

**Addis Ababa University**

**College of Technology and Built Environment**

**School of Biomedical Engineering**

**Development of Android-Based Autonomous Blind Examiner Mobile Application**

#### By Id

1. Alebel Melak UGR/5507/13
2. Amanuel Zewdu UGR/9255/13
3. Aragaw Hussen UGR/1178/13
4. Dagnachew Tilahun UGR/5514/13
5. Natnael Abayneh UGR/8829/13
6. Rumman Murad UGR/6778/13

In Partial Fulfillment of The Requirements for The Degree of Bachelor of Science in Biomedical Engineering

**Advisor: Mr. Bemnet Zelalem**

Addis Ababa, Ethiopia

June 2025

# ABSTRACT

Blind students often encounter significant challenges during regular exams. They usually need help from someone else to write their answers, which can take away their privacy and make them feel less independent. This project introduces a mobile app that helps blind students take exams on their own. The app reads out the questions using audio and lets students type their answers using a touch-based Braille keyboard on the screen. It also uses smart technology (machine learning) to check short, written answers by comparing their meaning with the correct ones. To stop cheating, the app blocks the bottom buttons on the phone so students can’t leave the app during the test. The app was made for Android phones and tested with real students. The results show that it works well, is easy to use, and helps blind students take exams in a fair and safe way giving them more freedom, confidence, and a better learning experience.

**Key Words:**

Assistive technologies, Braille input, Audio question delivery, Semantic similarity, Exam accessibility, visually impaired

# DECLARATION

We, the undersigned, hereby declare that this thesis, “Development of an Android-

Based Autonomous Blind Examiner Mobile Application”, has been prepared by us in partial fulfillment of the requirements for the Bachelor’s degree in Biomedical Engineering. We further confirm that all sources of information used in the preparation of this thesis have been properly acknowledged.

#### Name ID Signature

Natnael Abayneh UGR/8829/13 \_\_\_\_\_\_\_\_\_\_

Dagnachew Tilahun UGR/5514/13 \_\_\_\_\_\_\_\_\_\_ Amanuel Zewdu UGR/9255/13 \_\_\_\_\_\_\_\_\_\_ Aragaw Hussen UGR/1178/13 \_\_\_\_\_\_\_\_\_\_

Alebel Melak UGR/5507/13 \_\_\_\_\_\_\_\_\_\_

Rumman Murad UGR/6778/13 \_\_\_\_\_\_\_\_\_\_

This BSc thesis has been submitted for examination with my approval as an advisor.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bemnet Zelalem (MSc.)

# CERTIFICATE OF EXAMINATION

This is to certify that the thesis prepared by Natnael Abayneh, Dagnachew Tilahun, Amanuel Zewdu, Aragaw Hussen, Alebel Melak, Rumman Murad, entitled: “Development of Android based Autonomous Blind Examiner Mobile Application” submitted in partial fulfillment of the requirements for the Degree of Bachelor of Science in Biomedical Engineering complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

#### Signed by examining committee

Examiner\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Examiner \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Advisor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chief of Department or Undergraduate Program Coordinator

# ACKNOWLEDGMENTS

First and foremost, we would like to thank God for His guidance, strength, and blessings throughout this project.

We would also like to extend our deepest gratitude to our advisor, **Mr. Bemnet Zelalem (MSc.),** for his exceptional guidance and continuous support. His expertise and dedication have been invaluable throughout the course of this work.

Lastly, we would like to thank our evaluator for their constructive feedback and thoughtful critique, which have significantly contributed to the improvement and completion of this project.

**Table of Contents**

[ABSTRACT I](#_Toc58827)

[DECLARATION II](#_Toc58828)

[CERTIFICATE OF EXAMINATION III](#_Toc58829)

[ACKNOWLEDGMENTS IV](#_Toc58830)

[LIST OF FIGURES VII](#_Toc58831)

[ACRONYM X](#_Toc58832)

[CHAPTER ONE 1](#_Toc58833)

[INTRODUCTION 1](#_Toc58834)

[1.1 Background 1](#_Toc58835)

[1.2 Problem Statement 2](#_Toc58836)

[1.3 Hypothesis 3](#_Toc58837)

[1.4 Objective 3](#_Toc58838)

[1.4.1 General Objective 3](#_Toc58839)

[1.4.2 Specific Objectives 3](#_Toc58840)

[1.5 Scope of the Project 4](#_Toc58841)

[1.6 Significance 4](#_Toc58842)

[1.7 Organization of the Thesis 5](#_Toc58843)

[CHAPTER TWO 6](#_Toc58844)

[LITERATURE REVIEW 6](#_Toc58845)

[2.1. General Literature on Blind Examination Using Mobile Apps 6](#_Toc58846)

[2.2. Brail Input Method 7](#_Toc58847)

[2.3. Semantic similarity 11](#_Toc58848)

[2.4. Literature Gaps 16](#_Toc58849)

[2.5. Proposed Solution 17](#_Toc58850)

[CHAPTER THREE 18](#_Toc58851)

[METHODOLOGY 18](#_Toc58852)

[3.1. Methods 18](#_Toc58853)

[3.1.1. Data Collection 18](#_Toc58854)

[3.1.2. Public Datasets and Resources 18](#_Toc58855)

[3.1.3. Frontend/User Interface (UI) Development 19](#_Toc58856)

[3.1.4. Backend Development 20](#_Toc58857)

[3.1.5. Machine Learning Integration 22](#_Toc58858)

[3.1.6. Testing and Quality Assurance 23](#_Toc58859)

[3.2. Materials 23](#_Toc58860)

[3.2.1. Hardware Requirements 23](#_Toc58861)

[3.2.2. Software and Development Tools 24](#_Toc58862)

[3.2.3. Cloud Infrastructure and Services 24](#_Toc58863)

[3.2.5. Data Collection and Analysis Materials 25](#_Toc58864)

[3.2.6. Other Tools 26](#_Toc58865)

[CHAPTER FOUR 28](#_Toc58866)

[RESULTS AND DISCUSSIONS 28](#_Toc58867)

[4.1. Machine learning evaluation 47](#_Toc58868)

[4.1.1. Evaluation of Braille Character Recognizer 47](#_Toc58869)

[4.1.2. Evaluation Summary: Automatic Grading 50](#_Toc58870)

[CHAPTER FIVE 56](#_Toc58871)

[CONCLUSIONS 56](#_Toc58872)

[5.1. Challenges and Limitations 56](#_Toc58873)

[REFERENCES 58](#_Toc58874)

[APPENDIX 62](#_Toc58875)

[Code 62](#_Toc58876)

# LIST OF FIGURES

Figure 1.1 World report on vision .......................................................................... 1

Figure 2.1 Detailed interaction diagram of the Exam Voice system components .. 6

Figure 2.2 Top Level System Architecture for Voice-Based Online Examination

Syste................................................................................................................ 7

Figure 2.3 Sample Braille input is taken from a participant ................................... 8

Figure 2.4 Basic gestures ........................................................................................ 9

Figure 2.5 Braille-Gesture of Character A .............................................................. 9

Figure 2.6 Braille alphabet, letters a, b, c and d. .................................................. 10

Figure 2.7 A. App Interface and B. Tactile Interface ............................................ 10

Figure 2.8 Model for text similarity using different algorithms ........................... 12

Figure 3.1 Flowchart for the whole app ............................................................... 26

Figure 3.2 Flowchart for exam taking system ...................................................... 27

Figure 4.1 Login page ........................................................................................... 28

Figure 4.2 Admin dashboard ................................................................................ 28

Figure 4.3 Admin Dashboard: Create Student Account ....................................... 29

Figure 4.4 Admin Dashboard: Create Instructor Account .................................... 29

Figure 4.5Admin Dashboard - Manage Users (Students Filtered View) .............. 30

Figure 4.6 Admin Dashboard - Manage All Users View ...................................... 30

Figure 4.7: Admin Dashboard - System Settings ................................................. 31

Figure 4.8 Admin Dashboard - Manage Users (Instructors Filtered View) .......... 31

Figure 4.9 Admin Dashboard - Side Navigation Menu ........................................ 32

Figure 4.10 Admin Dashboard - Administrator Profile Page ............................... 32

Figure 4.12 Teacher Dashboard ............................................................................ 33

Figure 4.11 Contact Us ......................................................................................... 33

Figure 4.13 Exam Questions ................................................................................ 34

Figure 4.14 Date Selection Overlay / Teacher Dashboard with Calendar ............ 34

Figure 4.15 Time Picker Overlay / Teacher Dashboard with Time Selection ...... 35

Figure 4.16 Create New Exam / Add Questions Manually .................................. 35

Figure 4.17 Exam Basic Details ........................................................................... 36 Figure 4.18 Exam Document Upload Guidelines ................................................. 36

Figure 4.20 Teacher Dashboard Summary with Recent Exams ........................... 37

Figure 4.21 Teacher Dashboard Summary (Initial) .............................................. 37

Figure 4.22 Teacher Profile .................................................................................. 38

Figure 4.23 Student Results Section ..................................................................... 38

Figure 4.25 Add Question - Details Form ............................................................ 39

Figure 4.24 Add Question - Type Selection .......................................................... 39

Figure 4.26 Add Question - True/False Type ........................................................ 40

Figure 4.27 Add Question - Short Answer Type ................................................... 40

Figure 4.28 Add Question - Matching Type ......................................................... 41

Figure 4.29 Add Question - Fill in the Blank Type .............................................. 41

Figure 4.30 Add Question - Matching Pairs Input................................................ 42

Figure 4.31 Student Dashboard - Available Exams .............................................. 42

Figure 4.32 True False Question........................................................................... 43

Figure 4.33 Multiple Question .............................................................................. 43

Figure 4.34 Brail Input ......................................................................................... 44

Figure 4.35 Short Answer ..................................................................................... 44

Figure 4.36 Fill in The Blank ............................................................................... 45

Figure 4.37 Result Display ................................................................................... 45

Figure 4.38 Student Dashboard - Exam History ................................................... 46

Figure 4.39 Student Dashboard - Exam Details ................................................... 46

Figure 4.40 Training and Validation Performance of Braille Character Recognizer

...................................................................................................................... 47

Figure 4.41 Confusion Matrix of Braille Character Recognizer on Validation Data

...................................................................................................................... 48

Figure 4.42 Classification Report Heatmap (Character-Level Metrics) ............... 49

Figure 4.43 Distribution of Labels in Automatic Grading Dataset....................... 50

Figure 4.44 Confusion Matrix for Automatic Grading System ............................ 51

Figure 4.45 Distribution of Labels in Dataset ...................................................... 52

**LIST OF TABLES**

Table 1 Summary of references and Literatures reviewed. .................................. 15

Table 2 Evaluation on Training Set ...................................................................... 53

Table 3 Evaluation on Validation Set .................................................................... 53

Table 4 Per-Class Classification Report (Validation Set) ..................................... 54

Table 5 Comparison Summary: Training vs Validation ........................................ 54

# ACRONYM

AI: Artificial Intelligence

API: Application Programming Interface

ASAG: Automatic Short Answer Grading

ASR: Automatic Speech Recognition

BVI: Blind and Visually Impaired

CNN: Convolutional Neural Network

DT: Decision Tree

GPU: Graphics Processing Unit

IDE: Integrated Development Environment

JDK: Java Development Kit

JSON: JavaScript Object Notation

KNN: K-Nearest Neighbors

LSA: Latent Semantic Analysis

ML: Machine Learning

NB: Naive Bayes

NLP: Natural Language Processing

NoSQL: Non-Structured Query Language

OCR: Optical Character Recognition

POS: Part of Speech

RNN: Recurrent Neural Network SBERT: Sentence-BERT

SDK: Software Development Kit

SVM: Support Vector Machine

TTS: Text-to-Speech

UI: User Interface

UX: User Experience

WHO: World Health Organization

XML: Extensible Markup Language

# CHAPTER ONE

# INTRODUCTION

## 1.1 Background

Globally, there are 2.2 billion people with a vision impairment and 39 million who are blind according to the World Health Organization (WHO). These populations confront numerous barriers to fair and equal access to quality education, particularly for formal testing circumstances. Although many high-income countries have integrated flexible exam and classroom accommodations using assistive technology (for example, digital Braille, screen readers, and standardized test accommodations delivered through reading, responding, and recording devices), these resources are often too expensive or are not easily available to low-income countries. As a result, many blind students around the world continue to rely on human support during exams, which raises concerns regarding independence, fairness, and privacy.

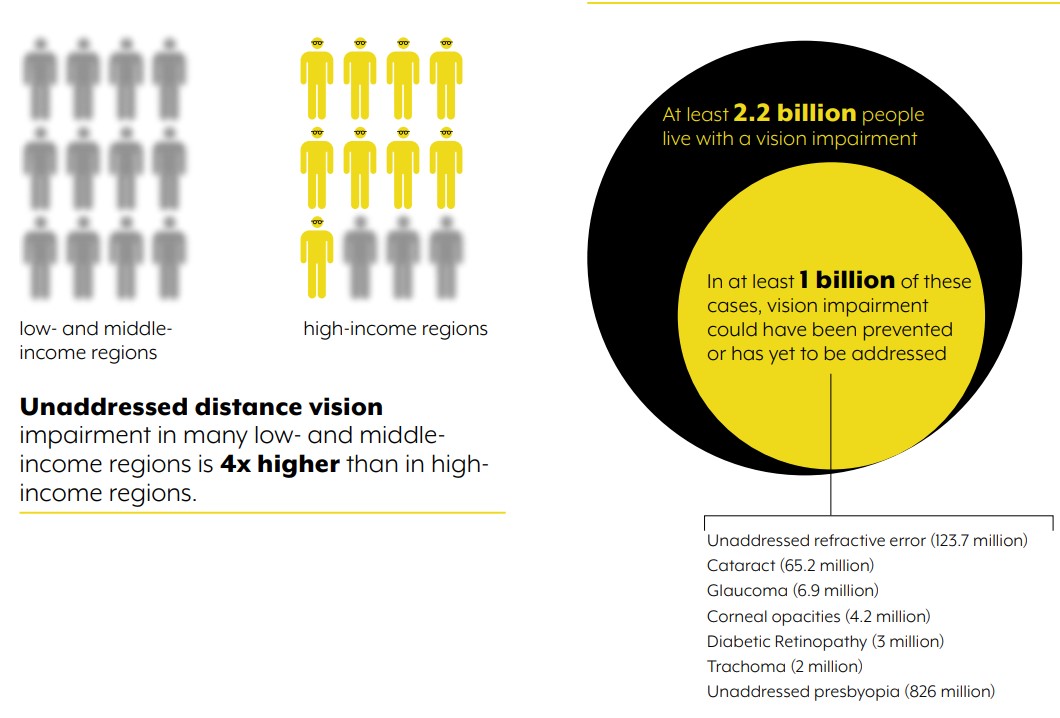


Figure 0.1 World report on vision [1]

The Ministry of Health and disability rights organizations say millions more suffer from low vision in Ethiopia where more than 800,000 are believed to be blind.[2] While the Ethiopian government has promulgated policies of inclusive education, which focuses on the mainstreaming of children with disabilities into primary and secondary schools and even university education, practicing these policies remains an area of difficulty. During tests in particular, blind students frequently have human scribes or personal assistants who read questions to them and record their answers. This not only limits the student’s autonomy but may also affect academic performance and introduce bias in evaluation. **[3]**

Also, the absence of affordable and accessible assistive Technology is considered one of the greatest challenges for blind students in Ethiopia. The majority of schools and universities do not employ digital Braille interfaces, screen readers, or standalone testing. The resources that are available are urban-centric; rural, underserved students are often left without the proper support. Therefore, there is a need for low-cost, locally customizable technological solutions that permit blind students to independently take an exam without requiring the assistance of another person.

To bridge this void, our research plans to design a mobile application, instead, allowing blind students to take examinations through a touch-based Braille input method with a voice-based readout of the questions. The system uses also ML to evaluate free-text answers and employs anti-cheating mechanisms to support the examination process. Through utilizing readily available mobile technology, this application provides a scalable, affordable means to assist visually impaired learners in Ethiopia. This initiative not only promotes academic independence but also aligns with national and global goals of inclusive, technology-driven education.

## 1.2 Problem Statement

Traditional examination systems cause great inconvenience to blind students as these traditionally rely on human assistance for reading questions and writing answers. Not only does this dependence introduce logistical challenges; it should also raise alarm bells about the problems of privacy, bias, transcription mistakes and fairness in assessment that are synonymous with human assessment. Although tools such as textto-speech, Braille input devices, OCR readers, and email-based exam systems currently available provide some support, they are all siloed, technically unreliable, and they are not integrated into the actual workflow of teachers. Challenges like inaccurate speech recognition in noisy environments, formatting inconsistencies, poor synchronization, and the absence of features like automated grading for descriptive answers or robust anti-cheating mechanisms further limit their effectiveness.

This concern has been highly noticeable at Addis Ababa University, 6 Kilo Campus, where visually impaired students face a difficult time in taking exams fairly to the same extent as non-impaired students. What we’re facing here on this campus is a microcosm of the wider problems confronted by blind students in all of Ethiopia where accessible and integrated examination materials are hard to come by.

To address these gaps, this project proposes a comprehensive, Android-based application that provides blind students with an accessible, non-visual exam interface combining audio output, Braille input, real-time feedback, intelligent answer evaluation using semantic similarity for short answer grading, and exam security by freezing Android navigation buttons to prevent app exit. The solution aims to ensure independent, fair, and secure test-taking for visually impaired students across Ethiopia.

## 1.3 Hypothesis

If visually impaired students are provided with a mobile-based examination system that integrates non-visual interaction methods—such as audio guidance and Braille input along with intelligent grading and anti-cheating mechanisms, then they will be able to take exams more independently, accurately, and securely compared to traditional human-assisted methods. This approach is expected to reduce reliance on intermediaries, minimize errors and bias, and improve the overall inclusivity and efficiency of examination processes.

## 1.4 Objective

### 1.4.1 General Objective

To design and develop an autonomous, mobile-based examination platform specifically intended for blind students, which enables seamless access to exam content through audio output, facilitates Braille-based answering via touch-screen input, and incorporates intelligent automated evaluation mechanisms to assess student responses effectively.

### 1.4.2 Specific Objectives

* To convert written exam questions into audio format.
* To Enable Braille-based answering through a mobile interface.
* To Translate Braille responses into readable text.
* To Evaluate descriptive answers using semantic similarity for short answer grading.
* To ensure the system is fully accessible and user-friendly

## 1.5 Scope of the Project

This project focuses on the development of an autonomous mobile application that allows visually impaired individuals to take exams independently and efficiently. The system will include functionality for securely fetching exam questions from a remote server, converting the questions into audio using text-to-speech technology, and allowing students to respond using a Braille input method. The Braille-based responses will be translated into text, which will then be automatically evaluated using a semantic similarity approach for short answer grading. The application will support only English language text-based questions, including multiple-choice and descriptive formats, and will be designed for use on mobile platforms. The scope of the project is limited to the core functions of audio-based question delivery, Braille method input processing, automated answer evaluation, and freezing the three bottom navigation buttons on Android devices to ensure exam security. It does not cover exams in other languages, complex visual content (e.g., diagrams or graphs), handwritten inputs, or subjects requiring advanced mathematical notation.

## 1.6 Significance

This project holds significant value when considering the advancement of inclusive education, and the issues visually impaired individuals face in standard examination halls. It provides examination privacy, independence, and fairness dashing our forebears’ ‘exam-cheating’ finger-pointing ground to dust. The software enables students with visual impairments to independently access exams using audio question reading and Braille method response, fostering a user-friendly, accessible testing environment. Furthermore, the integration of a semantic similarity approach for short answer evaluation introduces an intelligent and consistent grading system, reducing human bias and error. This project contributes to advancing assistive technologies and supports the global movement toward equal opportunities in education for people with disabilities.

## 1.7 Organization of the Thesis

The rest of this thesis is structured as follows. Chapter Two reviews related work in assistive technologies and examination systems for visually impaired students. Chapter Three explains the methods and materials used to develop the mobile application. Chapter Four presents the results and discusses the system’s performance and limitations. Chapter Five provides recommendations and outlines possible future improvements.

# CHAPTER TWO

## LITERATURE REVIEW

# 2.1. General Literature on Blind Examination Using Mobile Apps

Al-Eidarous et al.**[4]** developed Exam Voice, a mobile-based system that enables blind and visually impaired (BVI) students to take exams independently using Bluetooth earphones for audio question delivery and a tactile Braille-marked answer sheet. Students respond by placing split pins into labeled holes (A–D), allowing examiners to evaluate answers visually without needing Braille literacy. The system promotes inclusion by allowing BVI students to participate alongside sighted peers and removes dependence on human scribes. However, it only supports multiple-choice questions, with no input method for essays or short answers—limiting its use in broader assessment contexts and identifying a key area for future development.

