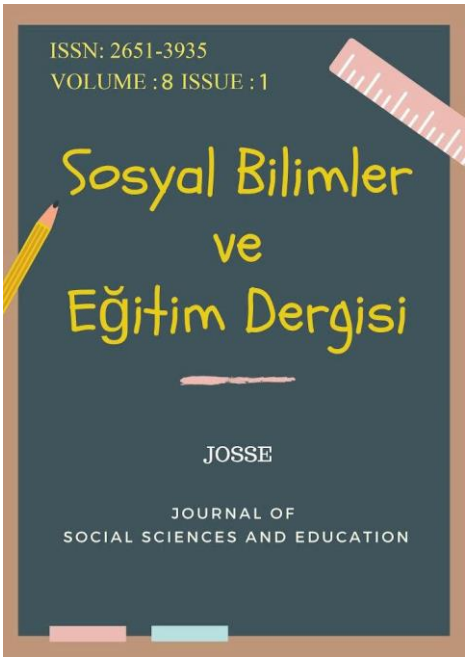
JOURNAL OF SOCIAL SCIENCES AND EDUCATION (JOSSE)



https://dergipark.org.tr/tr/pub/josse

Design of an iOS Mobile Application for the Automated Evaluation of Open-Ended Exams via Artificial Intelligence and Image Processing

** This study was produced from the paper presented at the International Symposium on Measurement, Selection and Placement held on October 04-06, 2024.*

Nazmi Ekin VURAL[1]
Yozgat Bozok University, Faculty of Communication, New Media and Communication, Asst. Prof.
ekin.vural@bozok.edu.tr
Orcid ID: 0000-0003-4198-0407

Article Type: Research Article
Received: 04.05.2025
Accepted: 31.05.2025
Published online: 31.05.2025

Citation: Vural, N. E. (2025). Design of an ios mobile application for the automated evaluation of open-ended exams via artificial intelligence and image processing. *Journal of Social Sciences and Education*, *8*(1), 1-35.

Design of an iOS Mobile Application for the Automated Evaluation of Open-Ended Exams via Artificial Intelligence and Image Processing*

Nazmi Ekin VURAL[1]

Yozgat Bozok University, Faculty of Communication, New Media and Communication

Abstract

Evaluating open-ended exams presents significant challenges in terms of time management and consistency in educational processes. This study aims to develop an iOS-based mobile application, *"Exam Reader"* to streamline the evaluation of handwritten open-ended exam responses by integrating visual recognition and language analysis tools, enabling educators to deliver timely and fair assessments. Developed using the Swift programming language, the application relies on two core technologies. First, handwritten student responses are converted into digital text using Optical Character Recognition (OCR) via the Google Cloud Vision API. These texts are then analyzed for clarity and coherence using the OpenAI API and GPT-4o model, ensuring that students' ideas are presented in a structured, accessible format for evaluation. Finally, the evaluation results and related data are provided to users in PDF format. Designed with a user-friendly interface, the application allows educators to quickly interpret responses and align them with expected learning outcomes through integrated language and image analysis tools. This system offers an innovative model for digitizing, standardizing, and automating open-ended exam evaluations, contributing to the systematic improvement of educational assessment processes. However, the application has limitations. Variations in handwriting and low-quality scans may reduce OCR accuracy, and AI-supported content analysis risks missing contextual nuances. Additionally, the system requires a stable internet connection, limiting offline functionality. Future enhancements, including advanced OCR models, multilingual support, and an offline mode, are planned to address these issues. The application developed in this direction is expected to make a significant contribution to the digitalization of educational assessment and to adapt to next-generation technologies.

Research Article

Received: 04.05.2025
Accepted: 31.05.2025
Published online: 31.05.2025

Keywords: Image processing, Handwriting recognition, AI, Cloud Vision API, OpenAI API, iOS.

* This study was produced from the paper presented at the International Symposium on Measurement, Selection and Placement held on October 04-06, 2024.

[1] Corresponding author:
Asst. Prof.
ekin.vural@bozok.edu.tr
Orcid ID: 0000-0003-4198-0407

Introduction

Open-ended exams are an essential educational tool that provides a detailed measure of students' knowledge, analytical thinking skills, and depth of understanding of topics. However, manual scoring of these exams is time-consuming and can lead to inconsistencies and errors due to the subjective judgment of human raters (Wiser et al., 2016, pp. 841-844). In the process of digitizing handwritten exam answers, the application of user-centered design principles plays a critical role in reducing the risk of errors and increasing process efficiency. According to Norman (2013, p. 72), principles such as discoverability, pointers, feedback, and conceptual models enable users to understand the system's functions and evaluate its interactions. For example, in an Optical Character Recognition (OCR) system, visual markers highlighting the area to be scanned (e.g., blinking frames) and immediate feedback (e.g., preview screens showing scanning accuracy) allow users to avoid erroneous data entry and speed up the process. Feedforward mechanisms guide users to correct actions through beacons and mappings, while feedback facilitates error correction by clearly communicating the results of actions. When these principles are designed to suit user needs in the digitization process, both accuracy and efficiency are increased. Traditional manual scoring methods pose substantial challenges due to their high labor and cost requirements, especially in large-scale exams.

As a solution to these problems, the development of automated scoring systems has accelerated since the 1990s. The first generation of automated scoring systems achieved limited success with rule-based approaches, but advances in machine learning techniques have brought notable innovations to this field (Shermis et al., 2013, p. 2). Modern machine learning systems, especially with supervised learning methods, can evaluate student responses with notable accuracy in terms of contextual accuracy, grammar, and conceptual integrity (Wiser et al., 2016, p. 842). For example, systems such as e-rater® and IntelliMetric™ support the digitization and automatic scoring of handwritten text with machine learning algorithms, while user-centered interfaces enable educators and students to use these systems with minimal learning curves (Burstein et al., 2013, p. 55; Schultz, 2013, p. 89). This integrated approach increases the accessibility and effectiveness of technology in educational settings, enabling users with different skill levels to interact with the system with confidence.

The digitization and evaluation of handwritten open-ended responses are facilitated by advanced technologies such as Tesseract OCR and Multidimensional Recurrent Neural Networks (MDRNN) (Smith, 2007, p. 629; Graves & Schmidhuber, 2009, p. 546). Tesseract OCR supports digitization by recognizing text through text line detection and connected

component analysis, while MDRNN offers high accuracy in language-independent transcription by recognizing different writing styles without dependency on specific languages or alphabets. User-centered design, with markers and feedback (Norman, 2013, p. 72), allows educators and students to learn the process quickly, while integration with automated essay evaluation (AEE) systems increases assessment accuracy (Shermis & Burstein, 2013, p. 2).

Transformer-based Large Language Models (LLMs), based on attentional mechanisms, evaluate open-ended responses quickly and consistently (Vaswani et al., 2017, p. 5999). These models provide individualized feedback in instruction by analyzing the grammar, fluency, and content of responses, thus identifying students' strengths and weaknesses. However, these technologies can carry ethical risks, such as generative AI tools enabling plagiarism or cheating, and biases in data sets leading to unfair assessments (Tanberkan et al., 2024, p. 144). These risks can be mitigated by adopting a participatory approach in which educators, students, and other stakeholders are actively involved in shaping the development and implementation of the system. Promoting AI literacy and fostering awareness of ethical use further empowers educational communities to engage with these technologies responsibly and reflectively.

In this study, we discuss a process that involves automatically scanning open-ended exam responses using an iOS-based mobile application, digitizing the handwritten content, and interpreting it with a large language model. The application recognizes students' handwritten answers through OCR technology, evaluates them using an AI-assisted scoring system, and generates the results in PDF format. This approach aims to offer an innovative contribution to assessment and evaluation practices within the educational technology literature.

1. Artificial Intelligence and Image Processing Based Automation Approaches in the Evaluation of Open-Ended Exams

1.1. Conceptual Framework and General Approach

This chapter systematically reviews recent technological advances in the field of automated assessment of open-ended exams. Innovative solutions such as machine learning, OCR, handwriting recognition, and large language models are employed to digitize, analyze, and score handwritten student responses. In this context, the approach offers multidimensional benefits, including time savings in assessment processes, reduced human error, and faster pedagogical feedback. Each subsection in this part offers a concrete illustration of how digital enhancement is shaping education, through an in-depth examination of technical components and practical implementation examples.

