text for comparison and receive a similarity score.

1.4.4.2 Stimulus/Response Sequence

Stimulus: User uploads an image.

Response: Process the uploaded image to extract the handwritten text content and compare it with the dataset using the image comparison algorithm.

Stimulus: User submits an image that contains text not found in the dataset.

Response: Provide feedback that the text is not recognized in the dataset, and the comparison score may not be available.

Stimulus: User submits an image with clear handwriting and high similarity to the dataset.

Response: Calculate a high similarity score and provide a positive message, possibly with encouraging feedback.

1.4.4.3 Functional Requirements

REQ-4.1: Users can upload a handwritten text image.
REQ-4.2: The app must use an image comparison algorithm to compare the uploaded image with the dataset.
REQ-4.3: The application must have a dataset of reference handwritten texts.
REQ-4.3: A similarity index (e.g, a percentage or score) should be calculated and displayed.

1.4.5 Teacher Dashboard

1.4.5.1 Description

The teacher dashboard provides instructors with the necessary functions to manage students, monitor their progress and prepare assignments which students can perform and get evaluated.

1.4.5.2 Stimulus/Response Sequence

Stimulus: Instructor logs in to the teacher dashboard.

Response: Display teacher dashboard with options for managing data sets, assignments, and student progress.

Stimulus: Instructor views a student's progress report.

Response: The system will display a detailed review of each student's performance, including submitted images, assigned exercises, and a list of enrolled students.

1.4.5.3 Functional Requirements

REQ-5.1: Teachers can create different assignments for the students.

REQ-5.2: Teachers can access a comprehensive overview of each student's progress, including submitted images, assigned exercises.

1.4.6 Requirements

Requirement #	Requirement	Component
R1	User registration and login	Mobile app (Flutter)
R2	Account types (student, teacher)	Mobile app (Flutter)
R3	Level selection and gameplay	Mobile app (Flutter), Django app
R4	Image capture and upload	Mobile app (Flutter)
R5	Image processing and comparison	Backend server (Django, Python libraries)
R6	Scoring and feedback generation	Backend server (Django, Python libraries)
R8	Progress tracking and visualization	Mobile app (Flutter), Firebase Database
R9	Leaderboard display	Mobile app (Flutter), Firebase Database
R10	Teacher dashboard for student management	Firebase Database
R11	Exercise upload and management	Firebase Database
R12	Enrollment requests and approvals	Firebase Database

Requirement #	Requirement	Component
R13	Authentication and authorization	Mobile app (Flutter), Firebase Database
R14	Security and privacy measures	Mobile app (Flutter), Django app, Firebase
R15	Performance and scalability	Backend server (Django, Python libraries), Firebase
R16	User interface design and usability	Mobile app (Flutter)
R17	Cross-platform compatibility (Android, iOS)	Mobile app (Flutter)

Table 1. Requirements Table

1.5 Other Non-functional Requirements

1.5.1 Performance Requirements

One of the primary NFRs of the WriteRight application is the system's performance requirement, since the project will analyze students' handwriting and produce findings. Interface screen loading times cannot be longer than three seconds. Users expect a fluid and responsive experience when interacting with the WriteRight program, whether they are accessing various functions or switching between modules. Prolonged loading times might cause user annoyance and disinterest. It is something I have noticed to be very important because if interface screens take a long time to load, users' happiness can not be guaranteed. Second, if no response is received after a certain amount of time has passed, the window is closed and a timeout message is displayed. It is important for the programme to inform the user of a server response or to simply notify the user of the progress when the program is performing a complex and time consuming action. This also assists in managing expectations by creating a barrier that reduces the likelihood of a user being displeased with a certain service. For instance, if a user is performing a handwriting analysis activity that involves server-side processing, and the processing operation takes time exceeding that estimated due to high server load and this disrupts the normal intended operation then a timeout message should be conveyed to the user. It is recommended to include a message in the form of a notification to inform the user or any associated party about the delay. In terms of scalability, the WriteRight application demonstrates solid performance, as thousands of user interactions are processed without any issue in the frontend and server-side backend. For instance, there might be high usage during times like when many users are using the handwriting analysis tool or are engaged in the game-based activities that allow users to enhance specific skills, the workload must be balanced in a way to achieve the best performance. This scaleability guarantees customers positive customer experiences in terms of latency and downtime with increased number of concurrent queries thus providing students with more uninterrupted learning as they make use of the resources in the application.

1.5.2 Safety Requirements

To reduce the risk on the part of the users on the safety and security of their data that they input in the WriteRight application, stringent measures are taken. Optimum security measures complement user-generated dispatches and performance data that are retained and transmitted. This encryption would utilize currently accepted methods and protocols to from access to and eavesdropping on the data whether stored or being transmitted.

1.5.3 Security Requirements

To address security issues the WriteRight Programme has a secure data handling policy, feature access control among others are among the security measures of the WriteRight Programme. Also, restricted and controlled user operations that based of the policies and permits set will be eliminated by policy and access control measures to prevent break-in and unauthorized access into the systems. user legitimacy would also require multiple passwords inputting or another type of input such as the password entry only option.

1.5.4 Software Quality Attributes

1.5.4.1 Availability

At the highest extent, WriteRight makes availability the absolute number one, which ensures servers are continuously running to make sure users never stay out for a single moment or second. As a result, due to this dedication to availability, customers can do their jobs in WriteRight at any time they feel like doing it thereby increasing their productivity levels and at the same time enhancing their experience and user satisfaction.

1.5.4.2 Clarity

WriteRight's prompts and interface elements are designed for instant and clear understanding by anyone who lacks technical expertise. By using plain language and simple design concepts, the program allows users to easily explore its features and functions, so that their engagement with it is enhanced and comprehension promoted.

1.5.4.3 Reliability

WriteRight really treasures reliability. Users can trust that the application will deliver reliable and accurate performance since it undergoes rigorous tests and validations which assures accuracy of handwriting analysis results and other functionalitiesibility

1.5.4.4 Intuitiveness

The principle of design of WriteRight, focus on ease of use where as possible a user will not need guides and tutorials. For WriteRight, known user interface patterns and common sense practices that enable users to complete tasks as easily as possible and move through the system have decreased the time taken to learn while increasing the level of satisfaction.

1.5.4.5 Informative

WriteRight follows a proactive approach to user communication by quickly warning users in case of any problem or error that they have experienced while using the program. Various error messages and notifications are availed to users, giving them information on the condition of the system that creates openness and confidence in the dependability of the program.

1.5.4.6 Correctness

WriteRight analyzes the handwriting through image processing techniques but sometimes reports errors due to limitations. It strives for constant improvement in accuracy over time.

1.5.4.7 Flexibility

WriteRight places high value on adaptability, hence making it easy to respond to emerging user needs and changes in technology. Such flexibility in the architecture and principles of modular design of WriteRight allow it to easily add new features and improvements while ensuring scalability and sustainability. WriteRight's adaptability helps it to remain relevant and valuable over time because it can respond to user feedback and market development.

1.5.5 Business Rules

You can only see your grades and stuff in the WriteRight app if youre actually taking the class. , like, info about how you write and how youre doing with it, is in this data. Only certain students will be able to see personal info to keep things private and confidential. Teachers can check out students past grades and give them tasks in the WriteRight app. But only the teacher who was given to each student will be able to see their answers and other private stuff. Thanks to this access control system, teachers can keep an eye on student progress without invading their privacy.

1.6 Sustainable Development Goals(SDGs)

WriteRight is committed to advancing several key United Nations Sustainable Development Goals, specifically **SDG 4** "Quality Education" through tools for improved learning and communication; **SDG 16** "Peace, Justice, and Strong Institutions" through data security and privacy; **SDG 9** "Industry, Innovation, and Infrastructure" through the leveraging of cutting-edge technologies and creativity; and addressing finally **SDG 10** "Reduced Inequalities" by provision for inclusivity and equality in access to educational resources, ultimately closing the digital divide.

Chapter 2. LITERATURE OVERVIEW

Educational sector targeting kids especially is declined towards speech and verbal content. In this digital age when everything is a click away handwriting education applications are emerging for improved learning experience. This review explores the potential of these apps to enhance handwriting skills across various user groups.

2.1 Industrial Background

In the age of ubiquitous digital communication, the ability to write legibly by hand remains a valuable skill. However, with the decline of pen-and-paper use in daily life, many students struggle to maintain legible and efficient handwriting. Handwriting can be critically important for an individual as it one of the tools for making an impression as well as reflects the degree of professionalism to some extent.

WrightRight creates an early market segregation where growing concern regarding declining handwriting skills creates an opportunity to cater a wide range of users, including **children** to develop proper letter formation and improve writing fluency in a fun and interactive way, **adults** to learn write new language they are learning as being multilingual is in trend these days, and **senior** citizens to maintain cognitive function and dexterity through targeted handwriting exercises.

2.2 Existing Solutions and Differentiating Factor

There is a lot of work done on extracting features from handwritten text but to compare and mark the extent to which the two writings are similar is pretty much new to explore. This is different from signature verification because in signature verification we have a set of true signatures and it is a classification problem where model can be changed while handwriting improvement has no fixed true or false variations and is never a classification problem.

Chapter 3. DESIGN AND DEVELOPMENT

3.1 Introduction

The Software Design Specification (SDS) introduction provides an overview of the complete SDS, including its goal, scope, definitions, acronyms, abbreviations, references, and overview. This document's purpose is to discuss in depth the functional and non-functional features of the WriteRight, which employs image processing techniques to score and mark the images uploaded by students. This document provides thorough descriptions and visualizations of the WriteRight.

3.2 Purpose

The Software Design Specification for WriteRight is a technical guide to help transform the vision of a gamified handwriting improvement system from UI widgets to image analysis algorithms that will seamlessly integrate together. The primary goal of the SDS is not only to instruct developers but to facilitate collaboration, avoid redundant work and ensure a final product that is not only an app but also an integrated system will be delivered. It contains all relevant details of how each component is both functional and delightful to use by students, teachers and every user in between. The SDS in short is the link between vision and implementation to make WriteRight serve its mission to render handwriting practice harmless, entertaining and finally effective.

3.3 Product Scope

"WriteRight" is a complete software solution for handwriting development. The app offers an interactive, gamified and user friendly environment for users of all ages to learn and improve handwriting skills through a fun and competitive space. The learning process is monitored in real-time and users are provided with immediate feedback and rewards. Teachers and Professionals have access to their own "Teacher Dashboard" where they can assign tasks and monitor students' progress. The technology behind the app is a Dart and Firebase based Flutter application, using Django as a backend server, and an image comparison algorithm to assess and compare users' handwriting. WriteRight works on Android and iOS devices and aims to be inclusive, effective and fun.

3.4 System Overview

“WriteRight” is a fun gamified app for improving our handwriting. For the mobile part, I used Flutter, backend is written in Django and, of course, Firebase for syncing data in real time. There is a ranking where users can see their progress compared to other users. The main feature of the product is giving users feedback based on their handwriting using an image comparison algorithm. “WriteRight” is multilingual, has Android and iOS platform support, levels and rewards that boost users’ motivation and much more… This app is a complete solution for better handwriting using technology and interactive learning.

3.4.1 Product Perspective

"WriteRight" is a part of educational technology, it is an application for improving handwriting skills. This mobile application is developed using Flutter and Django. "WriteRight" is a gamified application, that is easy in using, including functions like registration, monitoring progress, passing levels. The application is integrated with Firebase as a data storage, which allows to work with the data in real time. "WriteRight" is a complex solution, which goal is to create an engaging and useful application for improving handwriting skills for students and teachers.

3.4.2 Context Diagram

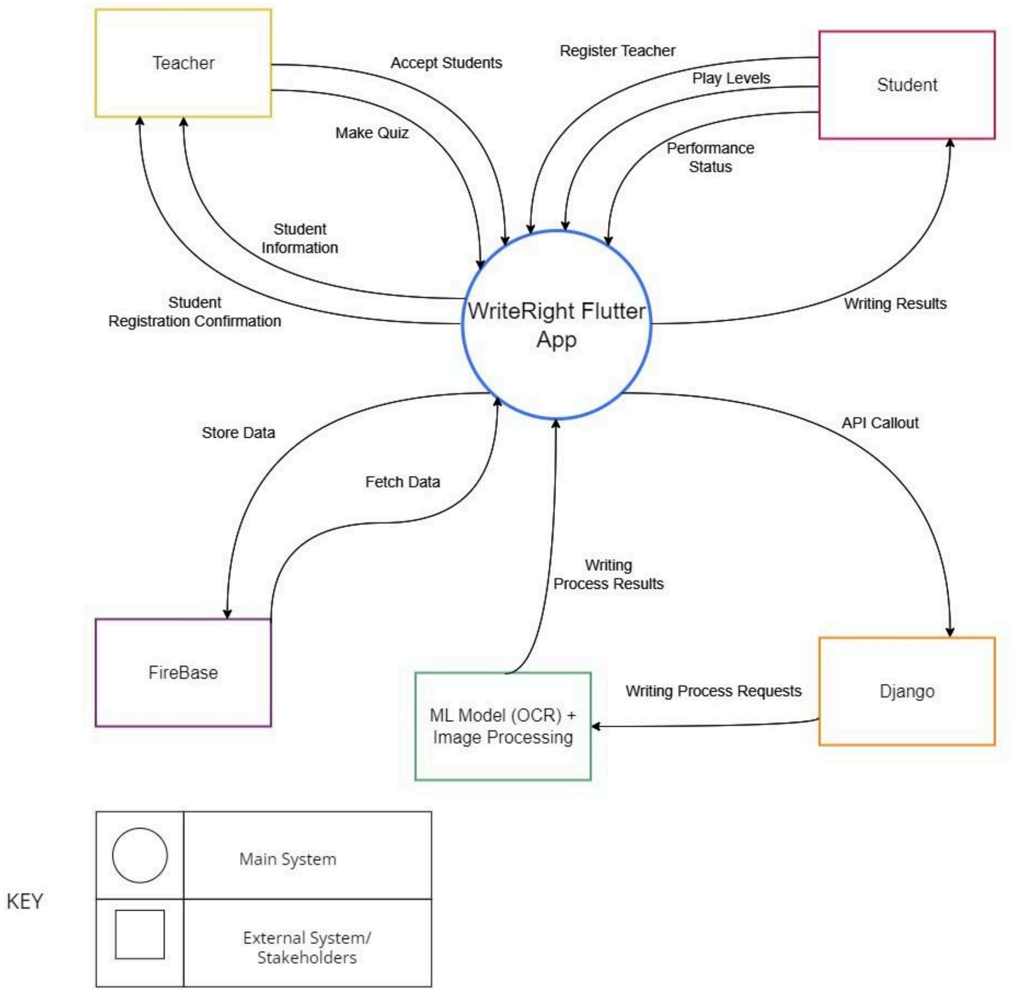


Fig 1 Context diagram presenting outlining external interactions and system boundaries.

The context diagram states the functionalities of WriteRight when users interact – the teachers can manage their students and exercises, while students can play different levels. All data is stored on Firebase and Django's APIs connecting the application to the DIP functions analyzing student's writing. The central system handles user registration, storing and fetching data, generating writing results, and integrating with Firebase and Django.

3.4.3 Data Flow Diagrams

3.4.3.1Student Module

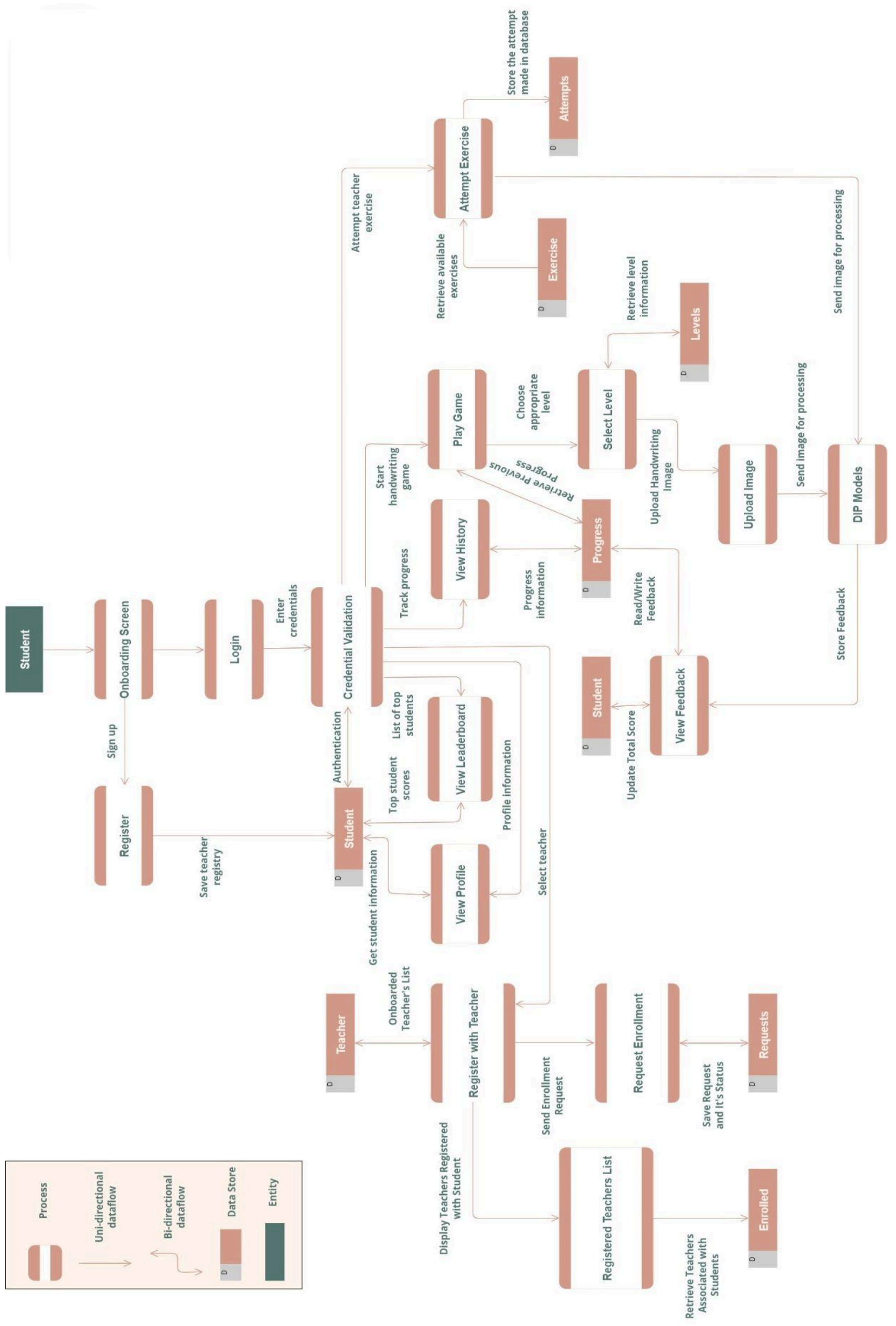


Fig 2: Data Flow Diagram for Student Module of WriteRight

The student journey at WriteRight application starts by students registering with their details and logging into the system. Once logged in, they can play different available gamified levels designed to enhance their handwriting skills. Upon selecting a level, an image is displayed on screen and students submit a handwritten text image trying to mimic it. Students can use the "Edit Profile" tab for profile management, "Gamified Map" for level progression and interactive GUI, "Leaderboard" to encourage healthy competition among students, "Register Teacher" for students to get an amazing experience with teachers, and "History" for a comprehensive performance overview.