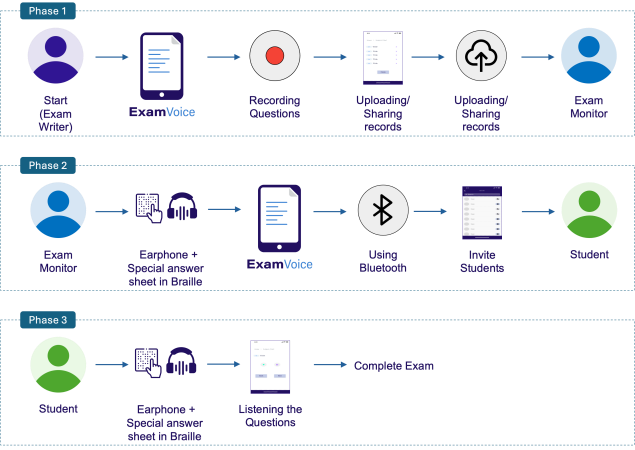


Figure 2.1.1 Detailed interaction diagram of the Exam Voice system components

Thowfeek **[5]** proposed a fully voice-controlled online exam system for visually impaired students, using speech synthesis to deliver questions and speech recognition to capture verbal answers. Students navigate the exam through voice commands (e.g.,

“Next,” “Repeat,” “Submit”), and answers—including essays—are auto-evaluated using a semantic similarity algorithm. The system enables hands-free, independent exam-taking and supports multiple question types. However, it is highly sensitive to noise, supports only English, and struggles with accent variations. A major gap arises when multiple students are in the same room, as spoken answers increase the risk of plagiarism or answer sharing, highlighting the need for improved privacy, noise filtering, and individualized testing environments.

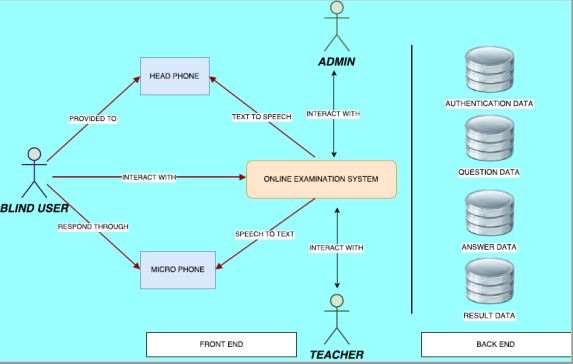


Figure 2.1.2 Top Level System Architecture for Voice-Based Online Examination

System

## 2.2. Brail Input Method

Shokat et al.[6] proposed a position-free Braille input technique for Android devices that enables visually impaired users to input only the necessary Braille dots anywhere on the screen, eliminating the need for fixed layouts and reducing memorization effort. A dataset of 1,284 images (Grade 1 Braille a–z) was collected from 24 visually impaired students and processed using deep learning models, with Google Net achieving the highest classification accuracy of 95.8%. The method outperformed traditional classifiers such as SVM, KNN, and Decision Trees. While the system enhances accessibility with minimal training and tactile feedback, it currently lacks support for Grade 2 Braille, real-time interaction, and error correction.

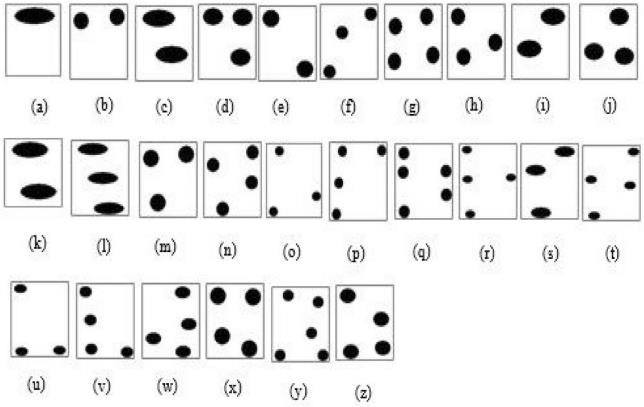


Figure 2.1.3 Sample Braille input is taken from a participant.

Niazi et al. **[7]** proposed a Touch Sensitive Keypad Layout, designed to enhance smartphone usability for blind and visually impaired users by using screen division and larger, strategically placed touch zones. Their system segments the smartphone screen into five functional areas, each dedicated to a core task like dialing, messaging, emailing, and internet access. The interface provides audio and haptic feedback and uses gestures for navigation and input. In a comparative study with 25 blind participants from Pakistan and Afghanistan, the proposed keypad significantly outperformed standard QWERTY and ordinary keypads in usability. It showed reduced semantic loss, fewer errors, and higher ease-of-use ratings, particularly in typing and dialing tasks. ANOVA testing confirmed statistical significance in improved performance. Users reported that the interface was easy to learn, reduced cognitive load, and allowed onehanded use with minimal finger movement. Future work includes further refinement of gesture-based controls and broader longitudinal evaluation.

Oliveira et al. **[8]** introduced BrailleType, a text-entry method that enables blind users to input Braille characters on touchscreen devices using large, easy-to-locate targets. Unlike multi-touch systems, BrailleType uses single-finger input and audio feedback to guide dot selection. Evaluated against VoiceOver with fifteen blind users, BrailleType was slower (1.45 WPM vs 2.11 WPM) but significantly more accurate (8.91% vs 14.12% error rate). Users found it easier to learn and use, particularly due to fewer errors from mis-taps. Feedback emphasized its accessibility and reduced cognitive load.

Planned improvements include adjustable timeouts and long-term usability evaluation.

Govindarajan **[9]** proposed Eyedroid, a layout-free Braille input method using swipe and tap gestures on Android devices to help visually impaired users input text without relying on-screen positions. Each character is formed with three gestures matching the Braille cell's rows, supported by voice feedback. Tested on 12 blind participants, the system showed a significant reduction in input time and improved accuracy after training. Eyedroid removes the need for screen navigation and is ideal for Brailleliterate users, though it lacks tactile feedback and support for non-Braille users. Future enhancements include adding tactile output and broader mobile functionalities.

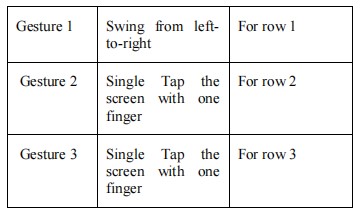
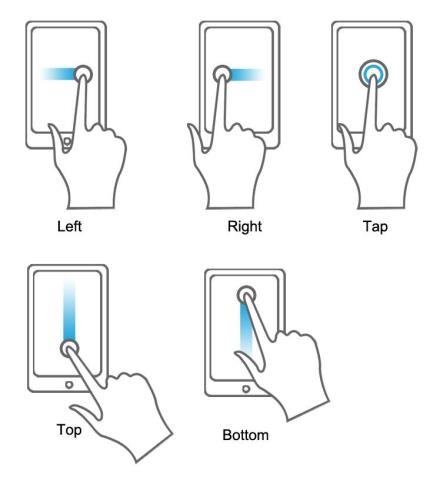


Figure 2.1.4 Basic gestures Figure 2.1.5 Braille-Gesture of Character

Siqueira et al.**[10]** developed BrailleÉcran, a text entry system for blind users that integrates an Android app with a 3D-printed tactile screen overlay to simulate a physical Braille interface. The app allows users to activate Braille dots via vertical finger swipes, confirmed through audio and vibrotactile feedback, eliminating the need for visual navigation. Users can input characters by touching corresponding Braille cells in any sequence (e.g., 12 or 21 for "b") and use gestures to confirm, delete, or send messages. The tactile overlay helps in spatial orientation, guiding users with raised paths and checkpoints. Though still a prototype tested by a single experienced blind user, the system shows promise for accessible smartphone use, and plans include multi-user testing and customizable interface covers.

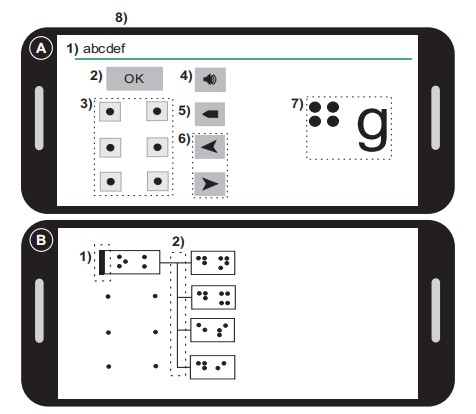
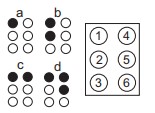


Figure 2.1.6 Braille alphabet, Figure 2.1.7 A. App Interface and B. letters a, b, c and d. Tactile Interface

Paisios et al. **[11]** proposed and evaluated four Braille-based text-entry methods for touchscreen smartphones aimed at improving accessibility for visually impaired users. The input methods include One-Finger, Split-Tap, Two-Finger, and Thumb-Typing, and were compared to an existing Nine-Digit method. A user study with 15 blind or legally blind participants found that the One-Finger method, which uses spatial taps to indicate Braille dots and employs a pattern-matching algorithm for interpretation, was the most intuitive and preferred. Users valued its natural feel and minimal training requirement. In contrast, the Split-Tap method, though more precise, was slower and frustrating. The Two-Finger and Thumb-Typing methods demanded more cognitive load and were harder to learn. Overall, the One-Finger method achieved a strong balance of simplicity and accuracy. Future improvements include dynamic adaptation to user behavior and customizable input intervals.

Mascetti et al. **[12]** proposed TypeInBraille, a gesture-based Braille input system using taps and flicks. Compared to the QWERTY keyboard, it improved typing speed and accuracy in both quiet and noisy environments. Tested with ten blind users, it proved effective after minimal training. A key limitation is its two-handed use, prompting the future development of thumb-only input.

Jayant et al. **[13]** introduced V-Braille, a novel tactile interface that enables deaf-blind users to perceive Braille characters on standard touchscreen phones using localized vibrations. V-Braille divides the touchscreen into six regions representing the six dots of a Braille cell and activates vibration when a user touches a region corresponding to a raised dot. A pilot study involving nine deaf-blind and blind users, with an average of 25.8 years of Braille experience, found that 90% of Braille characters were identified correctly with minimal training. Reading times ranged from 4.2 to 26.6 seconds per character, with the potential for significant speed gains with experience. Six of nine users successfully read and comprehended a full sentence. Participants praised the system's accessibility and suggested adding tactile overlays to improve area distinction. V-Braille was well-received, especially due to its use of off-the-shelf smartphones and potential as a cost-effective communication aid. Future development aims at integrating

Braille input methods and expanding to full communication systems

## 2.3. Semantic similarity

Qurashi et al.**[14]** proposed an NLP-driven document processing framework to automatically measure semantic similarity between safety-critical railway documents, eliminating the need for manual clause comparison and enhancing consistency across evolving rule sets. Using a dataset comprising the Enterprise Architecture (EA) model and the Operational Concept Document (OCD), the system employed pre-processing, tokenization, and Word2Vec-based vectorization, followed by the application of Jaccard and Cosine similarity metrics. Experimental results showed that Cosine similarity achieved higher accuracy in identifying equivalent rules, with scores reaching 100% in some cases, outperforming Jaccard’s character-based method. The solution was implemented using tools like Python, Gensim, and NLTK, and proved effective in identifying matching clauses despite structural variations. However, limitations remain in actor-role mapping, partial clause content in UML files, and lack of real-time document updates.

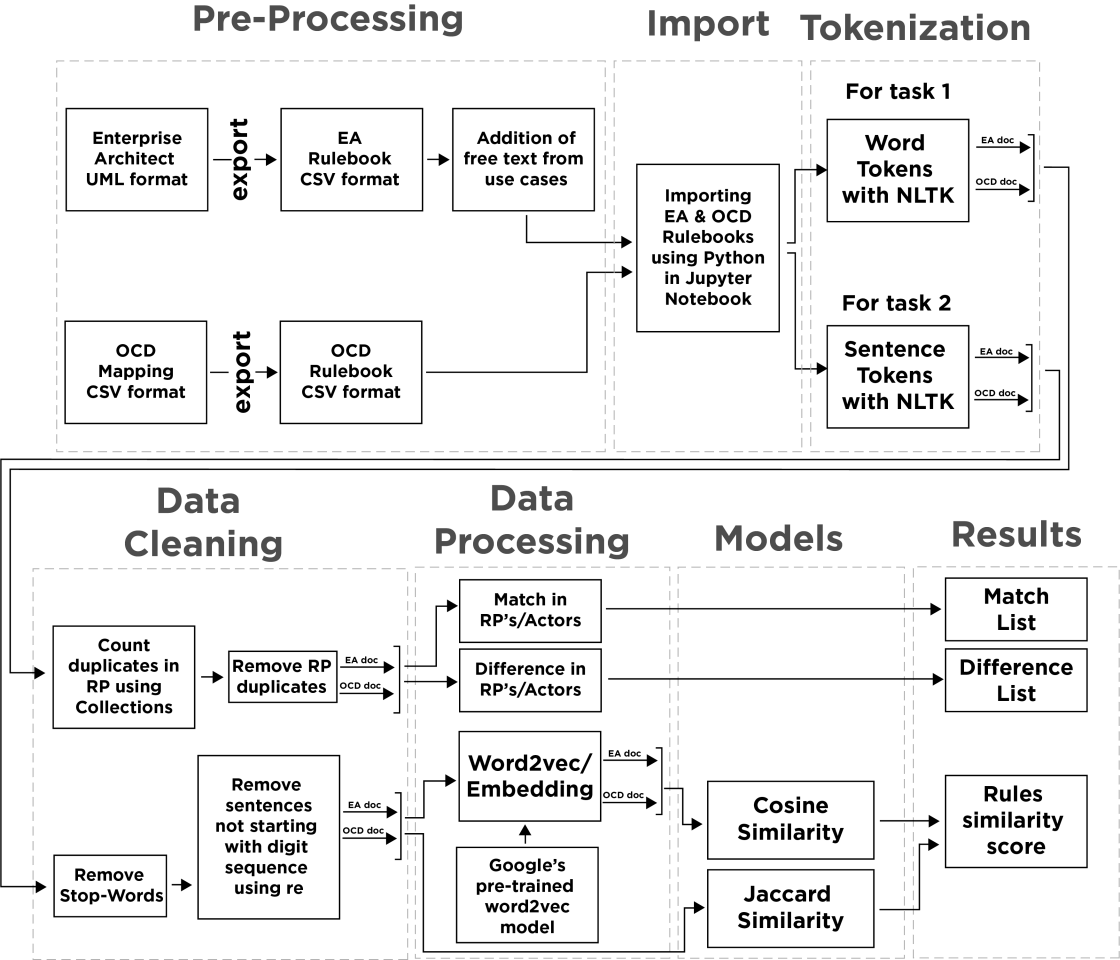


Figure 2.1.8 Model for text similarity using different algorithms

Mihalcea **[15]** proposed an unsupervised text similarity approach for automatically grading short student answers by comparing them with instructor-provided correct answers. The system evaluates multiple knowledge-based and corpus-based similarity measures, including WordNet and Latent Semantic Analysis (LSA), and introduces a novel feedback mechanism that enhances grading accuracy by incorporating topperforming student responses into the reference answer. Using a dataset of 630 short answers from undergraduate computer science students, the model achieved a peak Pearson correlation of 0.51, outperforming traditional tf-idf baselines and matching human annotator agreement levels. Although domain-specific corpora improved results significantly, the system’s effectiveness depends on answer variability and the richness of training data. Limitations include sensitivity to phrasing and lack of deep syntactic analysis, but the approach demonstrates strong potential for scalable, automated assessment in educational settings.

Galhardi & Brancher **[16]** conducted a systematic review of 44 studies employing machine learning methods for automatic short answer grading (ASAG), focusing on dataset characteristics, NLP preprocessing, learning algorithms, feature types, and system evaluations. The review identified 28 distinct datasets, with most works applying supervised learning approaches such as SVM, decision trees, and neural networks, along with preprocessing steps like lemmatization, POS tagging, and WordNet-based enrichment. Lexical features like n-grams and TF-IDF were most common, followed by syntactic structures and semantic measures such as LSA and entailment modeling. The evaluation showed the best-performing systems used ensemble learning and domain-specific models, achieving high accuracy on public datasets like ASAP-SAS and SciEntsBank. Despite progress, challenges remain in standardizing datasets and extending deep learning integration across diverse question types and languages.

Another **[17]** research proposes a novel deep learning approach for detecting semantic similarity between text pairs, specifically focusing on duplicate questions from Quora's dataset. The authors introduce a hybrid model combining Long Short-Term Memory Network (LSTM) and Convolutional Neural Network (CNN) to leverage both longterm dependency learning and local feature extraction. The model takes sentence pairs as input and, after minimal preprocessing including tokenization, stop word removal, and stemming, utilizes Google's word2vec for word embeddings. The experimental results demonstrate that this combined LSTM-CNN model achieved an accuracy of 87.50% on the Quora dataset, outperforming existing techniques and individual CNN or LSTM models in precision, recall, and F1 score.

This **[18]** systematic literature review provides a comprehensive overview of ShortText Semantic Similarity (STSS), highlighting techniques, challenges, and future directions. The authors detail the inherent difficulties in STSS, such as ambiguity, polysemy, sparsity, and various linguistic anomalies in short texts. They categorize deep learning techniques for STSS, focusing on Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and transformer models (BERT, ELMo, GPT-2), emphasizing their ability to generate high-level contextual representations. The review also identifies common datasets (e.g., Quora, Mohler, SciEntsBank) and discusses challenges in short question-answering systems (SQAS), including the lack of standardized datasets and difficulties in syntactic interpretation, while offering suggestions for future improvements.

Another **[19]** paper investigates methods for learning semantic similarity in very short text fragments, a challenge for traditional word overlap-based methods like TF-IDF due to limited shared vocabulary. The authors explore distributed word representations (word embeddings) and propose a novel hybrid approach that combines word embeddings with TF-IDF information. By weighing word vectors with their inverse document frequency (idf) values, the model reduces the impact of less informative, frequently occurring words, thereby enhancing the discriminative power between semantically similar and dissimilar short texts. Experiments conducted on a toy dataset of Wikipedia articles, using Euclidean distance as a metric, demonstrate that this importance-factor-based method significantly outperforms traditional TF-IDF and naive word embedding aggregation techniques in reducing error rates.

This **[20]** survey provides an overview of Semantic Textual Similarity (STS) methods, tools, and applications, categorizing measures into topological/knowledge-based, statistical/corpus-based, and string-based approaches. It emphasizes topological methods, particularly those using WordNet taxonomy, for their ability to disambiguate ambiguous words and understand semantic relations (e.g., IS-A, hypernym-hyponym). The paper also details various statistical measures like Latent Semantic Analysis (LSA), Explicit Semantic Analysis (ESA), and Normalized Google Distance (NGD), which build vector space models from corpora. Additionally, it covers string-based similarity measures (character-based like LCS and term-based like Cosine Similarity) and lists available software and tools. The authors propose a new language model-based semantic network combining WordNet taxonomy with unigram language models to enhance similarity detection.

Table 1 Summary of references and Literature reviewed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Activity** | **Methods** | **Results** | **Authors** | **Year** |
| 1 | Enhancing exam  accessibility for blind and visually impaired students | Android-based system with  Bluetooth audio, tactile Braille-marked sheets | 100% independence; 0% need for scribes; 80% inclusion; 60% grading simplification | Al-Eidarous et al. | 2024 |
| 2 | Voicecontrolled exam system for  visually impaired students | TTS, ASR,  semantic similarity grading | 95% voice  recognition;  90% exam  independence; supports essays | Thowfeek | 2021 |
| 3 | Positionfree Braille input on touchscreen devices | Braille dot input with  GoogLeNet CNN; compared to SVM, KNN,  DT | 95.8% accuracy; 10– 25% better than traditional classifiers | Shokat et al. | 2020 |
| 4 | Layout-free Braille input using swipe/tap gestures | Gesture-based input; Android with voice  feedback | Significant accuracy improvement post-training | Govindarajan | 2018 |
| 5 | BrailleÉcran tactile | Multi-point Braille input; tactile overlay; | 90% spatial accuracy; 85% input accuracy; | Siqueira et al. | 2016 |
|  | Braille input | haptic/audio feedback | full user  approval |  |  |
| 6 | Semantic similarity for grading descriptive answers | WordNet,  LSA, top-  performing answer enrichment | 0.51 Pearson  correlation; matches human grading | Mihalcea | 2009 |
| 7 | NLP for  semantic similarity in safety documents | Word2Vec +  Cosine/Jaccard similarity | Cosine reached up to 100%  accuracy | Qurashi et al. | 2020 |
| 8 | Machine learning review for short answer grading | Review of 44 studie using  SVM,  ensemble, and semanticmodel | Identified best practices; dataset standardization needed | Galhardi &  Brancher | 2018 |

## 2.4. Literature Gaps

Despite significant advancements in assistive technologies for visually impaired individuals, the reviewed literature reveals two major gaps that remain unaddressed. First, while several studies have proposed innovative Braille input methods such as position-free tapping interfaces and gesture-based systems these solutions are limited to text entry and lack integration with intelligent answer evaluation mechanisms. There is currently no unified mobile application that combines Braille-based input with automatic short-answer grading using semantic similarity models. This gap highlights the need for a comprehensive solution that enables both accessible input and intelligent assessment within a single platform.