1.2. Automation Studies

Open-ended exams effectively assess students' conceptual understanding and analytical thinking, but scoring these exams by human assessors is time- and labor-intensive. For instance, one study reported that it takes a human evaluator an average of four minutes to evaluate a single answer for nine different ideas, equating to approximately ten hours to evaluate five questions for a class of 30 students (Wiser et al., 2016, p. 841). To address these challenges, machine learning-based automated assessment systems have been developed. However, these systems have not yet fully matched the performance of human evaluators. In a competition on machine scoring of short-form constructed responses, the average squared weighted kappa value for human raters was 0.90, while the best machine algorithm achieved 0.76, indicating higher reliability among human raters (Shermis, 2015, p. 60).

Natural Language Processing (NLP) model developed for identifying research questions in Hebrew scientific reports demonstrated high accuracy in detecting their presence or absence (Ariely, Nazaretsky, & Alexandron, 2020, p. 565). This finding underscores the potential of NLP-based systems to accurately identify specific components of open-ended responses in automated assessment processes. However, the effectiveness of automated writing evaluation (AWE) systems in improving student writing is significantly influenced by pedagogical practices, with more favorable outcomes observed when AWE is integrated with human feedback during the drafting and revising process (Chen & Cheng, 2008, p. 107). Without sufficient teacher facilitation, reliance on AWE can lead to student frustration and limited improvement in writing quality, particularly in areas like coherence and content development.

Tools such as EvoGrader, which uses supervised machine learning to analyze open-ended student responses in biology education, assess concepts like natural selection with accuracy rates exceeding 90% (Wiser et al., 2016, pp. 841–847). While EvoGrader is effective for formative assessments, caution is advised when using it for high-stakes grading due to potential limitations in capturing nuanced responses. Similarly, e-rater® V.2, developed by Educational Testing Service (ETS), plays a significant role in the automated assessment of open-ended written texts. It conducts structural, grammatical, and content-based analyses using a single scoring model applicable to all prompts of an assessment (Attali & Burstein, 2006, pp. 2, 7–12). Key criteria, such as the organization and development of discourse elements and the proportion of grammatical, usage, and spelling errors, enhance the system’s validity. However, e-rater® V.2 has limitations, including vulnerability to irrelevant content (Attali & Burstein, 2006, p. 4), making it more suitable for standardized tests and classroom assessments when paired with human oversight.

In contrast, c-rater, also developed by ETS, focuses on short-answer structured responses and has achieved high reliability in specific contexts, with unweighted kappa values ranging from 0.55 to 0.94 for reading and math questions (Shermis, 2015, p. 49). Despite this, inconsistencies between machine scoring and human raters persist, particularly due to challenges like conceptual diversity and linguistic variations (Shermis, 2015, pp. 49, 62). Recent efforts to improve reliability involve hybrid models combining rule-based and machine learning approaches (Ramesh & Sanampudi, 2022, p. 2499). The rise of distance education during the COVID-19 pandemic further accelerated the need for automated evaluation of open-ended responses. Studies have shown that NLP techniques, combined with similarity measures like soft cosine similarity and machine learning algorithms such as Support Vector Machine (SVM), Random Forest, and Multinomial Naïve Bayes, can successfully evaluate descriptive responses, with Multinomial Naïve Bayes achieving 92% accuracy (Ahmed, Hina, & Asif, 2021, pp. 4887–4891).

Automated Essay Scoring (AES) systems offer the potential to increase consistency in assessment processes and provide faster feedback by reducing the workload of human raters (Ke & Ng, 2019, p. 6300). However, these systems are primarily developed with English-oriented datasets, limiting their applicability to morphologically rich languages (MRLs) (Ke & Ng, 2019, p. 6303). To address this, a study demonstrated that Hebrew biology open-ended questions can be successfully evaluated using machine learning and NLP techniques (Ariely, Nazaretsky, & Alexandron, 2020, p. 565). The AdaBoost.M1 algorithm demonstrated superior performance in automatically scoring open-ended physics questions in Turkish, achieving accuracy rates ranging from 0.65 to 0.99 across four university-level questions with varying complexity (Çınar et al., 2020, p. 3834). This study highlights the potential of machine learning to address the challenges of automated short-answer grading in morphologically rich languages, where linguistic complexity poses significant obstacles to traditional scoring systems. To enhance the practical applicability of AES systems, models must account for language and cultural differences and undergo continuous validation tailored to specific educational contexts (Attali & Burstein, 2006; Shermis, 2015; Ke & Ng, 2019, p. 6306).

The accurate digitization of handwritten student responses is a critical step in the automated evaluation of open-ended exam responses. Two key technologies, OCR and Handwriting Recognition, have seen substantial advancements in recent years, driven by deep learning techniques and mobile application developments.

1.3. Optical Character Recognition (OCR) Technologies

OCR systems aim to detect characters in printed or written documents and convert them into digital text. Tesseract, one of the open-source OCR engines, was developed by Hewlett-Packard between 1984 and 1994 and released as open source in 2005. Tesseract's innovative structure is based on features such as detecting lines of text through connected component analysis and easily detecting both black-and-white and inverted (black on white) text. Architecturally, it involves a two-stage recognition process (initial recognition and adaptive classification) with the identification of connected components (blobs), line and word alignment. A second pass improves accuracy by re-evaluating words recognized with low accuracy. This approach provides high performance on documents that contain scanning artifacts or different fonts (Smith, 2007, pp. 629, 633). The main stages in OCR processes include thresholding, edge detection, segmentation and morphological operations. These stages are especially critical for separating and clarifying the text from the background. Histogram equalization, grayscale and binary imaging techniques are also important steps that improve OCR performance (Gonzalez & Woods, 2008, pp. 197- 223).

Nowadays, OCR technologies play an effective role in the educational application of optical mark recognition (OMR) systems for mobile devices. High accuracy rates (90%-95%) are achieved by reading optical forms with mobile device cameras, supported by image processing techniques such as grayscale, noise reduction and edge detection (Turhan et al., 2023, pp. 177-178, 183). These systems demonstrate the successful integration of Optical Form Recognition and OMR applications on mobile platforms (Turhan, Bozkurt, & Şahin, 2023, p. 170). Thanks to cross-platform support, these applications, which work on Android, iOS and web browsers, contribute to digitalization and time saving in education.

1.4. Handwriting Recognition Technologies

Handwriting recognition, unlike OCR, aims to digitize text written in free form rather than in print. In particular, offline handwriting recognition requires processing only image data, which makes the process more complex.

One of the most important advances in this field is the integration of MDRNN and Connectionist Temporal Classification (CTC) techniques. It has been shown that the MDRNN structure can simultaneously model horizontal and vertical contexts in the image, thus providing robustness against local distortions. MDRNN utilizes the contextual information around each pixel to create a flexible representation against local variations from different angles. In particular, Multidimensional LSTM (MDLSTM) cells effectively model long-distance

dependencies and have been successful with irregular and continuously varying data such as handwriting (Graves & Schmidhuber, 2009, pp. 545-550).

Furthermore, in modern systems for handwriting recognition, image processing steps include segmentation and normalization before character detection (Shah & Yousaf, 2007, pp. 1-2). Especially for low-resolution image recognition, these preprocessing techniques support system performance by providing consistent data input. These methods have made it possible to perform recognition directly at the full line or paragraph level without prior letter or word segmentation. Indeed, MDRNN-CTC-based systems have outperformed traditional systems in Arabic handwriting recognition competitions with over 91% accuracy rates (Graves & Schmidhuber, 2009, pp. 545-551).

1.5. OCR and Handwriting Recognition on Mobile Platforms

The migration of image processing techniques to mobile devices has substantially increased the accessibility of automated test assessment applications in education. Optical reader systems developed for mobile platforms can provide fast feedback to the user by processing optical forms received through the camera in real time (Turhan et al., 2023, p. 171).

Artificial intelligence plays an important role in digital image processing, especially in mobile platforms, in steps such as segmentation and edge detection. Machine learning-based methods are increasingly used in applications such as healthcare and real-time image analysis on mobile devices. Also, artificial intelligence gains analytical capabilities as machine learning and deep learning methods process and interpret complex data patterns (Çeliker & Gürsoy, 2025, p. 113; Lepakshi, 2022, p. 402). For example, deep learning techniques provide high accuracy in areas such as cancer diagnosis in medical imaging, and these technologies are being integrated into mobile platforms (Hidayat et al., 2025, pp. 1-8). In addition, applications of artificial intelligence in the field of education can be indirectly supported by mobile technologies by providing solutions for individual needs through personalized learning systems (Keser Ateş et al., 2025, p. 19). AI-driven image processing techniques, such as filtering and edge detection, enhance image quality by reducing noise and extracting meaningful features, enabling accurate OCR for mobile-based educational assessment systems (Hussain et al., 2020, p. 245). Tools like OpenCV provide robust support for these processes, facilitating real-time analysis of handwritten student responses on mobile platforms.

