JobChecker: Automating Assessment of Handwritten Student Work

Alua Meiirbekova
Dias Murzagaliyev
Taikozha Turdyakyn
Timur Issamadiyev

Final Report

April 2025

Adviser: Professor Hashim Ali

Contents

Abstract

In educational institutions, the grading of handwritten student assignments still seems to be a time-consuming and prone-to-error process. Designed to improve grading efficiency, consistency, and accessibility for both teachers and students, JobChecker is an AI-powered tool meant to automate the evaluation of written student work. The system uses OpenAI's GPT-4 model to assess and generate comments on handwritten input and incorporates Optical Character Recognition (OCR) technology to transform it into digital form. A key aspect that distinguishes JobChecker is its holistic integration of advanced OCR for handwritten text with the sophisticated contextual understanding of the GPT-4 model, enabling accurate assessment directly from handwritten submissions – a capability not commonly found in existing automated grading solutions.

JobChecker supports multiple subjects and sections, provides multilingual feedback, and is accessible via web and mobile applications. The frontend and mobile interfaces use ReactJS and SwiftUI respectively for a responsive user experience; its backend architecture is driven by Java Spring Boot and PostgreSQL. The project includes a comparative evaluation of several state-of-the-art language models, highlighting GPT-4's better contextual understanding and API accessibility.

Along with an analysis of technical difficulties faced during implementation, this report covers the design, development, and performance evaluation of JobChecker. By reducing the manual workload on teachers and providing instant feedback to students, JobChecker shows how artificial intelligence might modify educational processes.

1 Introduction

1.1 Motivation

The manual evaluation of handwritten student assignments continues to be a significant difficulty in modern educational settings. In remote learning environments or large classrooms, the delivery of meaningful feedback is frequently delayed due to the time-intensive nature of grading. This issue is particularly acute. Such delays adversely impact student learning, as prompt and constructive feedback is essential for academic advancement. Furthermore, the grading process is prone to human irregularities, such as scoring fluctuation and errors due to exhaustion, which compromise the dependability of evaluations.

Given the swift progress in artificial intelligence (AI) and educational technology, the automation of assignment assessment offers a promising alternative. The JobChecker project aims to rectify the inefficiencies of conventional grading by integrating Optical Character Recognition (OCR) methods with large language models (LLMs) to accurately

assess handwritten student submissions and provide timely, high-quality comments.

1.2 Project Objectives

JobChecker achieves its main goal by The main aim of JobChecker is to automate the assessment of handwritten tasks with advanced AI techniques. The system is specifically engineered to precisely extract handwritten text utilizing OCR capabilities and to produce pedagogically effective feedback through interaction with OpenAI's GPT-4 language model. JobChecker aims to provide immediate feedback to students via user-friendly web and mobile applications, while also supporting educators by offering editable AI-generated evaluations. The system is intended to enhance student engagement through timely and constructive commentary and to improve instructional efficiency by reducing manual grading burdens.

1.3 Scope

The existing implementation of JobChecker is designed for high school tasks necessitating brief to moderate written responses. This encompasses open-ended inquiries typically present in areas such as language arts, history, and fundamental STEM fields. The system now emphasizes English-language input and output but is designed to accommodate multilingual capabilities in future versions.'

JobChecker features a comprehensive full-stack design, comprising a scalable backend, adaptive frontend interfaces, and cloud-based media storage. Essential elements comprise a teacher dashboard for assignment oversight and AI feedback moderation, a student site for submission uploads and assessment receipt, and a cohesive AI pipeline that facilitates OCR-based text extraction and GPT-4-driven analysis.

1.4 Target Audience

The JobChecker system is built to support a wide range of people in education. For teachers, it helps speed up grading while ensuring fairness and clarity in how students are assessed. Students get structured, real-time feedback that helps them understand their progress and improve. Schools and universities can use JobChecker as a scalable tool to manage assessments—whether in traditional classrooms or online settings.