Second, most existing voice-based exam systems [4] focus on audio question delivery and speech recognition but struggle with privacy, noise interference, and academic integrity, especially in shared testing environments. These systems are not designed to provide secure, individualized exam sessions or prevent cheating. Moreover, they often rely on cloud services for processing, which raises concerns about data privacy and reliability in low-resource settings. Therefore, a critical gap exists in developing a privacy-preserving and secure mobile exam system tailored for visually impaired students.

## 2.5. Proposed Solution

To address the challenges faced by visually impaired students during examinations, this project proposes the development of an Android-based mobile application that allows users to take exams independently through a fully non-visual interface. The application delivers exam questions in audio format using text-to-speech (TTS) technology and enables students to respond using a position-free Braille input method on a touchscreen. With this design, the visual help and human presence are removed from the exam process altogether which helps promote fairness, privacy, and academic independence at its best. It has a navigation lock function to limit users use of the back, home, and recent apps buttons during the testing, and to restrict them from accessing other applications or the device to prevent them from cheating or tampering, they cannot open any other apps while the exam is ongoing.

In addition to its accessible user interface, the proposed solution integrates machine learning for the automated grading of descriptive answers. By using the SentenceBERT (SBERT) model, the system calculates the semantic similarity between student responses and reference answers, providing an intelligent and consistent grading mechanism that reduces human bias. The backend, developed using Firebase, manages secure data storage, user authentication, and real-time synchronization of exam data. The entire system is designed to be low-cost, scalable, and adaptable to the local educational context in Ethiopia, supporting English-language exams for blind students across various institutions. This solution bridges critical gaps in assistive educational technology and sets the foundation for a more inclusive and independent academic assessment environment.

# CHAPTER THREE

# METHODOLOGY

This chapter details the methodological approach undertaken in the development of an Android-based Autonomous Blind Examiner Mobile Application. The project focuses on creating an inclusive and independent exam-taking experience for visually impaired students through the design of a non-visual user interface, intelligent grading features, and robust security mechanisms. The methodology encompasses data collection, frontend and backend development, machine learning integration, and rigorous testing to ensure accessibility, usability, and reliability.

## 3.1. Methods

The project follows a structured methodology to develop an Android application that facilitates autonomous exam-taking for visually impaired students. This includes the design of a non-visual user interface and the integration of intelligent grading and security features.

### 3.1.1. Data Collection

Data for the autonomous mobile examination application was primarily sourced from existing literature and publicly available datasets. This included pre-existing Braille input datasets for training the Braille recognition system and compiled short-answer exam questions and reference answers from educational materials for automatic grading based on semantic similarity.

### 3.1.2. Public Datasets and Resources

To support both the Braille input recognition and automatic short answer grading features, a combination of public datasets and resources from existing literature was utilized.

Braille Input: For Braille input, the project referenced datasets such as those used by Shokat et al. who collected over 1,200 Grade 1 Braille character images from visually impaired users to train deep learning models for accurate Braille recognition. These datasets informed the design and training of the position-free touchscreen Braille input system implemented in this study.

Grading System: Publicly available educational datasets from platforms like Kaggle were employed, containing short-answer question-answer pairs across subjects to train and validate the semantic similarity model. Additionally, pre-trained models such as Sentence-BERT (SBERT) from Hugging Face were integrated to enhance grading accuracy by capturing meaning beyond exact wording. Literature on machine learningbased grading techniques (Galhardi & Brancher, 2018) and semantic analysis (Mihalcea, 2009) also guided dataset selection and model evaluation strategies.

### 3.1.3. Frontend/User Interface (UI) Development

The mobile application's front end was developed using Android Studio, prioritizing accessibility, simplicity, and independence for visually impaired users. The user interface eschews visual elements, relying instead on audio feedback and touch-based interactions. A [21] Position-Free Touch Screen-Based Braille Input Method was implemented, allowing users to tap anywhere on the screen in sequences that match the six-dot Braille system.

The initial setup involved installing Android Studio according to the official guide, which included the Android SDK,[22] emulator, and development tools. SDK and JDK directories were configured, and necessary licenses were accepted. A new project was created with accessibility libraries and voice interface support. Key features like Textto-Speech (TTS) [23] for question delivery and gesture detection for Braille input were then added, laying the groundwork for an inclusive and interactive exam experience for blind students.

###### 3.1.3.1. Implementation

The front end was developed using Kotlin in Android Studio, chosen for its efficiency, safety features, and smooth integration with Android development tools. Its concise syntax facilitated code readability and maintainability.

**Non-Visual Interface:** All screens were designed using XML layout files from the Android SDK. Each layout was carefully structured to support interaction without relying on visuals, with functionality and navigation handled through audio guidance and gesture-based input.

**Braille Input System:** A core frontend feature is the Braille input system, enabling users to tap anywhere on the screen in patterns representing Braille characters. This position-free design enhances usability for visually impaired users who may not be able to locate fixed buttons. The logic for this system was implemented using Kotlin.

**Text-to-Speech (TTS):** TTS technology was integrated to read out questions, instructions, and feedback aloud, which is essential for independent navigation through the exam.

**Exam Security:** To enhance user experience and exam security, a mechanism was added to disable the device’s navigation buttons (Home, Back, and Recent Apps) [24]during exams, preventing unauthorized exits from the application during a test session.

### 3.1.4. Backend Development

The backend of the mobile application ensures smooth user management, secure data storage, and real-time synchronization between the exam system and the user interface. Firebase, [25] a Google platform offering cloud-based backend services, was utilized for these features. Firebase was selected due to its ease of integration with Android Studio, real-time database capabilities, and built-in support for authentication and secure data access, all crucial for a reliable exam system for visually impaired students.

###### 3.1.4.1. Firebase Integration

Firebase was integrated into the project through Android Studio using the official Firebase SDK. The following services were utilized:

**Firebase Authentication:** This feature facilitates secure user logins (students and administrators) using email and password credentials, managing access and userspecific exam records.

**Cloud Firestore:** Firebase's real-time NoSQL database was used to store and retrieve exam questions, user responses, and evaluation results. Real-time data synchronization is crucial for consistency during exam sessions.

**Firebase Storage:** This service was used to store static files such as audio instructions or future media assets.

**Firebase Realtime Database Rules:** Secure access control rules were implemented to restrict data access to authenticated users, ensuring privacy and data integrity. The Firebase console provided a user-friendly interface for monitoring user activity, managing data collections, and debugging application issues efficiently.

###### 3.1.4.2. Backend Implementation

Logic was developed in the backend to handle various tasks, including user registration, question delivery, and response recording. All communication between the front end and Firebase was managed through Kotlin [26] code within the application’s activity and fragment classes. Upon student login, Firebase Authentication verifies their identity.

Once authenticated, the application fetches exam questions from Firestore based on the selected subject or exam code. These questions are then converted to speech by the frontend’s Text-to-Speech system.

As the user provides answers through the Braille input method, each answer is saved locally and simultaneously updated in the Firestore database. This dual approach ensures responses are not lost in case of connectivity issues, as they are queued and synced once a stable connection is re-established.

Firebase’s timestamping features were used to record when each question was answered, enabling future enhancements like time tracking and performance analytics.

Firebase’s real-time features facilitated responsive behavior, allowing, for instance, an administrator to monitor exam progress or receive submitted answers in real time.

### 3.1.5. Machine Learning Integration

Machine learning is a core component of the application, enabling intelligent interaction and automated grading essential for independent exam-taking by visually impaired students. Two major areas of machine learning integration were implemented: Braille input recognition and semantic similarity-based grading of descriptive answers. Public datasets, pre-trained models, and custom-collected data were utilized to train and evaluate model performance, ensuring accurate and context-aware system behavior. The integration of machine learning improves the accessibility, usability, and scalability of the application.

###### 3.1.5.1. Braille Input Integration

To facilitate position-free Braille text entry on touchscreens, a custom Braille input system was developed and integrated using machine learning models trained on Grade 1 Braille data. A dataset of Braille dot patterns was collected from visually impaired users, simulating real-world touchscreen interactions. These inputs were processed using deep learning models, informed by prior research such as[6], where GoogLeNet and RNN architectures demonstrated high accuracy in recognizing Braille characters. The model detects user taps anywhere on the screen and classifies them into corresponding Braille characters without requiring fixed input positions. This approach minimizes memorization effort and enhances user comfort and input speed, enabling blind users to interact fluidly with the exam system.

###### 3.1.5.2. Text Similarity Analysis

Machine learning is a core component of the application, enabling intelligent interaction and automated grading essential for independent exam-taking by visually impaired students. Two major areas of machine learning integration were implemented:

Braille input recognition and semantic similarity-based grading of descriptive answers. A Convolutional Neural Network (CNN) model [27] was used for Braille input recognition due to its strong performance in spatial pattern detection. For automated grading, Sentence-BERT (SBERT) [28] was employed in combination with cosine similarity to measure the semantic closeness between student answers and reference responses. Public datasets, pre-trained models, and custom-collected data were utilized to train and evaluate model performance, ensuring accurate and context-aware system behaviour. The integration of machine learning improves the accessibility, usability, and scalability of the application.

### 3.1.6. Testing and Quality Assurance

The project involved thorough testing of the application's functionalities, including Braille input accuracy, audio question delivery, automated grading, and overall performance.

###### 3.1.6.1. Functionality Testing

Both manual and automated testing were performed to detect and fix bugs. Manual tests covered Braille input, audio output, grading accuracy, and exam security. Automated tests validated backend processes, data synchronization, and system stability under different conditions.

###### 3.1.6.2. Usability Testing

The interface and features were refined by enhancing audio guidance, input sensitivity, and navigation to ensure a user-friendly experience.

## 3.2. Materials

### 3.2.1. Hardware Requirements

**Testing Devices:** Samsung M15 5G, different Google emulators

**Braille Displays:** BRLTTY-compatible Braille displays (40-cell and 80-cell), Humanware Brailliant BI 40X, and Freedom Scientific Focus 14 Blue.

**Audio Equipment:** Professional headphones for audio testing, external speakers for group testing sessions, and audio recording equipment for user interviews.

**Haptic Testing Devices:** Various Android devices with different haptic feedback capabilities and tactile feedback measurement tools.

### 3.2.2. Software and Development Tools

**Development Environment:** The primary IDE used was Android Studio. The programming language was Kotlin, and the build system was Gradle. Version control was managed with Git and a GitHub repository. Project management utilized Jira for task tracking and Confluence for technical documentation.

**Frameworks and Libraries:** The application integrated various Android and thirdparty libraries. Key components included core Android and Jetpack Compose libraries for UI and navigation [29], Firebase for backend services, and TensorFlow Lite for machine learning functionalities. Additionally, libraries for accessibility and security features were incorporated to enhance the user experience and protect data.

**Machine Learning Tools:**

**Model Development:** TensorFlow, Keras, Python, and Jupyter Notebook were used for model development and experimentation.

**Data Processing:** NumPy, Pandas, OpenCV (for image processing), and Scikit-learn (for data analysis) were used for data processing.

**Model Optimization:** TensorFlow Lite Converter [30], TensorFlow Model Optimization Toolkit, and quantization and pruning tools were employed for model optimization.

### 3.2.3. Cloud Infrastructure and Services

Firebase Services: The application utilized Firebase Auth for user management, Cloud

Firestore for real-time data synchronization, [31] Firebase Storage for file management, Firebase Analytics for usage tracking, Firebase Crashlytics for error reporting, and Firebase Performance for app optimization.

Additional Cloud Services: Google Cloud Platform's Compute [32]Engine was used for ML model training, Cloud Storage for dataset management, and Cloud Functions for serverless operations. CloudFlare was used for global content delivery, and Google Analytics along with custom monitoring solutions was employed for monitoring.

##### 3.2.4. Testing and Quality Assurance Tools

Automated Testing Tools: Unit testing employed JUnit and Mockito. UI testing used Espresso and Compose Testing. Accessibility testing included Android Accessibility Scanner, Espresso Accessibility Checks, and custom accessibility validation tools. Performance testing utilized Android Profiler, Firebase Performance Monitoring, and custom performance benchmarking tools.

Manual Testing Tools: Screen reader testing involved TalkBack (various versions) [33] and Voice Assistant integration testing. Accessibility evaluation utilized WAVE (Web Accessibility Evaluation Tool), axe DevTools for mobile, and Color Contrast Analyzers. Security testing involved OWASP Mobile Security Testing Guide tools, custom penetration testing scripts, and security vulnerability scanners.

### 3.2.5. Data Collection and Analysis Materials

**Research instruments:** User interview guides (structured and semi-structured), usability testing protocols (task-based scenarios), accessibility evaluation checklists (WCAG 2.1 compliance),[34] performance measurement tools (custom metrics collection frameworks), and user satisfaction and feedback questionnaires were used.

**Data Analysis Software:**

**Statistical Analysis:** Python with SciPy and StatsModels [35] were used for quantitative analysis.

**Qualitative Analysis**: NVivo and Atlas.ti was used for qualitative data analysis, coding, and theme identification.

**Visualization**: Tableau, Python Matplotlib Seaborn, and D3.js were used for data visualization.

### 3.2.6. Other Tools

**Canva:** Canva was used to draw flowcharts for the project, leveraging its simple interface and ready-made templates for creating clear, organized diagrams, and facilitating team collaboration.

**EndNote:** for referencing

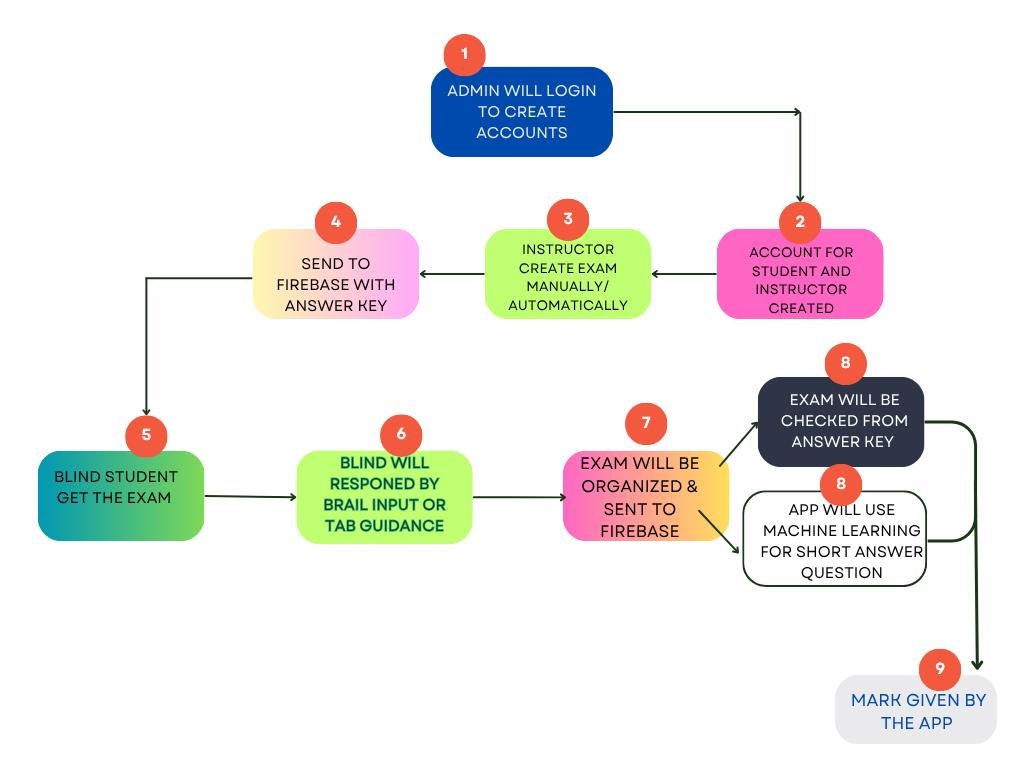


Figure 0.1 Flowchart for the whole app

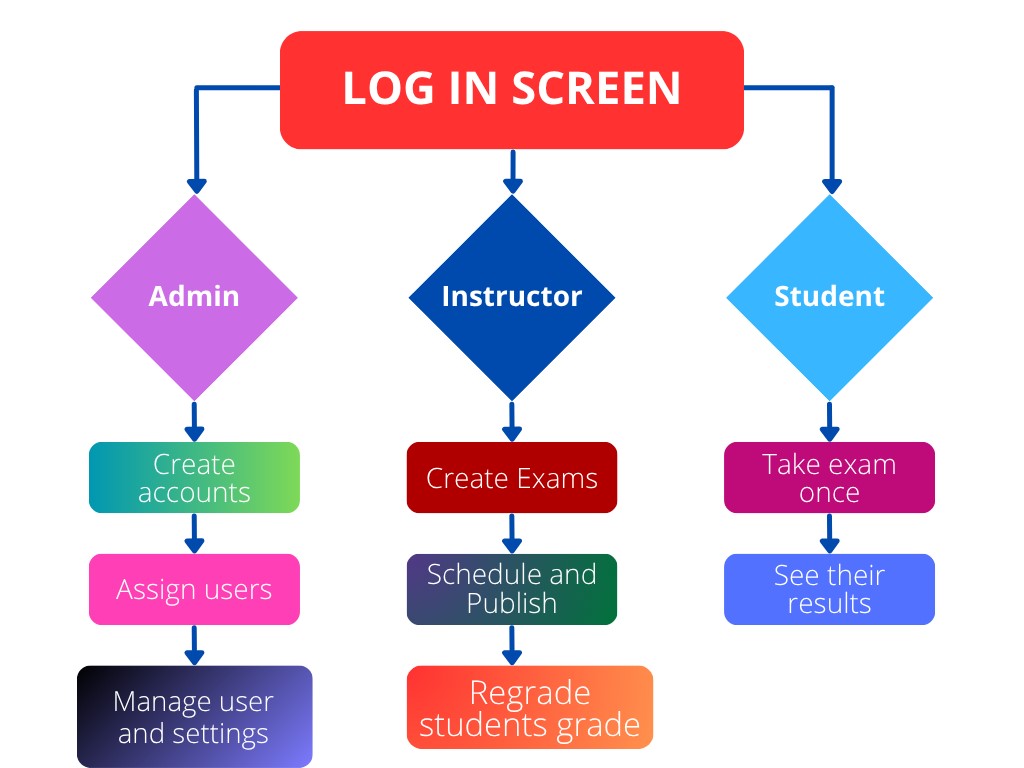


Figure 0.2 Flowchart for exam-taking system

# CHAPTER FOUR

# RESULTS AND DISCUSSIONS

This chapter presents the outcomes of our application testing and what they mean in real use. The mobile app worked as intended blind students can able to listen to questions through audio and respond using the Braille input system without needing anyone's help. The navigation was smooth, and the system’s security features, like locking the home and back buttons, ensured students stayed focused during the exam. Instructors and admins also found the system easy to use for managing exams and tracking student performance.