The integration of artificial intelligence techniques in mobile image processing systems has made it possible to use advanced features such as automatic target recognition, document classification, and fault tolerance (Azizi et al., 2024, p. 133). These developments contribute to

the more accurate and faster operation of mobile-based open-ended exam assessment applications in educational technologies.

1.6. Use of Artificial Intelligence and Large Language Models in Education

The rapid development of artificial intelligence technologies in recent years has led to major shifts in the field of education. In particular, new generation artificial intelligence applications such as LLMs are reshaping many processes in education such as assessment, evaluation, content production and individualized learning. Computational language tools now allow learning processes to be more attuned to individual student needs, helping educators identify strengths and areas for improvement. Early automated writing evaluation systems demonstrated the potential to save teachers' time and encourage student revision, though their effectiveness was limited by superficial revisions without pedagogical support (Warschauer & Grimes, 2008, p. 33). These findings underscore the importance of integrating modern large language models with effective teaching strategies to enhance assessment and learning outcomes in education. In particular, LLMs such as ChatGPT reduce the workload of educators in automatic scoring of open-ended exams and intelligent assessment processes, provide objective results, and improve student test scores by 30% and reduce self-reported anxiety levels by 20% (Tanberkan et al., 2024, pp. 140-142; Wang et al., 2024, p. 1). However, the capacity of these models to produce high-quality text increases ethical risks such as plagiarism and cheating, threatening the fairness of the assessment processes. Furthermore, issues such as algorithm biases and personal data privacy complicate the use of LLMs in education. To mitigate these risks, authentic exam designs, plagiarism detection tools and ethical use trainings are recommended; at the same time, it is emphasized that theoretical frameworks such as flow theory should be used to develop AI-supported learning environments (Karadağ, 2023, pp. 829-831; Wang et al., 2024, p. 15).

The transformer architecture lies at the heart of the success of large language models. Vaswani et al. (2017: pp. 5999-6015) showed in their study *"Attention Is All You Need",* that this architecture based on the attention mechanism brings notable changes in language modeling and significantly increases learning speed and model performance by parallel processing on data. Thanks to the Transformer structure, much longer textual relations could be learned compared to previous models and there were noticeable differences in language processing tasks.

OpenAI's GPT-2 model, which was developed based on this architecture, achieved notable success in multi-task learning and demonstrated human-like performance in NLP tasks

by exhibiting capabilities such as zero-shot learning (Radford et al., 2019). Radford et al. showed that the model can be effective on different tasks such as question answering, summarization, and translation, simply by training it on large datasets.

The BERT (Bidirectional Encoder Representations from Transformers) model stands out as an innovation in NLP. Transformer-based models have been reported to provide notable progress in individualized assessment processes in education. Devlin et al. (2018, pp. 1-6) stated that BERT generates contextual representations of texts more efficiently thanks to its bidirectional attention mechanism and outperforms previous methods in many language processing tasks. BERT's success offers the potential for wide application, especially in areas such as semantic analysis of open-ended responses in education, content assessment, and individualized review of student work. Furthermore, Cui and Liang (2024, pp. 8-15) demonstrated that BERT supports educational processes by producing fast, reliable and objective results in complex tasks such as translation quality assessment. This shows that BERT offers both scalable and in-depth analysis in education.

Another development that supports the use of artificial intelligence and large language models in education is deep learning methods. LeCun, Bengio, and Hinton (2015, p. 436) emphasize that deep learning innovates tasks such as natural language understanding and content generation through its ability to extract high-level features from complex data and learning abstract representations. These methods have enabled LLMs to learn abstract conceptual knowledge with multi-layered structures. However, the use of LLMs in education brings some risks. Tanberkan et al. (2024, pp. 143-144) draw attention to ethical violations, data bias, accuracy problems and the risk of artificial intelligence systems producing hallucinatory (misleading) content. Therefore, it is emphasized that AI applications in education should be designed within the framework of transparency, fairness and reliability principles.

As a result, artificial intelligence and large language models have the potential to enhance teaching, assessment, and evaluation processes in education. Applications such as automatic item generation, automatic assessment and personalized feedback can make educational processes more efficient and individualized (Karadağ, 2023, p. 828). In particular, the *"hallucinatory snowball effect",* where LLMs generate additional false assertions to support false answers, can lead to reliability issues in education (Zhang et al., 2023, pp. 1-2). To mitigate these risks, increasing users' AI literacy and developing pedagogical strategies will support the ethical and efficient integration of technology in educational settings (Karadağ, 2023, p. 833).

1.7. Problem Definition

In traditional assessment processes, manual evaluation of open-ended examinations incurs high time and labor costs, as well as problems such as inconsistencies and risks of error due to subjective interpretations of the assessors. Especially in educational settings with large class sizes, detailed review of written responses is often not possible, which reduces the quality of feedback and delays learning processes. Automated assessment of open-ended exams is a critical need in the field of assessment and evaluation. Such solutions require not only accurate recognition of the written text, but also reliable analysis of the contextual and conceptual correctness of the answers. In particular, the digitization of handwritten responses necessitates the coordinated use of OCR and artificial intelligence-based text analysis techniques.

1.8. Purpose and Scope of the Study

The main objective of this study is to ensure that handwritten open-ended exam answers are automatically scanned, digitized and evaluated based on content using an artificial intelligence-supported model in a fast, accurate and reliable manner through a mobile-based system. To this end, an OCR process supported by image processing techniques and an automatic scoring mechanism using LLMs were designed.

The scope of the study is based on three main technical components:

First, students' handwritten responses are scanned using a mobile device camera and converted into digital text using open-source solutions such as the Tesseract OCR engine. Second, the resulting digital text is analyzed for spelling and phrasing corrections using large language models and is then given a meaningful structure. Finally, a success score for the student's answer is generated through a comparative analysis with the correct answers and evaluation criteria and presented to the evaluator as a PDF.

The study was limited to open-ended questions of a specific structure. Answers containing complex graphs, figures or tables were excluded from the scope of this system. Furthermore, the evaluation process will be based on content integrity and conceptual accuracy; stylistic or aesthetic aspects (e.g. spelling, page layout) will not be taken into account. By improving the reliability of handwriting recognition, this study aims to reduce human error in measurement and evaluation processes, ease the workload of instructors in educational settings, and increase the capacity to provide faster feedback to students.

Method

The mobile application *"Exam Reader"* developed in this study integrates a range of modern technologies for the automatic evaluation of open-ended exam answers (Table 1). The application process consists of scanning the exam papers, digitizing them with OCR, content analysis supported by artificial intelligence, and automatic scoring.

1. Application Development Process and General Structure

The iOS-based mobile application developed in this study combines the processes of automatically digitizing handwritten answers of open-ended exams, evaluating them with artificial intelligence-supported content analysis and presenting the results to the user in a holistic structure. The application was coded in Swift programming language using Apple's official development environment Xcode. The code block, consisting of 1,028 lines in total, is structured with a modular architecture approach.

1.1 Core Technologies Used

Table 1

The Main Technologies Employed in the Application

Component	Description
Swift	The programming language used for development
VisionKit	Document scanning and image acquisition
CoreImage	Image filtering and preprocessing
Google Cloud Vision API	OCR (handwriting recognition) processes
OpenAI GPT-4o	Text correction and content scoring
PDFKit & QuickLook	PDF generation and display
UIKit & SwiftUI	Building user interfaces

The mobile application developed in this study was designed to deliver both technical functionality and a user-friendly experience. Built using the Swift programming language in the Xcode environment, it is fully compatible with iOS versions 17.5.3 and 18.2.

The primary development objectives of the application were defined as follows:

Accuracy: High-precision digitization of handwritten exam responses.

Consistency: Objective and standardized scoring based on predefined criteria.