3.4.3.2 Teacher Module

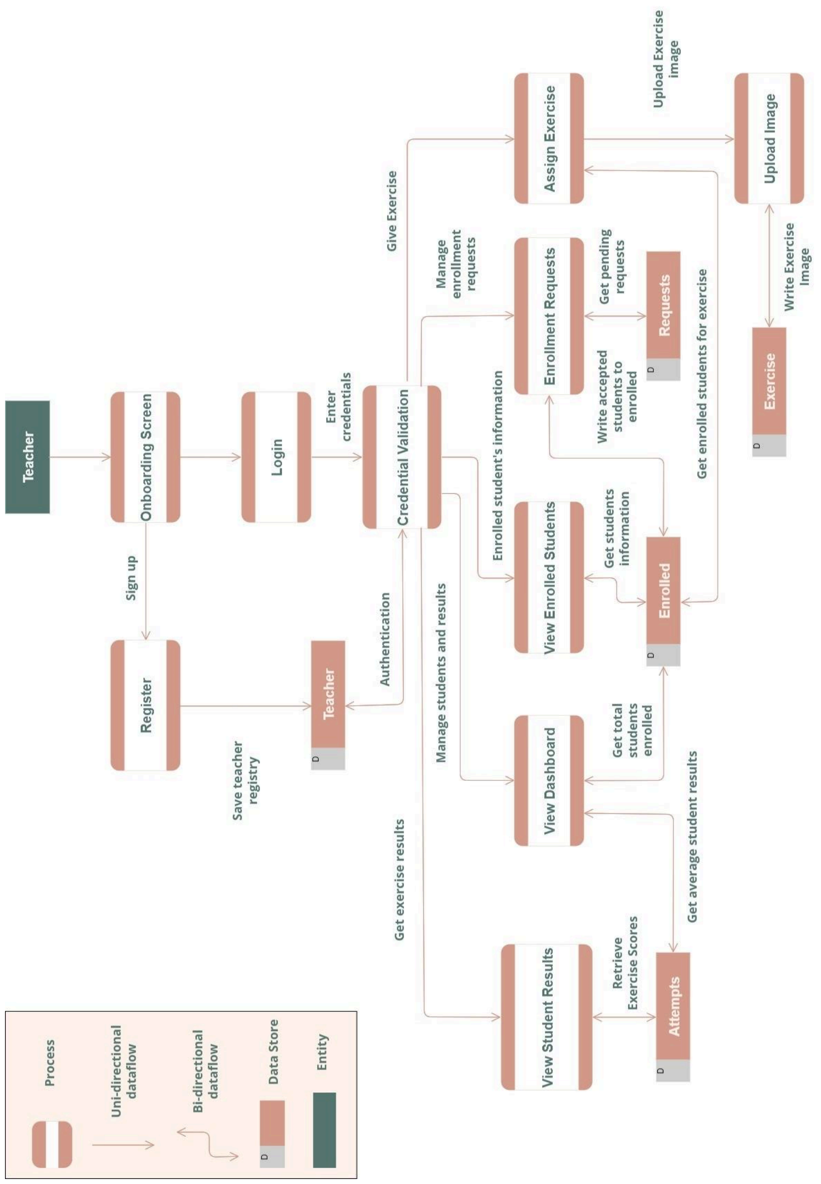


Fig 3: Data Flow Diagram for Teacher Module of WriteRight

The teacher's journey at WriteRight application commence by entering their credentials and are redirected to a comprehensive dashboard. Teachers can seamlessly create assignments, specifying instructions and deadlines. The system securely stores these assignment details, ensuring efficient management. Teachers can also view a list of enrolled students, Additionally, teachers can accept enrollment requests from students who want to get associated with them, enhancing the collaborative and educational aspects of the platform.

3.4.4 Product Functions

3.4.4.1 Client Side

- **User Registration:** Users can register, log in, and reset their password.
- **Tracking History:** Students can view performance trends, celebrate milestones, and track progress.
- **Gamification and Level System:** Students can progress through levels, and earn XP points giving an overall game experience.
- **Image Upload and Comparison:** Students are supposed to upload images, which are then compared and a score is given along with a real-time feedback.
- **Register with Teacher:** Students can search and register with the teacher they want to keep as a mentor.
- **Attempt Exercise:** Students can attempt teacher-assigned exercises for practice.
- **Assign Exercise:** The teacher can upload and assign exercises to students to improve their handwriting.
- **View Results:** The teacher can view the assigned exercise's results.
- **Handle Student Request:** The teacher can accept or reject a student's request.

3.4.4.2 Server Side

- **User Authentication:** Authenticate and authorize users, ensuring data confidentiality.
- **Firebase Real-time Database:** Synchronize data across devices, and manage user profiles and progress.
- **Digital Image Processing (DIP):** Images are preprocessed, OCR extracts the text, and DIP models analyze the image producing results.

3.5 System Architecture

The application's architecture is designed to provide a seamless handwriting improvement experience. The architecture is a client-server model, where the user's mobile device (front end) interacts effortlessly with the application's backend. The engaging user interface, developed in Flutter offers effortless navigation through features like registration, progress tracking, and gamified learning. The robust backend, powered by Django and supported by Firebase handles important tasks such as authentication, data storage, and image processing using sophisticated DIP models. There is a unique channel for DIP functions. A solid and scalable foundation for WriteRight is created through this seamless integration of user-friendly design with complex backend technology to facilitate smooth and impactful handwriting excellence experience.

3.6 Architectural Design

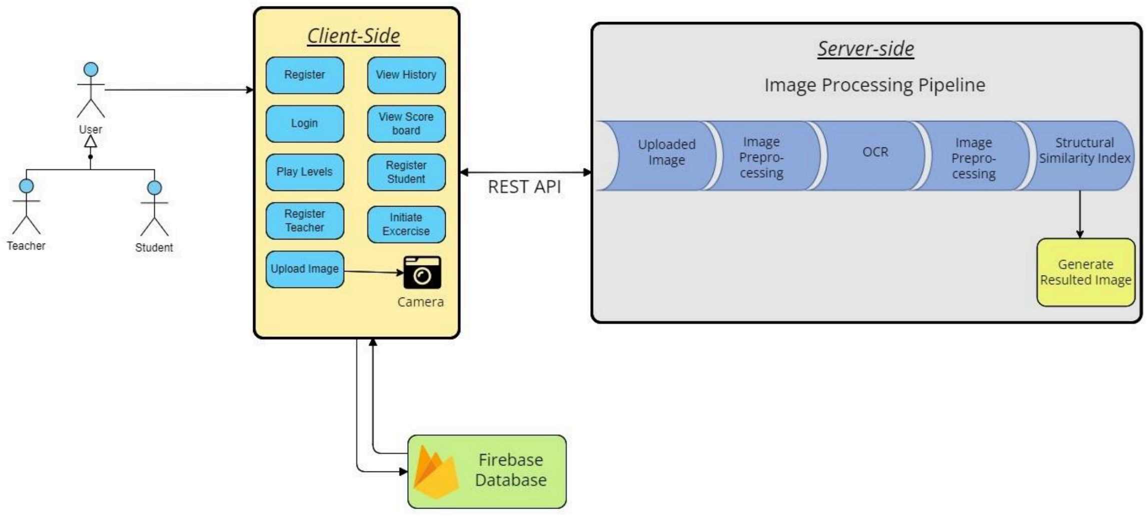


Fig 4: Client-Server Architecture Diagram of WriteRight

In the context of WriteRight, two layers have been identified: the client layer and the server layer based on the DIP model where the flutter application is the client and the DIP model is the server. The DIP model is piped and filtered internally to enable a smooth and efficient processing of images received from the app for scoring. This pipeline just ensures that data is well managed within the DIP model with much ease. It takes feedback of the DIP model, using which allows the user to monitor the changes in their handwriting.User Profiles and performances are retrieved from Firebase.

3.6.1 Decomposition Description

3.6.1.1 Module Decomposition

3.6.1.1.1 Deployment Diagram

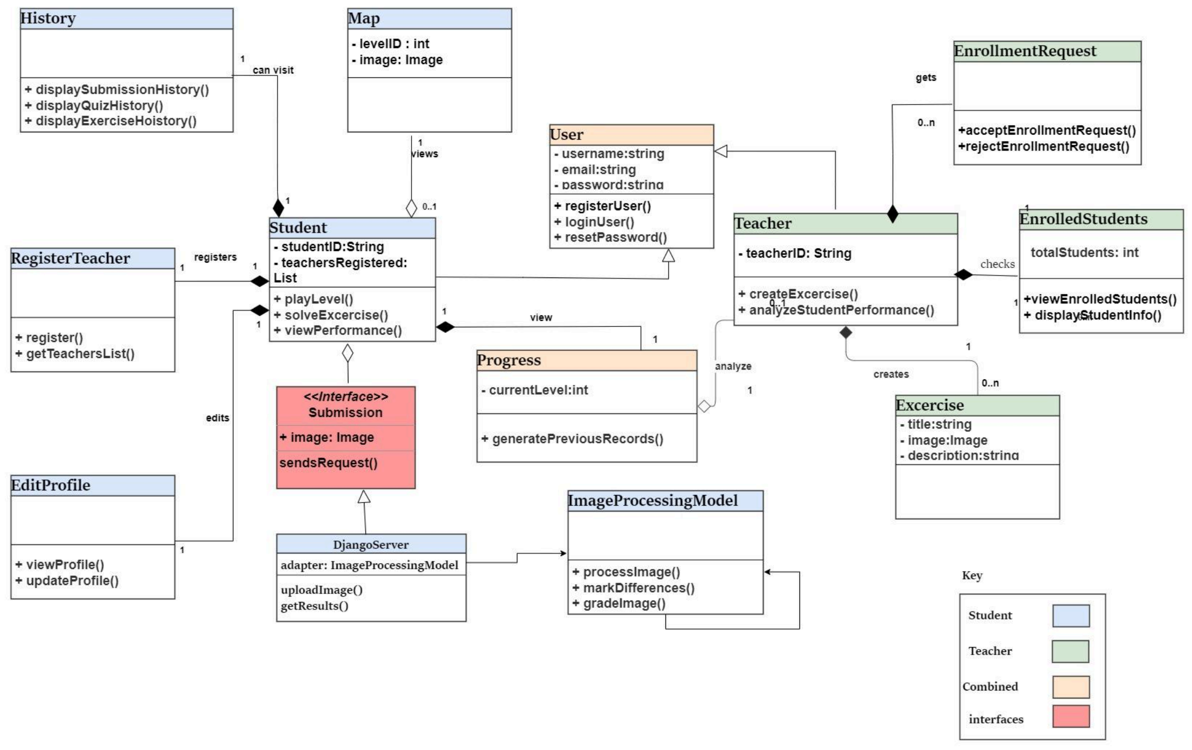


Fig 5: Detailed Class Diagram of WriteRight

The WriteRight class diagram captures essential components and interactions within the system. Users differentiated as teachers and students, engage in activities such as assignment management, enrollment handling, and participating in gamified learning levels. The system seamlessly integrates functionalities such as assignment creation, submission processing, and level progression. This modular structure facilitates efficient data management, user-specific operations, and an engaging learning experience within the WriteRight application.

3.6.1.1.2 Overview of Modules

An overview of the modules mentioned in Fig 5 is given below:

1. **User:** Represents students and teachers, allowing registration and role-based feature access.
2. **Student:** Caters to individuals seeking personalized handwriting improvement.
3. **Submission:** Facilitates the submission and analysis of handwritten text images.
4. **Teacher:** Serves educators, enabling assignment management and student progress monitoring.
5. **Exercise:** Manages exercises created by teachers for students or classes.

6. **Progress:** Tracks and displays student's performance over time.
7. **Image Processing Model:** Employs algorithms to analyze and compare handwritten text images.
8. **Edit Profile:** Allows users to modify and update their profile information.
9. **Register Teacher:** Enables students to request and register with specific teachers.
10. **Map**: Provides a visual representation, possibly for gamification levels or user locations.
11. **DjangoServer:** The Django Model accepts requests, seamlessly forwarding them to the DIP Model for further processing and analysis of handwritten text images.
12. **EnrolledStudents:** Teachers can view all enrolled students in their class
13. **EnrollmentRequests:** Teachers can reject or accept students' received enrollment requests.

3.6.1.1.3 Server Modules

The functionalities on the server side for the WriteRight app are as follows:

1. **Authentication Module:** Handles user authentication processes.
2. **API Server:** Serves as the gateway for licensed users to access the application's functionality.
3. **Database Management Module:** Manages data stored in the Firebase database.
4. **Django Framework**: Used to develop the server-side application.
5. **Optical Character Recognition(OCR)**: Extracts text from images for comparison.
6. **Structural Similarity Index Matrix(SSIM):** Measures the similarity between image regions and marks the areas for improvement.

3.6.1.2 Process Decomposition

1. Use Case Diagram

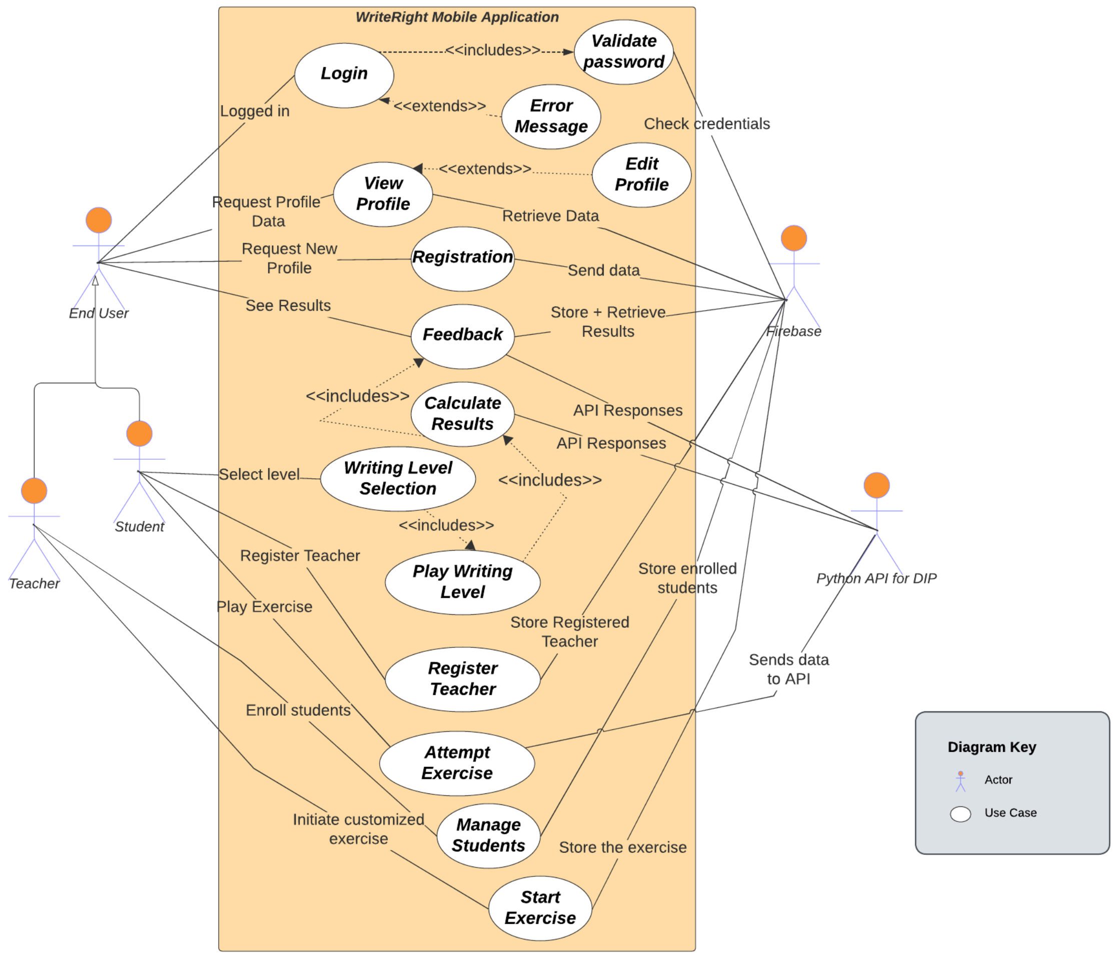


Fig 6: Use Case Diagram of the WriteRight Application

The WriteRight use case diagram shows how users log in, sign up, and handle their profiles. Users also play gamified levels with feedback processed by the DIP model. Managing enrollments is easy for users, while teachers take care of exercise and student requests. The DIP model helps with image processing, making feedback better, and highlighting WriteRight's user-friendly and educational features.

2. Sequence Diagram

● Student Module

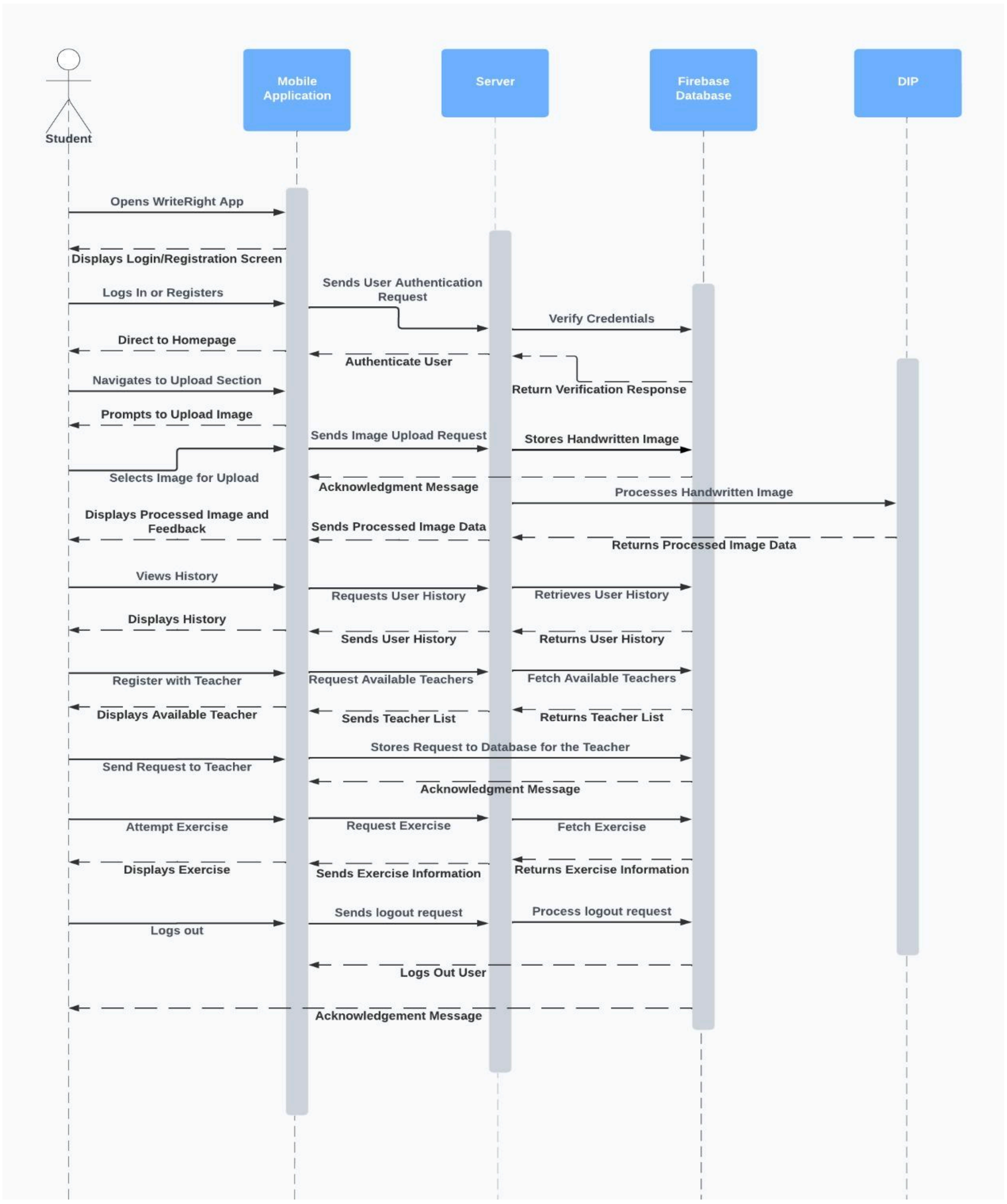


Fig 7: Sequence Diagram of Student Module for WriteRight

The WriteRight app begins with the student logging in or registering, leading to server authentication. Once authenticated, students can upload handwritten images, triggering an upload request to the server. The server processes the image and returns the data to the app, displaying the processed image and feedback. Students can access their image history, view available teachers, attempt exercise, or log out. Logging out prompts a request to the server for logout, ensuring secure and efficient user interactions, including image uploads, history viewing, and interactions with teachers and exercises.

i. Login

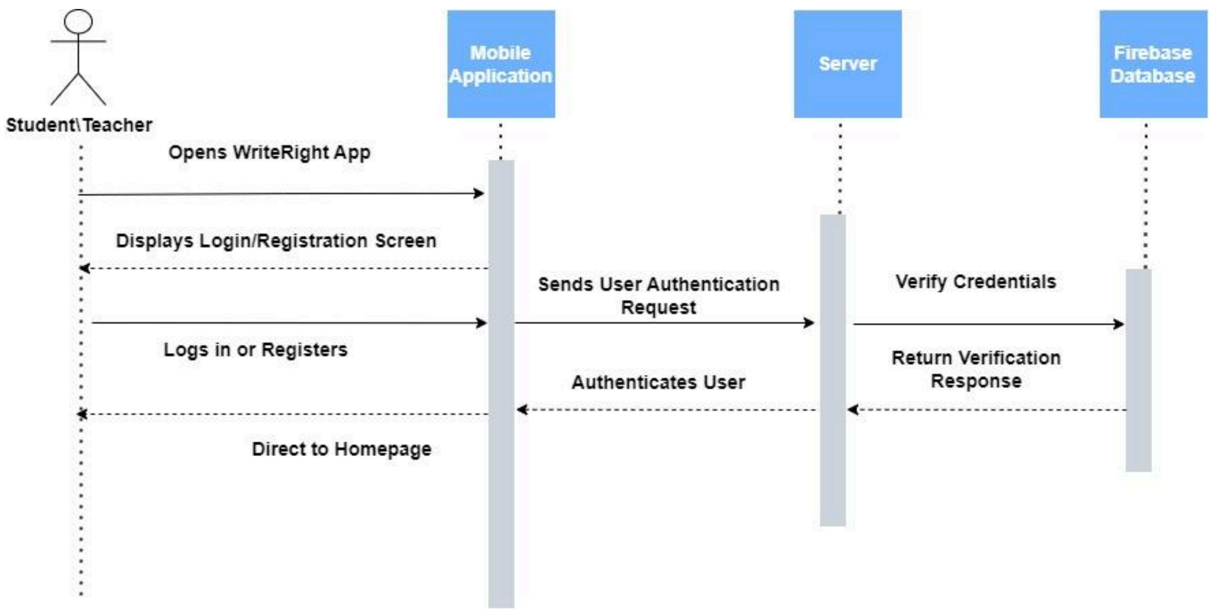


Fig 8: Sequence Diagram of Login for Student Module

The diagram starts with the user opening the app, which then displays a login or registration screen. The user enters their credentials and the app sends them to Firebase for authentication. The Firebase checks the user's credentials and if they are valid, the server sends an authentication confirmation back to the app. The app then directs the user to the Home Screen.