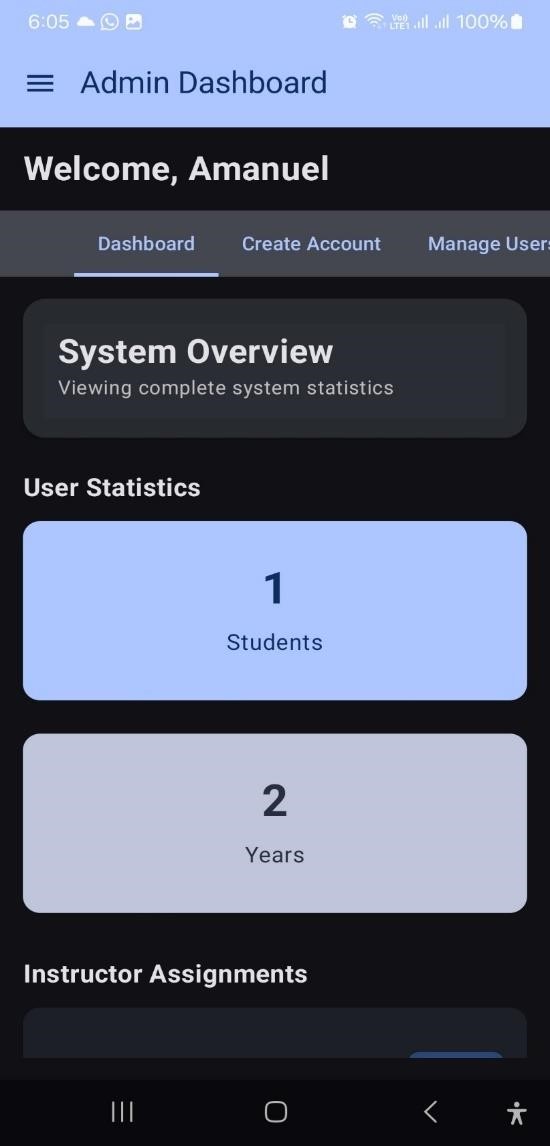
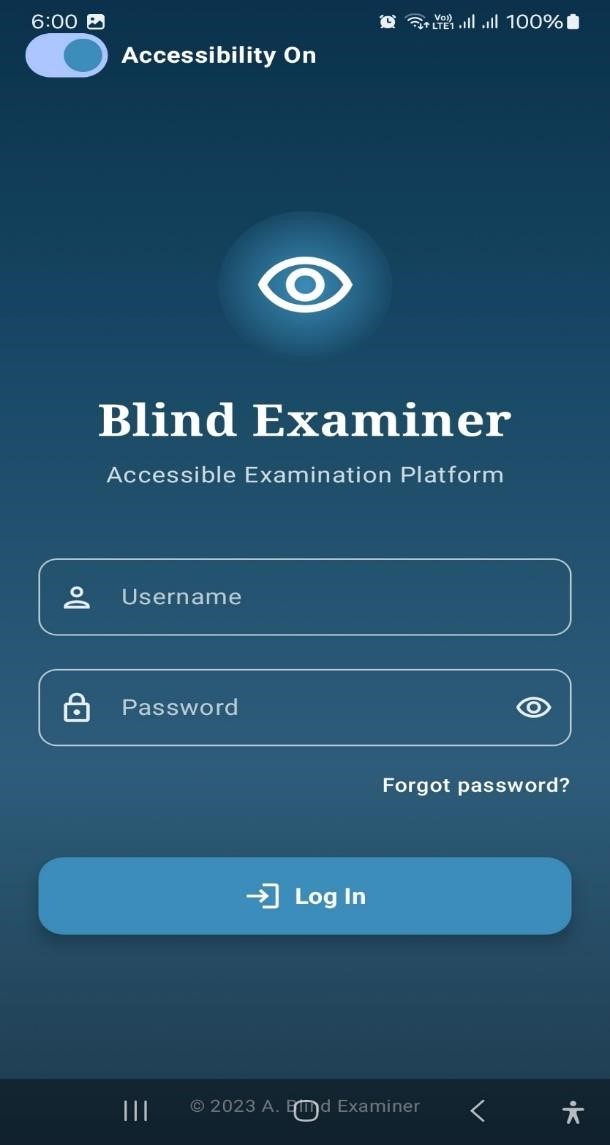


Figure 0.1 Login page Figure 0.2 Admin dashboard

Figure 4.1 Displays the Blind Examiner: Accessible Examination Platform login page, emphasizing accessibility. Figure 4.2 shows Amanuel's Admin Dashboard, providing a system overview with user statistics (1 student, 2-year metric) and sections for instructor assignments, indicating a comprehensive educational management system.

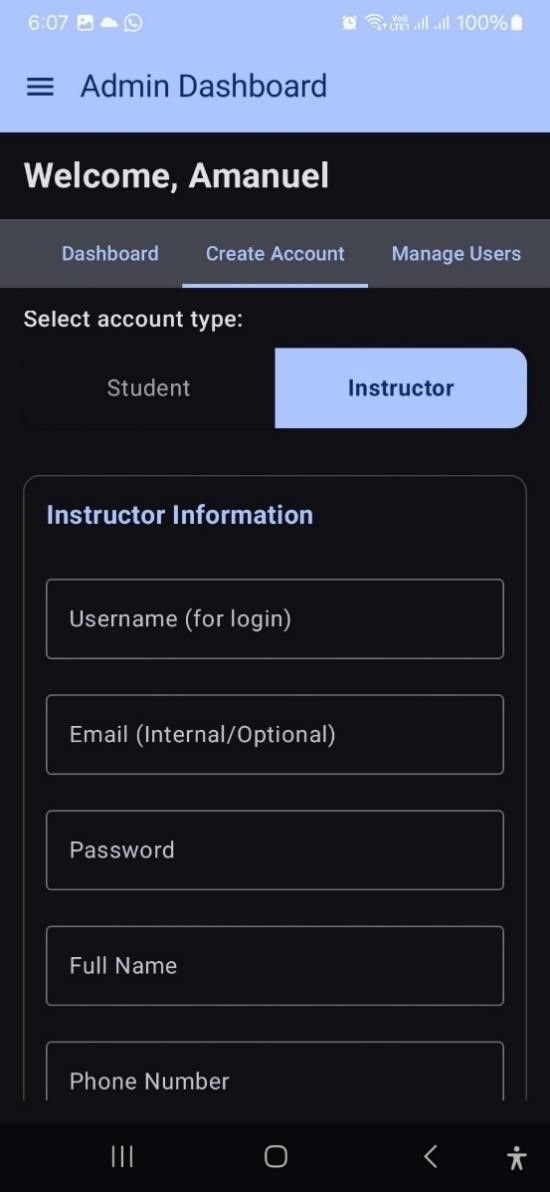
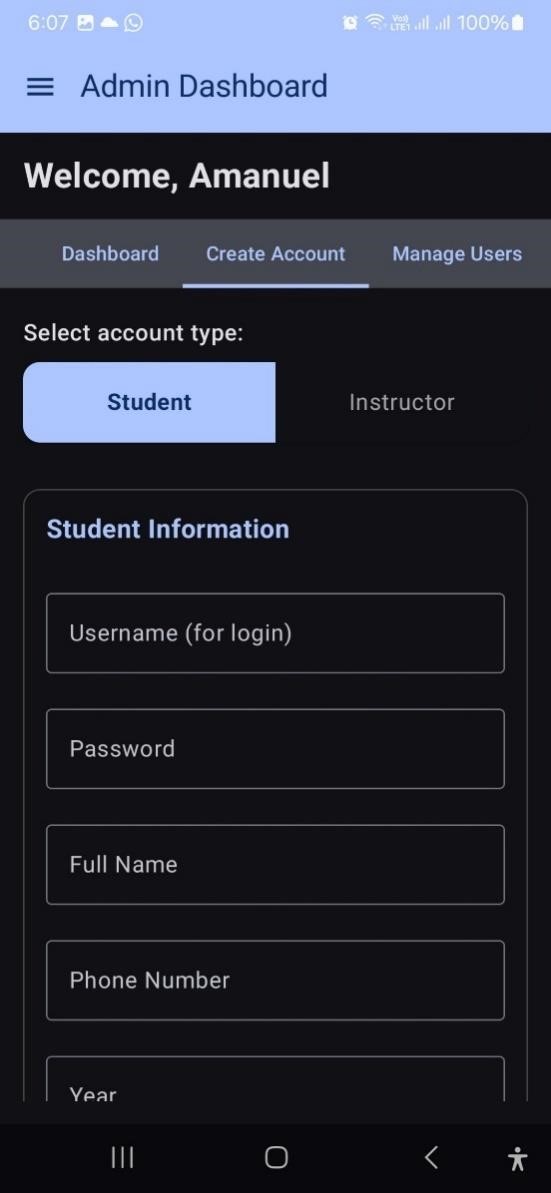


Figure 0.3 Admin Dashboard: Create Figure 0.4 Admin Dashboard: Create

Student Account Instructor Account

Figure 4.3 shows the interface configured for creating a "Student" account, prompting for Student Information including Username (for login), Password, Full Name, Phone Number, and Year. In contrast, Figure 4.4 illustrates the interface for creating an "Instructor" account, requiring Instructor Information such as Username (for login), Email (Internal/Optional), Password, Full Name, and Phone Number.

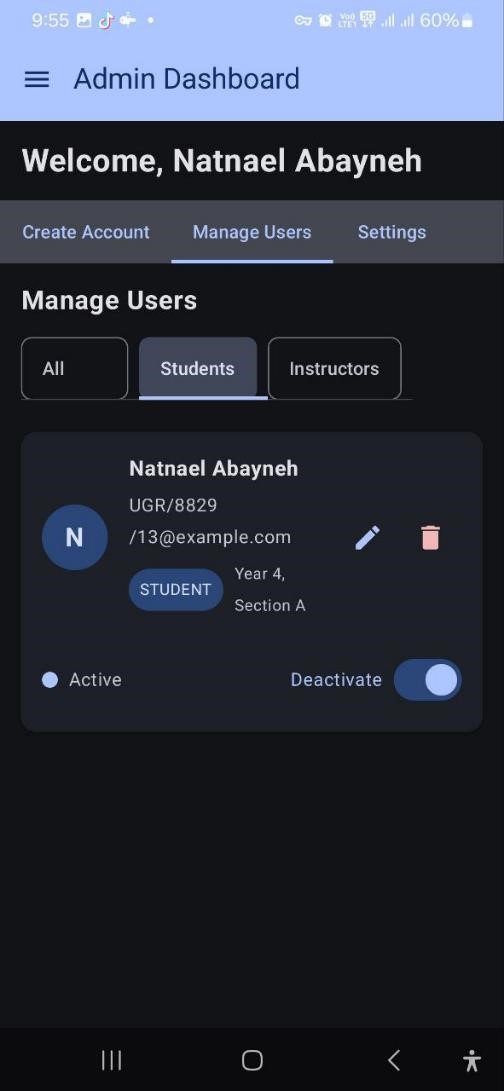
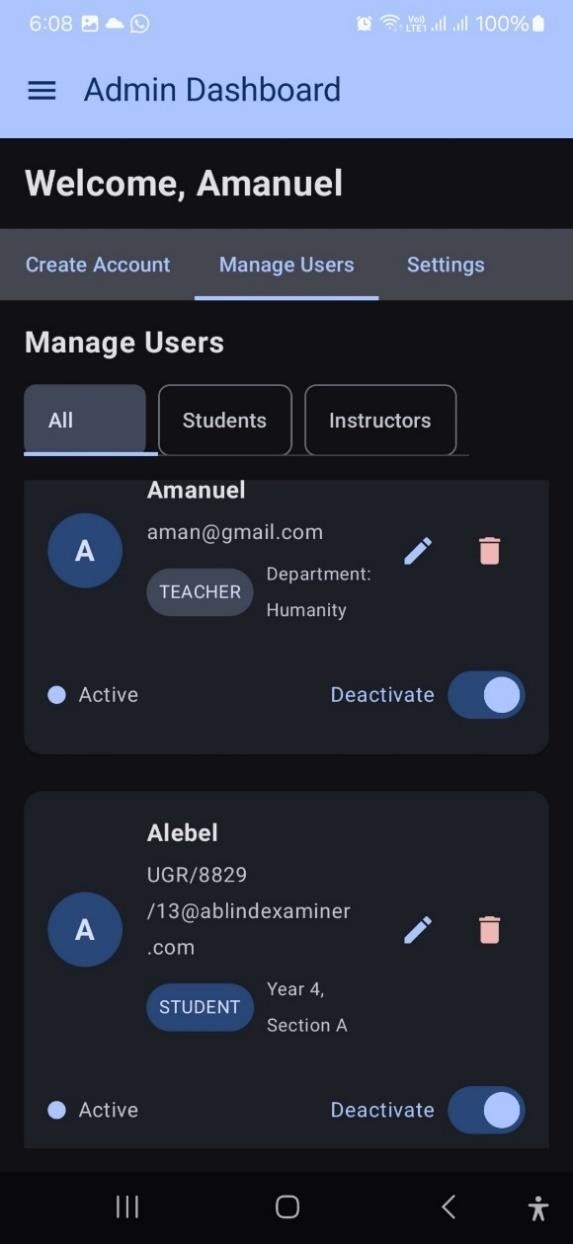


Figure 0.6 Admin Dashboard - Manage Figure 0.5Admin Dashboard - Manage

All Users View Users (Students Filtered View)

Figure 4.6 displays the Admin Dashboard's Manage Users section, showing a view of "All" users, including both teachers and students with their respective management options. Figure 4.5 illustrates the Manage Users section, specifically filtered to show "Students," focusing on the student account of the logged-in administrator, Natnael Abayneh.

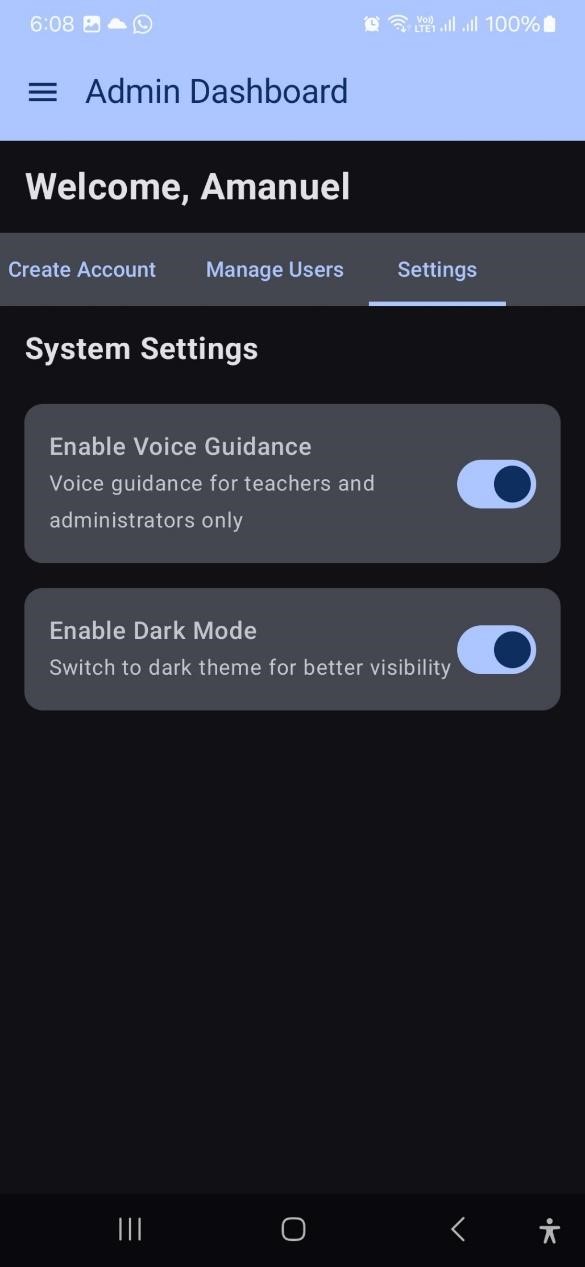
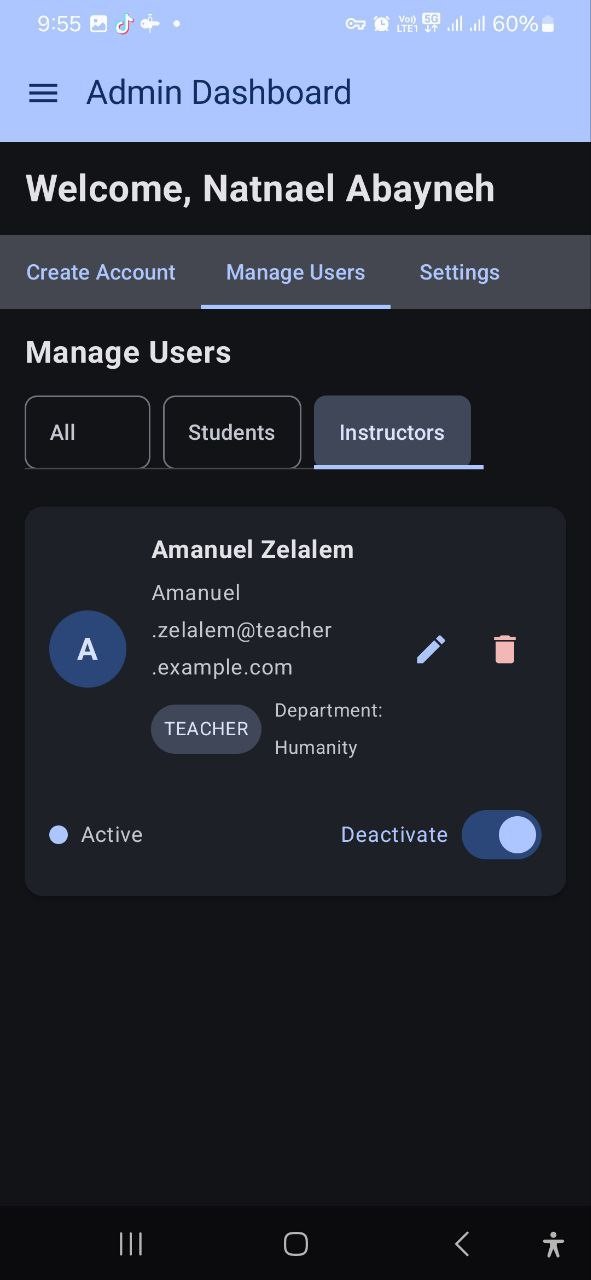


Figure 0.8 Admin Dashboard - Manage Figure 0.7: Admin Dashboard - System

Users (Instructors Filtered View) Settings

Figure 4.8 displays the Admin Dashboard's "Manage Users" section, filtered to show "Instructors," where a teacher's account is listed with various management options, In contrast, Figure 4.7 illustrates the "Settings" section of the Admin Dashboard, presenting system-wide options such as enabling voice guidance and dark mode.