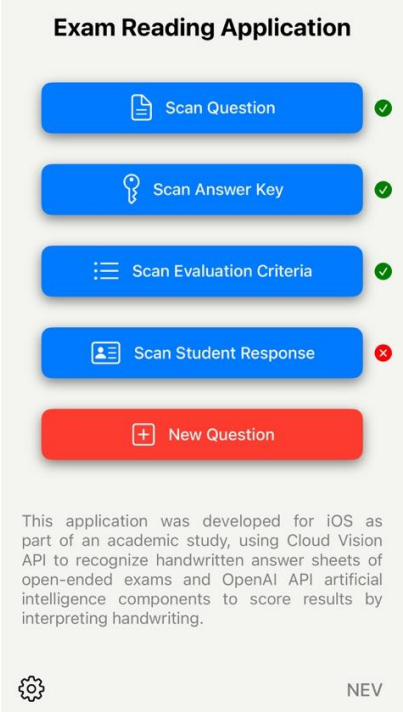
Usability: Intuitive step-by-step guidance for users without technical expertise.
Accessibility: A flexible interface adaptable to various device sizes and user needs.

The app integrates Apple's VisionKit and CoreImage for image processing, Google Cloud Vision API for OCR, and OpenAI's GPT-4o model for text editing and content analysis. PDF outputs are generated using PDFKit for transparent evaluation results. The user interface combines UIKit and SwiftUI, with button-based navigation and consistent visual/textual feedback to enhance user experience. Error handling and API communication prioritize secure data flow and clear error messages.

1.2. Interface Design and Button Workflow

Picture 1

User Interface



The application's user interface is designed to simplify the exam evaluation process for users, such as educators, by providing a clear and intuitive workflow. Four main buttons guide users through the process: "Scan Question", "Scan Answer Key", "Scan Evaluation Criteria", "Scan Student Response" and "New Question" (Picture 1). Each button triggers a specific action, such as capturing a question or student answer using the device's camera. Visual feedback—a green checkmark (✔) for successful scans or a red cross (✕) for errors—ensures users can track their progress. The interface dynamically updates button availability based on

the current step, preventing errors and enhancing usability. For example, the "Scan Student Response" button is only enabled after the question-and-answer key are scanned. This design minimizes the learning curve, making the application accessible to users without technical expertise.[1]

1.3. Workflow Management and Resetting State

The application manages the evaluation process through a structured workflow that ensures seamless transitions between steps, such as scanning questions, answer keys, evaluation criteria, and student responses. After completing an evaluation or starting a new one, the system resets all data and interface components to their initial state, allowing users to begin a fresh cycle without errors ("New Question" button Picture 1). This reset process clears stored information, such as scanned texts and button statuses, and prepares the interface for the next evaluation. For example, an educator can evaluate one student's exam and automatically begin a new session for another without manual input. This streamlined approach enhances efficiency and reduces the risk of data overlap, making the application practical for repeated use in educational settings.[2]

2. Image Processing and OCR Process

The application processes scanned exam papers to convert handwritten responses into digital text, enabling automated evaluation. This process begins with image processing, where the captured images—such as questions, answer keys, evaluation criteria, or student responses—are enhanced for clarity. Adjustments like brightness correction and noise reduction ensure the text is legible, improving the accuracy of subsequent steps. Following this, OCR technology, powered by Google Cloud Vision API, transforms the handwritten text into machine-readable digital text. The OCR process is optimized for the Turkish language by setting a language hint to "Turkish," which enhances recognition accuracy for Turkish handwriting. Once the text is digitized, it is ready for further analysis and evaluation. This seamless integration of image processing and OCR allows the application to handle handwritten exams efficiently, saving time for users like educators while maintaining high accuracy in text recognition.[3]

[1] For technical details on the interface implementation, see Appendix 1.
[2] For technical details on the reset process, see Appendix 2.
[3] For technical details on the image processing and OCR implementation, see Appendix 3.

2.1. Image Acquisition

The application begins the evaluation process by capturing images of exam papers, such as questions, answer keys, evaluation criteria, or student responses, using the device's camera. This image acquisition step ensures that handwritten content is accurately recorded for further processing. Once captured, the images undergo preprocessing to enhance their quality and improve text recognition accuracy. This involves adjustments like increasing brightness, enhancing contrast, and reducing noise, which make the handwritten text clearer and easier to digitize. For instance, a faintly written student response can be made more legible through these enhancements, ensuring reliable results in the subsequent OCR step. This process allows the application to handle a variety of handwriting styles and paper conditions, making it practical for real-world educational settings.[4]

2.2. Use of OCR Technology

The application uses OCR technology to convert handwritten exam responses into digital text, enabling automated evaluation. This step relies on Google Cloud Vision API, a powerful tool that accurately recognizes text in images, including handwritten content. After the exam papers are scanned and preprocessed, the API analyzes the images and extracts the text, such as student answers or evaluation criteria. To ensure high accuracy for Turkish handwriting, the OCR process is configured with a language hint set to "Turkish," which helps the API better interpret Turkish characters and grammar. Once the text is digitized, it is ready for further analysis, such as comparison with the answer key or evaluation by system. This OCR process ensures that handwritten responses are reliably converted into a format suitable for automated scoring, saving time and reducing manual effort for users like educators.[5]

2.3 Handwriting Recognition and Text Editing Process

The student responses transferred into digital format through the OCR process generally contain various spelling and grammatical errors and thus must be edited before becoming meaningful and suitable for evaluation. In this context, the raw texts obtained from handwriting were linguistically and structurally improved using OpenAI's GPT-4o model within the application. This process ensures grammatical integrity and also increases the accuracy of AI-supported scoring.

[4] For technical details on image acquisition, see Appendix 4.
[5] For technical details on the OCR implementation, see Appendix 5.

2.3.1 Handwriting Recognition Technologies

The application leverages handwriting recognition technologies to accurately interpret and digitize handwritten exam responses, a critical step in automating the evaluation process. Building on the OCR process, this step uses advanced algorithms to specifically handle the variability of handwriting styles, such as different letter shapes or writing clarity. The Google Cloud Vision API, which powers the OCR, includes handwriting recognition capabilities that are fine-tuned for diverse scripts, including Turkish. This ensures that even challenging handwritten texts—such as those with cursive writing or faint ink—are correctly converted into digital text. The digitized text is then prepared for further analysis, such as comparison with the answer key or AI-based scoring. This technology enables the application to process a wide range of handwritten responses efficiently, reducing the manual workload for users like educators and ensuring consistent evaluation results.[6]

2.3.2 Post-OCR Text Editing

After the OCR process converts handwritten exam responses into digital text, the application performs post-OCR text editing to ensure the extracted text is accurate and suitable for evaluation. This step addresses common OCR errors, such as misrecognized characters or formatting issues, which can occur due to handwriting variations or image quality. The application uses OpenAI's GPT-4o model to intelligently refine the text by correcting errors, standardizing formatting, and ensuring the text aligns with the expected context (e.g., a student's response to a specific question). For example, if the OCR misinterprets a handwritten "5" as an "S," GPT-4o can correct this based on the context of the question. To provide transparency, the application calculates the percentage of changes made to the original OCR output using the Levenshtein distance method, which measures the number of edits needed to transform the original text into the refined version. This percentage is reported in the final PDF (e.g., "Text edited by AI by X%"), allowing users to understand the extent of modifications. This ensures the digitized text is reliable for automated scoring, enhancing the accuracy of the evaluation for users like educators.[7]

2.4 Content Analysis, Evaluation, and PDF Result Generation

This section outlines the integrated process of analyzing, evaluating, and presenting student responses in a comprehensive and transparent manner. The application performs content

[6] For technical details on the handwriting recognition implementation, see Appendix 6.
[7] For technical details on the post-OCR text editing implementation, see Appendix 7.

analysis, scores responses based on predefined evaluation criteria, and compiles results into a visually accessible PDF report, ensuring accuracy, consistency, and usability for educators and students.