ii. Upload Image

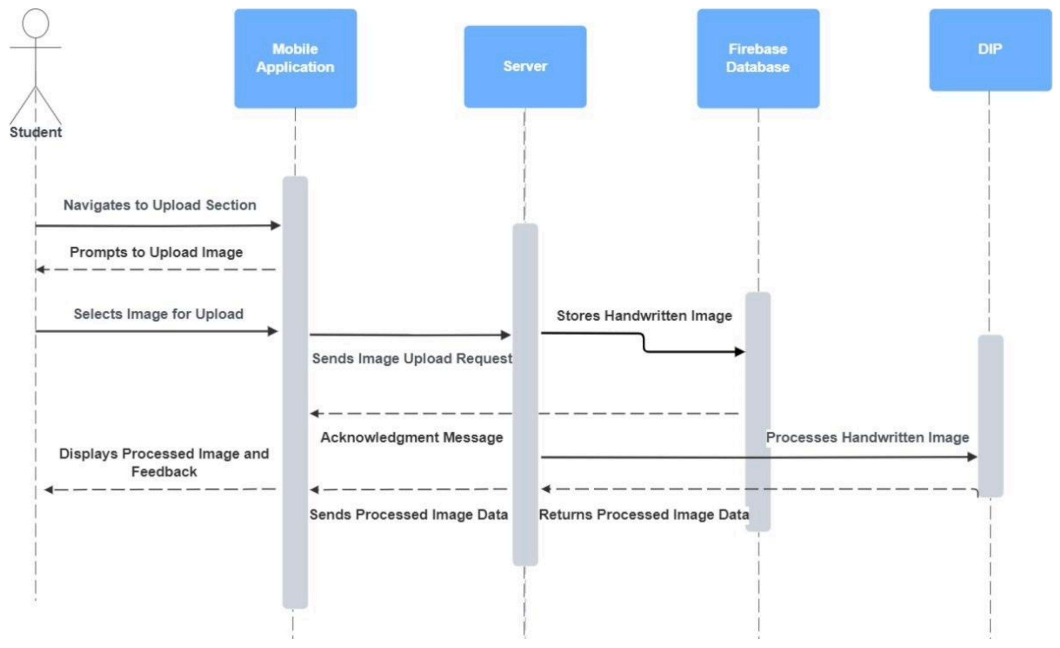

Student
Mobile Application
Server
Firebase Database
DIP
Navigates to Upload Section
Prompts to Upload Image
Selects Image for Upload
Sends Image Upload Request
Stores Handwritten Image
Acknowledgment Message
Processes Handwritten Image
Displays Processed Image and Feedback
Sends Processed Image Data
Returns Processed Image Data

Fig 9: Sequence Diagram to Upload Image for Student Module

The Above diagram shows the flow for uploading an image in writeRight. The student will navigate to a Pacific level in the game where he/she will upload the image. The image will be stored in the database and it will be sent to the DIP model for scoring as well. The model will send back scores and feedback which will be displayed to the user.

iii. View history

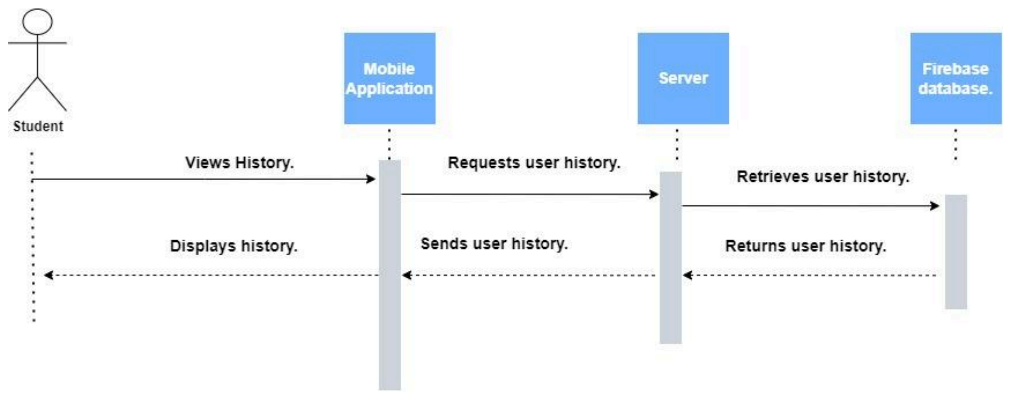


Fig 10: Sequence Diagram of View History for Student Module

Students can review their learning journey using the "View History" screen. Using Firebase, the app retrieves and displays a comprehensive history, including completed levels and performance metrics. This feature encourages self-assessment and enhances the user experience by providing valuable insights into the user's handwriting improvement journey.

iv. Enroll With Teacher

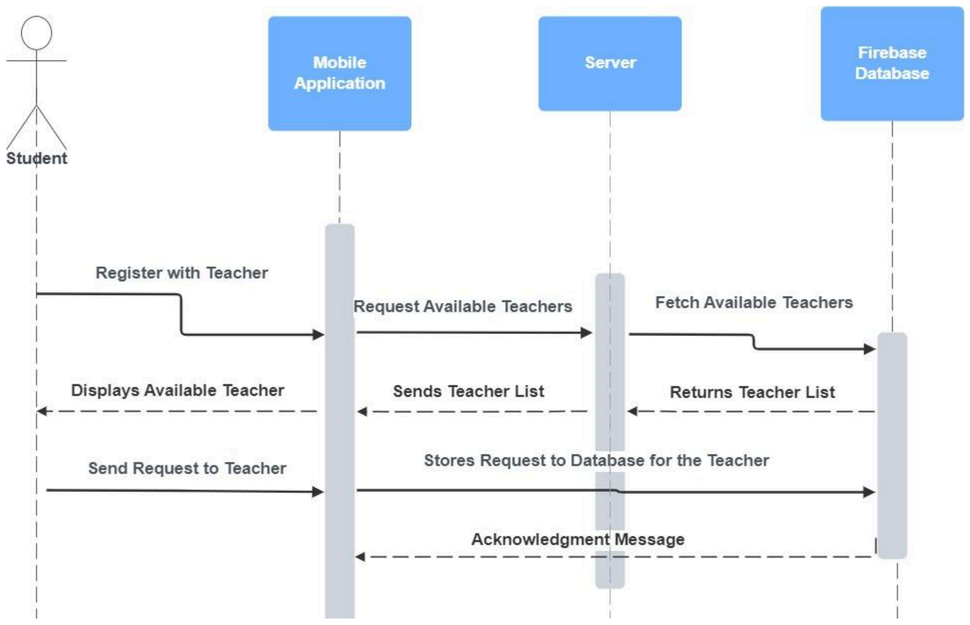


Fig 11: Sequence Diagram of Register with Teacher for Student Module

To register with a teacher, students go to the ‘Register Teacher’ screen on the application. The application sends a request to Firebase to fetch all available teachers. Firebase processes this request and returns a list of all available teachers, which is then displayed to the student on the ‘Register Teacher’ screen. Students can then send requests to register with a specific teacher. The student’s request will be stored in Firebase. This process allows students to easily find and register with a suitable teacher.

v. Attempt exercise:

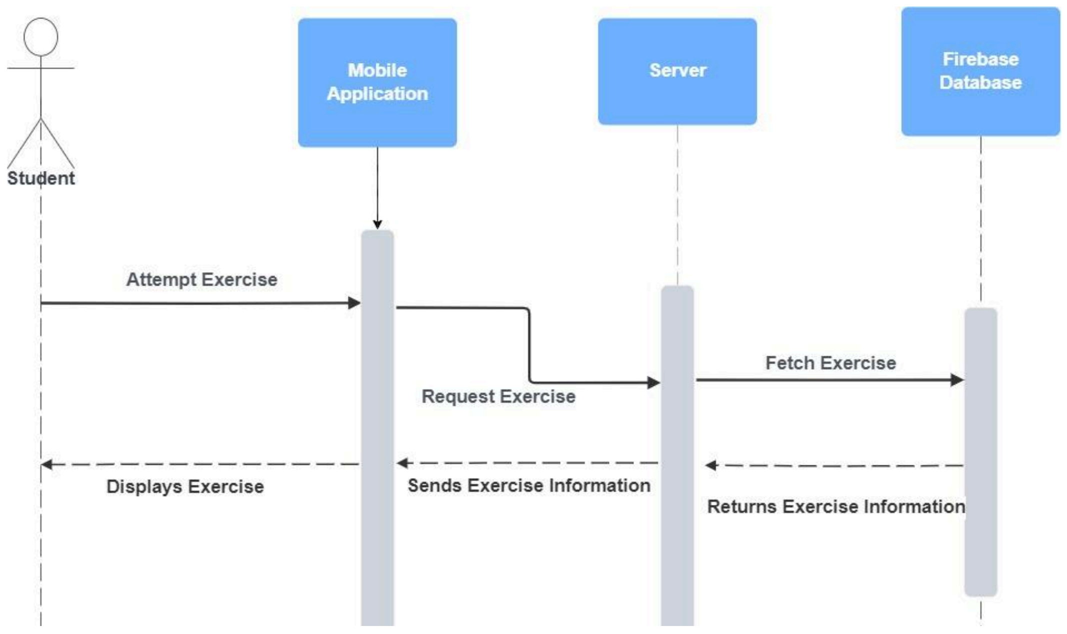


Fig 12: Sequence Diagram of Attempt Exercise for Student Module

To take an exercise, students will go to the exercise screen in the application. A request is then sent to Firebase to check for any exercises that are available or pending. Firebase responds with a list of available exercises. Students can then select ‘attempt’ to start taking an exercise.

vi. Log out

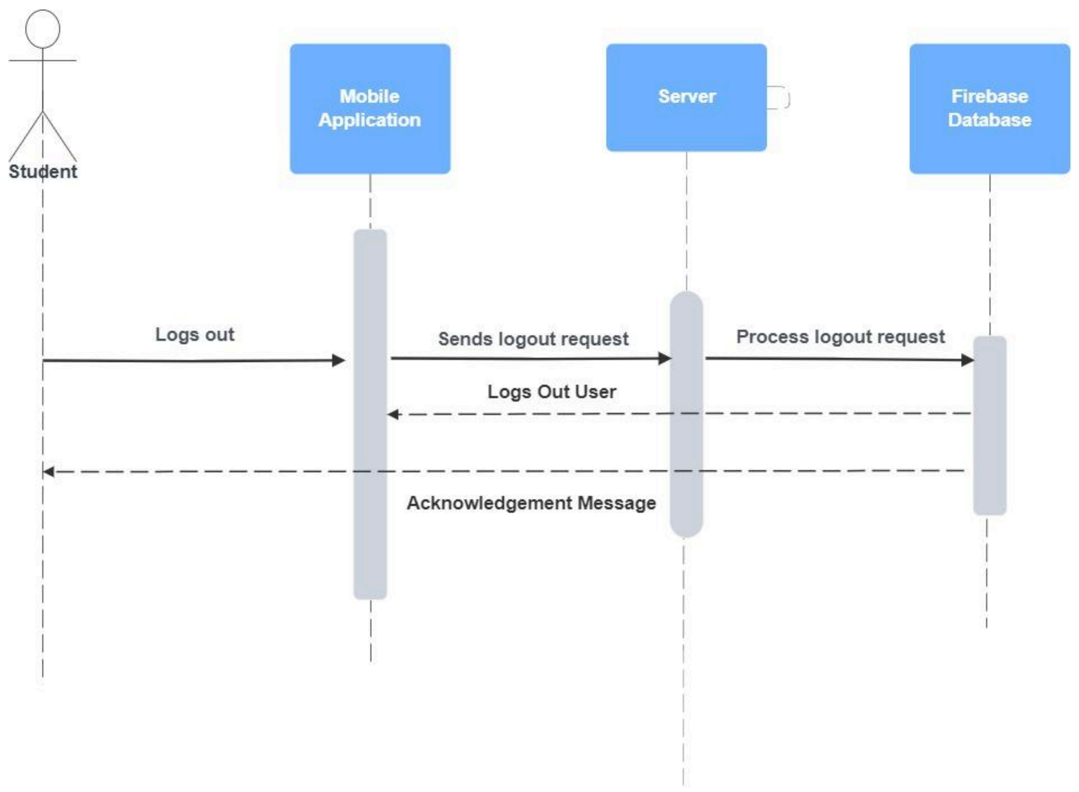


Fig 13: Sequence Diagram for Logout of Student Module

Users have the option to log out of the application by simply clicking on the 'logout' button. Once this button is clicked, a request is sent to Firebase to initiate the logout process. Firebase, in turn, handles this request and effectively logs the user out of the application.

- **Teacher Module**

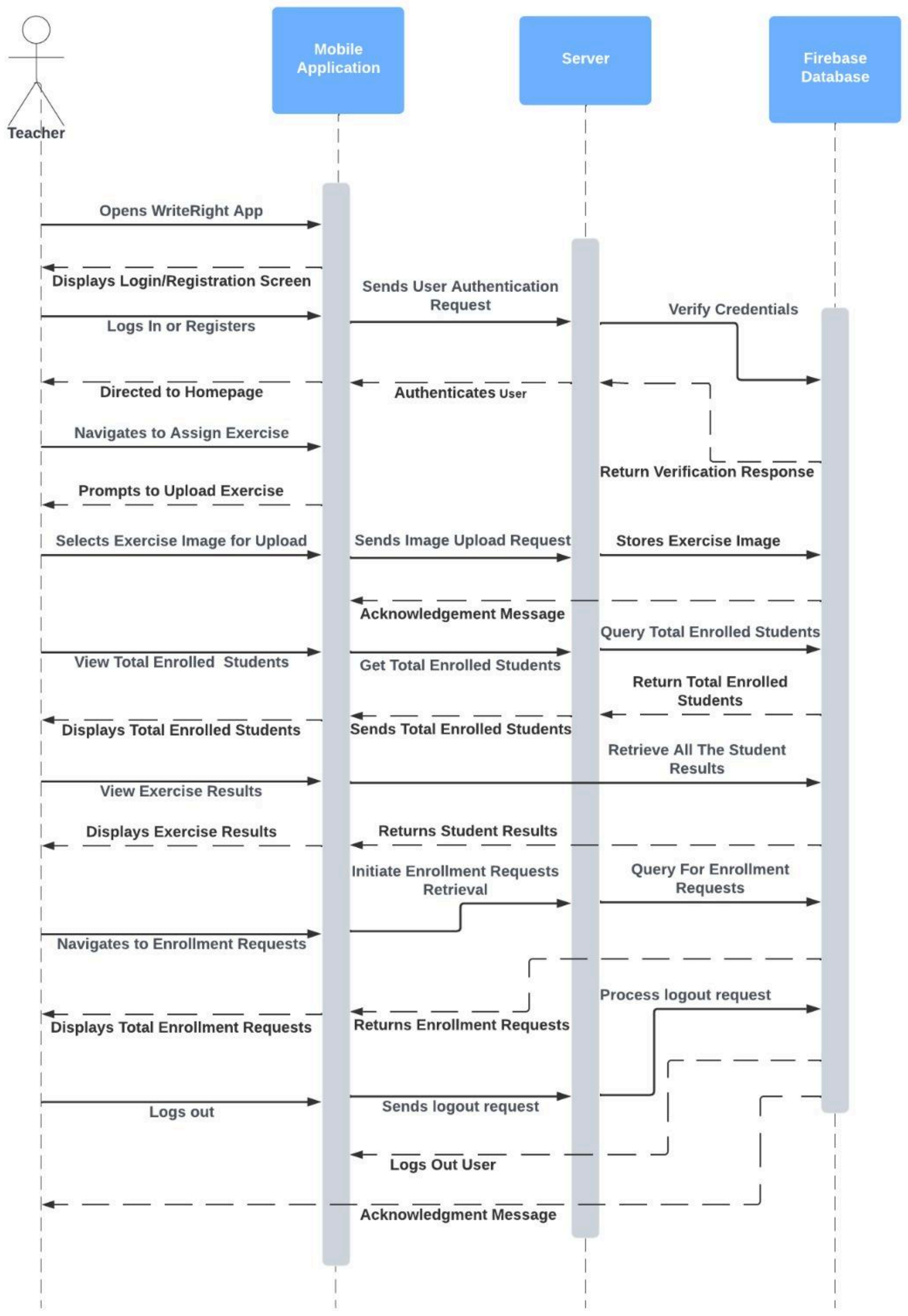

Teacher
Mobile Application
Server
Firebase Database
Opens WriteRight App
Displays Login/Registration Screen
Logs In or Registers
Sends User Authentication Request
Verify Credentials
Directed to Homepage
Authenticates User
Return Verification Response
Navigates to Assign Exercise
Prompts to Upload Exercise
Selects Exercise Image for Upload
Sends Image Upload Request
Stores Exercise Image
Acknowledgement Message
Query Total Enrolled Students
View Total Enrolled Students
Get Total Enrolled Students
Return Total Enrolled Students
Displays Total Enrolled Students
Sends Total Enrolled Students
Retrieve All The Student Results
View Exercise Results
Displays Exercise Results
Returns Student Results
Initiate Enrollment Requests Retrieval
Query For Enrollment Requests
Navigates to Enrollment Requests
Process logout request
Displays Total Enrollment Requests
Returns Enrollment Requests
Logs out
Sends logout request
Logs Out User
Acknowledgment Message

Fig 14: Sequence Diagram of Teacher Module for WriteRight

The WriteRight app begins with teachers logging in or registering on the mobile platform, leading them to a dashboard. From the dashboard, teachers can manage various tasks, including assigning exercises, viewing enrolled students, and handling enrollment requests. To assign an exercise, teachers upload an image, which the app processes and stores in the database. After confirming the extracted text, teachers send the processed image to the server along with a request for the total enrolled students. The server retrieves and returns the count, displayed on the app. To handle enrollment requests, teachers query the server, which provides the requests for their class. When done, teachers can log out, triggering a server logout and closing the app session.

i. Login

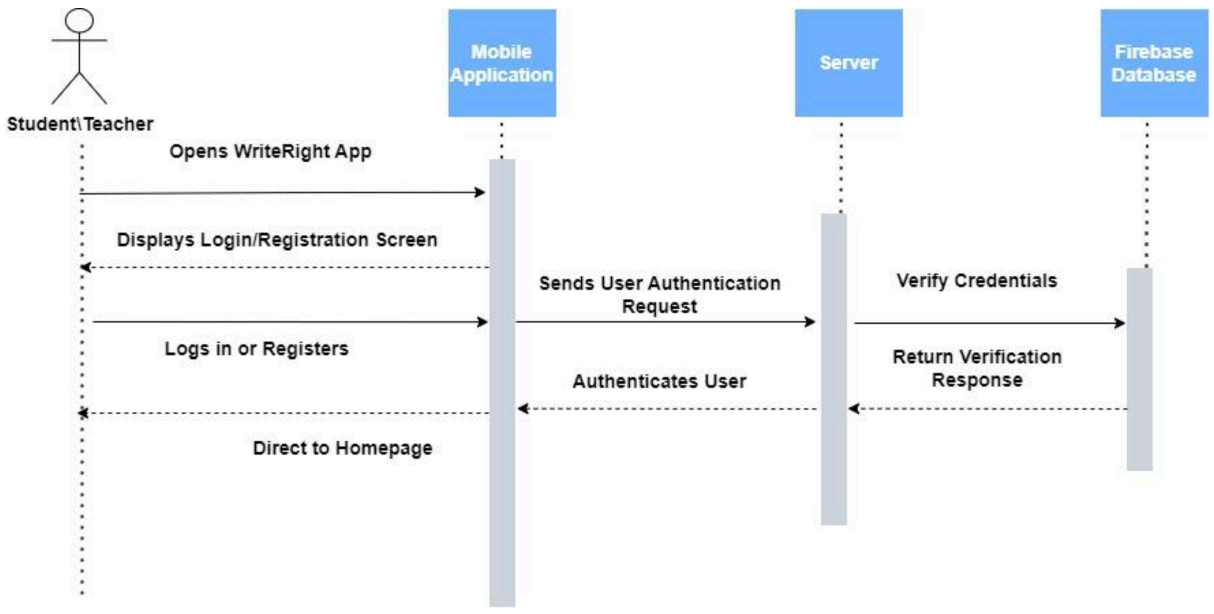


Fig 15: Sequence Diagram of Login for Teacher Module

The diagram starts with the user opening the app, which then displays a login or registration screen. The user enters their credentials and the app sends them to Firebase for authentication. When the user's credentials are verified by the Firebase database, the server authenticates the app and delivers it back to the app. After that, the app redirects the user to the main dashboard.