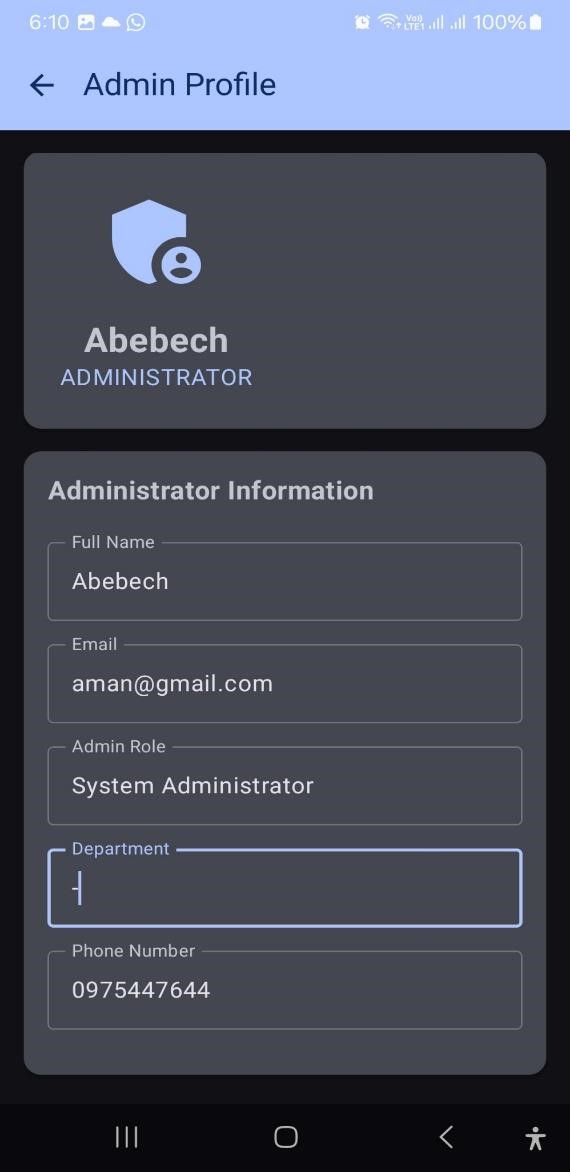
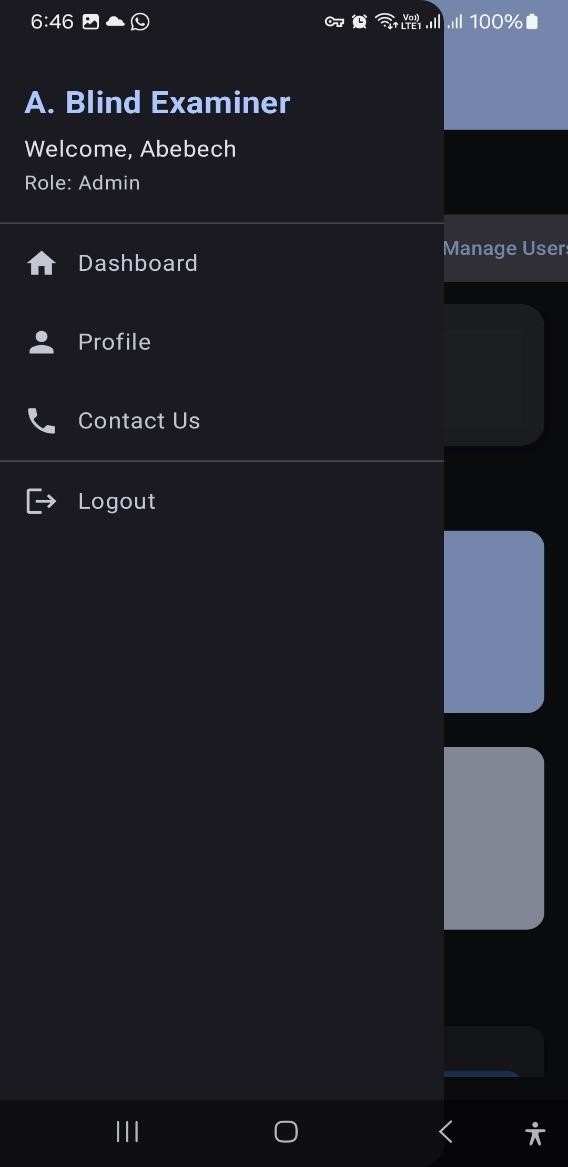


Figure 0.9 Admin Dashboard - Side Figure 0.10 Admin Dashboard -

Navigation Menu Administrator Profile Page

Figure 4.9 presents a side menu from the "Admin Dashboard" of the "Blind Examiner" application, welcoming an administrator and offering navigation options such as Dashboard, Profile, Contact Us, and Logout. Figure 4.10, conversely, displays the "Admin Profile" section, showing an administrator's details and providing editable fields for Administrator Information including Full Name, Email, Admin Role (System Administrator), Department, and Phone Number.

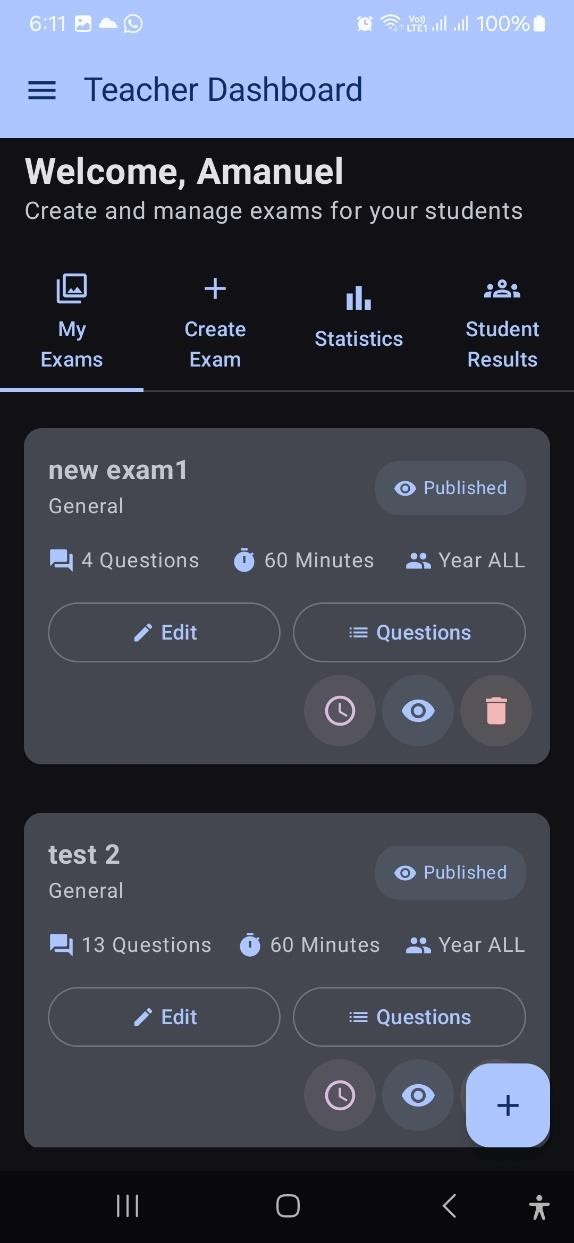
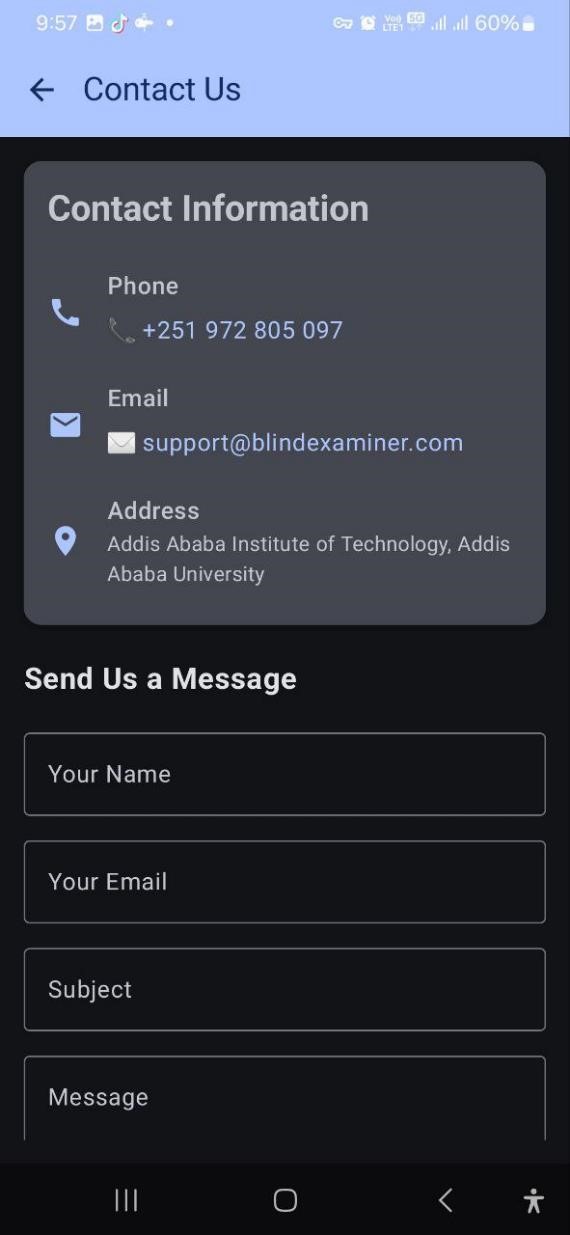


Figure 0.12 Contact Us Figure 0.11 Teacher Dashboard

Figure 4.11 shows the "Contact Us" page of the "Blind Examiner" application, detailing contact information like a phone number, email, and address, alongside a form for sending messages. Figure 4.12 displays the Teacher Dashboard, presenting tools for creating and managing exams, including a list of published exams with their respective details and action buttons.

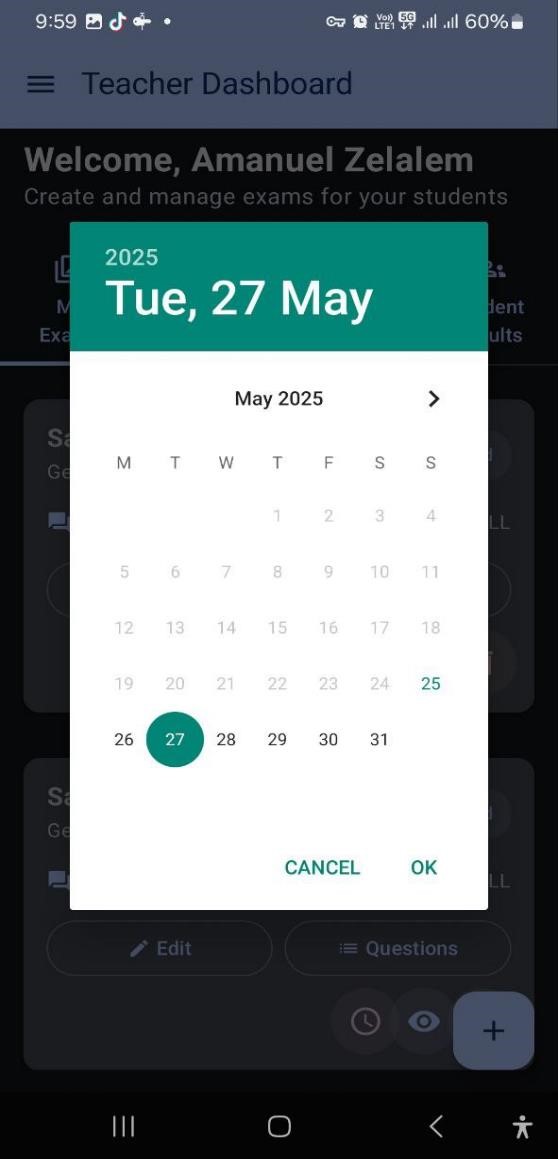
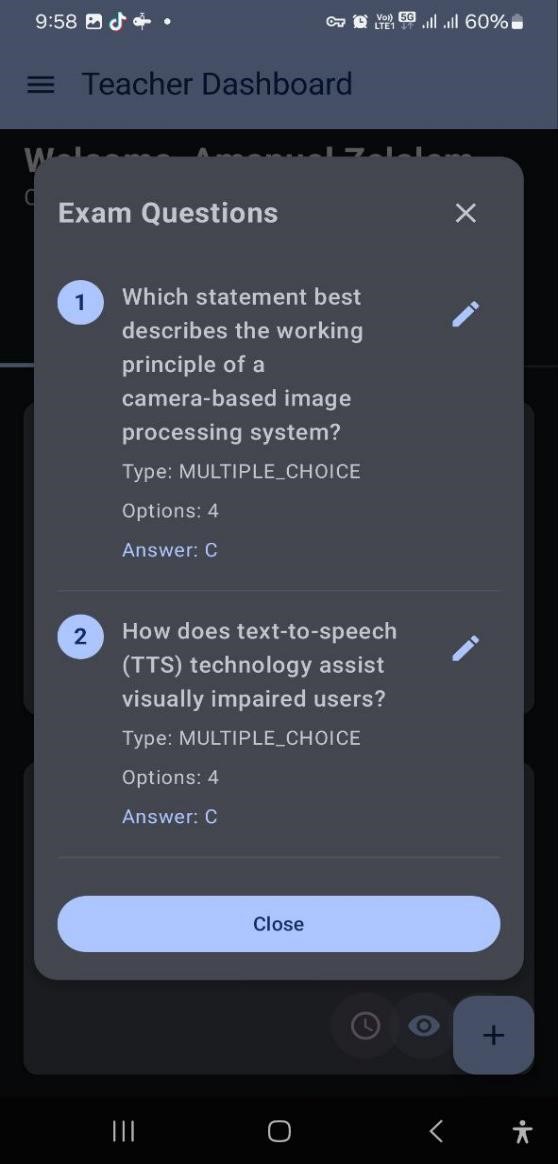


Figure 0.13 Exam Questions Figure 0.14 Date Selection Overlay /

Teacher Dashboard with Calendar

Figure 4.13 displays a pop-up titled "Exam Questions," showcasing a list of individual exam questions complete with their text, type, number of options, and answer, along with an edit option for each and a "Close" button. Figure 4.14 presents the "Teacher Dashboard" overlaid with a date selection calendar, highlighting "Tuesday, May 27, 2025," and offering "CANCEL" and "OK" options.

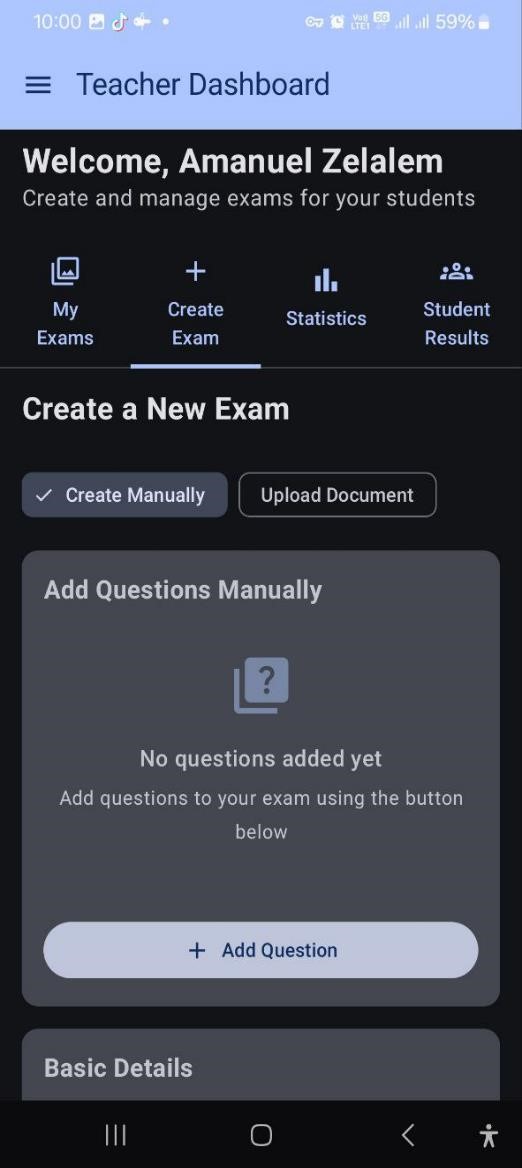
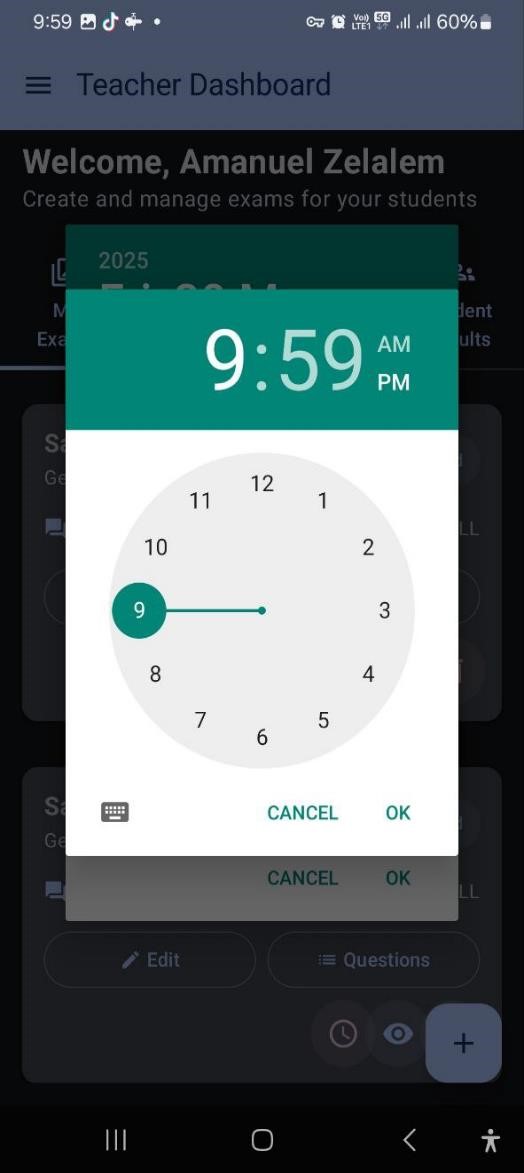


Figure 0.15 Time Picker Overlay / Figure 0.16 Create New Exam / Add

Teacher Dashboard with Time Questions Manually

Figure 4.15 shows the Teacher Dashboard overlaid with a time picker pop-up, presenting 9:59 on an analog clock face with AM/PM toggle and CANCEL and OK options. Figure 4.16, also from the Teacher Dashboard, displays the Create a New Exam section, offering options to Create Manually (with an Add Question button for manual input) or Upload Document.

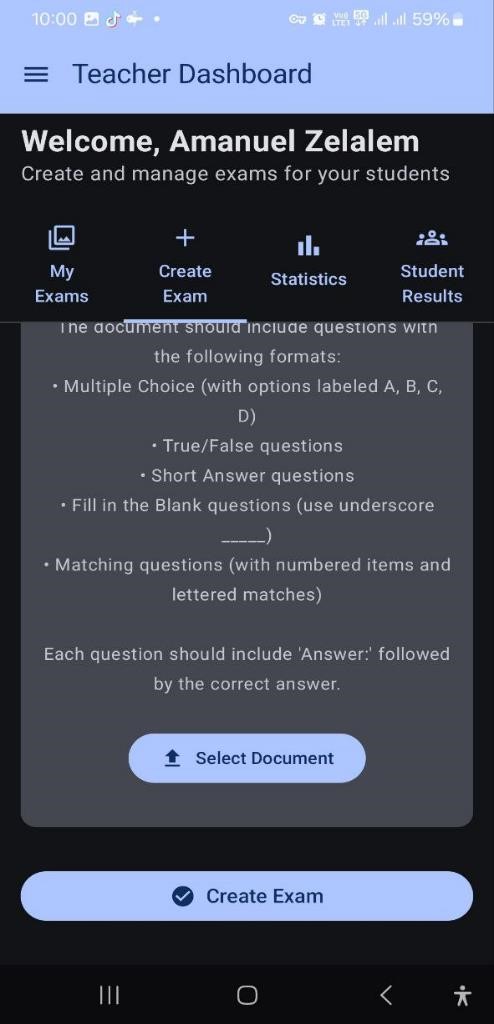
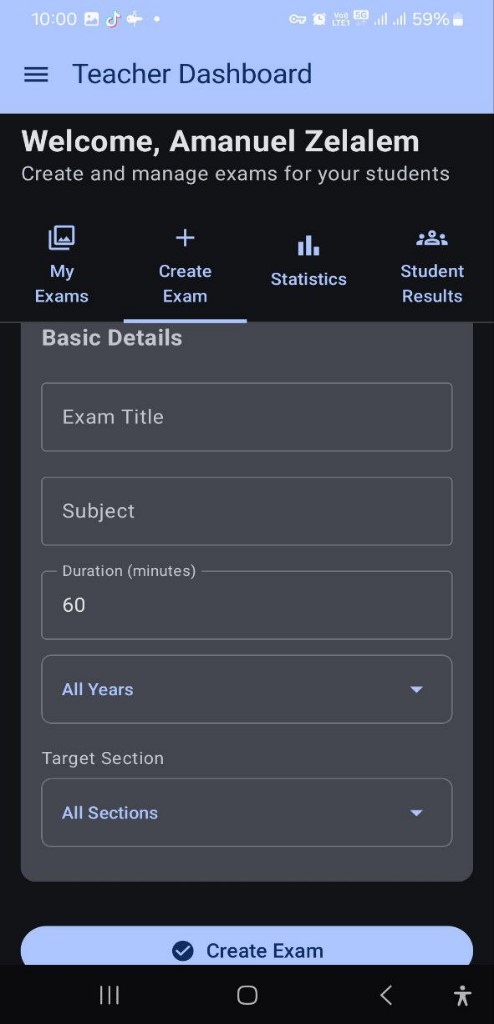


Figure 0.17 Exam Basic Details Figure 0.18 Exam Document Upload

Guidelines

Figure 4.17 displays the Teacher Dashboard's Basic Details section for exam creation, showing input fields for title, subject, duration, year, and section, concluding with a Create Exam button. Figure 4.18, also on the Teacher Dashboard, outlines document upload requirements for exam creation, specifying supported question formats and answer instructions, and includes the Select Document and Create Exam buttons.