2.4.1 Scanning and Defining Evaluation Criteria

The application enables users, such as educators, to scan and define evaluation criteria, which form the foundation for automated scoring. Users scan a document, such as a rubric or scoring guide, outlining expected answers and corresponding points. Using OCR, the application extracts these criteria into digital text, ensuring the system understands specific scoring requirements, such as key concepts or phrases. For example, a history exam question might require mentioning "the Industrial Revolution" and "its economic impact" for full points. This process aligns student responses with learning goals, promoting consistency while allowing nuanced interpretation. By simplifying the assessment process, it contributes to fairer and more transparent grading. [8]

2.4.2 AI-Based Analysis of Student Responses

Once student responses are digitized and refined through OCR and post-OCR text editing, the OpenAI GPT-4o model analyzes them against the predefined evaluation criteria. The system compares each response to key learning goals, suggesting a score based on conceptual understanding, accuracy, and completeness. For instance, if a history question requires "the Industrial Revolution" and "its economic impact," GPT-4o identifies these elements and assigns points accordingly. The system also provides a confidence estimate for each evaluation, offering transparency to educators. This AI-based analysis ensures objective, efficient, and consistent grading, reducing educator workload while maintaining fairness. [9]

2.4.3 Scoring, Feedback, and PDF Report Generation

At the final stage, the application compiles evaluation results into a comprehensive PDF report using the PDFKit library, presenting scores, feedback, and transparent insights in a professional, archivable format. Each response is scored based on how well it meets the evaluation criteria, with supportive indicators such as the degree of text editing (e.g., 15% for OCR corrections) and evaluation confidence (e.g., 95%). For example, a report might show a score of 8/10 for a response, noting the inclusion of "the Industrial Revolution" but missing "its

[8] For technical details on scanning and defining evaluation criteria, see Appendix 8.
[9] For technical details on the AI-based analysis implementation, see Appendix 9.

economic impact." The report includes feedback for educators, highlighting student strengths and areas for improvement, and for students, offering constructive comments like, "Consider adding economic effects for a complete answer." Generated with AI support, this feedback ensures consistency with scoring criteria and supports tailored instruction and student learning. The PDF format ensures readability, shareability, and archivability, allowing educators to distribute or store results efficiently. This integrated process enhances evaluation efficiency, transparency, and actionable outcomes. [10]

In conclusion, the developed mobile application enhances the efficiency of open-ended exam evaluation by reducing processing time and improving assessment consistency. It offers a new model for assessment and evaluation that emphasizes speed, accuracy, and transparency.

Results and Recommendations

The iOS-based mobile application developed within the scope of this study adopts an integrated approach to automatically scanning, digitizing, correcting, and evaluating students' handwritten answers in open-ended exams through content analysis. Considering the increasing need for digitalization of assessment and evaluation processes in education, it is evident that such applications offer notable advantages such as speed, consistency and objectivity.

The application offers a key advantage by reducing the time required for manual assessment processes. Reliable handwriting recognition using the Google Cloud Vision API, followed by editing and evaluation using the OpenAI GPT-4o model, increased both the speed and accuracy of the evaluation process. The high accuracy rates offered by LLMs for content analysis made it possible to obtain consistent results even when compared to human raters. In addition, a simple and user-friendly interface design was adopted in the development of the application, and wide compatibility with mobile devices was ensured so that different user groups could easily use the application. The creation of PDF-based results reports provided educators with the opportunity for long-term data storage and transparent communication with students.

However, some limitations of the system were also identified. Differences in the legibility of handwriting, especially during the OCR process, could lead to fluctuations in recognition performance. Individual differences in handwriting and low-quality scans sometimes reduced the accuracy of text recognition, which could affect the results of the

[10] For technical details on scoring and result generation, the PDF report generation implementation and the feedback generation implementation see Appendix 10.

automatic assessment. It was also considered that AI-assisted content analysis runs the risk of not fully understanding the context. As emphasized in the literature, artificial intelligence models can sometimes make incomplete or misleading assessments based on superficial clues. Another limitation of the system is that it requires an uninterrupted internet connection for full performance. The lack of offline modes limits the usability of the application, especially in rural or low bandwidth areas, as access to the Google Cloud Vision API and OpenAI API can be difficult.

In terms of future development, supporting the application with more advanced OCR models is an important goal. In particular, integration with advanced handwriting recognition systems such as MDRNN can be achieved to improve recognition accuracy. In addition, providing multilingual support will enable students from different language groups to be included in the system. In this context, the integration of multilingual models should be planned so that educational institutions can effectively use the application in different language environments. The development of an offline mode of use has also been identified as an important area of future work. In particular, developing a hybrid structure where local OCR and evaluation engines can be run on the device will increase the independence and accessibility of the application.

In conclusion, the mobile application developed in this study presented an innovative approach in the field of automatic assessment of open-ended exams. It has made a substantial contribution in terms of providing speed, objectivity and transparency in measurement and evaluation processes in education and has strongly served the goals of supporting student learning and reducing educator workload. This model, created with the combination of artificial intelligence and mobile technologies, has laid a solid foundation for future studies in the field of educational technologies and paved the way for new development opportunities.

References

Ahmed, F., Hina, R., & Asif, M. (2021). Evaluation of descriptive answers of open-ended questions using NLP techniques. *Turkish Journal of Computer and Mathematics Education, 12*(10), 4887–4896. https://doi.org/10.1109/ICCIS54243.2021.9676405

Ariely, N., Nazaretsky, T., & Alexandron, G. (2021). Machine learning and Hebrew NLP for automated assessment of open-ended questions in biology. https://doi.org/10.1007/s40593-021-00283-x

Ariely, M., Nazaretsky, T., & Alexandron, G. (2020). First steps towards NLP-based formative feedback to improve scientific writing in Hebrew. In A. N. Rafferty, J. Whitehill, V. Cavalli-Sforza, & C. Romero (Eds.), *Proceedings of the 13th International Conference on Educational Data Mining (EDM 2020)* (pp. 565–568). Retrieved from https://osf.io/preprints/edarxiv/pe5ky

Attali, Y., & Burstein, J. (2006). Automated Essay Scoring With e-rater® V.2. The Journal of Technology, Learning and Assessment, 4(3). Retrieved from https://ejournals.bc.edu/index.php/jtla/article/view/1650

Azizi, S., Mahmoudi, F., Alizadehsani, R., & Nahavandi, S. (2024). Image processing and artificial intelligence for apple detection and localization: A comprehensive review. *Artificial Intelligence Review, 57*(2), 123–150. https://doi.org/10.1016/j.cosrev.2024.100690

Burstein, J., Tetreault, J., & Madnani, N. (2013). The E-rater® automated essay scoring system. In M. D. Shermis & J. Burstein (Eds.), *Handbook of automated essay evaluation: Current applications and new directions* (pp. 55–67). Routledge. Retrieved from https://psycnet.apa.org/record/2013-15323-004

Chen, C. F. E., & Cheng, W. Y. E. (2008). Beyond the design of automated writing evaluation: Pedagogical practices and perceived learning effectiveness in EFL writing classes. *Language Learning & Technology, 12*(2), 94–112. Retrieved from https://www.lltjournal.org/item/10125-44145/

Cui, Y., & Liang, M. (2024). Automated scoring of translations with BERT models: Chinese and English language case study. *Applied Sciences, 14*(5), 1925. https://doi.org/10.3390/app14051925

Çeliker, N., & Gürsoy, S. (2025). Artificial intelligence in human resource management: A bibliometric analysis on trends, prospects and future research agenda. *The Journal of Business Science, 13*(1), 97-120. https://doi.org/10.22139/jobs.1594699

Çınar, A., Kara, A., Koç, H., & Kılıç, S. (2020). Machine learning algorithm for grading open-ended physics questions in Turkish. *Education and Information Technologies, 25*(5), 3821–3844. http://doi.org/10.1007/s10639-020-10128-0

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint* arXiv:1810.04805. https://arxiv.org/abs/1810.04805

Gonzalez, R. C., & Woods, R. E. (2008). *Digital image processing* (3rd ed.). Pearson Prentice Hall. Retrieved from https://sde.uoc.ac.in/sites/default/files/sde_videos/Digital%20Image%20Processing%203rd%20ed.%20-%20R.%20Gonzalez,%20R.%20Woods-ilovepdf-compressed.pdf

Graves, A., & Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in Neural Information Processing Systems, 21*, 545–552. Retrieved from https://www.cs.toronto.edu/~graves/nips_2008.pdf

Hidayat, E. Y., Hastuti, K., & Muda, A. K. (2025). Artificial intelligence in digital image processing: A bibliometric analysis. *Intelligent Systems with Applications, 25*, 200466. https://doi.org/10.1016/j.iswa.2024.200466

Hussain, S., Dixit, P., & Hussain, M. S. (2020). Image processing in artificial intelligence. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 6*(5), 244–249. https://doi.org/10.32628/CSEIT206542

Karadağ, N. (2023). The impact of artificial intelligence on online assessment: A preliminary review. *Journal of Educational Technology & Online Learning*, 6(4), 822-837. https://doi.org/10.31681/jetol.1351548

Ke, Z., & Ng, V. (2019). Automated essay scoring: A survey of the state of the art. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)* (pp. 6300–6308). International Joint Conferences on Artificial Intelligence. https://doi.org/10.24963/ijcai.2019/879