ii. Assign Exercise

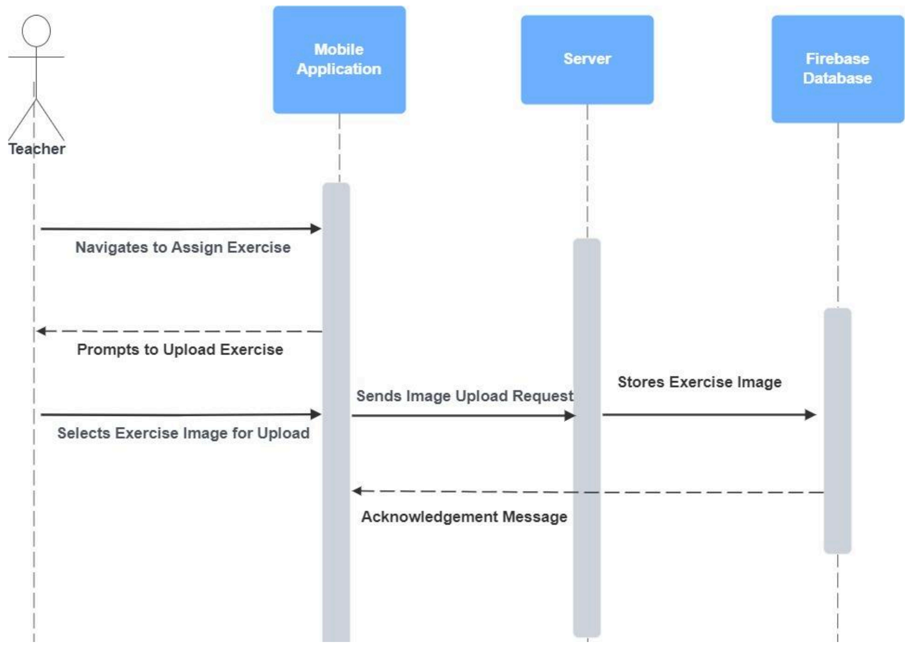

Teacher
Mobile Application
Server
Firebase Database
Navigates to Assign Exercise
Prompts to Upload Exercise
Selects Exercise Image for Upload
Sends Image Upload Request
Stores Exercise Image
Acknowledgement Message

Fig 16: Sequence Diagram of Assign Exercise for Teacher Module

The teacher is navigated to the "Assign Exercise" section of the mobile application. There they can upload an exercise, and upon uploading/drawing the desired image, the app sends an upload request to the server. The server receives the image and stores it in the Firebase database.

iii. Enrolled Students

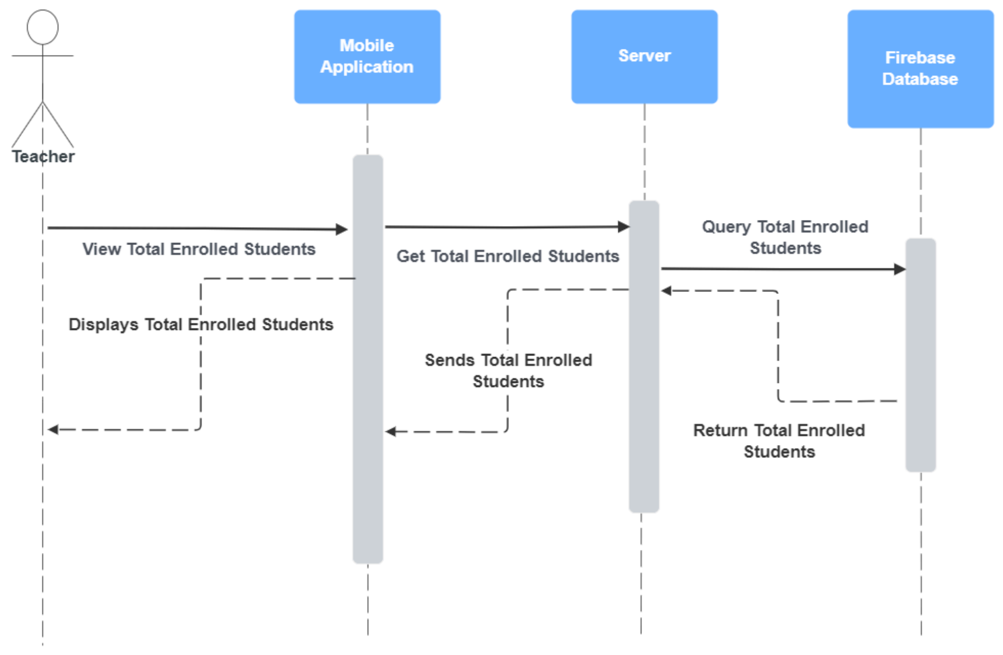

Teacher
Mobile Application
Server
Firebase Database
View Total Enrolled Students
Get Total Enrolled Students
Query Total Enrolled Students
Displays Total Enrolled Students
Sends Total Enrolled Students
Return Total Enrolled Students

Fig 17: Sequence Diagram of Enrolled Students for Teacher Module

This diagram shows a teacher tracking the number of students enrolled in a class. On moving to the screen where the list of students is shown, a request is made to the server to fetch all the enrolled students with the teacher. The data is fetched from firebase and returned, which is then displayed to the teacher.

iv. Exercise Results

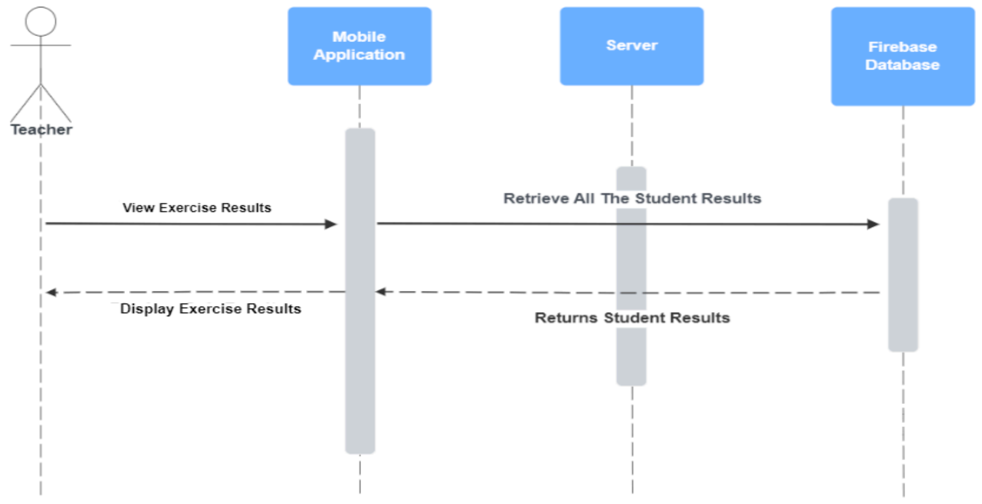


Fig 18: Sequence Diagram of Exercise Results for Teacher Module

This diagram covers the sequence of a teacher viewing exercise results students have submitted. The teacher uses the frontend to send a request to the server to view exercise results. The server then retrieves the results from the database and sends them back to the app. The app displays the results to the teacher.

v. Enrollment Requests

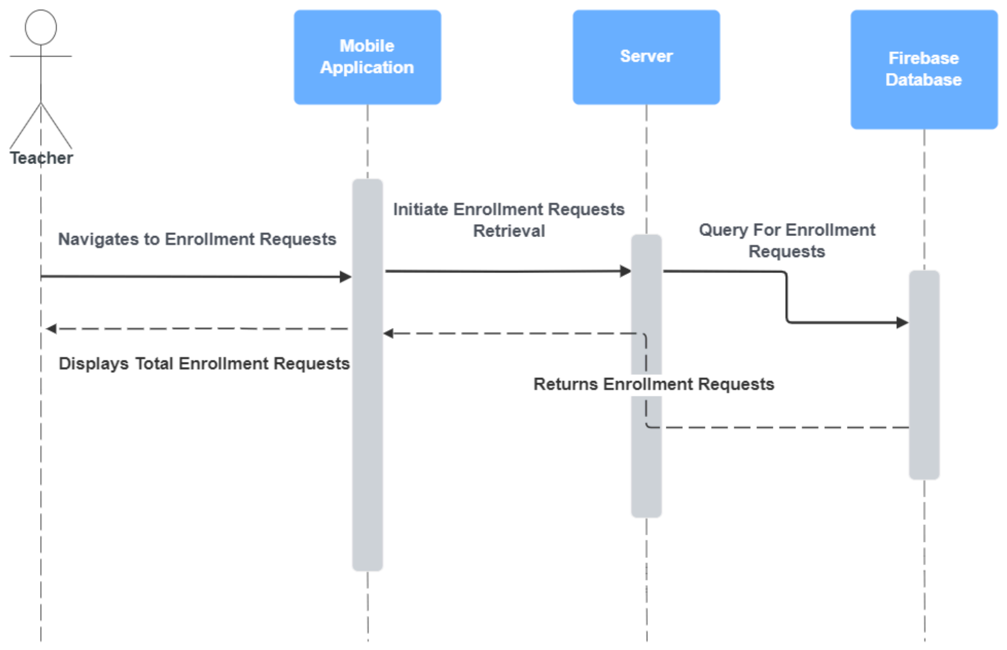

Teacher
Mobile Application
Server
Firebase Database
Navigates to Enrollment Requests
Initiate Enrollment Requests Retrieval
Query For Enrollment Requests
Returns Enrollment Requests
Displays Total Enrollment Requests

Fig 19: Sequence Diagram of Enrollment Requests for Teacher Module

The process starts when the teacher opens the app and navigates to the "Enroll Student" section. They can view the pending student requests, and accept or reject the requests. The app then sends a request to the server to enroll the student.

vi. Logout

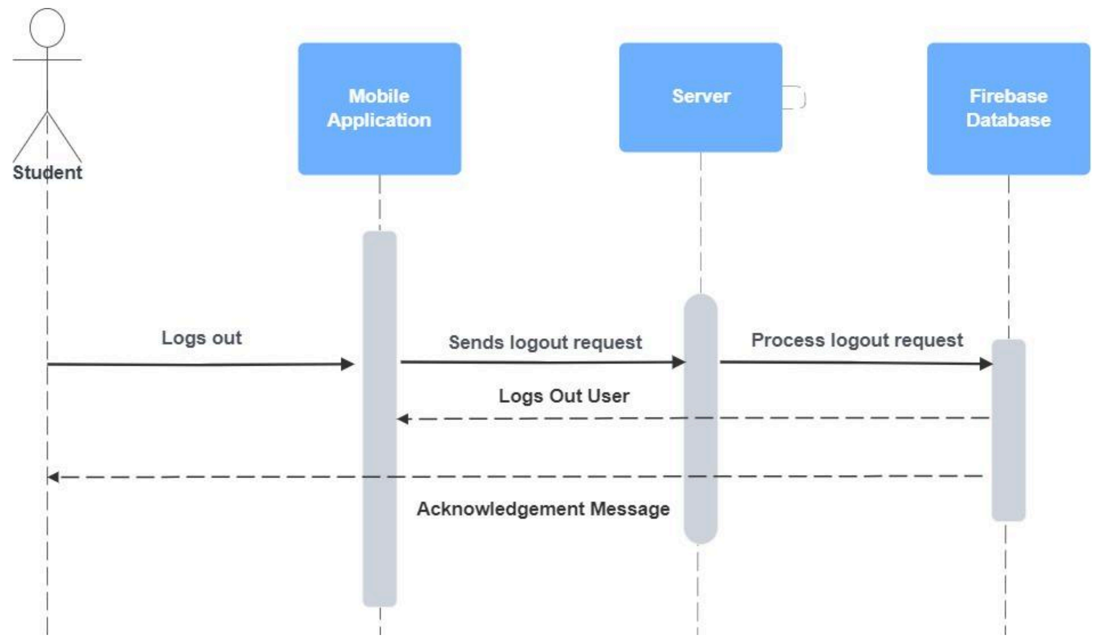


Fig 20: Sequence Diagram of Logout for Teacher Module

Simply clicking on the 'logout' button will allow users access to log out of the application; within seconds, a correspondence will be sent to Firebase in order for it to start the process of logging the user out.Firebase, in turn, handles this request and effectively logs the user out of the application. This means the user will no longer be able to access their account or any protected data until they log in again.

3. Activity Diagram

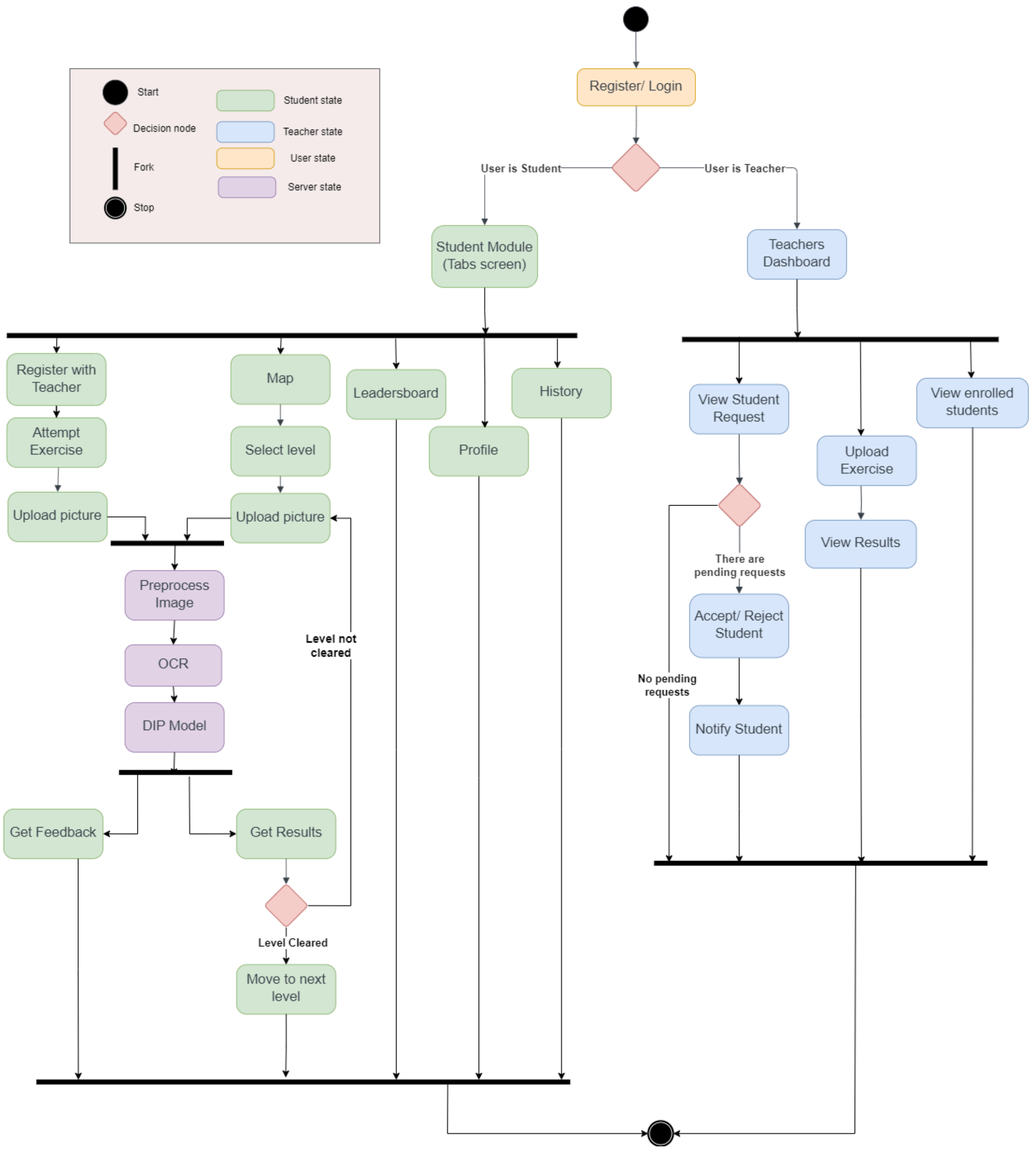


Fig 21: Detailed Activity Diagram of WriteRight

The activity diagram of writeRight shows the basic flow of the application. The app is divided into 2 parts, the teacher module and the student module. The teacher module provides functions like enrolling students and giving exercises. Students can access functionalities like play level, attempt exercise, or register with the available teachers.

i. Login

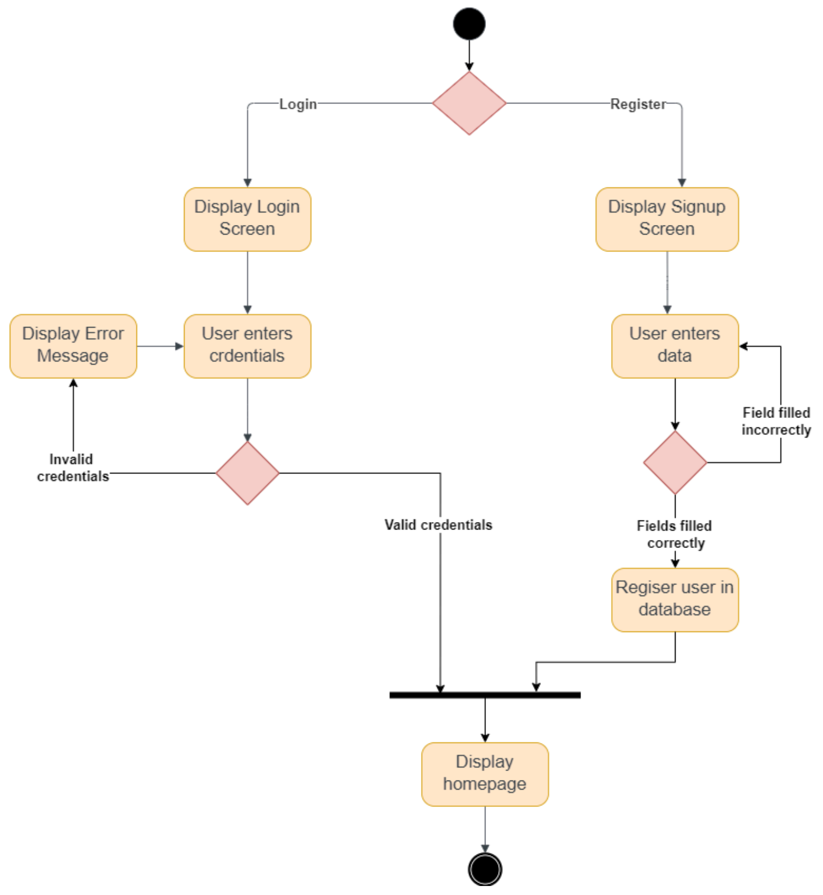


Fig 22: Activity Diagram of Login Flow for WriteRight

This activity diagram shows the flow of authentication in our app. Users can login or register in our app. If a user fails to provide correct credentials, he/she will need to enter credentials again until the right credentials are provided. Similarly, our app will check if all fields are correctly filled at the time of registration. Users will be prompted to fill fields again until they are filled correctly. After all fields are filled correctly, users will be registered.

ii. Play Level

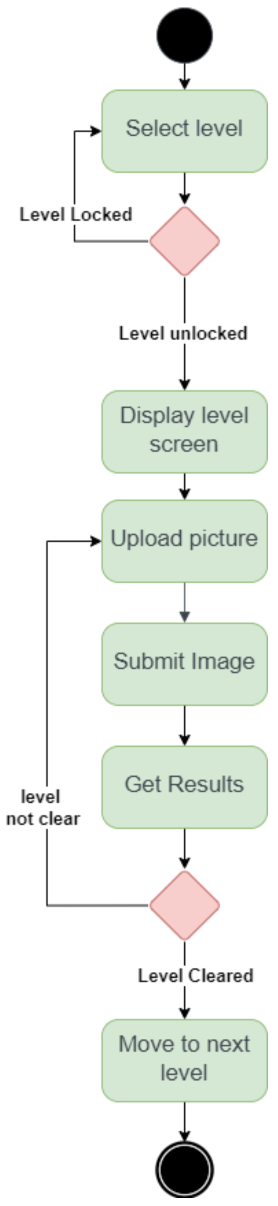


Fig 23: Activity Diagram of Play Level for Student Module

This diagram shows the flow to play a level in our game. User will upload an image and get results. The user cannot move to the next level unless he/she meets a specific threshold of score.