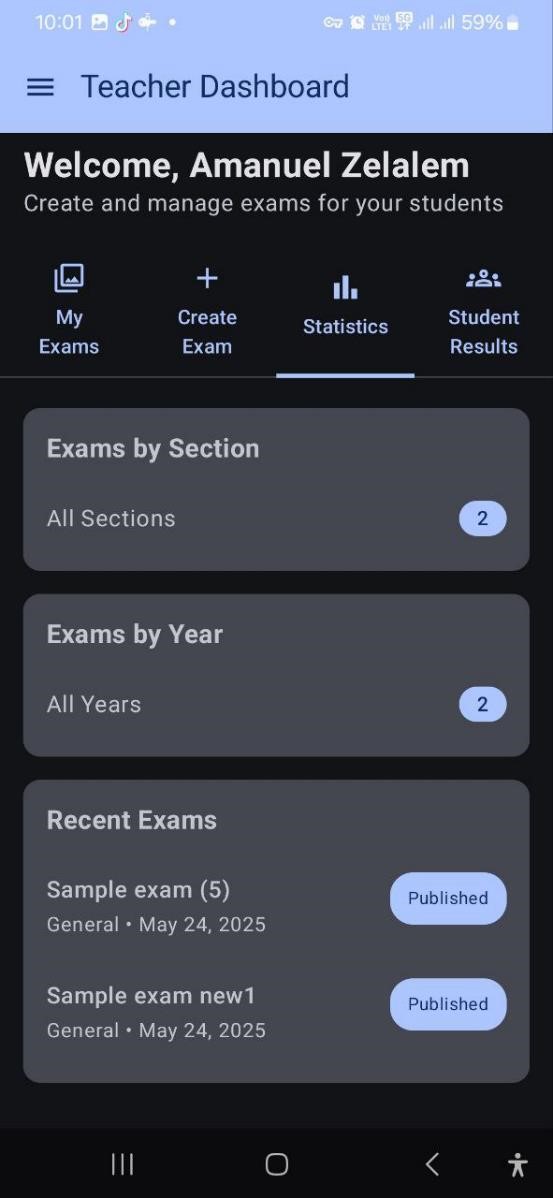
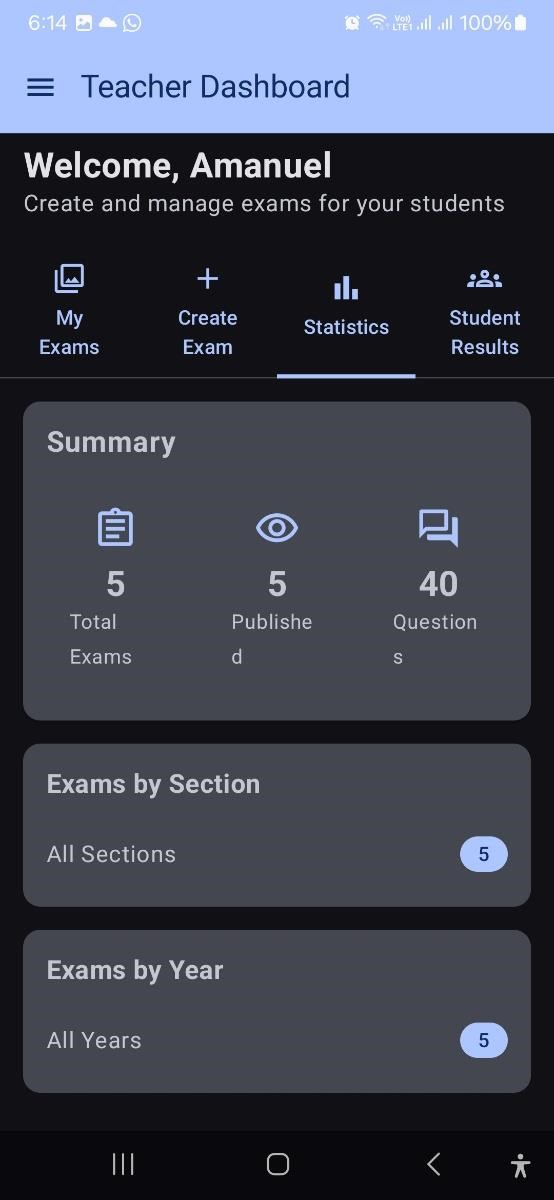


Figure 0.20 Teacher Dashboard Summary Figure 0.19 Teacher Dashboard

(Initial) Summary with Recent Exams

Figure 4.20, also from the Teacher Dashboard, presents a similar overview but with different counts for exams by section and by year, and additionally lists Recent Exams including their names, general category, date, and published status. Figure 4.19 displays the Teacher Dashboard's Summary section, showing totals for exams, published exams, and questions, along with categorized counts for exams by section and by year.

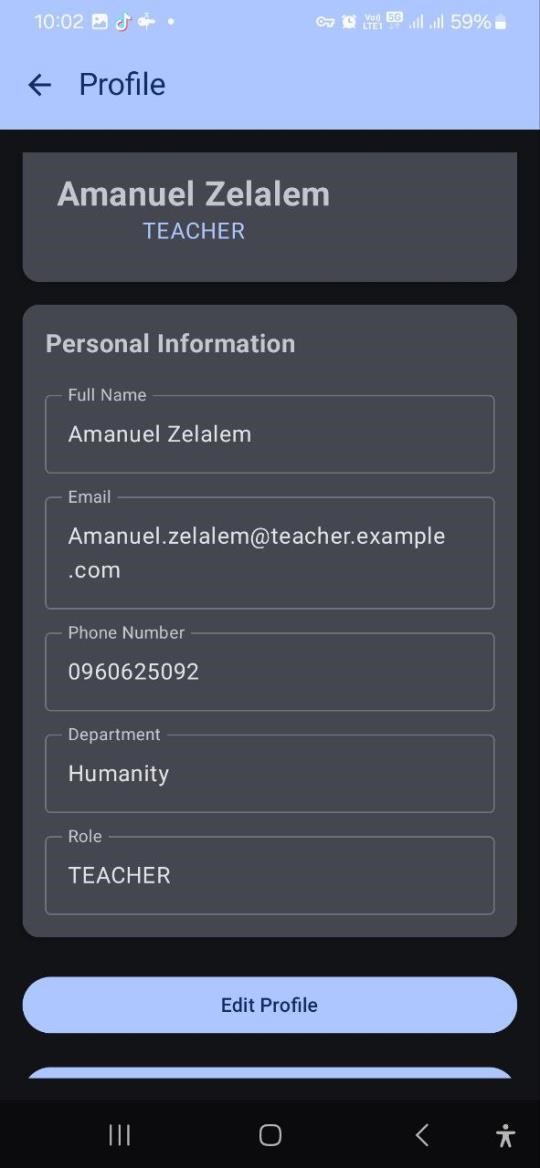
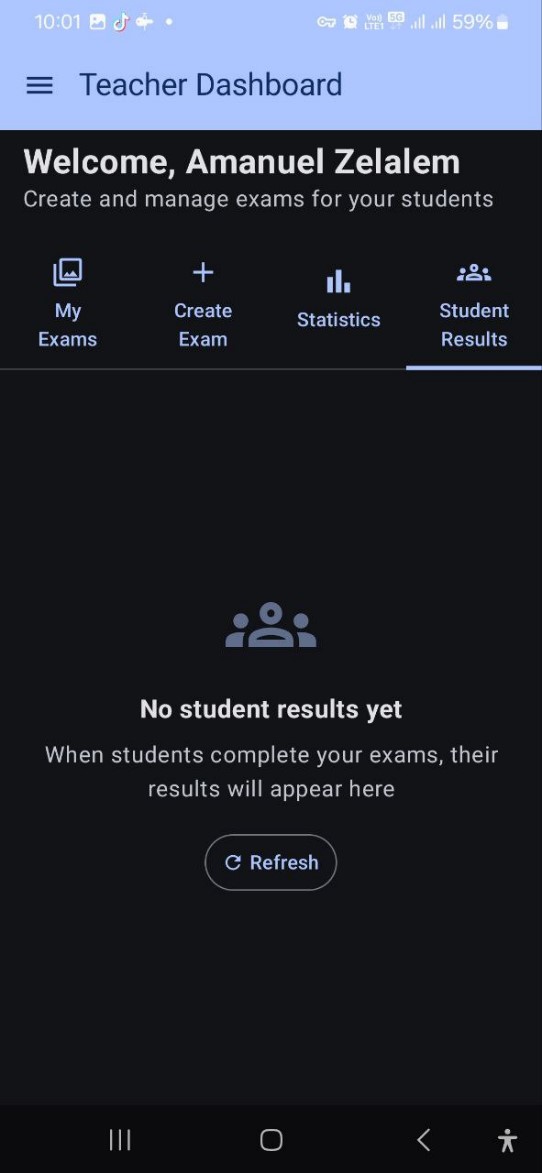


Figure 0.22 Student Results Section Figure 0.21 Teacher Profile

Figure 4.22, conversely, presents the Profile section, showing a teacher's personal information including full name, email, phone number, department, and role, with an Edit Profile button at the bottom. Figure 4.21 displays the Teacher Dashboard's Student Results section, indicating that no student results are currently available and inviting the user to refresh the page once students complete their exams.

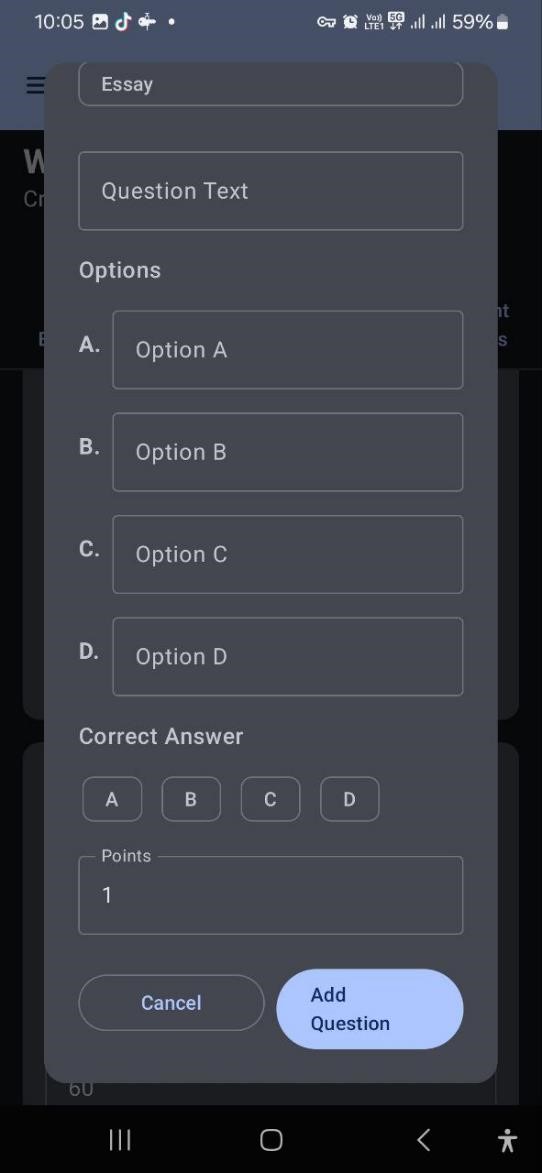
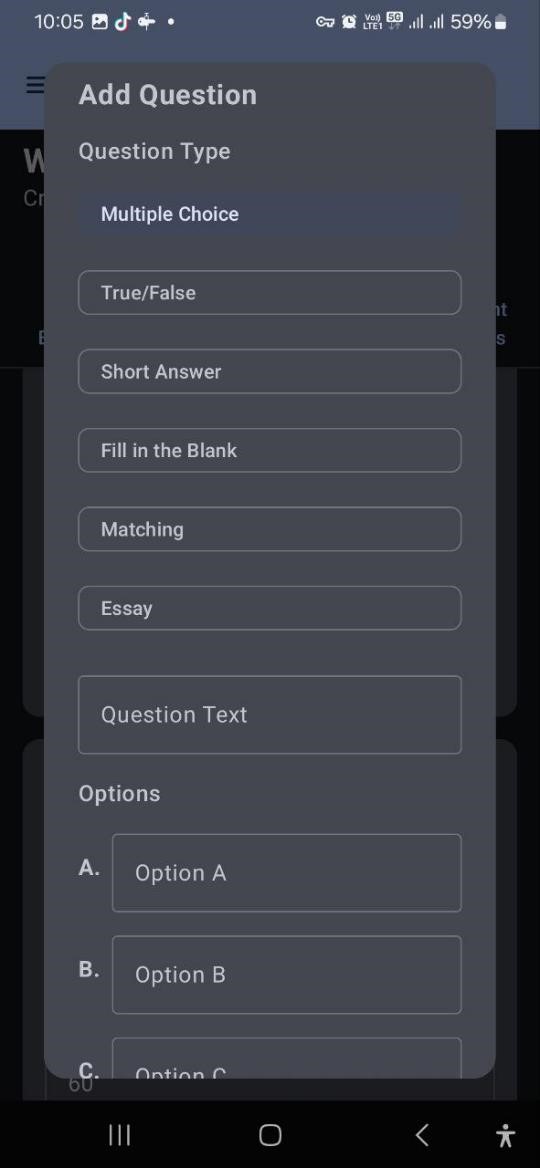


Figure 0.23 Add Question - Details Figure 0.24 Add Question - Type

Form Selection

Figure 4.24, a continuation or specific type view, shows fields for Question Text and Options A through D, includes a section for selecting the Correct Answer, an input for Points (set to 1), and concludes with Cancel and Add Question buttons. Figure 4.23 displays an Add Question interface, allowing the selection of a Question Type from options such as Multiple Choice, True/False, Short Answer, Fill in the Blank, Matching, and Essay, followed by input fields for the Question Text and various Options (A, B, C, D).

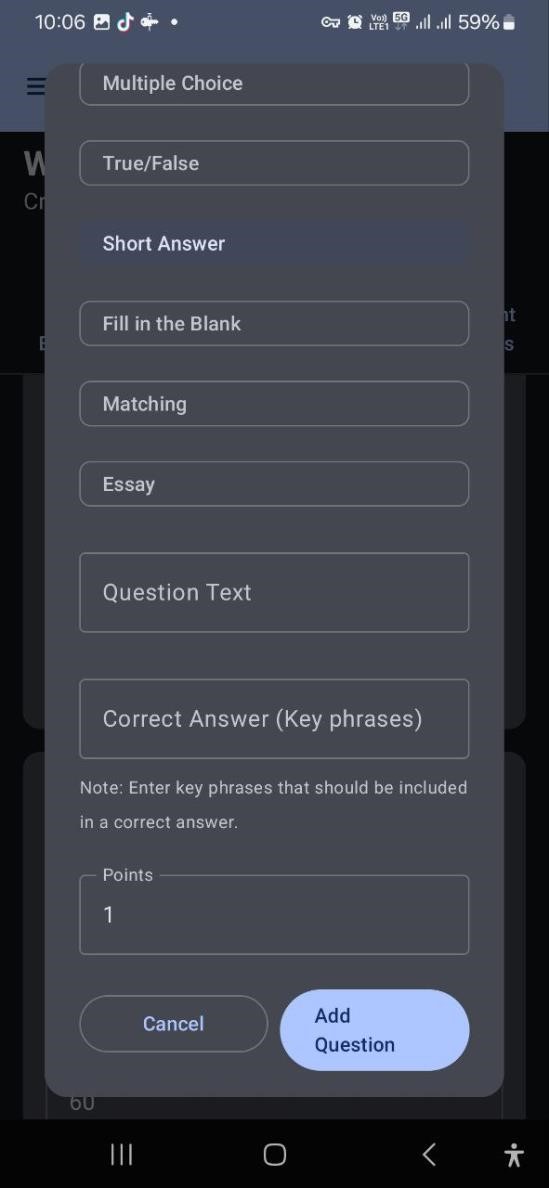
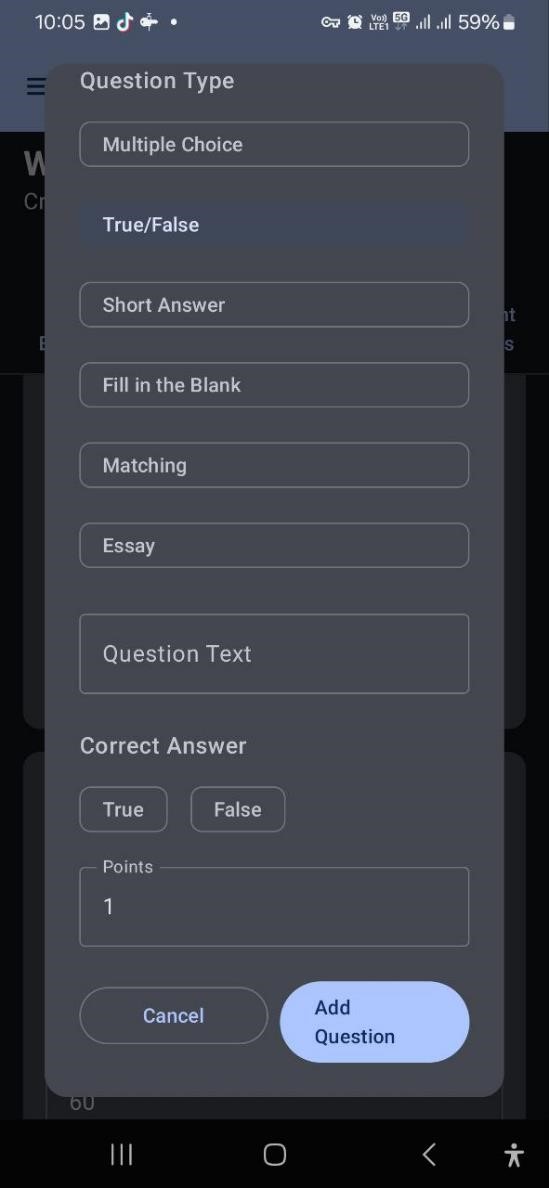


Figure 0.26 Add Question - True/False Figure 0.25 Add Question - Short

Type Answer Type

Figure 4.25 shows the Add Question interface configured for a True/False question, including fields for question text, true/false answer selection, points, and action buttons. Figure 4.26, also on the Add Question interface, similarly displays question type options and question text, but focuses on setting up a Short Answer question with a key phrases input for the correct answer, points, and action buttons.

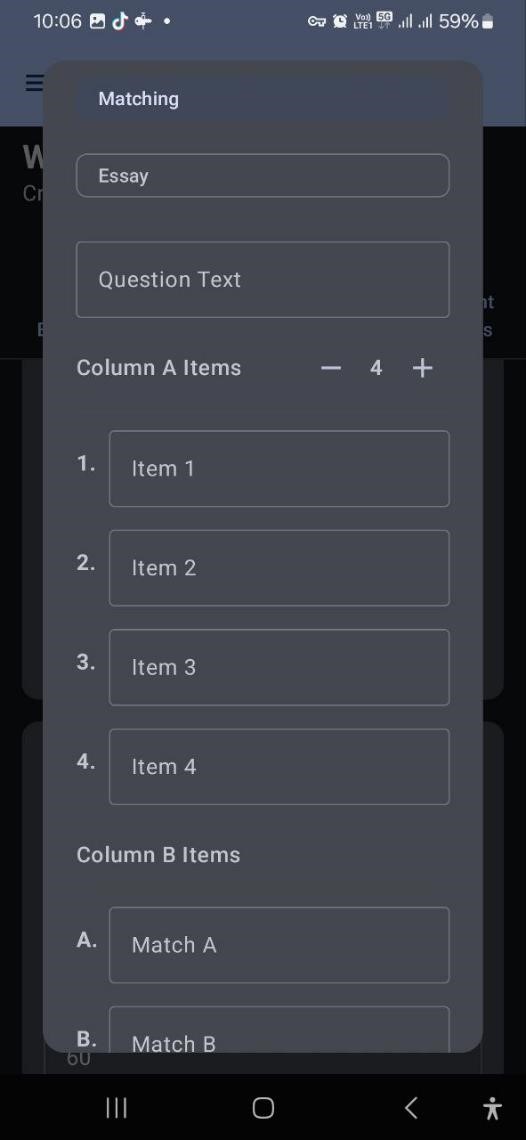
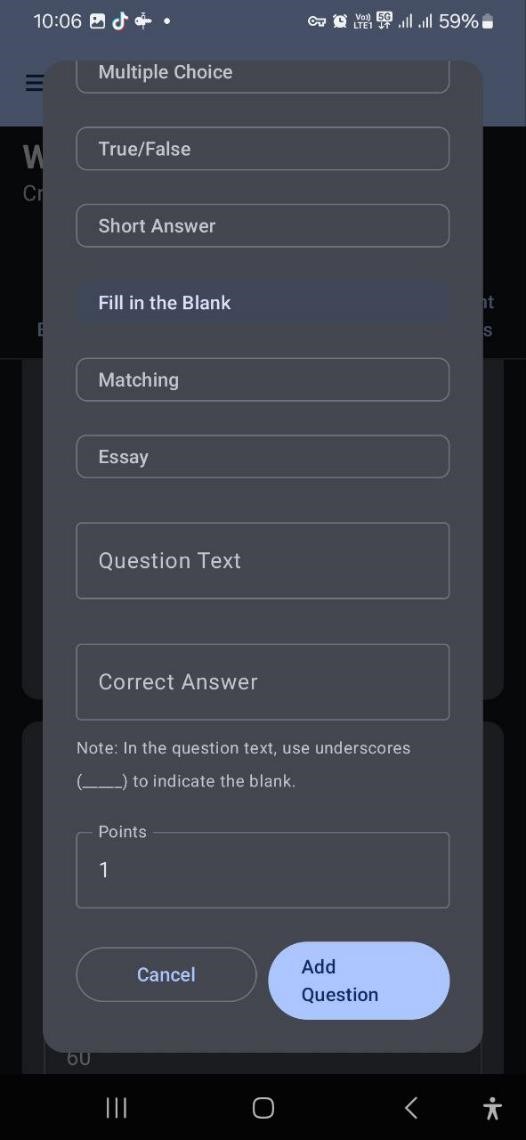


Figure 0.28 Add Question - Fill in Figure 0.27 Add Question - the Blank Type Matching Type

Figure 4.28 shows the Add Question interface configured for a Fill-in-the-blank question, including fields for question text, correct answer with blank indicators, points, and action buttons. Figure 4.27, also on the Add Question interface, displays the setup for a Matching question, featuring a question text field and sections for defining Column A and Column B items.