Keser Ateş, S., Kaleci, F., & Erdoğan, A. (2025). Artificial intelligence in education: A bibliometric analysis. *Ahmet Keleşoğlu Faculty of Education Journal (AKEF), 7*(1), 14–36. https://doi.org/10.38151/akef.2025.147

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lepakshi, V. A. (2022). Machine learning and deep learning based AI tools for development of diagnostic tools. In A. Parihar, R. Khan, A. Kumar, A. K. Kaushik, & H. Gohel (Eds.),

Computational approaches for novel therapeutic and diagnostic designing to mitigate SARS-CoV-2 infection (pp. 399–420). Academic Press. https://doi.org/10.1016/B978-0-323-91172-6.00011-X

Norman, D. A. (2013). *The design of everyday things* (Rev. & expanded ed.). Basic Books. https://jnd.org/books/the-design-of-everyday-things-revised-and-expanded-edition/

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI.* Retrieved from https://openai.com/research/language-unsupervised

Ramesh, D., & Sanampudi, S. K. (2022). An automated essay scoring systems: A systematic literature review. *Artificial Intelligence Review, 55*, 2495-2527. https://doi.org/10.1007/s10462-021-10068-2

Shermis, M. D. (2015). Contrasting state-of-the-art in the machine scoring of short-form constructed responses. *Educational Assessment, 20*(1), 46–65. https://doi.org/10.1080/10627197.2015.997617

Shermis, M. D., Burstein, J., & Apel Bursky, S. (2013). Introduction to automated essay evaluation (AES) NLP. In M. D. Shermis & J. Burstein (Eds.), *Handbook of automated essay evaluation: Current applications and new directions* (pp. 1–15). Routledge. https://doi.org/10.4324/9780203122761

Schultz, M. T. (2013). The IntelliMetric™ automated essay scoring engine – A review and an application to Chinese essay scoring. In M. D. Shermis & J. Burstein (Eds.), *Handbook of automated essay evaluation: Current applications and new directions* (pp. 89–98). Routledge. https://doi.org/10.4324/9780203122761

Shah, F. T., & Yousaf, K. (2007). Handwritten digit recognition using image processing and neural networks. *Proceedings of the World Congress on Engineering 2007, Vol I (WCE 2007).* Retrieved from https://www.iaeng.org/publication/WCE2007/WCE2007_pp648-651.pdf

Smith, R. (2007). An overview of the Tesseract OCR engine. *Proceedings of the Ninth International Conference on Document Analysis and Recognition*, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

Tanberkan, H., Özer, M., & Gelbal, S. (2024). Impact of artificial intelligence on assessment and evaluation approaches in education. *International Journal of Educational Studies and Policy, 5*(2), 139–152. Retrieved from https://dergipark.org.tr/en/pub/ijesp/issue/90802/1659826

Turhan, S., Bozkurt, M., & Şahin, D. Ö. (2023). Development of mobile application based optical mark reader system using image processing techniques. *KMÜ Mühendislik ve Doğa Bilimleri Dergisi, 5*(2), 169–190. https://doi.org/10.55213/kmujens.1386520

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser L. & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 5998–6008. https://arxiv.org/abs/1706.03762

Wang, S., Wang, F., Zhu, Z., Wang, J., Tran, T., & Du, Z. (2024). Artificial intelligence in education: A systematic literature review. *Expert Systems with Applications, 252*, 124167. https://doi.org/10.1016/j.eswa.2024.124167

Warschauer, M., & Grimes, D. (2008). Automated writing assessment in the classroom. *Pedagogies: An International Journal, 3*(1), 22–36. https://doi.org/10.1080/15544800701771580

Wiser, M. J., Anderson, C. W., & Sadler, T. D. (2016). Comparing human and automated evaluation of open-ended student responses to questions of evolution. *Research in Science Education, 46*(6), 841–867. https://doi.org/10.1162/978-0-262-33936-0-ch025

Zhang, M., Press, O., Merrill, W., Liu, A., & Smith, N. A. (2023). How language model hallucinations can snowball. https://arxiv.org/abs/2305.13534

Appendixes

Appendix 1

Technical Implementation of User Interface

This appendix provides the technical details of the user interface implementation described in Section 2.1.2. The user interface is built using Swift and UIKit, with buttons created and managed as follows:

The user interface is implemented using Swift and UIKit. Buttons are created with the following code:

```
scanQuestionButton = createStyledButton(title: "Scan Question", icon:
UIImage(systemName: "doc.text"), action: #selector(scanQuestion))
scanAnswerButton = createStyledButton(title: "Scan Answer Key", icon:
UIImage(systemName: "key"), action: #selector(scanAnswer))
```

Button states are managed using a custom enumeration:

```
enum ButtonState {
    case notEnabled
    case enabled
    case clicked
}
```

The updateButtonStates() function dynamically adjusts button states:

```
func updateButtonStates() {
    setButtonState(button: scanQuestionButton, state: questionText.isEmpty ?
.enabled : .clicked)
    ...
}
```

-Button Creation: The createStyledButton function generates buttons with specific titles, icons (e.g., document and key symbols), and actions (e.g., triggering the scanQuestion or scanAnswer methods).

-State Management: The ButtonState enumeration defines three states (notEnabled, enabled, clicked) to control button availability and visual feedback during the workflow.

-Dynamic Updates: The updateButtonStates() function ensures buttons reflect the current step of the process, such as enabling the next button only after a successful scan.

Appendix 2

Technical Implementation of Workflow Management and Resetting State

This appendix provides the technical details of the reset functionality described in Section 2.1.3. The code below illustrates how the application resets data and interface components to their initial state, ensuring a seamless transition between evaluation cycles.

The startNewQuestion() function is used to reset all relevant data and user interface components at the end of an evaluation process or when starting a new one:

```
@objc func startNewQuestion() {
    questionText = ""
    answerText = ""
    ...
    updateButtonStates()
    resetCheckIndicators()
}
```

-Data Reset: The variables questionText and answerText are cleared to ensure no residual data from the previous evaluation remains.

-Interface Update: The updateButtonStates() function adjusts the state of the buttons (e.g., enabling/disabling them based on the workflow stage).

-Visual Feedback Reset: The resetCheckIndicators() function clears visual feedback indicators (e.g., green checkmarks or red crosses) to prepare the interface for the next cycle.

Appendix 3

Technical Implementation of Image Processing and OCR Process

This appendix provides the technical details of the image processing and OCR functionality described in Section 2.2. The following code and configurations illustrate how the application enhances captured images and converts handwritten text into digital text using Google Cloud Vision API.

Image Preprocessing

Before OCR, the application preprocesses captured images to improve text recognition accuracy. The preprocessImage function adjusts image properties such as brightness and noise levels:

```
func preprocessImage(_ image: UIImage) -> UIImage {
    guard let ciImage = CIImage(image: image) else { return image }
    let filter = CIFilter(name: "CIColorControls")!
    filter.setValue(ciImage, forKey: kCIInputImageKey)
    filter.setValue(1.1, forKey: kCIInputBrightnessKey) // Increase brightness
    filter.setValue(0.5, forKey: kCIInputContrastKey)   // Adjust contrast
    let outputImage = filter.outputImage!
    let context = CIContext()
    let cgImage = context.createCGImage(outputImage, from: outputImage.extent)!
    return UIImage(cgImage: cgImage)
}
```

-Brightness and Contrast Adjustment: The CIColorControls filter increases brightness and adjusts contrast to enhance text visibility.

-Noise Reduction: Implicit noise reduction is achieved through these adjustments, improving OCR accuracy.

OCR with Google Cloud Vision API

The OCR process converts handwritten text into digital text using Google Cloud Vision API. The API request is configured with a language hint to optimize for Turkish handwriting:

```
func performOCR(image: UIImage, completion: @escaping (String?) -> Void) {
```

```
    guard let base64String = image.jpegData(compressionQuality:
0.8)?.base64EncodedString() else {
        completion(nil)
        return
    }

    let parameters: [String: Any] = [
        "requests": [
            [
                "image": ["content": base64String],
                "features": [["type": "DOCUMENT_TEXT_DETECTION"]],
                "imageContext": ["languageHints": ["tr"]]
            ]
        ]
    ]

    // API call implementation (omitted for brevity)
    // On success, the extracted text is returned via the completion handler
}
```

-Image Encoding: The image is converted to a Base64 string to be sent to the API.

-Language Hint: The "languageHints": ["tr"] parameter ensures the API prioritizes Turkish language recognition, improving accuracy for Turkish handwriting.

-Feature Type: The "type": "DOCUMENT_TEXT_DETECTION" setting instructs the API to focus on extracting text from document-like images.