iii. Register With Teacher

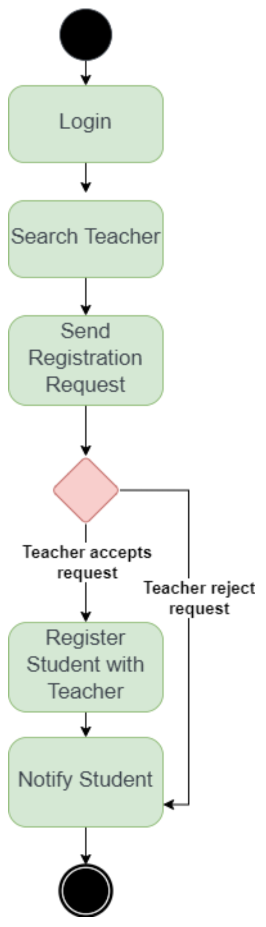


Fig 24: Activity Diagram for Registering With Teacher for Student Module

This diagram shows the flow for a student to register with a teacher. Students can search for a teacher and send enrollment requests to the teacher. Teachers can accept or reject a request. Students will be notified about the status of the request.

iv. Attempt Exercise

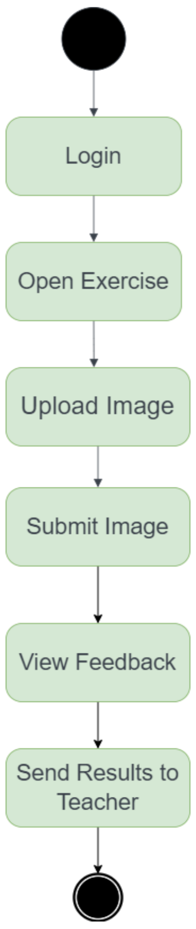


Fig 25: Activity Diagram for Attempting Exercise for Student Module

This diagram shows the flow for a student to attempt an exercise given by the teacher. Students will complete the exercise and upload its image which will be evaluated by our algorithm.

v. Handle Requests

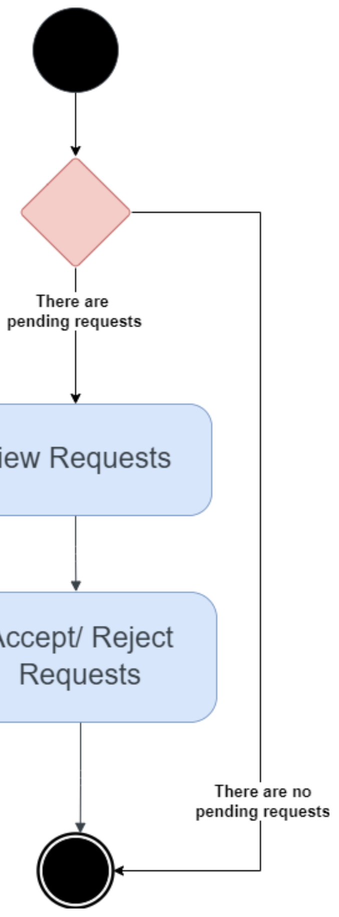


Fig 26: Activity Diagram for Handling Student Enrollment Requests

This diagram shows the flow of the teacher module to enroll students with them. The teacher will check if there are any pending requests. Then he can accept or reject them.

vi. Upload Exercise

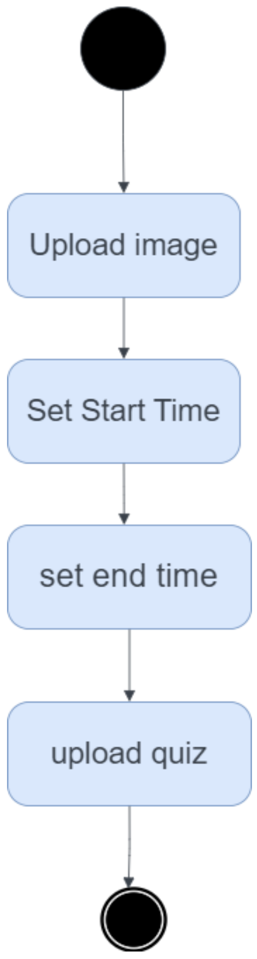


Fig 27: Activity Diagram for Uploading Exercise for Teacher Module

This diagram shows the flow for a teacher to upload an exercise. The teacher will upload the image that students need to copy and then set the start and end times for the exercise. Finally, the teacher will upload the exercise.

4. Package Diagram

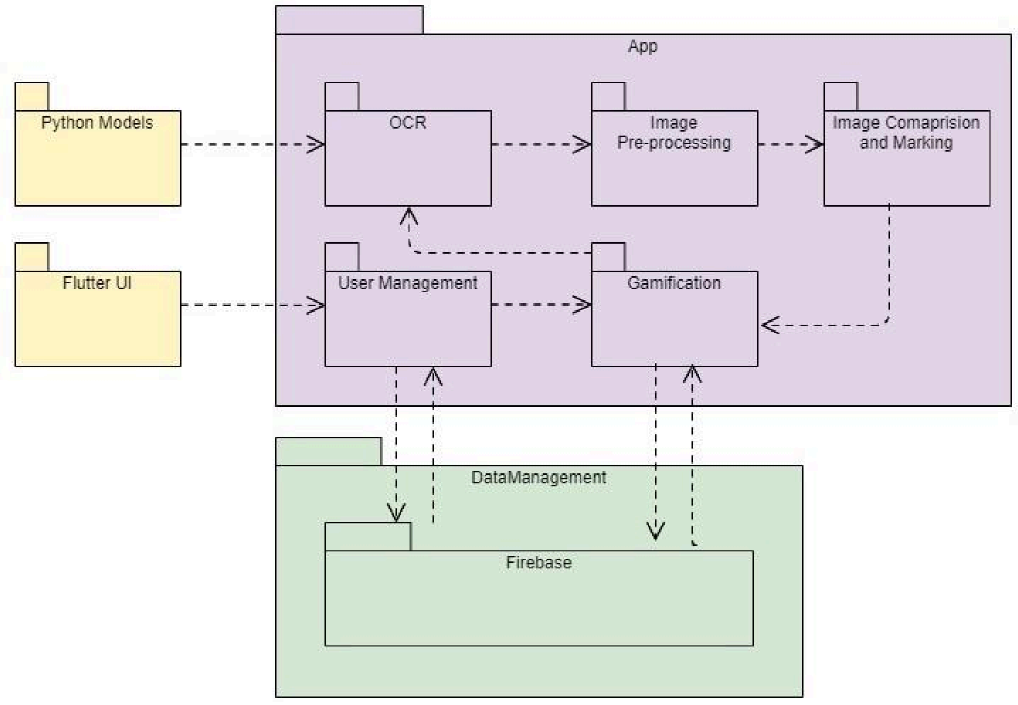


Fig 28: Package Diagram Representing Granular Modules for WriteRight

Here our application is depicted in a comprehensive package diagram. It comprises distinct packages for seamless functionality. The "User Interface" package encapsulates the Flutter UI app, ensuring an intuitive user experience. "Data Management" is facilitated by Firebase, ensuring efficient storage and retrieval of user data. The "Gamification" package enhances user engagement through image preprocessing and scoring. The "OCR Processing" package incorporates a Python model for seamless text extraction from images. The "Image Comparison" package employs a DIP model to score and compare handwritten images, providing users with valuable feedback. Together, these packages create a well-organized and efficient system for a feature-rich and user-friendly experience.

5. Deployment Diagram

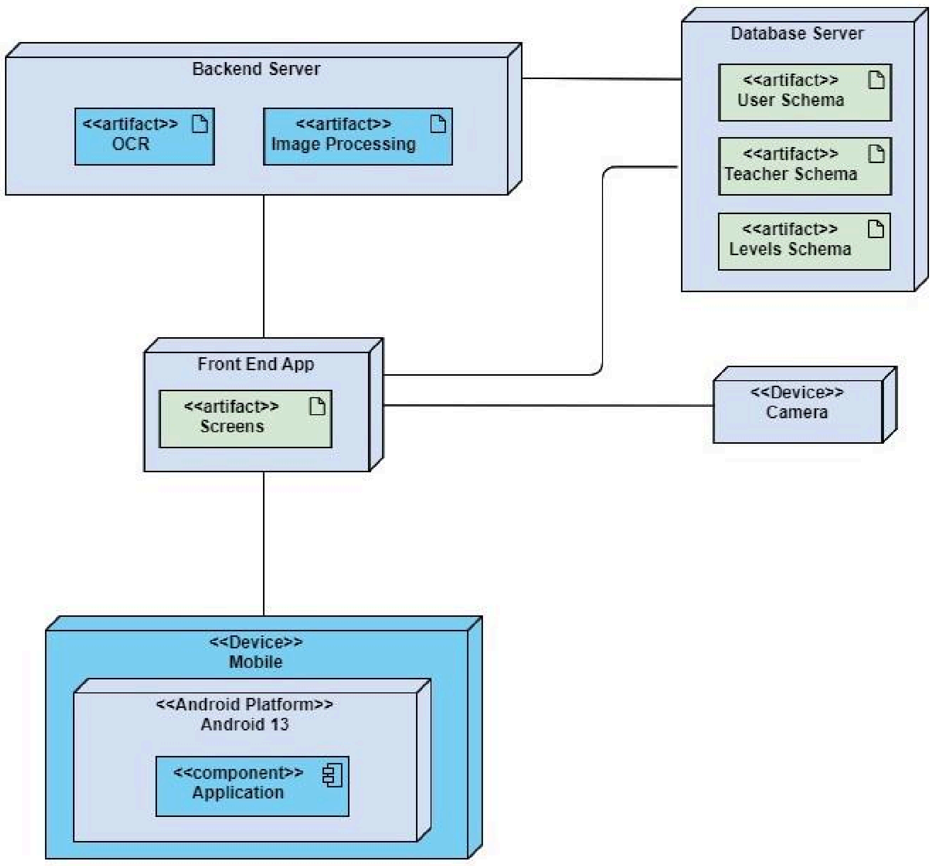

Backend Server
<<artifact>> OCR
<<artifact>> Image Processing
Database Server
<<artifact>> User Schema
<<artifact>> Teacher Schema
<<artifact>> Levels Schema
Front End App
<<artifact>> Screens
<<Device>> Camera
<<Device>> Mobile
<<Android Platform>> Android 13
<<component>> Application

Fig 29: Deployment Diagram Representing Deployment Modules for WriteRight

Within the WriteRight application's deployment architecture, a robust backend server orchestrates critical functions like image processing and OCR. This backend seamlessly communicates with a database server housing user profiles, teacher-student associations, and diverse gamification levels. The mobile application serves as the user interface, allowing users to effortlessly capture images using their device's camera. These images are uploaded, connecting with the backend server for efficient processing. This setup ensures a cohesive and streamlined user experience throughout the application.

3.7 Design Rationale

The design rationale for "WriteRight" centers on 3 design patterns mainly. Each oneof the patterns is picked from all 3 categories - Structural, Creational, and Behavioral design patterns providing all necessary coverage to code reusability, flexibility, and assignment of responsibilities.

3.7.1 Singleton Pattern:

It ensures that there is only a single point of access, while that single point is globally available for all clients. We need strict control over our globally running ML Models on servers, therefore it is mandatory to ensure a single point of access even if available globally. In our case, when the user interacts with the ML Model via Flutter, the server serves as the single access point while the ML models are deployed with global access.

3.7.2 Adapter Pattern:

It ensures incompatible interfaces to collaborate. Since our application is cross-platform (android and iPhone), it must be flexible enough, and the front-end application needs to communicate with the ML Models therefore Adapter pattern is to be implemented. *In our case, we use the Django server as an adapter to make the Flutter interface and ML Scripts compatible to run.*

3.7.3 Observer Pattern:

This pattern lets define a subscription mechanism to notify multiple objects about any events that happen to the object they're observing. Our application is event-driven and the observer pattern inherits event-driven characteristics by default, hence, it becomes the first choice for WriteRight. *In our case, ML scripts are only run when a student uploads the image and the Django server observes the interface for this change.*

.

Chapter 4. Data Design

4.1 Data Description

WriteRight's data revolves around two core facets: user information and handwriting samples. User information includes login credentials, progress data, and achievements, facilitating personalized learning and motivation. Handwriting samples, captured as images, are analyzed using image processing algorithms to generate valuable insights like similarity scores and areas for improvement. This data feedback loop fuels the core gameplay and empowers users to track their progress, while simultaneously enriching the app's ability to adapt and provide increasingly accurate feedback over time.

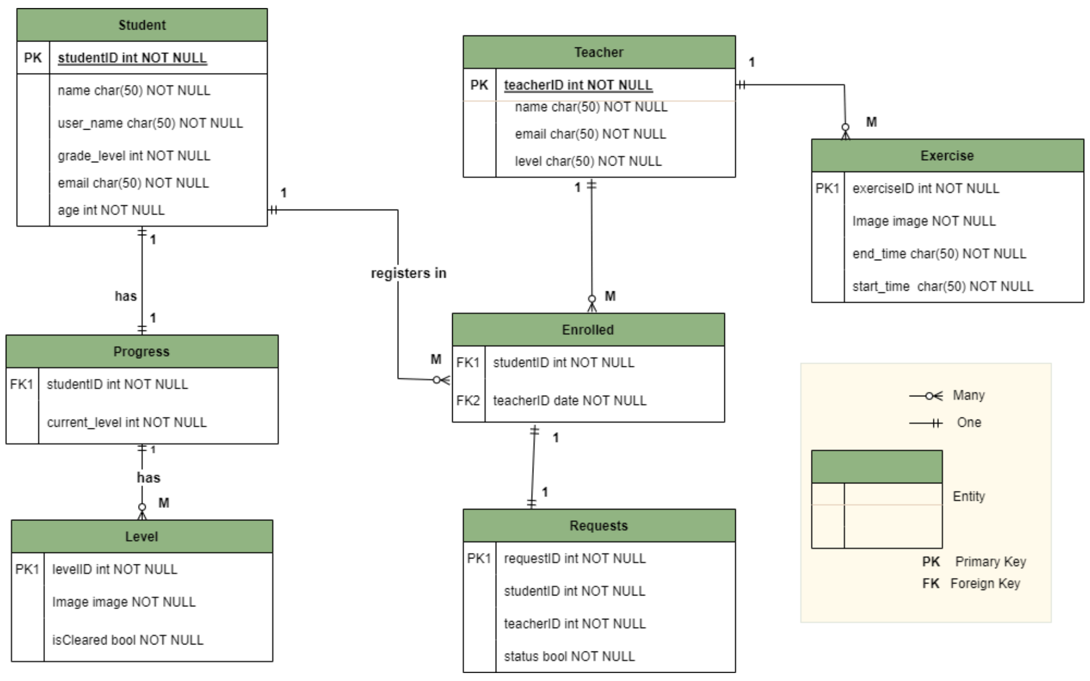


Fig 30: Entity Relation Diagram for the Database Entities in the WriteRight

4.2 Data Dictionary

We will use the Firebase real-time database for our app. The database of our app will consist of the following collection:

Collection	Key	Document	Subcollection

Students	studentId	Name (string)
		username(string)
		age(int)
		email(string)
		score(int)	

Teachers	TeacherId	Name(string)
		level(string)
		email(string)	

Enrolled	enrollId	studentId(string)
		TeacherId(string)	

Progress	ProgressId	studentId(string)	levelId(string)
			Score(int)
			stars(int)

Exercise	ExerciseId	ImageUrl(string)
		startTime(string)
		endTime(String)	

Levels	LevelId	ImageUrl(string)
		levelNumber(int)
		isCleared(bool)	

Request	RequestId	studentId(string)
		teacherId(string)
		status(string)	

Attempts	AttemptId	studentId(string)
		teacherId(string
		ImageUrl(string)	

4.2.1.1 Server-side.

1. extract_frames(fps, vid.video).images => this function is a part of the **Frame_Extraction** class and is used to divide the video into video based on fps parameter.
2. get_features(frames.images).string => this function is a part of the **Feature_Extraction** class and is used to extract the required features from the frames.

3. recognize _face(featuredata.frames).video=> this function is a part of the **Facial Recognition** and it feeds the feature data into the model which processes it and classifies the face.

4. analyze_activity(featuredata.frames).video=>this function will analyze activity such violence detection, number of people etc.

5. recognize_number_plate(featuredata.frames).video=> this function will detect cars and will read their number plates at entrance and exit.

6. notify(processedvideo.video).string, video => this function takes in the processed video and returns the relevant notification and resultant video.

7. rcvvideo().video => this function is a part of the **rest_api** class and is used to receive video from the application.

8. sendresults(notification, processedvideo). string => this function is part of the **rest_api** class and is used to transmit the results.

4.2.1.2 Application side.

1. main().void => this function is used to start the web application.
2. get_video(address.string).boolean => this function is a part of the Video_Input class which is basically tasked with taking in the user's input and returns true/false if video input is successful or not.
3. send_error_notification().string => the function returns an error notification in case video input fails.
4. rcv_notifcations().string, video => this function is a part of the Application class and it is tasked with receiving the results from the server.

4.3 Component Diagram

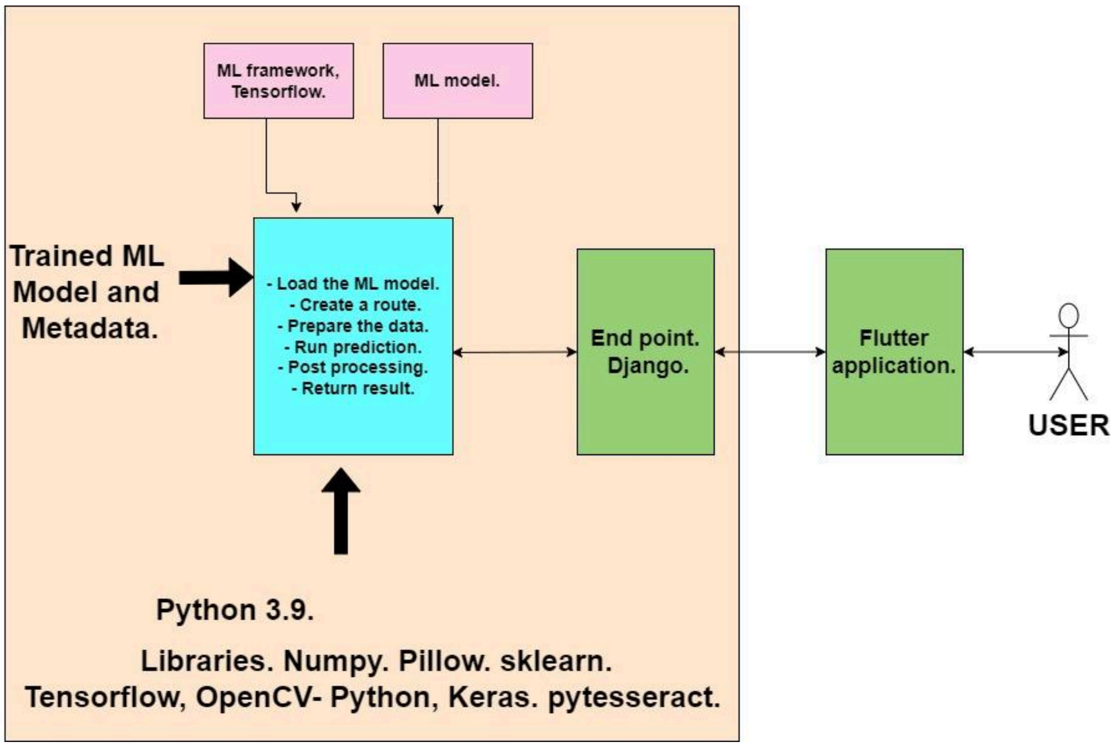


Fig 31: Overview of Component Design for WriteRight

The above diagram lays out the major components of our project. Machine learning workflow consists of Python libraries used to create an ML model for an application.
Data is collected from various sources and then preprocessed to make it suitable for training the ML model. Then this preprocessed data is used to train the ML model. The trained model is deployed to production where it can be used to make predictions.

The Django app is being hosted on a web server for user access via web browsers. A connection is established between the Flutter app and the Django app's API to utilize the ML model's predictions.