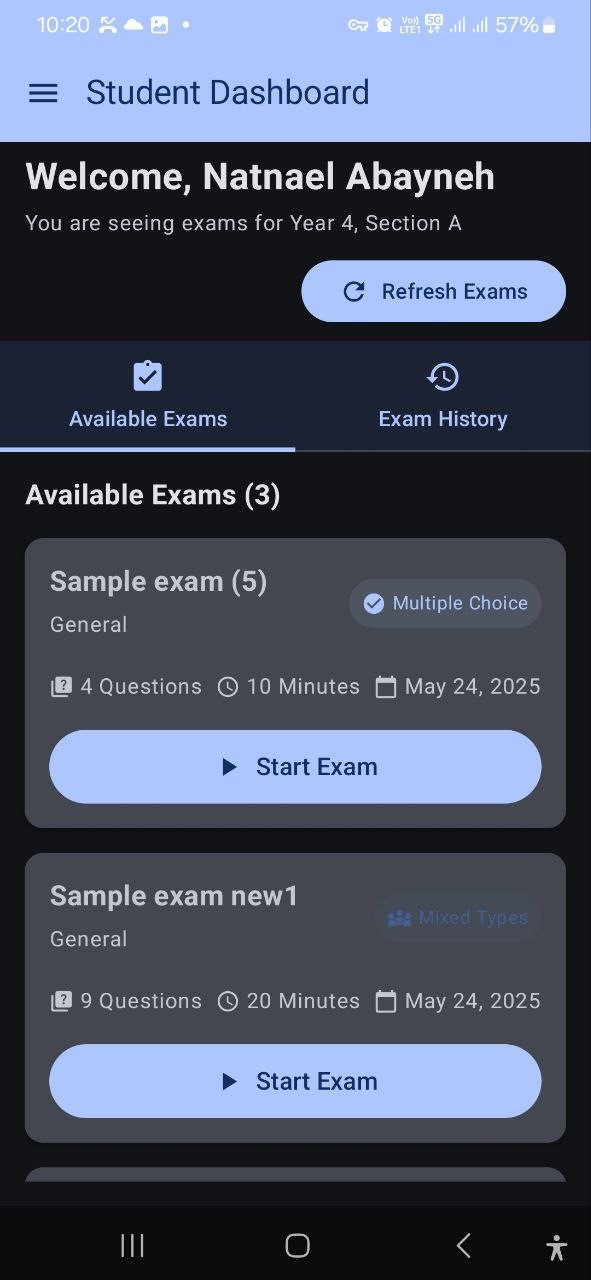
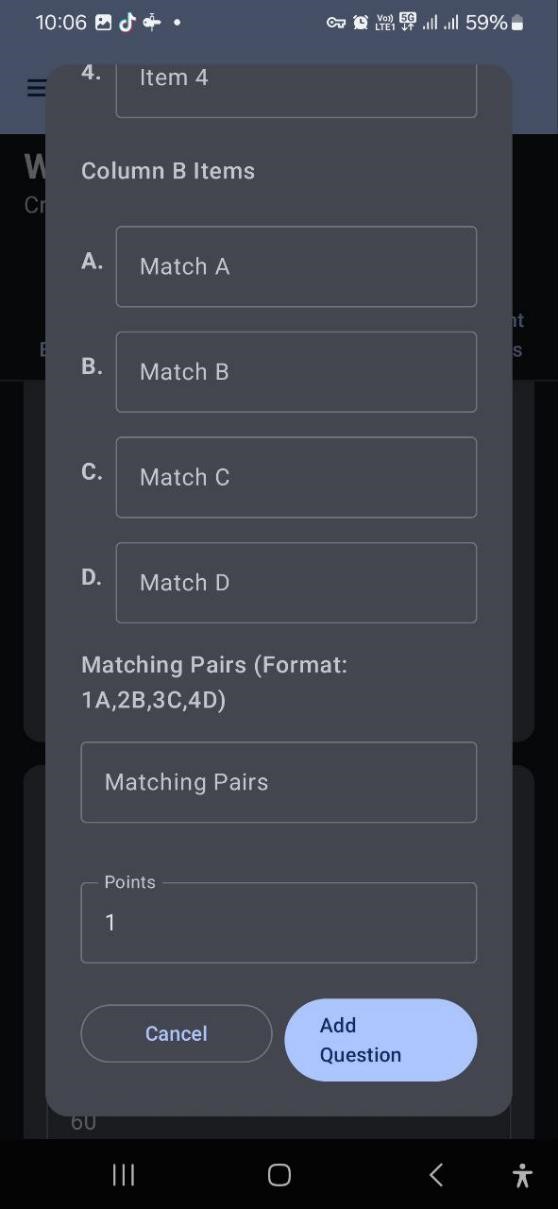


Figure 0.30 Add Question - Matching Figure 0.29 Student Dashboard -

Pairs Input Available Exams

Figure 4.29 completes the Add Question interface for a Matching question, showing Column B item definitions, a matching pairs input, points, and action buttons. Figure 4.30, the Student Dashboard, displays available exams for a student's year and section, listing exam details like question count, duration, and type, with a Start Exam button for each, and options to refresh or view exam history.

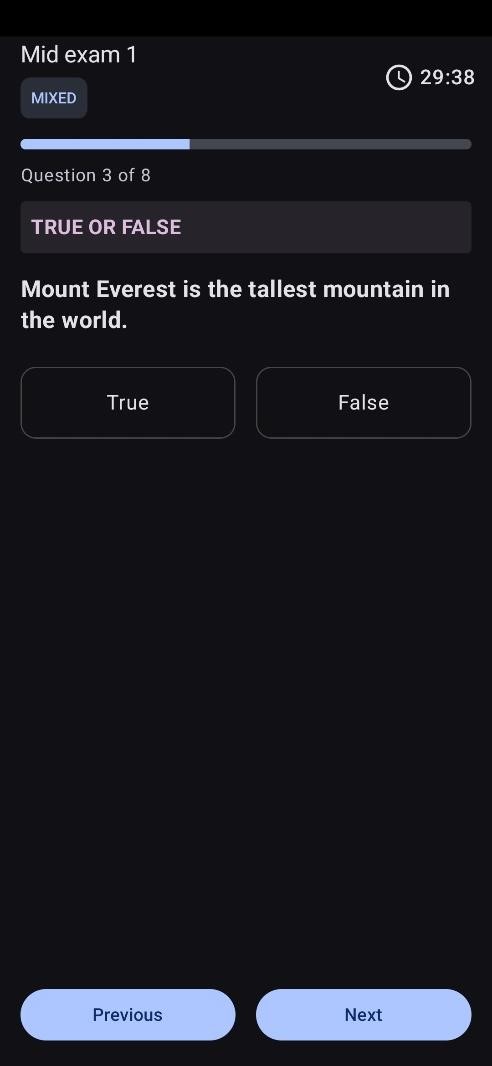
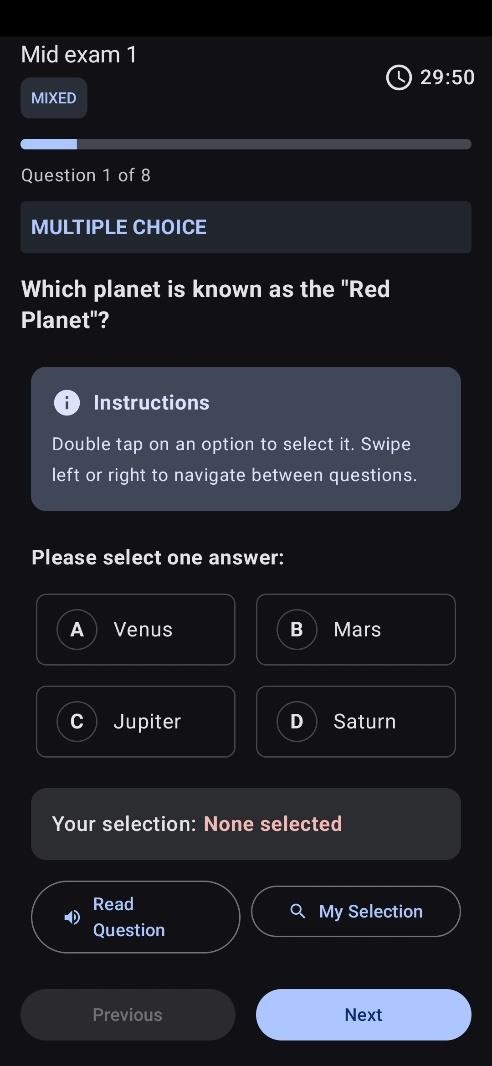


Figure 0.32 Multiple Question Figure 0.31 True False Question

Figure 4.32 shows Multiple-choice questions. In this exam screen, part blind can hear questions as well as options. He can navigate to each part using a swiping technique which is familiar to them. When the blind student Swipe to the left, he will go downward and vice versa. To select any button, he must double click anywhere on the part that the tts read it. In this way, he can take the exam. Figure 4.31 shows the same setup with True or False question variation.



Figure 0.34 Short Answer Figure 0.33 Brail Input

Figure 4.34 is a snapshot of a Short answer question page. In this part, the blind can do the same. Since it is a short answer question, he will use the Braile input button. When he touches the braille input button he will be directed to the Braille screen. Figure 4.33 This is a braille screen. On this screen, he can write like a normal braille writing that they write it using a slit. In this part Two machine learning models are integrated inside it. Braille character recognition model and Semantic short answer analysis. Both models are tenser flow machine learning models. For recognizing braille code and evaluating short-answer respectively

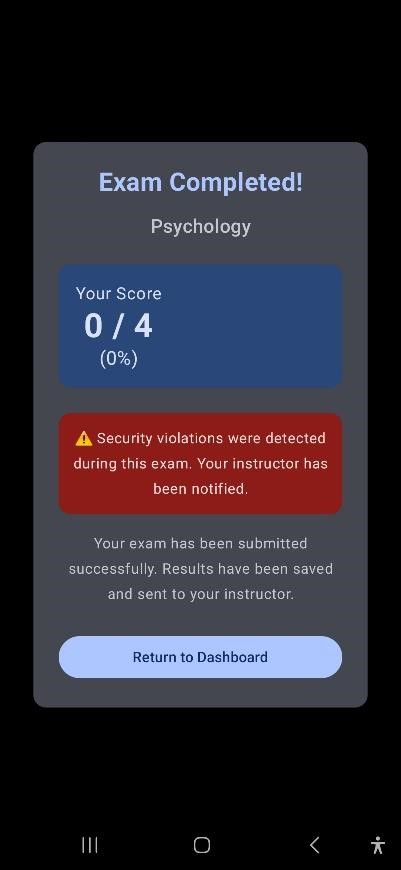
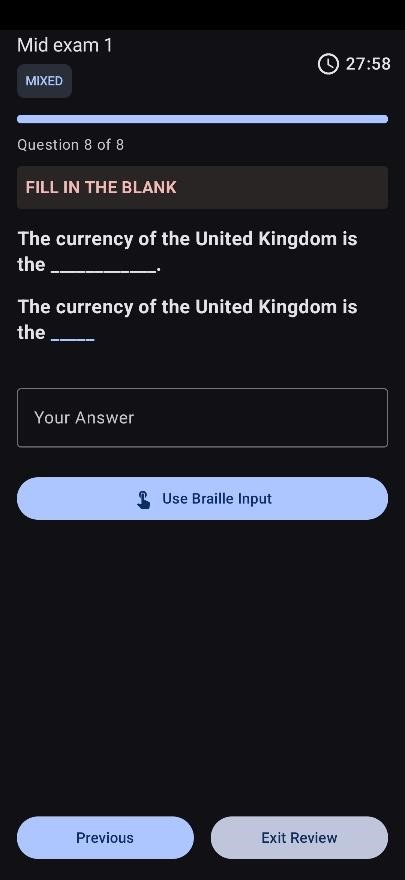


Figure 0.35 Fill in The Blank Figure 0.36 Result Display

Figure 4.35 shows a fill-in-the-blank question from a Mid exam 1 with a timer set at 27:58. The question, “The currency of the United Kingdom is the”, requires the user to input their answer. There's an option to Use Braille Input and navigation buttons for Previous and Exit Review. Figure 4.36 displays an "Exam Completed!" screen for a Psychology exam. The user's score is 0/4 (0%). A prominent red alert states, "Security violations were detected during this exam. Your instructor has been notified." Below this, it confirms, "Your exam has been submitted successfully. Results have been saved and sent to your instructor." There's a button to "Return to Dashboard."

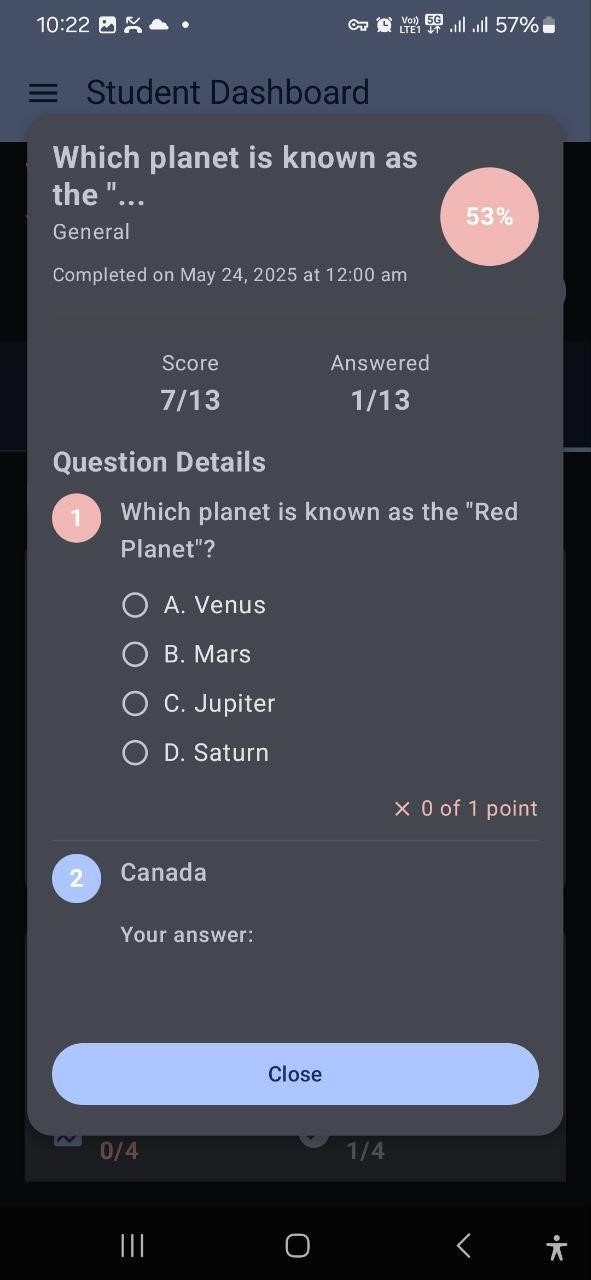
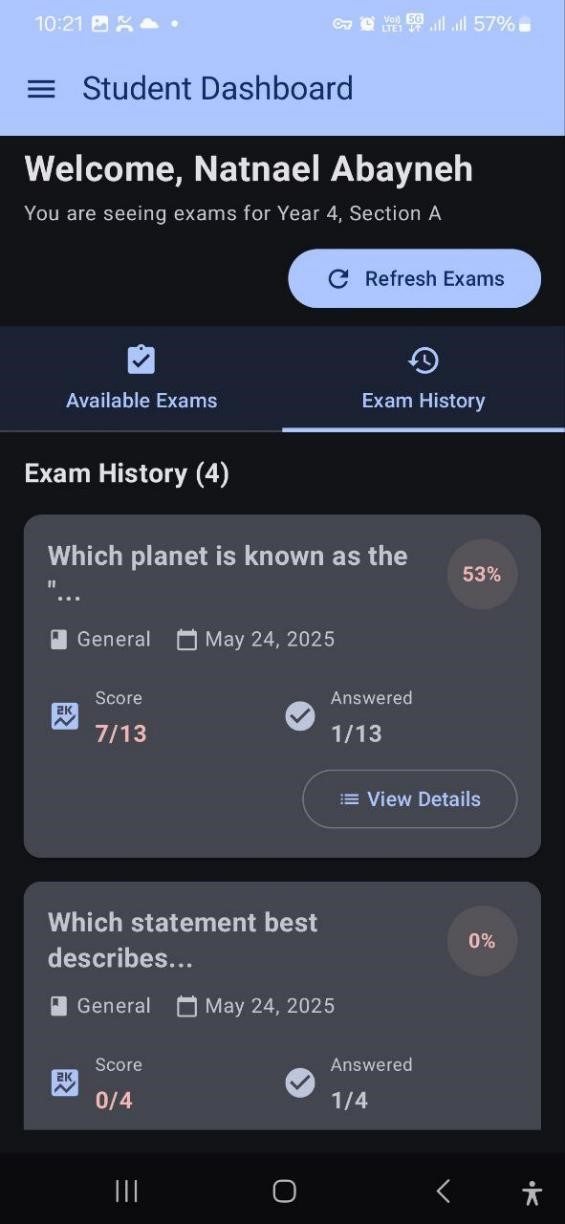


Figure 0.37 Student Dashboard - Exam Figure 0.38 Student Dashboard -

History Exam Details

Figure 4.37 displays the Student Dashboard's Exam History, summarizing past exams with their scores and progress. Figure 4.38 presents a detailed view of a completed exam from the Student Dashboard, showing the overall score and answered count, followed by individual question details with respective answers and options.

Generally, all of the student dashboard uses a swiping navigation system to go from one tab to another.

## 4.1. Machine learning evaluation

### 4.1.1. Evaluation of Braille Character Recognizer

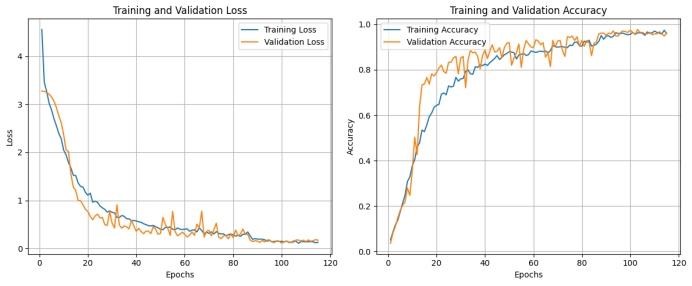


Figure 0.39 Training and Validation Performance of Braille Character Recognizer

The graphs show that as the model learns, its errors (loss) are steadily going down for both the data it's seen and new data, which is great. At the same time, its ability to correctly identify Braille characters (accuracy) is consistently going up and staying high, meaning it's learning well and can apply that knowledge to new Braille inputs.