Appendix 4

Technical Implementation of Image Acquisition

This appendix provides the technical details of the image acquisition functionality described in Section 2.2.1. The following code illustrates how the application captures images using the device's camera and preprocesses them to improve text recognition accuracy.

Image Acquisition

The application uses the iOS VNDocumentCameraViewController to capture images of exam papers, providing a user-friendly scanning interface:

```
func startDocumentScan() {
    let scanner = VNDocumentCameraViewController()
    scanner.delegate = self
    present(scanner, animated: true, completion: nil)
}

func documentCameraViewController(_ controller: VNDocumentCameraViewController,
didFinishWith scan: VNDocumentCameraScan) {
    guard scan.pageCount > 0 else {
        dismiss(animated: true, completion: nil)
        return
    }
    let image = scan.imageOfPage(at: 0)
    dismiss(animated: true) {
        self.processImage(image)
    }
}
```

-Camera Interface: The VNDocumentCameraViewController provides an intuitive interface for users to scan documents, automatically detecting page edges and capturing high-quality images.

-Image Retrieval: The didFinishWith delegate method retrieves the scanned image for further processing.

Appendix 5

Technical Implementation of OCR Using Google Cloud Vision API

This appendix provides the technical details of the OCR implementation using Google Cloud Vision API, as described in Section 2.2.2. The following code illustrates how the application converts preprocessed images into digital text.

OCR Implementation

The performOCR function sends the preprocessed image to Google Cloud Vision API and retrieves the extracted text:

```
func performOCR(image: UIImage, completion: @escaping (String?) -> Void) {
    guard let base64String = image.jpegData(compressionQuality:
0.8)?.base64EncodedString() else {
        completion(nil)
        return
    }

    let parameters: [String: Any] = [
        "requests": [
            [
                "image": ["content": base64String],
                "features": [["type": "DOCUMENT_TEXT_DETECTION"]],
                "imageContext": ["languageHints": ["tr"]]
            ]
        ]
    ]

    // API call implementation (omitted for brevity)
    // On success, the extracted text is returned via the completion handler
}
```

-Image Encoding: The image is converted to a Base64 string using jpegData with a compression quality of 0.8, preparing it for the API request.

-Language Hint: The "languageHints": ["tr"] parameter optimizes the API for Turkish handwriting, improving recognition accuracy for Turkish characters and grammar.

-Feature Type: The "type": "DOCUMENT_TEXT_DETECTION" setting instructs the API to focus on extracting text from document-like images, such as exam papers.

-Completion Handler: The completion closure returns the extracted text (or nil if the process fails), which is then used for further analysis in the evaluation pipeline.

Appendix 6

Technical Implementation of Handwriting Recognition Technologies

This appendix provides the technical details of the handwriting recognition implementation described in Section 2.3.1. The following code and configurations illustrate how the application uses Google Cloud Vision API to recognize and digitize handwritten text, building on the OCR process.

Handwriting Recognition with Google Cloud Vision API

The performOCR function, previously introduced for OCR, is utilized here with specific configurations to optimize for handwriting recognition:

```
func performOCR(image: UIImage, completion: @escaping (String?) -> Void) {
    guard let base64String = image.jpegData(compressionQuality:
0.8)?.base64EncodedString() else {
        completion(nil)
        return
    }

    let parameters: [String: Any] = [
        "requests": [
            [
                "image": ["content": base64String],
                "features": [["type": "DOCUMENT_TEXT_DETECTION"]],
                "imageContext": ["languageHints": ["tr"]]
            ]
        ]
    ]

    // API call implementation (omitted for brevity)
    // On success, the extracted text is returned via the completion handler
}
```

-Handwriting Optimization: The "type": "DOCUMENT_TEXT_DETECTION" feature of Google Cloud Vision API includes built-in capabilities for handwriting recognition, adept at handling variability in handwriting styles such as cursive or faint text.

-Language Hint for Turkish: The "languageHints": ["tr"] parameter ensures the API is optimized for Turkish handwriting, improving recognition accuracy for Turkish-specific characters and grammar.

-Image Encoding: The image is converted to a Base64 string with a compression quality of 0.8, ensuring efficient transmission to the API while maintaining quality for text recognition.

-Completion Handler: The completion closure returns the digitized text (or nil if the process fails), which is then used for further analysis in the evaluation pipeline.

Appendix 7

Technical Implementation of Post-OCR Text Editing

This appendix provides the technical details of the post-OCR text editing implementation described in Section 2.3.2. The following code and configurations illustrate how the application uses OpenAI's GPT-4o model to refine OCR-extracted text and calculate the percentage of changes made during the editing process.

Text Editing with GPT-4o

The editOCRText function sends the OCR-extracted text to GPT-4o for refinement, correcting errors and standardizing the output:

```
func editOCRText(_ ocrText: String, completion: @escaping (String?) -> Void) {
    let parameters: [String: Any] = [
        "model": "gpt-4o",
        "prompt": "Correct and standardize the following OCR-extracted text for
clarity and context: \(ocrText)",
        "max_tokens": 500,
        "temperature": 0.3
    ]

    // API call implementation (omitted for brevity)
    // On success, the refined text is returned via the completion handler
}
```

-Model Selection: The "model": "gpt-4o" parameter specifies the use of OpenAI's GPT-4o, which is adept at understanding context and correcting text errors.

-Prompt Design: The "prompt" instructs GPT-4o to correct and standardize the OCR text, ensuring the output aligns with the expected context (e.g., a student response).

-Control Parameters: The "max_tokens": 500 limits the response length, and "temperature": 0.3 ensures a more deterministic and precise correction process.

-Completion Handler: The completion closure returns the refined text (or nil if the process fails), which is then used for evaluation.

Change Percentage Calculation

The calculateChangePercentage function computes the percentage of changes made to the original OCR text, which is reported in the final PDF for transparency:

```
func calculateChangePercentage(original: String, processed: String) -> Int {
    let distance = levenshteinDistance(a: original, b: processed)
    return Int(round((Double(distance) / Double(original.count)) * 100))
}
```

-Levenshtein Distance: The levenshteinDistance function calculates the edit distance between the original OCR text (original) and the refined text (processed), measuring the number of changes made.

-Percentage Calculation: The change percentage is computed as the ratio of edits to the original text length, rounded to the nearest integer, and reported in the final PDF (e.g., "Text edited by AI by X%").

Appendix 8

Technical Implementation of Scanning and Defining Evaluation Criteria

This appendix provides the technical details of the scanning and defining evaluation criteria implementation described in Section 2.4.1. The following code illustrates how the application captures the evaluation criteria document, processes it using OCR, and prepares the criteria for automated scoring.

Scanning the Evaluation Criteria Document

The application uses the iOS VNDocumentCameraViewController to capture the evaluation criteria document, similar to the image acquisition process for other components:

```
func scanEvaluationCriteria() {
    let scanner = VNDocumentCameraViewController()
    scanner.delegate = self
    present(scanner, animated: true, completion: nil)
}

func documentCameraViewController(_ controller: VNDocumentCameraViewController,
didFinishWith scan: VNDocumentCameraScan) {
    guard scan.pageCount > 0 else {
        dismiss(animated: true, completion: nil)
        return
    }
    let image = scan.imageOfPage(at: 0)
    dismiss(animated: true) {
        self.processCriteriaImage(image)
    }
}
```

-Camera Interface: The VNDocumentCameraViewController provides an intuitive interface for users to scan the evaluation criteria document, automatically detecting page edges and capturing a high-quality image.

-Image Retrieval: The didFinishWith delegate method retrieves the scanned image, which is then passed to the processCriteriaImage function for further processing.

Extracting Evaluation Criteria Using OCR

The scanned image is processed using the previously defined performOCR function to extract the evaluation criteria as digital text:

```
func processCriteriaImage(_ image: UIImage) {
    let preprocessedImage = preprocessImage(image) // Apply preprocessing
    performOCR(image: preprocessedImage) { extractedText in
        guard let text = extractedText else {
            // Handle error (e.g., display red cross visual feedback)
            return
        }
        self.evaluationCriteria = text
        // Further processing to parse criteria into a structured format (if
needed)
    }
}
```

-Preprocessing: The preprocessImage function (detailed in Appendix 4) enhances the image quality by adjusting brightness and contrast, ensuring better OCR accuracy.

-OCR Processing: The performOCR function (detailed in Appendix 5) uses Google Cloud Vision API to extract the text from the scanned image, with a language hint set to “Turkish” for optimal recognition.