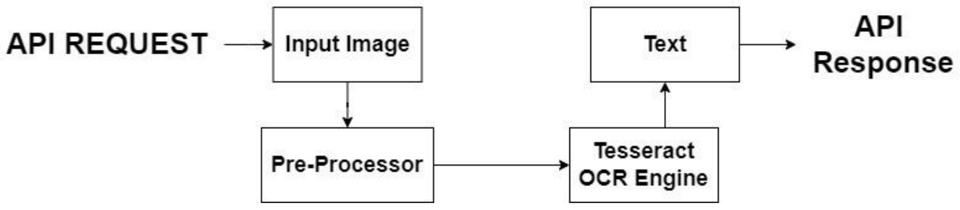


Fig 32: Working of OCR for DIP Model of WriteRight

The above diagram shows the working of our OCR for character recognition. Firstly, the image will be preprocessed and then the Tesseract OCR engine will extract text and provide it as API response.

4.3.1 Mobile Application Module

This module performs all the front-end tasks for WriteRight application System which includes student Registration, solving exercises, view leaderboard and solve assignments provided by teacher. This module provides the base for successful working of the server module.

4.3.1.1 Staff/Admin Side

4.3.1.1.1 Student Registration

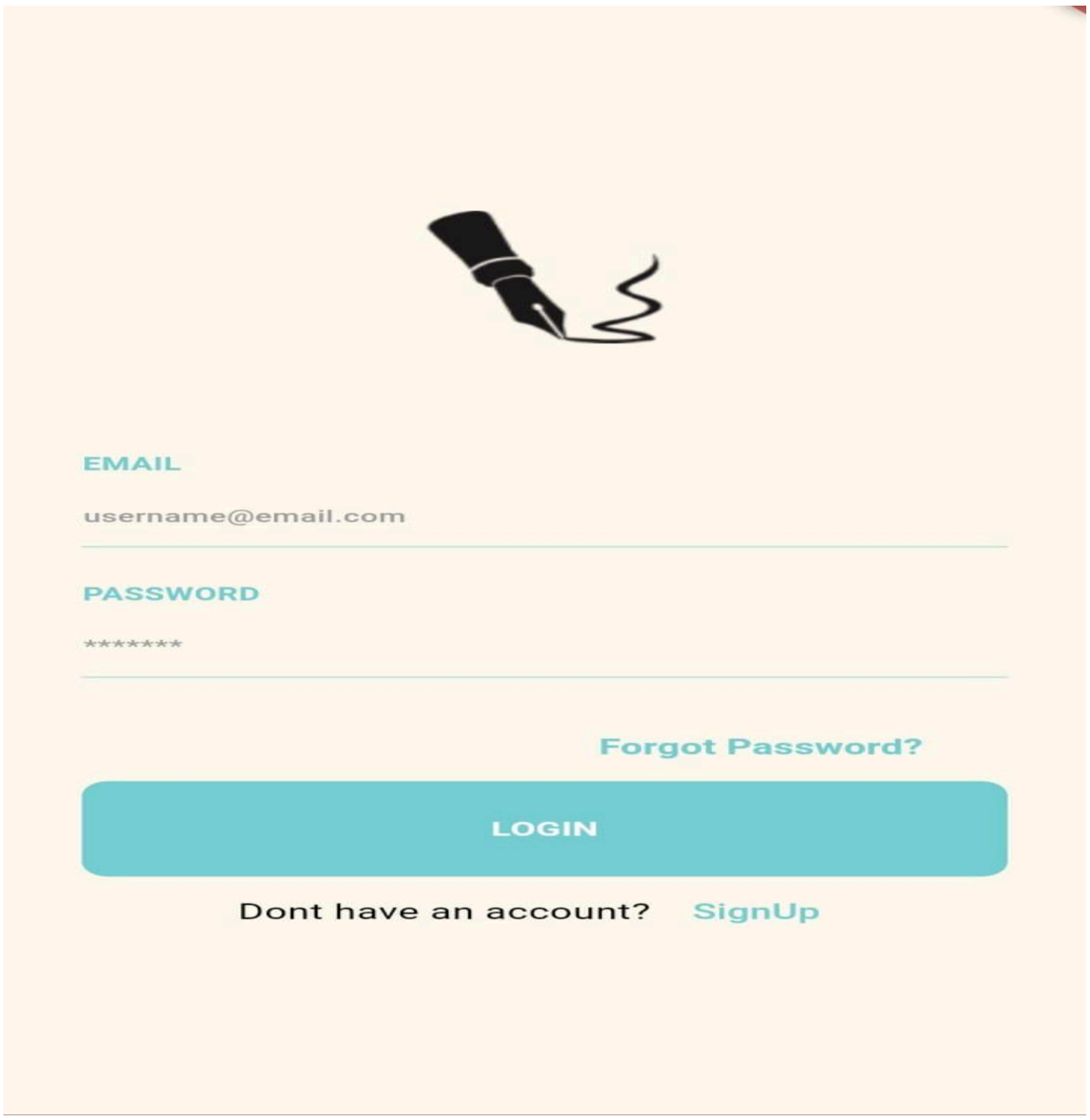


Fig 33: Login Page/Screen of WriteRight

Identification	Name. User Registration/Login Location. Mobile Application Module
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document. **Registration Requirement** The system shall be able to register new users.. **Description** This feature enables the system to register new users and depending on their role whether teacher or student is then further moved to their respective dashboard.
Function	This component of system interfaces with registration of students.
Subordinates	No subordinates
Dependencies	This component is independent module and runs in parallel for registration of new users.
Interfaces	No interfaces as registration process don't need to interface with any other module.
Resources	**Hardware.** Mobile. **Software.** Web Browser (Chrome, Firefox, IE), database for storing student`s images.

Processing	Images will further process after registration for attendance, sleep and smoking detection.
Data	This component uses students` data for fetching registration which includes name, registration number and images.

4.3.1.1.2 View Level Map

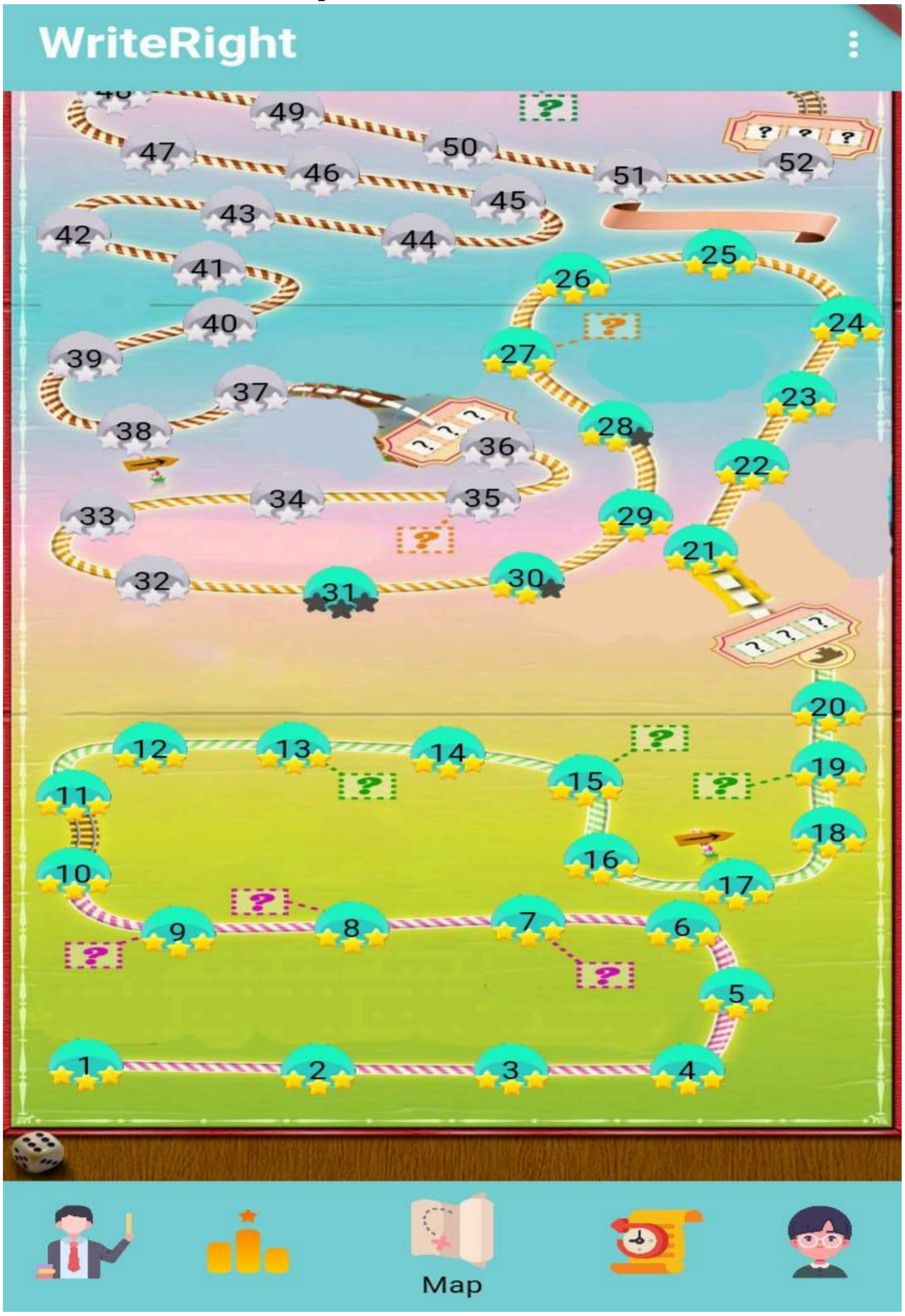


Fig 34: Level selection screen of WriteRight

Identification	View Level Map

Type	User Interface Component
Purpose	This component fulfills the following requirement from the Software Requirements Specification Document: The system shall allow users to view a map that displays the levels available in the gamification feature. **Description** This feature enables users to view a map that displays the levels available in the gamification feature.Students can select level to write and check that particular character. It ensures that users are logged in before accessing this feature.
Function	Display a map showing the available levels in the gamification feature.
Subordinates	None.
Dependencies	User authentication system to verify login status. Gamification data repository for level information.
Interfaces	● Input: User login status, user interaction with the gamification section. ● Output: Map displaying available levels for selection. Also displays which level is completed.
Resources	● User Interface: Frontend component to display the level map. ● Backend Server: To fetch level data which is a sample image associated with the level from the gamification feature.
Processing	● Verify user login status. ● Navigate to the gamification section upon user request. ● Fetch and display the level map from the backend server. ● Prompt the user to log in if not already authenticated .

Data	● Input Data: User login credentials, user actions. ● Output Data: Level map, login prompt (if necessary)

4.3.1.1.3 Play Level/upload image

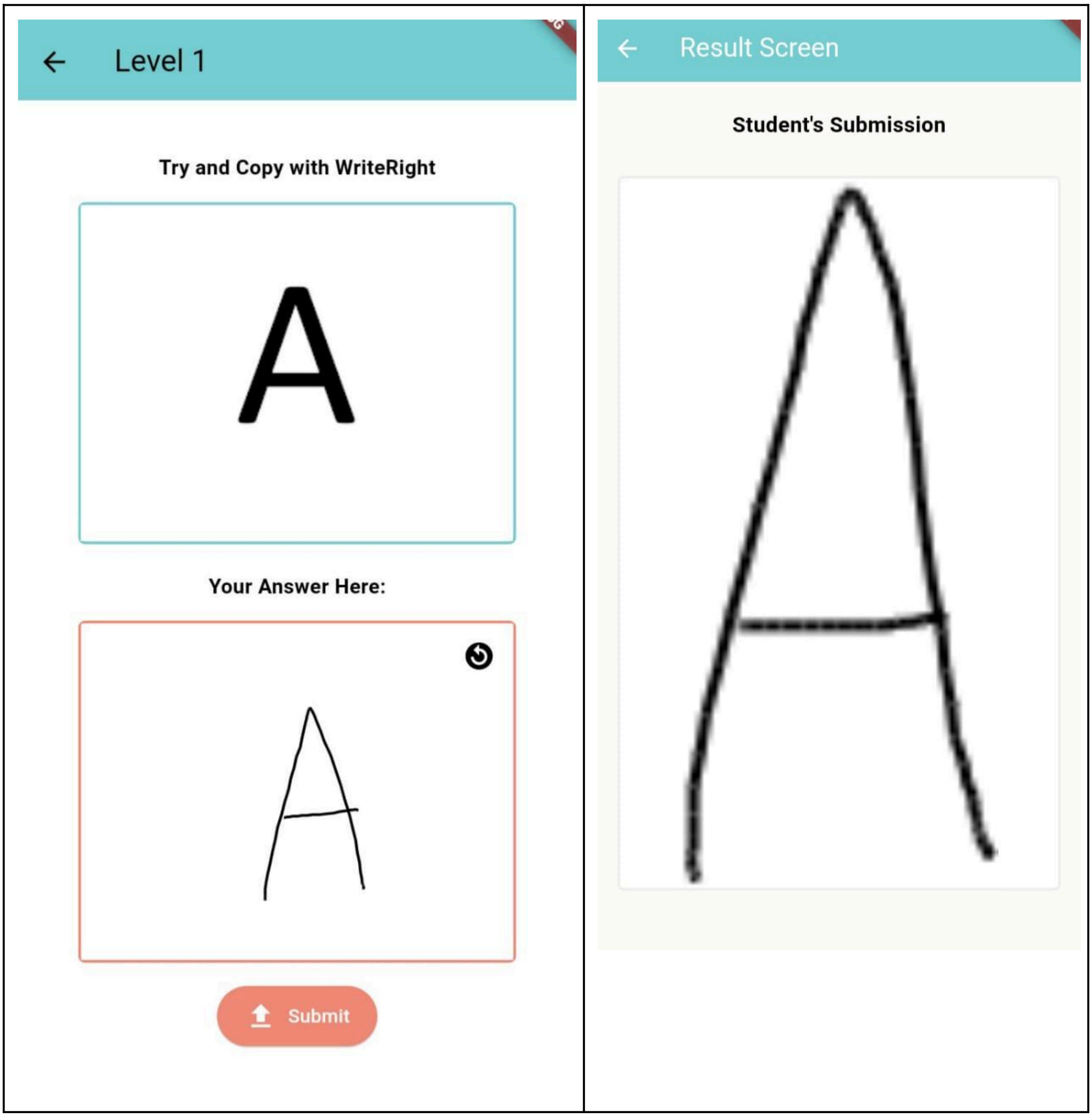


Fig 35: Sample character and student's submission screens of WriteRight

Identification	Name. Play Level. Location. Mobile Application Module
Type	User Interaction Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document. ● **The system shall allow users to upload an image of handwritten text for a specified level and receive a score based on the uploaded image.** **Description** This feature enables users to play a specific level by uploading an image of handwritten text associated with that level. The system processes the image and provides a score, allowing users to progress through the levels based on their performance.
Function	Enable users to upload a handwritten text image for a specific level and receive a score.
Subordinates	None.
Dependencies	● User authentication system to verify login status. ● Gamification data repository for level information and text samples. ● Image processing system to analyze the uploaded handwritten text.
Interfaces	User login status, user selection of level, uploaded image of handwritten text.
Resources	**User Interface:** Frontend component for level selection, image upload, and score display. **Backend Server:** To fetch level data, process uploaded images, and calculate scores. **Image Processing System:** For analyzing the quality and content of uploaded handwritten text images.

Processing	● Display the handwritten text for the selected level. ● Accept and upload an image of the user's handwritten text. ● Process the uploaded image to calculate a score.
Data	**Output Data:** Handwritten text display, calculated score, feedback on image quality, level progression status.

4.3.1.1.4 Calculate and View Results

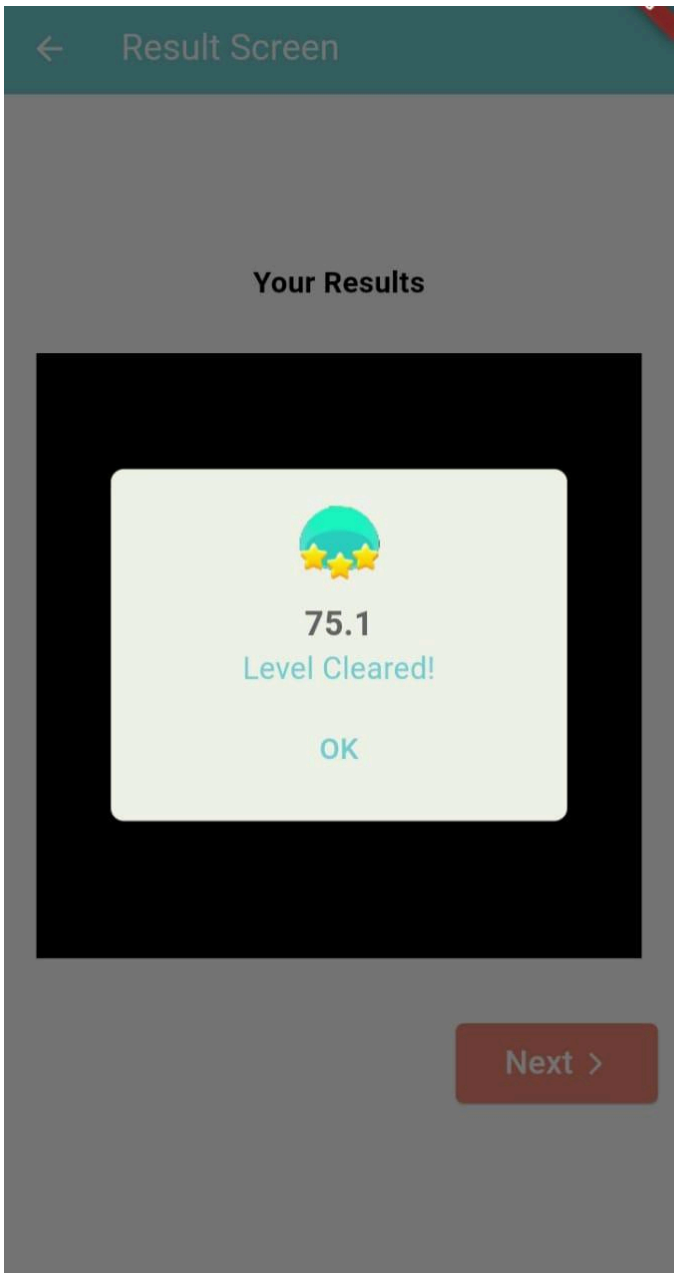
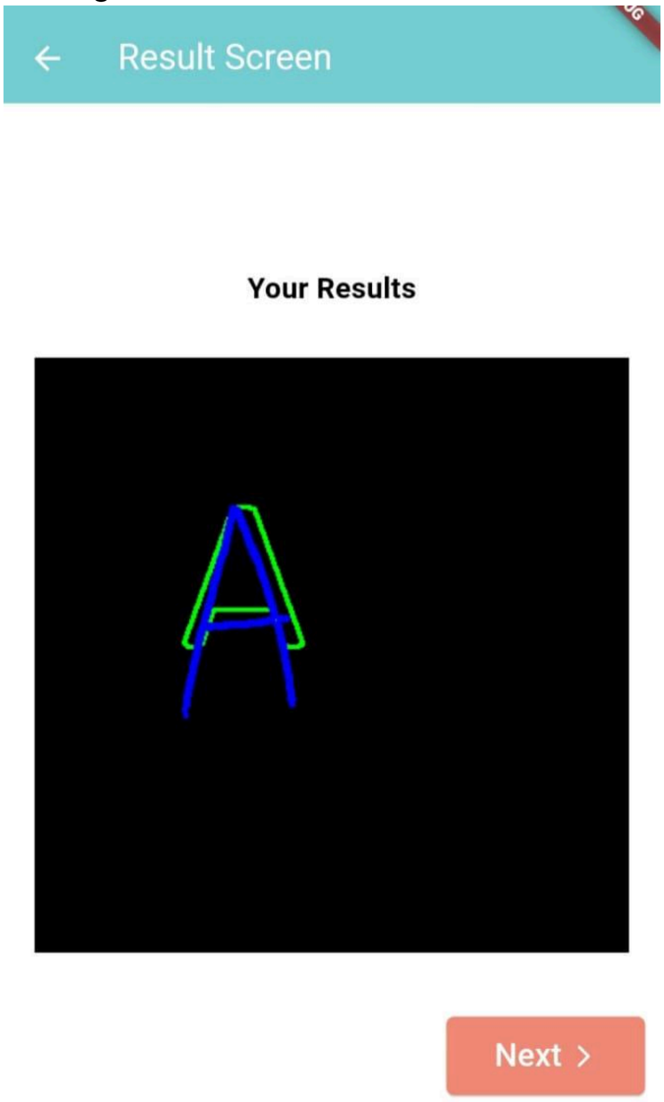
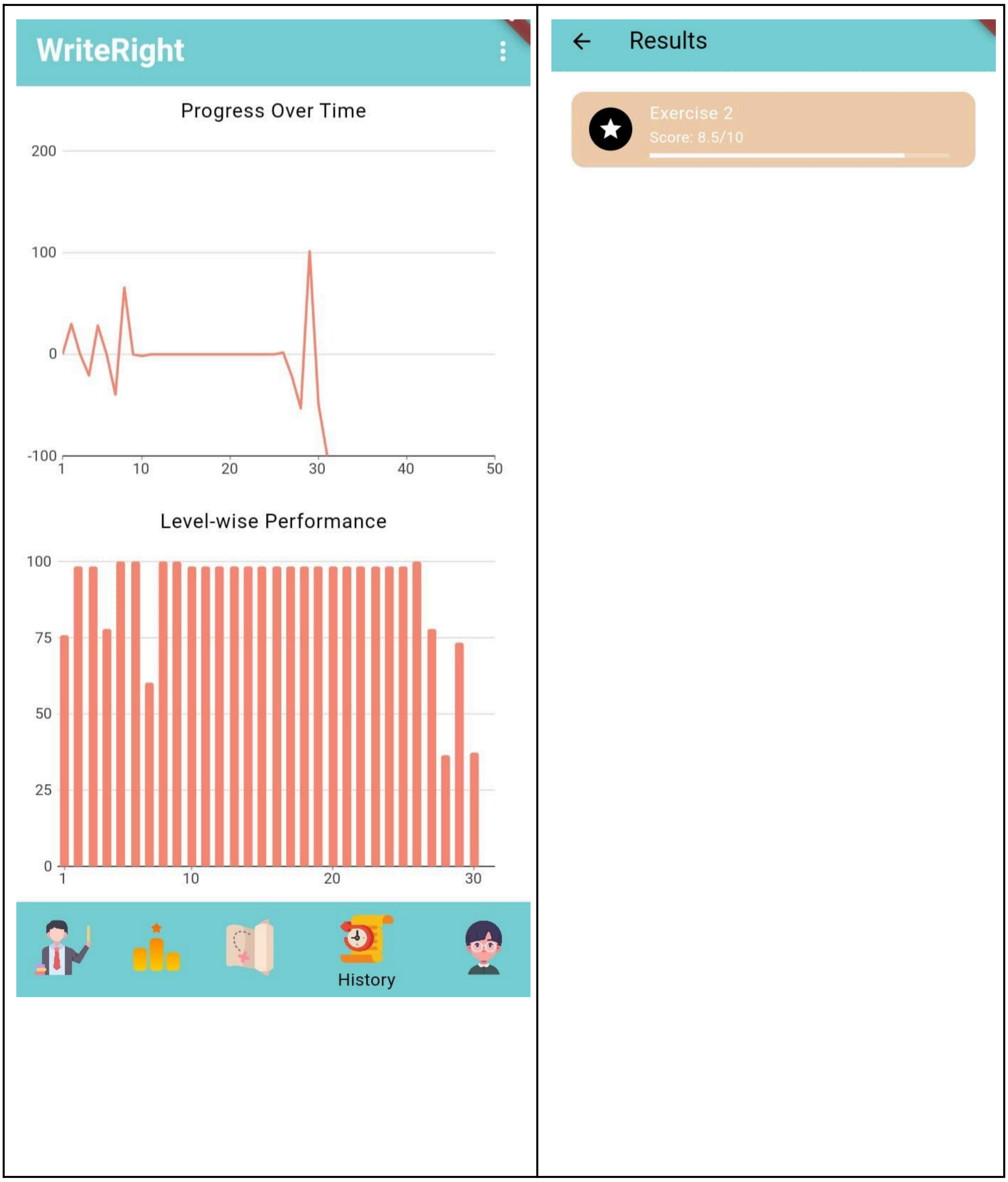
Score  Result Screen Your Results 75.1 Level Cleared! OK Next	Marking 