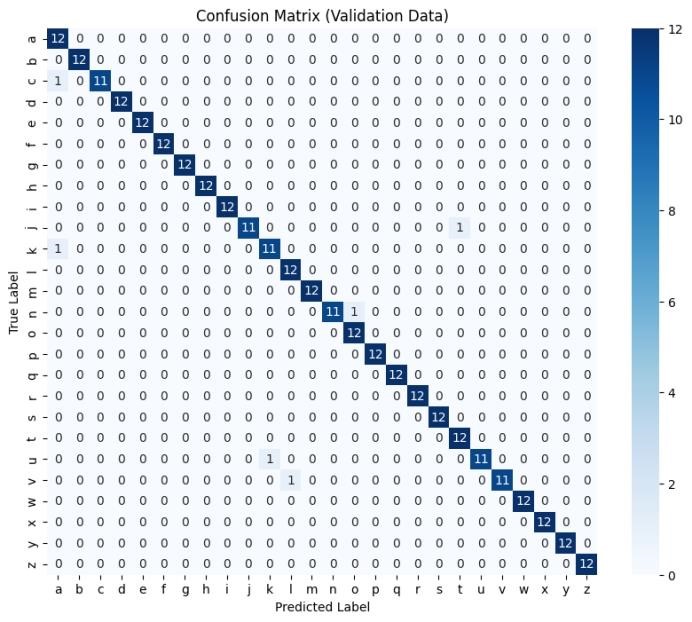


Figure 0.40 Confusion Matrix of Braille Character Recognizer on Validation Data

The image displays a confusion matrix generated from the validation data of a Braille Character Recognizer. This matrix visually represents the model's performance in classifying each Braille character (a-z). The diagonal elements, which are predominantly dark blue and show values of '12' or '11', indicate the number of instances where the model correctly predicted the character. For example, 'a' was correctly classified 12 times, 'b' 12 times, and so on. The very low number of off-diagonal entries, specifically '1' for 'j' misclassified as 'k' and 'u' misclassified as 't', suggests that the model exhibits excellent accuracy with minimal misclassifications across the different Braille characters, performing very well on the validation dataset.

**Validation Performance:**

**Validation Accuracy:** 96.15%

#### Validation Loss: 0.1433

Evaluated using Val generator on the validation dataset (10/10 steps, 23ms/step).

##### 4.1.1.1. Classification Report

The model was tested on 26 Braille characters (letters a to z), with 12 samples per class, totaling 312 validation samples.



Figure 0.41 Classification Report Heatmap (Character-Level Metrics)

This heatmap provides a detailed classification report for each Braille character (a-z), displaying their respective Precision, Recall, and F1-Score. The color intensity indicates the metric's value, with darker blues representing higher scores (closer to 1.0) and lighter yellows indicating lower scores (closer to 0.86). Overall, the heatmap is dominated by dark blue, signifying excellent performance across most characters and metrics, with many values at or near 1.0. A few lighter spots, such as for 'a' in Precision (0.86) and 'c', 'j', 'n', 'u', and 'v' in Recall (0.92), indicate slight variations in performance, but generally, the model demonstrates high accuracy and balanced performance for the Braille character recognition task.

Overall Metrics:

* Accuracy: 98%
* Macro Average: Precision: 0.98 | Recall: 0.98 | F1-Score: 0.98
* Weighted Average**:** Precision: 0.98 | Recall: 0.98 | F1-Score: 0.98

### 4.1.2. Evaluation Summary: Automatic Grading

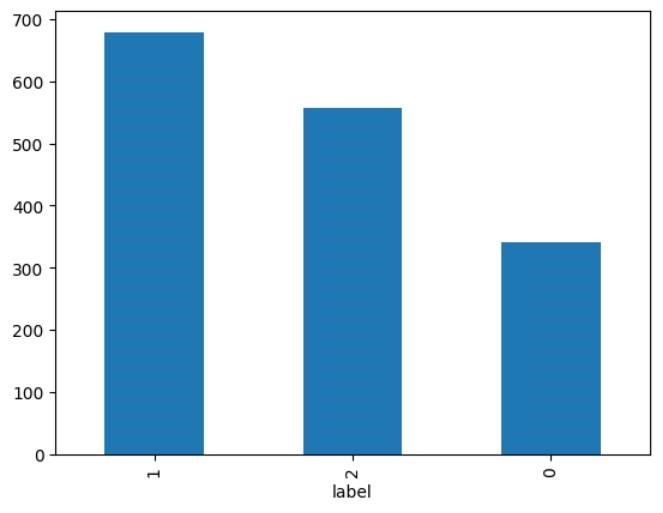


Figure 0.42 Distribution of Labels in Automatic Grading Dataset

This bar chart displays the distribution of labels within a dataset, likely representing grades or categories in an automatic grading system. The 'label' axis shows three distinct categories: '1', '2', and '0'. Label '1' is the most frequent, with a count close to 700. Label '2' follows, with a frequency slightly above 550. Label '0' is the least frequent, with a count of just under 350. This indicates an imbalanced dataset where category '1' is significantly more represented than '0', which could have implications for model training and evaluation, particularly for an automatic grading system where the distribution of grades might reflect the difficulty or commonality of certain score ranges.

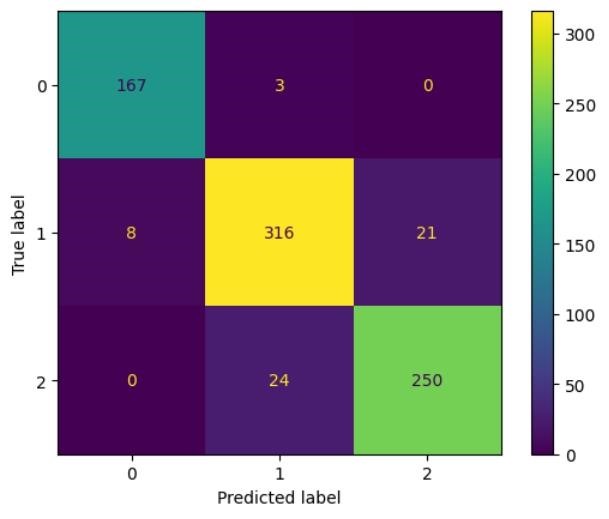


Figure 0.43 Confusion Matrix for Automatic Grading System

This image displays a confusion matrix, for an automatic grading system or a multiclass classification task with three categories (0, 1, and 2). The diagonal elements show the number of correct predictions for each class: 167 instances of true label 0 were correctly predicted as 0, 316 instances of true label 1 were correctly predicted as 1, and 250 instances of true label 2 were correctly predicted as 2. The off-diagonal values represent misclassifications. For example, 3 instances of true label 0 were mistakenly predicted as 1, 8 instances of true label 1 were predicted as 0, 21 instances of true label 1 were predicted as 2, and 24 instances of true label 2 were predicted as 1. Overall, the high values on the diagonal indicate generally good performance, but there are some noticeable confusions, particularly between classes 1 and 2, and to a lesser extent between 0 and 1.

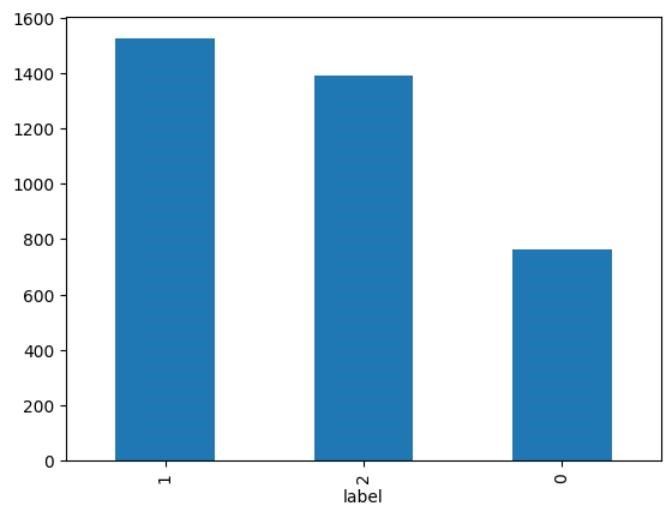


Figure 0.44 Distribution of Labels in Dataset

This bar chart shows the distribution of three different labels (0, 1, and 2) within a dataset. Label '1' is the most frequent category, with its count being just shy of 1550. Label '2' is the second most frequent, with a count of approximately 1400. Finally, label '0' is the least frequent, with its count being around 750. This visualization indicates an imbalance in the dataset, where labels '1' and '2' are significantly more represented than label '0'.

Model Type: TFBert For Sequence Classification

Framework: TensorFlow (initialized from PyTorch weights)

Export Path: ./asag\_pytorch\_export

Training Epochs: 8

The PyTorch weights have been successfully transferred and initialized in TensorFlow, making the model ready for inference without requiring additional training.

###### 4.1.2.1. Evaluation of Training Set

Table 2 Evaluation of Training Set

|  |  |
| --- | --- |
| Metric | Value |
| Loss | 0.0183 |
| Accuracy | 0.9940 |
| F1 Score (Weighted) | 0.9940 |
| Cohen's Kappa | 0.9907 |
| Runtime | 26.72 seconds |
| Samples/sec | 137.75 |
| Steps/sec | 8.64 |

**Interpretation**: The training metrics are nearly perfect, showing that the model has learned the training data extremely well.

###### 4.1.2.2. Evaluation of Validation Set

Table 3 Evaluation of Validation Set

|  |  |
| --- | --- |
| Metric | Value |
| Loss | 0.4867 |
| Accuracy | 0.9290 |
| F1 Score (Weighted) | 0.9289 |
| Cohen's Kappa | 0.8896 |
| Precision (Weighted) | 0.9288 |
| Recall (Weighted) | 0.9290 |
| Runtime | 5.82 seconds |
| Samples/sec | 135.49 |
| Steps/sec | 8.59 |

**Interpretation**: These values confirm strong generalization. A slight performance drop from training is expected and within a healthy range.

###### 4.1.2.3. Per-Class Classification Report (Validation Set) Table 4 Per-Class Classification Report (Validation Set)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1 Score | Support |
| 0 | 0.95 | 0.98 | 0.97 | 170 |
| 1 | 0.92 | 0.92 | 0.92 | 345 |
| 2 | 0.92 | 0.91 | 0.92 | 274 |
| Overall  Accuracy | — | — | 0.93 | 789 |
| Macro Avg | 0.93 | 0.94 | 0.93 | 789 |
| Weighted Avg | 0.93 | 0.93 | 0.93 | 789 |

###### 4.1.2.4. Observations

Class 0 demonstrates the highest individual performance, with overall performance remaining consistent across all three classes, ensuring no single class dominates the distribution and thereby helping to avoid bias.

###### 4.1.2.5. Comparison Summary: Training vs Validation

Table 5 Comparison Summary: Training vs Validation

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Training Set | Validation Set | Gap |
| Accuracy | 0.9940 | 0.9290 | -0.0650 |
| F1 Score  (Weighted) | 0.9940 | 0.9289 | -0.0651 |
| Cohen's Kappa | 0.9907 | 0.8896 | -0.1011 |

###### 4.1.2.6. Analysis

A slight gap suggests some over-fitting, but it is not excessive, as validation scores remain strong, indicating the model generalizes well.

Based on the current metrics, the model achieves very high training and validation performance, with a Cohen’s Kappa Score of 0.8896 on validation demonstrating excellent agreement beyond chance; minor overfitting is acceptable and expected in deep learning, though regularization or dropout could be considered for future tuning.

# CHAPTER FIVE

# CONCLUSIONS

This thesis successfully develops an Android-based mobile application to enhance exam accessibility for visually impaired students, addressing critical gaps in existing assistive technologies. The application features audio question delivery, a position-free Braille input method, and an intelligent automated grading system using semantic similarity (Sentence-BERT) for short answers, while also incorporating anti-cheating mechanisms by locking navigation buttons during exams. The evaluation shows high accuracy in Braille character recognition and robust performance in automatic grading, demonstrating the app's effectiveness in providing an independent, fair, and secure testing environment for blind students in Ethiopia and potentially beyond.

## 5.1. Challenges and Limitations

Despite its innovative features, the development of this Android-based autonomous blind examiner mobile application encountered specific challenges and limitations, particularly concerning machine learning integration and project timeline constraints. While the project aimed to integrate Braille-based input with intelligent answer evaluation, a significant hurdle was the lack of a pre-existing unified mobile application that combined these specific functionalities, necessitating novel development. Furthermore, existing voice-based exam systems, though offering audio question delivery and speech recognition, often fall short in addressing privacy, noise interference, and academic integrity, issues that the current project sought to overcome by implementing secure, individualized exam sessions and anti-cheating mechanisms. Beyond these technical integration complexities, the project also faced time constraints, particularly due to overlaps with final and exit examinations, which inherently limited the available development and testing period. This time pressure could potentially affect the depth of feature implementation or the breadth of testing scenarios, impacting the overall refinement of the application.

#### 5.2. Future Works and Recommendations

The future development of this Android-based autonomous blind examiner mobile application should focus on addressing existing limitations and expanding its capabilities to further enhance accessibility and educational equity for visually impaired students.

**Key Recommendations**:

* **Expand Language Support:** Currently, the application supports only Englishlanguage text-based questions. Future work should involve integrating support for additional languages, especially local languages in regions like Ethiopia, to broaden its applicability and reach a wider student population.
* **Offline Capability**: Investigate the feasibility of incorporating limited offline functionality for exam taking, particularly for areas with unreliable internet access, with subsequent synchronization when a connection is available.
* **Wider Deployment**: Following successful pilot deployments, strategize for wider deployment to other educational institutions and potentially across different regions, adapting the application to various local contexts and accessibility requirements.

# REFERENCES

1. W. H. Organization, "World report on vision," in *World report on vision*, 2019.
2. A. Ababa, "Ministry of Education," *Education Management Information Systems,* 2017.
3. N. Zerihun and D. Mabey, "Blindness and low vision in Jimma Zone, Ethopia:

results of a population-based survey," *Ophthalmic epidemiology,* vol. 4, no. 1, pp. 19-26, 1997.

1. W. Al-Eidarous, A. Alsiyami, M. Aljabri, S. Alqethami, and B. Almutanni, "ExamVoice: Innovative Solutions for Improving Exam Accessibility for Blind and Visually Impaired Students in Saudi Arabia," *Applied Sciences,* vol. 14, no. 19, 2024, doi: 10.3390/app14198813.
2. N. Thowfeek, "fully voice-controlled online exam system for visually impaired students," 2021.
3. S. Shokat, R. Riaz, S. S. Rizvi, A. M. Abbasi, A. A. Abbasi, and S. J. Kwon, "Deep learning scheme for character prediction with position-free touch screenbased Braille input method," *Human-centric Computing and Information Sciences,* vol. 10, no. 1, 2020, doi: 10.1186/s13673-020-00246-6.
4. B. Niazi, S. Khusro, A. Khan, and I. Alam, "A Touch Sensitive Keypad Layout for Improved Usability of Smartphones for the Blind and Visually Impaired Persons," in *Artificial Intelligence Perspectives in Intelligent Systems*,

(Advances in Intelligent Systems and Computing, 2016, ch. Chapter 38, pp.

427-436.

1. J. Oliveira, T. Guerreiro, H. Nicolau, J. Jorge, and D. Gonçalves, "BrailleType: unleashing braille over touch screen mobile phones," in *Human-Computer Interaction–INTERACT 2011: 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part I 13*, 2011: Springer, pp. 100-107.
2. S. a. Govindarajan, "a layout-free Braille input method using swipe and tap gestures on Android devices " 2018.
3. J. Siqueira *et al.*, "BrailleÉcran: A Braille Approach to Text Entry on Smartphones," presented at the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), 2016.
4. N. Paisios, A. Rubinsteyn, and L. Subramanian, "Mobile brailler: Making touch-screen typing accessible to visually impaired users," in *Workshop on Frontiers in Accessibility for Pervasive Computing. ACM*, 2012: Citeseer.
5. S. Mascetti, C. Bernareggi, and M. Belotti, "TypeInBraille: quick eyes-free typing on smartphones," in *Computers Helping People with Special Needs: 13th International Conference, ICCHP 2012, Linz, Austria, July 11-13, 2012, Proceedings, Part II 13*, 2012: Springer, pp. 615-622.
6. C. Jayant, C. Acuario, W. Johnson, J. Hollier, and R. Ladner, "V-braille," presented at the Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility, 2010.
7. A. W. Qurashi, V. Holmes, and A. P. Johnson, "Document Processing: Methods for Semantic Text Similarity Analysis," presented at the 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), 2020.
8. M. a. Mihalcea, "an unsupervised text similarity approach for automatically grading " 2009.
9. L. B. Galhardi and J. D. Brancher, "Machine Learning Approach for Automatic Short Answer Grading: A Systematic Review," in *Advances in Artificial Intelligence – IBERAMIA 2018*, (Lecture Notes in Computer Science, 2018, ch. Chapter 31, pp. 380-391.
10. M. Mansoor, Z. u. Rehman, M. Shaheen, M. A. Khan, and M. Habib, "Deep

Learning based Semantic Similarity Detection using Text Data," *Information*

*Technology And Control,* vol. 49, no. 4, pp. 495-510, 2020, doi:

10.5755/j01.itc.49.4.27118.

1. Z. H. Amur, Y. Kwang Hooi, H. Bhanbhro, K. Dahri, and G. M. Soomro, "ShortText Semantic Similarity (STSS): Techniques, Challenges and Future Perspectives," *Applied Sciences,* vol. 13, no. 6, 2023, doi:

10.3390/app13063911.

1. C. De Boom, S. Van Canneyt, S. Bohez, T. Demeester, and B. Dhoedt, "Learning semantic similarity for very short texts," in *2015 ieee international conference on data mining workshop (icdmw)*, 2015: IEEE, pp. 1229-1234.
2. G. Majumder, P. Pakray, A. Gelbukh, and D. Pinto, "Semantic Textual Similarity Methods, Tools, and Applications: A Survey," *Computación y Sistemas,* vol. 20, no. 4, 2016, doi: 10.13053/cys-20-4-2506.
3. S. Shokat, R. Riaz, S. S. Rizvi, I. Khan, and A. Paul, "Characterization of English braille patterns using automated tools and RICA based feature extraction methods," *Sensors,* vol. 22, no. 5, p. 1836, 2022.
4. H. R. Esmaeel, "Apply android studio (SDK) tools," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 5, no. 5, 2015.
5. T. Dutoit, "High-quality text-to-speech synthesis: An overview," *Journal Of Electrical And Electronics Engineering Australia,* vol. 17, no. 1, pp. 25-36, 1997.
6. S. Hoober and E. Berkman, *Designing mobile interfaces*. " O'Reilly Media, Inc.", 2011.
7. C. Khawas and P. Shah, "Application of firebase in android app development-a study," *International Journal of Computer Applications,* vol. 179, no. 46, pp. 49-53, 2018.
8. B. G. Mateus, "Towards high-quality Android applications development with Kotlin," Université Polytechnique Hauts-de-France, 2021.
9. C. Li and W. Yan, "Braille recognition using deep learning," in *Proceedings of the 4th International Conference on Control and Computer Vision*, 2021, pp. 30-35.
10. N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084,* 2019.
11. B. G. Asefa, "Building Android Component Library Using Jetpack Compose," 2022.
12. J. Pandey and A. R. Asati, "Lightweight convolutional neural network architecture implementation using TensorFlow lite," *International Journal of Information Technology,* vol. 15, no. 5, pp. 2489-2498, 2023.
13. Y. Sukmana and Y. Rosmansyah, "The use of cloud firestore for handling realtime data updates: An empirical study of gamified online quiz," in *2021 2nd international conference on electronics, communications and information technology (CECIT)*, 2021: IEEE, pp. 1239-1244.
14. E. Bisong and E. Bisong, "Google cloud machine learning engine (cloud mle)," *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners,* pp. 545-579, 2019.
15. A. Rodrigues, K. Montague, H. Nicolau, and T. Guerreiro, "Getting smartphones to talkback: Understanding the smartphone adoption process of blind users," in *Proceedings of the 17th international acm sigaccess conference on computers & accessibility*, 2015, pp. 23-32.
16. M. Akram, G. A. Ali, A. Sulaiman, and M. ul Hassan, "Accessibility evaluation of Arabic University websites for compliance with success criteria of WCAG 1.0 and WCAG 2.0," *Universal Access in the Information Society,* vol. 22, no. 4, pp. 1199-1214, 2023.
17. P. Lemenkova, "Testing linear regressions by StatsModel Library of Python for oceanological data interpretation," *Aquatic Sciences and Engineering,* vol. 34, no. 2, pp. 51-60, 2019.

# APPENDIX

## Code

The code for the whole application has been uploaded to the GitHub repository at the following link:

https://github.com/Natiabay/Autonomous-Blind-Examiner-Mobile-application.git