-Criteria Storage: The extracted text is stored in the evaluationCriteria variable, ready for use in the AI-based evaluation process. Additional parsing may be applied to structure the criteria (e.g., into key phrases and associated points).

Appendix 9

Technical Implementation of AI-Based Analysis of Student Responses

This appendix provides the technical details of the AI-based analysis implementation described in Section 2.4.2. The following code and configurations illustrate how the application uses OpenAI’s GPT-4o model to evaluate student responses and assign scores based on defined evaluation criteria.

AI-Based Evaluation with GPT-4o

The evaluateStudentResponse function sends the digitized student response and evaluation criteria to GPT-4o for scoring:

```
func evaluateStudentResponse(_ response: String, criteria: String, completion:
@escaping (Int, Double) -> Void) {
    let parameters: [String: Any] = [
        "model": "gpt-4o",
        "prompt": """
            Evaluate the following student response based on the given criteria.
Assign a score out of 10 and provide a confidence score (0 to 1) for your
evaluation.
            Criteria: \(criteria)
            Response: \(response)
            Return the score and confidence in the format: 'Score: X, Confidence:
Y'.
        """,
        "max_tokens": 150,
        "temperature": 0.5
    ]

    // API call implementation (omitted for brevity)
    // Example response parsing:
    // let result = "Score: 8, Confidence: 0.95"
    // Parse result to extract score (8) and confidence (0.95)
    completion(8, 0.95) // Placeholder values for illustration
}
```

-Model Selection: The "model": "gpt-4o" parameter specifies the use of OpenAI’s GPT-4o, which is capable of understanding and evaluating complex text based on given criteria.

-Prompt Design: The "prompt" instructs GPT-4o to evaluate the student response against the criteria, assign a score out of 10, and provide a confidence score between 0 and 1.

-Control Parameters: The "max_tokens": 150 limits the response length, and "temperature": 0.5 balances creativity and precision in the evaluation process.

-Completion Handler: The completion closure returns the assigned score (e.g., 8) and confidence score (e.g., 0.95), which are used in the final report.

Example Evaluation

For a history question with the criteria "Mention the Industrial Revolution and its economic impact (10 points total: 5 for each element)," and a student response "The Industrial Revolution led to increased factory production and economic growth," GPT-4o might evaluate as follows:

-Identifies "Industrial Revolution" (5 points).

-Identifies "economic growth" as the economic impact (5 points).

-Returns: Score: 10, Confidence: 0.95.

Appendix 10

Technical Implementation of Scoring, Feedback, and PDF Report Generation

This appendix provides the technical details of the scoring and result generation implementation described in Section 2.4.3. The following code illustrates how the application compiles scores, evaluation metrics, and feedback into a PDF report.

Compiling Scores and Metrics

The generateEvaluationResult function aggregates the scores, confidence levels, and text editing percentage for inclusion in the final report:

```
func generateEvaluationResult(response: String, score: Int, confidence: Double,
editPercentage: Int) -> String {
    let resultSummary = """
        Student Response: \(response)
        Score: \(score)/10
        Confidence: \(confidence * 100)%
        Text Edited by AI: \(editPercentage)%
    """
    // Additional feedback can be generated by GPT-4o if needed (omitted for
brevity)
    return resultSummary
}
```

-Result Summary: The function formats the evaluation results into a string, including the student's response, assigned score (e.g., 8/10), confidence level (e.g., 95%), and the percentage of text edits made during post-OCR processing (e.g., 15%).

-Extensibility: Additional AI-generated feedback could be appended to the summary, such as suggestions for improvement, if required.

PDF Report Generation

The createPDFReport function generates a PDF document containing the evaluation results:

```
func createPDFReport(resultSummary: String) -> URL? {
    let pdfRenderer = UIGraphicsPDFRenderer(bounds: CGRect(x: 0, y: 0, width: 595,
height: 842)) // A4 size
    let data = pdfRenderer.pdfData { context in
        context.beginPage()
        let attributes: [NSAttributedString.Key: Any] = [
            .font: UIFont.systemFont(ofSize: 12),
            .foregroundColor: UIColor.black
        ]
        let text = NSAttributedString(string: resultSummary, attributes: attributes)
        text.draw(in: CGRect(x: 20, y: 20, width: 555, height: 802))
    }

    let                                    fileURL                                    =
FileManager.default.temporaryDirectory.appendingPathComponent("EvaluationReport.pdf
")
    try? data.write(to: fileURL)
    return fileURL
}
```

-PDF Rendering: The UIGraphicsPDFRenderer creates an A4-sized PDF document, rendering the resultSummarytext with specified formatting (e.g., font size, color).

-File Storage: The PDF data is saved to a temporary file (EvaluationReport.pdf), and the file URL is returned for sharing or storage.

-Formatting: The text is drawn within a defined rectangle to ensure proper layout on the page.

Compiling the Report Data

The generateEvaluationResult function aggregates the evaluation data into a formatted string for inclusion in the PDF report:

```
func generateEvaluationResult(response: String, score: Int, confidence: Double,
editPercentage: Int) -> String {
    let resultSummary = """
        Student Response: \(response)
        Score: \(score)/10
        Confidence: \(confidence * 100)%
        Text Edited by AI: \(editPercentage)%
    """
    // Additional feedback can be generated by GPT-4o if needed (omitted for
brevity)
    return resultSummary
}
```

-Result Summary: The function formats the evaluation results into a string, including the student's response, assigned score (e.g., 8/10), confidence level (e.g., 95%), and the percentage of text edits made during post-OCR processing (e.g., 15%).

-Extensibility: Additional AI-generated feedback, such as suggestions for improvement, could be appended to the summary if required.

Feedback Generation with GPT-4o

The generateFeedback function uses GPT-4o to create tailored feedback based on the student's response and evaluation criteria:

```
func generateFeedback(response: String, criteria: String, score: Int, completion:
@escaping (String, String) -> Void) {
    let parameters: [String: Any] = [
        "model": "gpt-4o",
        "prompt": """
            Provide feedback for an educator and a student based on the following:
            Evaluation Criteria: \(criteria)
            Student Response: \(response)
            Score: \(score)/10
            For the educator: Highlight areas of strength and weakness in the
response, and suggest teaching focus.
            For the student: Provide constructive feedback to improve their answer.
            Return the feedback in the format: 'Educator Feedback: ... | Student
Feedback: ...'
        """,
        "max_tokens": 200,
        "temperature": 0.7
    ]

    // API call implementation (omitted for brevity)
    // Example response parsing:
    // let result = "Educator Feedback: The student correctly identified the
Industrial Revolution but missed its economic impact. Consider focusing on this in
future lessons. | Student Feedback: You did well mentioning the Industrial
Revolution, but try to include its economic effects for a complete answer."
    // Parse result to extract educatorFeedback and studentFeedback
    completion("The student correctly identified the Industrial Revolution but
missed its economic impact. Consider focusing on this in future lessons.", "You did
well mentioning the Industrial Revolution, but try to include its economic effects
for a complete answer.") // Placeholder values for illustration
}
```

-Model Selection: The "model": "gpt-4o" parameter specifies the use of OpenAI's GPT-4o, which is capable of generating context-aware feedback.

-Prompt Design: The "prompt" instructs GPT-4o to generate feedback for both the educator and student, focusing on strengths, weaknesses, and actionable suggestions based on the response, criteria, and score.

-Control Parameters: The "max_tokens": 200 limits the response length, and "temperature": 0.7 allows for a balanced level of creativity in feedback generation.

-Completion Handler: The completion closure returns two feedback strings: one for the educator and one for the student, which are then included in the PDF report.

Integration into the PDF Report

The feedback is appended to the result summary before PDF generation, as part of the generateEvaluationResult function:

```
func generateEvaluationResult(response: String, score: Int, confidence: Double,
editPercentage: Int, educatorFeedback: String, studentFeedback: String) -> String {
    let resultSummary = """
```

```
        Student Response: \(response)
        Score: \(score)/10
        Confidence: \(confidence * 100)%
        Text Edited by AI: \(editPercentage)%
        Educator Feedback: \(educatorFeedback)
        Student Feedback: \(studentFeedback)
    """
    return resultSummary
}
```

-Feedback Inclusion: The educatorFeedback and studentFeedback strings are added to the resultSummary, which is then used by the createPDFReport function to generate the final PDF.

-Formatting: The feedback is presented clearly in the PDF report, ensuring both educators and students can easily access and understand the insights.