Fig 36: Results and marking displayed after comparing

Identification	Calculate and View Results
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document. The system shall process uploaded images of handwritten text using Digital Image Processing (DIP) algorithms to calculate results. The results are then displayed on user screen. **Description** This feature processes uploaded images of handwritten text using DIP algorithms. It calculates results, including a similarity score and improvement tips, and displays these results on the screen.
Function	Display results from DIP algorithms.
Subordinates	None.
Dependencies	● User authentication system to verify login status. ● Image processing system for analyzing uploaded images. ● Algorithmic modules for calculating similarity scores and generating improvement tips.
Interfaces	None.
Resources	**User Interface:** For displaying calculated results on the screen.
Processing	Display attendance after server processing on the basis of facial recoginition.
Data	● Accept and upload a clear image of the user's handwritten text. ● Process the uploaded image using DIP algorithms.

	● Calculate results, including a similarity score and improvement tips. ● Display the results on the screen for the user.

4.3.1.1.5 View Student Performance



Exercise 2
Score: 8.5/10

Fig 37: Performance tab for Student in WriteRight

Identification	View Child Performance
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document. The system shall allow parents to view the performance of their children over time. **Description** This feature enables parents to view the performance data of their children, tracking progress and improvements over time. The system presents historical performance data in an easily understandable format.
Function	Display the historical performance data of children for parents to review.
Subordinates	None.
Dependencies	Student performance data repository for historical data.
Interfaces	Display of historical performance data.
Resources	**User Interface:** Frontend component for displaying performance data. **Backend Server:** To fetch historical performance data of students. **Database:** Repository of student performance data.
Processing	● Enable selection of a specific child whose performance data is to be viewed. ● Retrieve and display the historical performance data for the selected child. ● Present the data in an easily understandable format, including charts and graphs showing performance over time.

Data	This component uses following information of the application. - Time to fetch notifications from server.

4.3.1.2 Teacher Dashboard module

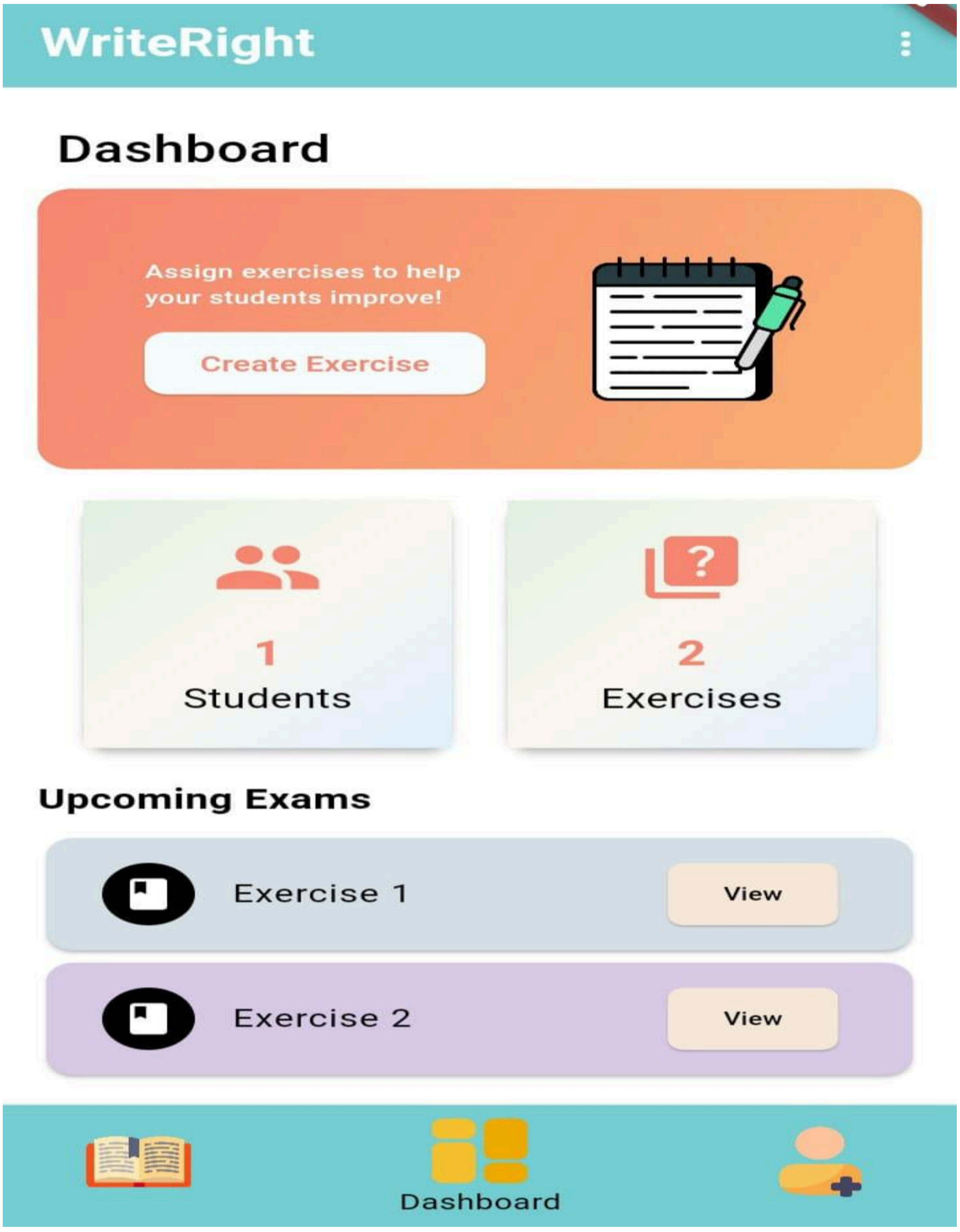


Fig 38: Dashboard module for Teacher in WriteRight

Identification	Teacher Module Dashboard
Type	Component
Purpose	This component fulfils following requirement from Software Requirements Specification Document. ● The system shall provide teachers with an overview and quick access to main features for managing students and exercises. **Description** The Teacher Module Dashboard is the main control panel for teachers using the WriteRight application. It allows teachers to create and assign exercises, manage students, and view upcoming exams. The dashboard provides quick statistics on the number of students and exercises, and easy navigation to different sections of the application.
Function	Provide teachers with an overview and quick access to main features for managing students and exercises.
Subordinates	None.
Dependencies	● User authentication system to verify teacher login status. ● Exercise management system to handle creation and tracking of exercises. ● Student management system to track student statistics.
Interfaces	Display of statistics, exercises, and navigation options for teacher.
Resources	**User Interface:** Frontend component for displaying dashboard elements. **Backend Server:** To fetch data related to students and exercises. **Database:** Repository of student and exercise data.
Processing	● Verify teacher login status. ● Display quick statistics on the number of students and exercises. ● Provide a button to create new exercises. ● List upcoming exams or exercises with options to view details.

	● Allow navigation to different sections of the application through a bottom navigation bar.
Data	Input Data: Teacher login credentials, user interactions (e.g., creating exercises, viewing exams). Output Data: Display of statistics (number of students and exercises), list of upcoming exams, navigation options.

4.3.2 Server-Side Module

This module performs all the back-end functionalities related to pre-processing of videos, feature extraction, classification, ML algorithms implementation and decision making. Feature extraction is the main input for Algorithm implementation component.

In the next chapter all of the techniques used in server side are discussed in detail.

Chapter 5. Implementation And Testing

5.1 System Overview

WriteRight" is an engaging, gamified handwriting improvement app. It uses Flutter for mobile development, Django for the backend, and Firebase for real-time data sync. Users can track progress and get feedback on their handwriting using an image comparison algorithm. The app, available on Android and iOS, uses levels and rewards to motivate users, offering a comprehensive solution to improve handwriting through technology and interactive learning.

5.2 Preprocessing

Preprocessing is essential for assuring the correctness and dependability of the metric's outcomes while preparing pictures for image comparison score computation. Prior to calculating the image comparison score, images frequently go through a number of preprocessing stages designed to improve their quality and highlight key structural elements.
Following are several reasons why it is crucial to preprocess the image before performing structural similarity index (SSIM) comparison:

- **Noise Reduction:** Images can have noise from lots of different places, such as bad sensors, compressed data or light when the picture was taken. To clean up such noise before further analysis researchers use filters (for example 'median filter'). For proper comparison it is important to reduce noise since any additional dissimilarity may affect the structural similarity index measure (SSIM): because this index is sensitive to even small changes between pixels.
- **Feature Enhancement:** Preprocessing methods like thresholding and Laplacian filtering are examples of how to draw attention to the important edges and features in an image. This makes the SSIM comparison less concerned with irrelevant details or noise and more with what is regarded as structural information. Preprocessed images are used to make comparisons that are more accurate since they highlight these important structural elements. Similarly, structural motif similarity between images is quantified using SSIM.
- **Standardization of Input:** Before conducting SSIM comparison, it is important to optimize the input images, ensuring they match and have homogenous properties. The preprocessing stage removes variances in image dimensions and color channels by reducing the photos to a common size (800x800 in this case) and converting them to grayscale. This increases the consistency and reliability of the SSIM comparison.

Following are the important steps of preprocessing before comparison.

5.2.1 Converting to grayscale:

The original image is converted from a color (BGR) representation to a grayscale representation. This simplifies the image to a single channel, making it easier to process and analyze.

5.2.2 Binarization:

To transform the grayscale image into a binary image, a thresholding procedure is used. Pixels are set to their maximum intensity (255) when their intensity values are above a threshold (in this case, 100), and to zero when they are below.

5.2.3 Noise Reduction with Median Filtering:

Median filtering is applied to the binary image to remove noise and small artifacts. It replaces each pixel's value with the median value in its neighborhood. The kernel size (5x5 in this case) determines the size of the neighborhood.

5.2.4 Laplacian Filtering:

The Laplacian filter is applied to highlight edges and features in the image. It calculates the second derivative of the image and is commonly used for edge detection.

5.2.5 Inversion of Laplacian Image:

The Laplacian image can be made inverted by deducting its values from 255. Then, the contrast will be increased and the edges will be easier to see.

One of the image processing techniques, Laplacian filtering, is utilized in edge identification and feature enhancement. The method calculates the second derivative of the picture intensity function. In the case of an image, the Laplacian operator is customarily modeled as a convolution kernel which is a two-dimensional operator.

This approach reveals the structural components of the image and emphasizes regions within the picture which have rapidly varying intensities, typically corresponding to borders. For a Laplacian-filtered frame, positive numbers show greater change of intensity while negative ones denote less modification within certain regions considerably. At the same time, applying a Laplacian filter means bettering image sharpness and clarity through emphasizing edges or objects with marked changes in intensity. Many image processing tasks require this enhancement like texture analysis, edge recognition and image sharpening; In order to boost structural information Laplacian filtering is essential at last, so as to enhance the general picture quality which will lead to later analysis and interpretation of pictures.

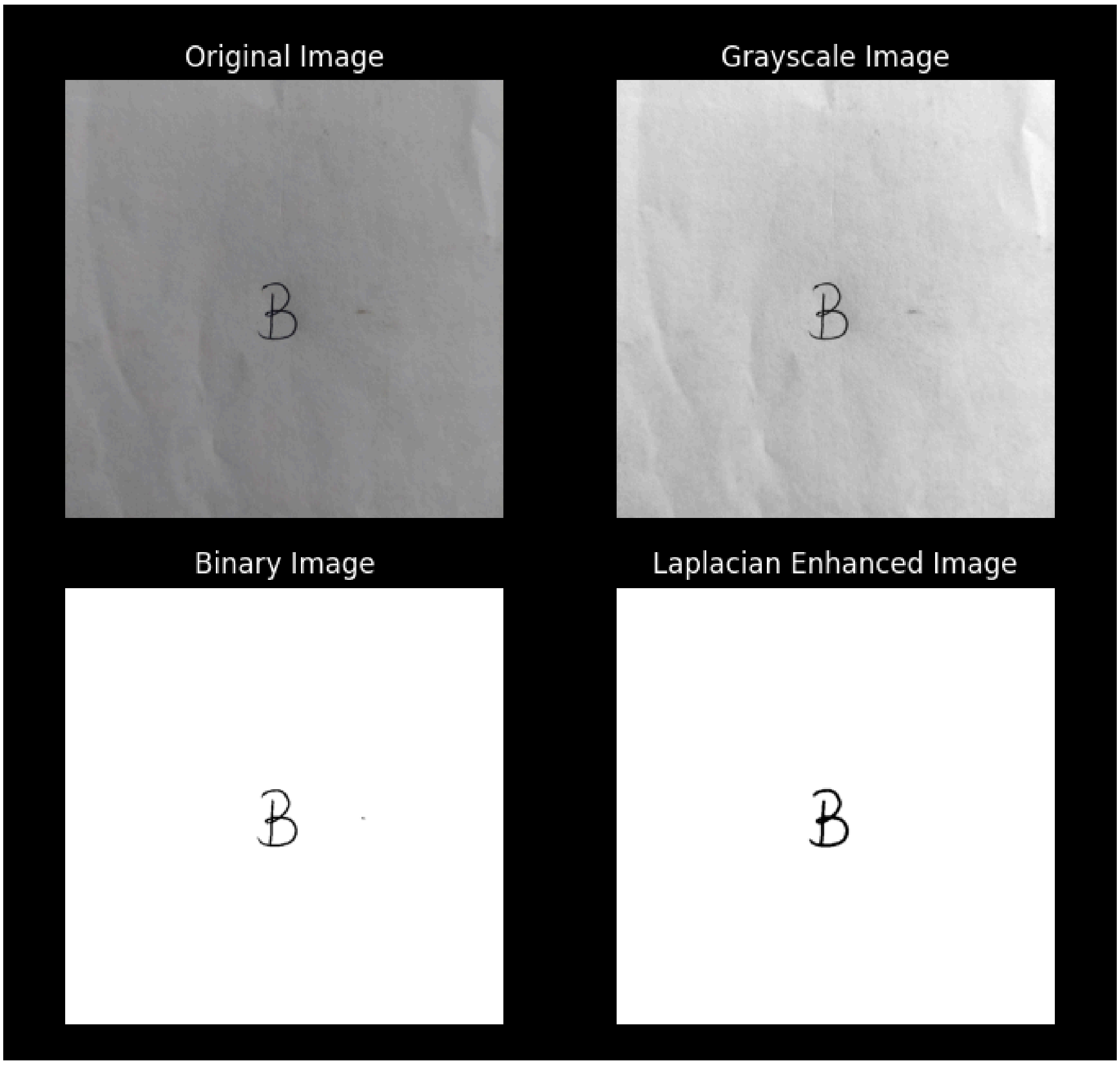


Fig 39: Preprocessing of images before comparing and other algorithms.

5.3 Structural Similarity Index:

After the image has been pre-processed, it goes to a stage of contrast involving comparison to an exemplar image that resides within the system. In this matching exercise, the Structural Similarity Index (SSIM) is applied to assess the degree of resemblance between the image that has been pre-processed and a standard. A common statistic used to assess how similar two images appear visually while taking structure, contrast, and brightness into consideration is called SSIM.

In image processing, the Structural Similarity Index (SSIM) is a popularly recognized metric used to compare two different pictures. The SSIM finds out how similar the photographs are in content and structure, as in preprocessing. We should look at the most important steps involved in SSIM computation after the preprocessing period:

5.3.1 Windowing and Division into Patches:

Rather than processing the entire image at once, SSIM operates on local image patches. There are smaller spots in the distorted as well as the reference photos. This stage enhances the precision of SSIM's similarity judgment by capturing local structural data.

5.3.2 Local Mean Calculation:

Every patch is computed to give the mean intensity value which shows the luminance or brightness level of that particular patch. The mean value serves as a reference point in terms of evaluating local similarities in structures between reference and deformed images.

5.3.3 Local Variance and Covariance Calculation:

We calculate the local differences in intensity values and how the reference and distorted patches change together. Variance tells us how the pixel intensities are spread out within a patch, while covariance measures how the pixel values in the two patches move in relation to each other. These statistics help us understand the local texture and contrast information.

5.3.4 Luminance, Contrast, and Structure Comparison:

SSIM, a measure for evaluating luminance similarity, contrast similarity, and structure similarity with respect to mean, variance, and covariance values, calculates three terms. Luminance similarity quantifies how well related patches match with regard to their average brightness levels. We can evaluate whether a given patch is similar in terms of texture/detail if we measure contrast level (contrast term preview). SSIM is able to observe patterns present between neighbouring pixels (structure term exapnasion).

5.3.5 Aggregation Across Patches:

The total SSIM index is obtained by calculating the average local similarity measures acquired from each patch such that, the mean SSIM value of all patches is usually presented as a final result giving a comprehensive analysis of how alike are all pre-processed images.

Although SSIM is useful, its inherent limitations prevent it from providing exact results every time. It may not be able to fully capture picture quality even though SSIM offers valuable insights into structural similarity. Complex scenes, severe distortions, and personal taste in colours are some examples of elements that can make it difficult for SSIM to measure similarity effectively. Furthermore, the particular preprocessing methods used and the properties of the images under comparison may have an impact on how well SSIM performs. Therefore, for a complete understanding of image quality, even if SSIM is a useful statistic for assessing image quality, it's important to recognise its limitations and supplement its findings with other metrics and subjective evaluations.

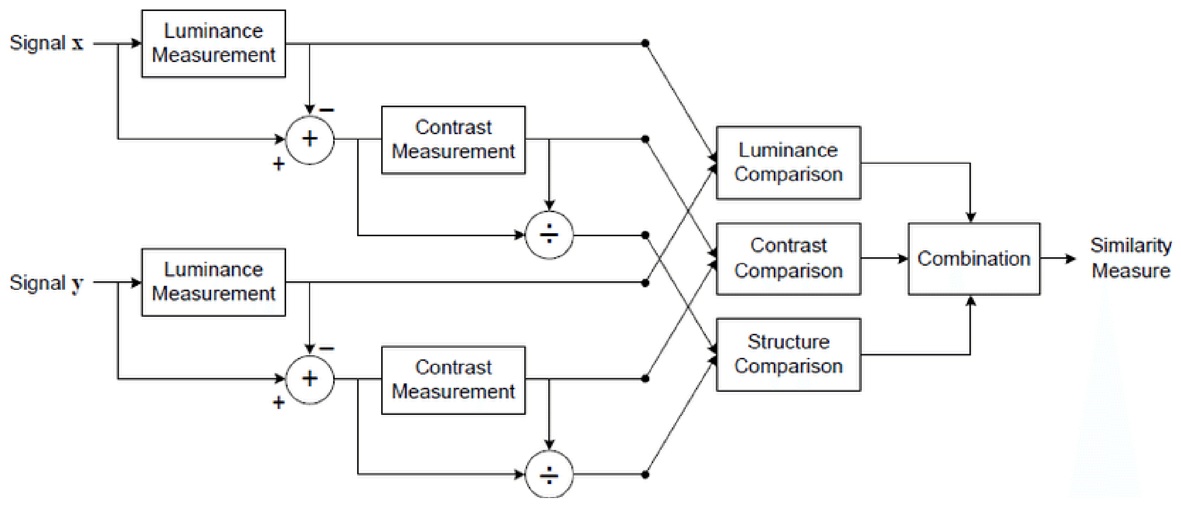


Fig 40: Working of ssim for image comparison results.

5.4 Marking and Placing Images

The preprocessing begins by converting the images to grayscale to simplify processing. Edge detection locates areas of rapid intensity change, enhancing edges. Edge density, the ratio of edge pixels to total pixels, is calculated to determine whether dilation is needed for contour detection. Contours, representing object boundaries, are extracted, and the largest contours are identified. Contours are resized while maintaining aspect ratios for consistency. Resized contours are drawn on a canvas for visual comparison. This process aids in aligning and visually inspecting contours, highlighting shapes and differences.

5.4.1 Edge Detection with Canny:

The edges of the image correspond to locations where the Canny algorithm detects rapid intensity changes. It uses gradient-based techniques, examining the pixel intensity gradients in the image, to pinpoint these modifications. The end product is a binary image with highlighted edges that are displayed as white lines on a black backdrop. This method works well for extracting important structural information from the picture, which is needed for later contour identification and shape analysis.

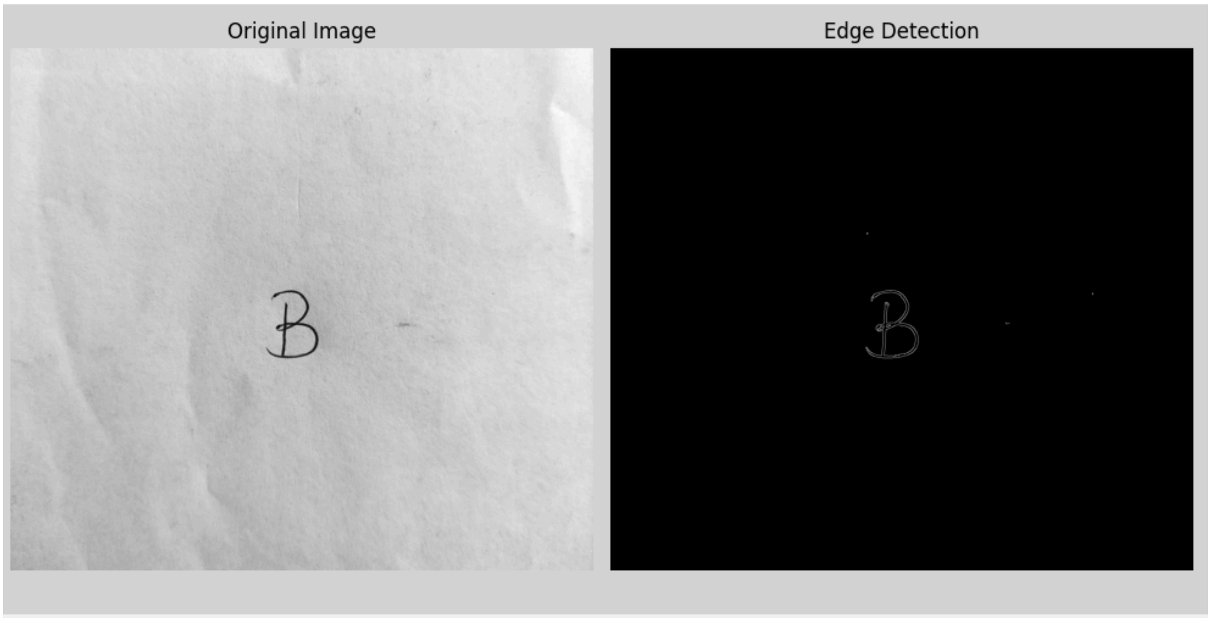


Fig 41: Edge detection in preprocessed image

5.4.2 Edge Density Calculation:

Edge density calculation is a technique used in image analysis to evaluate the prominence of edges within an image. This is done to determine the strength of edges in the image. Sometimes the strength of edges is not enough to determine and define the required boundary. The total number of pixels in the image is calculated at the same time as the number of edge pixels. Next, divide the number of edge pixels by the total number of pixels to get the edge density. This produces a ratio that shows how much of the image's surface is made up of edges. Whereas a lower density denotes smoother sections with fewer visible edges, a higher density signifies a larger frequency of edges, indicating more detailed or structured parts in the image. Image with less edge density is then dilation which is explained in next step.

5.4.3 Conditional Dilation:

Depending upon the edge density we apply dilation which eases the task of edge detection. The morphological process of dilation thickens the borders between objects. Depending on the determined edge density, it can be used to enhance or remove edges and features as per the appropriate edge density. In the pictures where edge prominence is different, this step assists the algorithm in contour detection. However, in image processing, dilation is a morphological operation that expands an object's size by adding more pixels to it. It involves using a kernel – or a small binary image which is commonly referred to as a structuring element- to traverse the image, followed by the setting of each pixel's intensity value to the greatest intensity value that has been identified in the neighborhood recognized by the structuring element. This process is an efficient way of increasing areas of high intensity within the image, through the thickening or enlarging of certain features. This means that extent of dilation depends on size and shape of the structuring element; large elements lead to substantial dilation of objects.

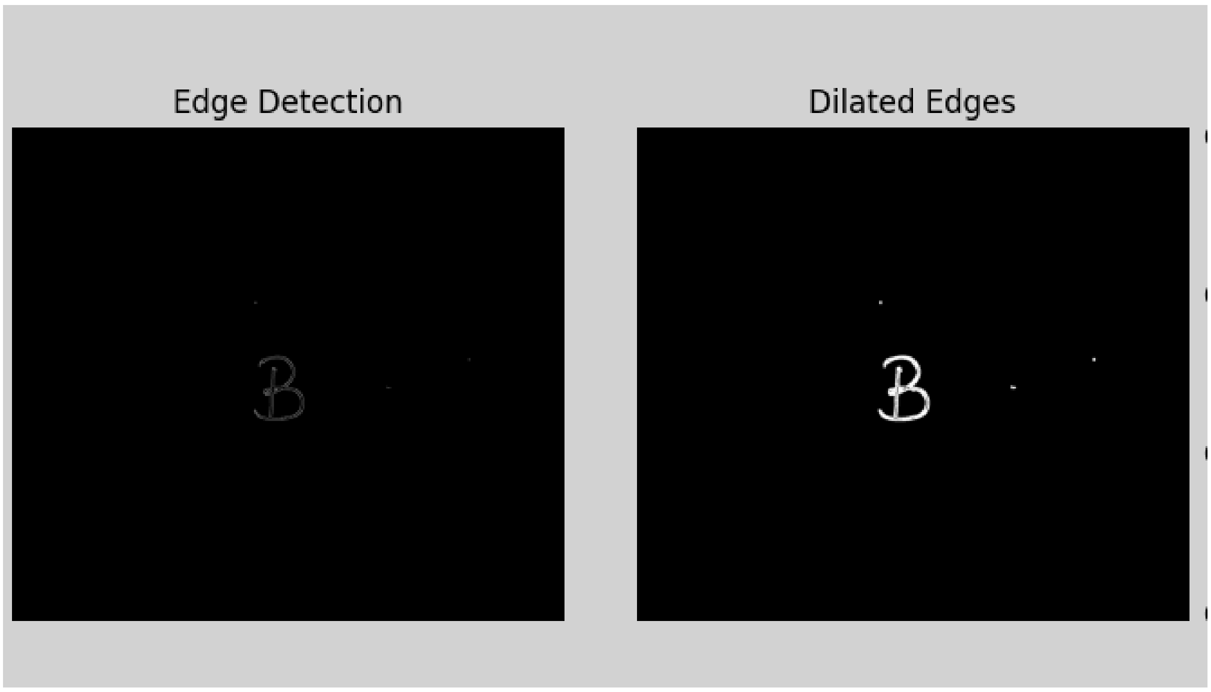


Fig 42: Dilation of edges

5.4.4 Contour Extraction and Selection:

Many techniques of image processing involve contoured pictures to differentiate the forms present in an image. From edge-detected pictures, contours, which connect places of continual shades along boundaries of constant characteristics, are obtained using the cv2. findContours() tool. These are the base work that builds up to the tasks of shape analysis and item detection later on the process. It is therefore important to establish the method of calculating the highest contour in order to enhance the analysis to a further degree. This step isolates the most important form in the image by choosing the contour with the largest area among the contours that have been found. This targeted strategy simplifies analysis by focusing on the most salient characteristic for closer inspection and contrast. To facilitate additional analysis or processing, the biggest contour—which is the object of interest—undergoes actions like scaling or alignment.

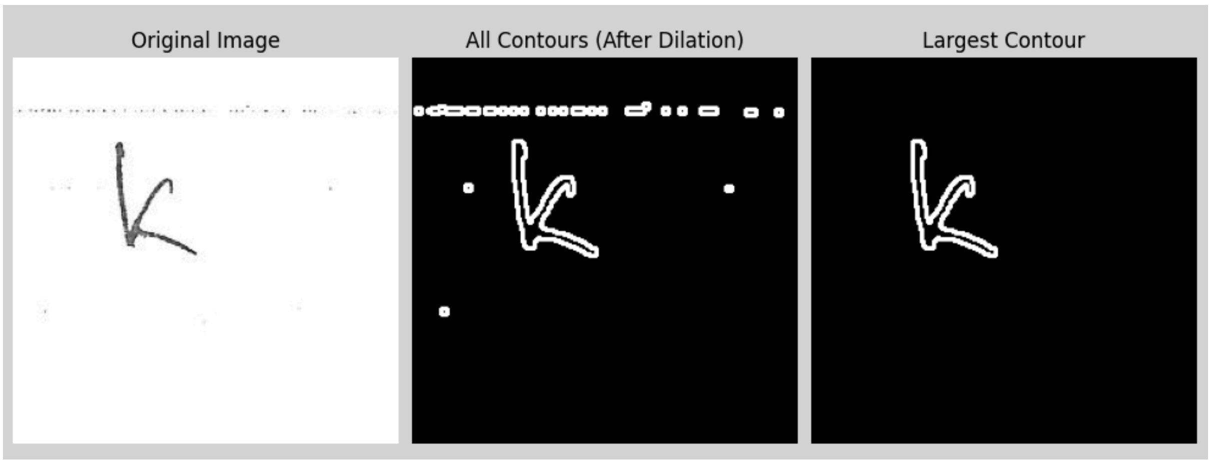


Fig 43: Displaying largest selected contour

Now we can see that the largest contour shown is the thing of interest and we will map these contours onto each other. The largest contour is typically the most significant shape in the image. To do this, we take this step by choosing the one wide line above all others—it lets us concentrate more on what matters most while leaving out distractions.

5.4.5 Resizing Contours with Aspect Ratio Preservation:

To compare different forms and achieve accurate results when matching photos to proper forms, the largest contours are resized while maintaining their aspect ratios. Resizing avoids distortion as well as maintains the correct dimensions of the shapes through maintaining the aspect ratio of the contours which is the ratio of width to height. This means that the shapes of different sizes or different orientations can be compared without necessarily developing bias due size differences. Students need to understand that by comparing both images using a standard size, it becomes easy to distinguish between the two and this leads to improved understanding of the different aspect of the two shapes.

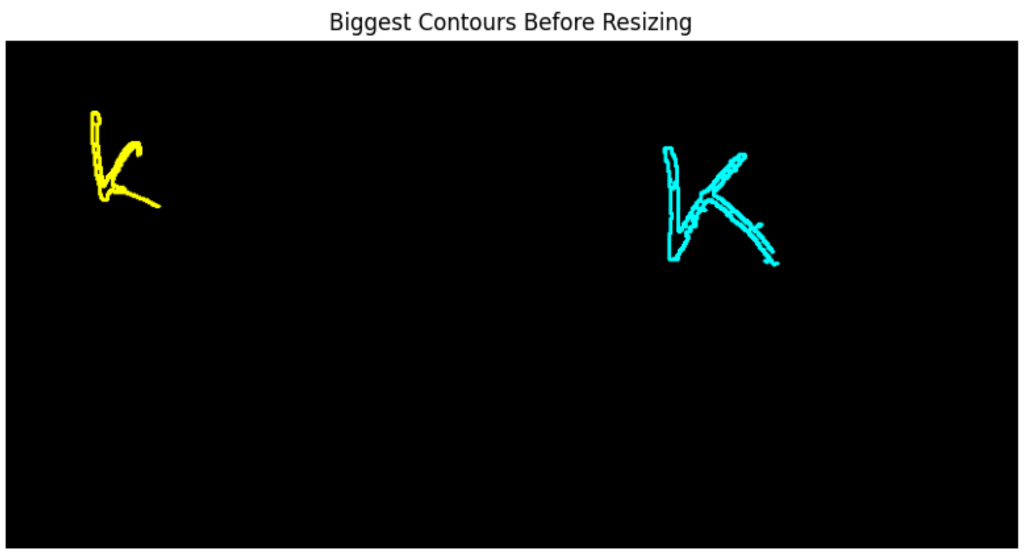


Fig 44: Largest contours with unequal sizes not suitable for marking

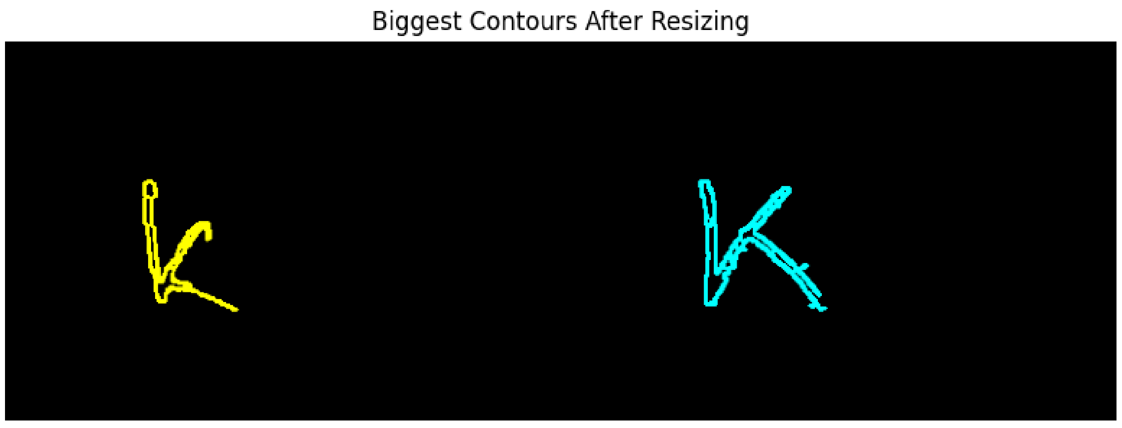


Fig 45: Resized contours with aspect ratio. Suitable for marking

5.4.6 Drawing Contours on a Canvas :

Meanwhile, in the domain of image analysis, understanding of shapes and patterns requires parallel visualisation. For example, imagine when a student aligns his/ her image with that of an instructor or a model image with the purpose of seeking differences and probable change enhancements. Such is the effect that the contour drawing on a canvas produces. To create the teacher's or sample image, the cv2.drawContours() function must be used to compile modified contours on a blank canvas; altered versions (from the students) are positioned next to the direction picture for comparison, while different areas are lined-up with these images for corrections. Different colors are applied to distinguish between two sets of lines drawn on the map: those coming from learners’ work and those from the teacher’samework. With the use of this visual tool, students can easily identify areas in which their work deviates from the required quality, giving them the confidence to rectify any errors and improve their abilities. As a result, creating contours on a canvas is an effective teaching tool that promotes deeper understanding and focused learning.

Chapter 6. Conclusion and Future Works

WrightRight is just a prototype and there is a lot of untouched domains to be explored. WrightRight is a base line variant and it can turn out to be the first handwriting improvement application in Pakistani market once the future worked is achieved. Due to the scope defined in SRS and limitations to implement research based solutions, the development of WriteRight is limited and the scope can be increased in the interest of research based domain.

Creating Animations to teach require Adope Animate tool. Learning the tool and developing the animations in house will be quite expensive in terms of expanding scope and a threat to time constraints. For the dedicated OCR, a large dataset of handwritten english alphabets and in future words and sentences is required to be gathered and creating a customized dataset requires a lot of time and effort along with a good number of volunteers to write. Improving the OCR and introducing words and sentences to OCR will also help character limitation to be bypassed. For better response time, algorithms used in the backend must be replaced with customized algorithm requiring skills in computer vision.

Sr No.	**Feature**	**Module**	**Expected Cost**	**Inhouse/ External**
1	Animated Tutorials	Student/ Teacher	$500	External
2	Faster Response Time	Student	$200	Inhouse
3	Dedicated OCR	Student	$800	Inhouse
4	No character limitation	Student/ Teacher	$400	Inhouse

Table 2. Table stating future work that can be made

References

1. *Image Processing Techniques for Handwriting Analysis: A Review, P. S. Hiremath and S. B. Hiremath, International Journal of Computer Applications, 2014:*https://www.researchgate.net/publication/345364858_Image_Processing_Techniques_A_Review

2. *IEEE Standard 830-1998 (R2019), IEEE Recommended Practice for Software Requirements Specifications, 1998, Reaffirmed 2019:* http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

3. *Gamification and Education: A Literature Review, Ilaria Caponetto, Jeffrey Earp and Michela Ott, ITD‐CNR, Genova, Italy, 2014:*https://www.itd.cnr.it/download/gamificationECGBL2014.pdf

4. *Secure File Upload and Storage Practices, National Institute of Standards and Technology (NIST), 2023:*https://www.nist.gov/publications/security-considerations-exchanging-files-over-internet

5. *API Security Best Practices, OWASP, 2023:* https://owasp.org/API-Security/editions/2023/en/0x11-t10/

6. *United Nations. (2023). Sustainable Development Goals. Retrieved from https:*https://www.un.org/sustainabledevelopment/sustainable-development-goal

7. SHRESHTA, N., PANTHA, R. R., NAGARKOTI, S., & DHUNGANA, S. (2023). *A INTEGRATED EDUCATIONAL TOOL:* https://elibrary.tucl.edu.np/handle/123456789/18860

8. Koç, H., Erdoğan, A. M., Barjakly, Y., & Peker, S. (2021, March). *UML diagrams in software engineering research: a systematic literature review. In Proceedings (Vol. 74, No. 1, p. 13). MDPI:* https://ceur-ws.org/Vol-1078/paper1.pdf

9. Evans, A. S. (1998, October). *Reasoning with UML class diagrams. In Proceedings. 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques (pp. 102-113). IEEE:* https://ieeexplore.ieee.org/abstract/document/766304

10. Fayed02.*Handwritten Isolated English Character Dataset:* https://www.kaggle.com/datasets/fayed02/handwritten-isolated-english-character-dataset

11. *Refactoring.Guru (2014-2024). Design Patterns:* https://refactoring.guru/design-patterns/

12. Pranjal Datta. *All about Structural Similarity Index (SSIM): Theory + Code in PyTorch:* *https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e*