1.5 Significance

JobChecker shows how modern AI can make a real difference in education. It tackles a common challenge—academic assessment—and turned into a meaningful learning journey for our team. At the beginning of the academic year, the majority of us possessed minimal

experience in full-stack development or artificial intelligence. During the development of JobChecker, we acquired practical expertise in backend development, user interface design, prompt formulation, and the deployment of cloud-based systems.

This project exemplifies global initiatives to enhance education through digitalization and personalization. Utilizing sophisticated feedback mechanisms such as JobChecker, we seek to enhance educational outcomes, alleviate the workload of educators, and increase the accessibility of quality education for all.

2 Literature Review and Background

2.1 Automated Grading Systems

The practice of automated grading has been in existence for some time already. The initial grading methods depended on both multiple-choice scanning procedures and keyword search algorithms. Numerous machine learning breakthroughs have developed systems which evaluate freely written texts so Automated Essay Scoring (AES) systems operate in standardized assessments including GRE and TOEFL [1].

The majority of automated grading applications use typed inputs primarily. The research community has not thoroughly investigated the process of grading handwritten student responses. Handwritten entry brings increased complexity to assessment processes because handwriting variations and scan quality variations require strong OCR and preprocessing algorithms to function effectively according to Burrows [2].

2.2 Optical Character Recognition

The conversion from text document images through scanning into a format computers can process occurs through OCR technologies. Tesseract OCR maintains its position as one of the most accurate free Open-Source OCR systems that Hewlett-Packard developed and Google currently supports. With capability to recognize more than 100 languages the system also supports some forms of handwriting. The performance of OCR becomes superior due to preprocessing operations which include noise reduction and skew correction and binarization techniques particularly for handling handwritten text according to Smith (2007) [3].

The accuracy rates of Tesseract OCR range between 85–98% for structured handwritten text based on its quality level according to Sporici et al. (2020). The results produced by this process are sufficient to allow NLP models to detect context patterns needed for meaningful feedback generation.

2.3 Natural Language Processing with GPT-4

OpenAI developed the transformer-based large language model which they call GPT-4. The model demonstrates strong capability in context interpretation and text production together with content assessment. The assessment system uses GPT-4 to evaluate student submissions through an evaluation of their grammar along with an assessment of coherence and logical structure and their relationship to the given assignment.

Because it flexibly processes diverse inputs GPT-4 proves suitable for assessing content converted from handwritten materials through Optical Character Recognition. The model achieves multitype question analysis through zero-shot and few-shot learning mechanisms which require few adjustments in its setup [4]. Multiple classroom trials employing LLMs present positive evidence regarding both student response assessment quality and teaching engagement according to Jacobs et al. (2024) [5].

2.4 Challenges in Handwritten Assessment

Modern technological improvements in AI and Optical Character Recognition (OCR) do not solve all the persistent problems that exist in handwritten assignment evaluation processes. OCR engine accuracy stands as the main problem when dealing with multiple handwriting variations.

The process of text recognition becomes compromised when OCR engines encounter irregular shapes of letters together with inconsistent spacing and slant within the handwriting. Subsequent natural language processing evaluations end up receiving false information as a result of these OCR recognition errors [6]. Contextual ambiguity which naturally exists within handwritten responses creates a second problem. Automated systems encounter difficulties understanding text meaning because extracted information usually does not have proper punctuation or standardized formatting. The subtle or complex nature of certain answers becomes especially difficult to assess through this method.

The use of AI evaluation systems raises important challenges regarding their absence of bias along with their inability to maintain fairness. GPT-4 demonstrates impressive capability but is likely to enqueue or strengthen current biases present in its training data. Equitable evaluation assessments face increased risks when serving students who have different linguistic and cultural backgrounds. To achieve fairness in evaluations there needs to be an active auditing process accompanying bias reduction methods as well as a human monitoring system for critical decision points.

The protection of student data remains a major worry in educational institutions. The protection of student-sensitive information requires multi-layered encryption security measures including data protection standards which must follow General Data Protection

Regulation (GDPR). Every stage of system development in educational settings must follow ethical principles to guarantee trust and responsibility in environments where students put their trust. The solution of these complex issues requires systematic solutions which combine preprocessing analysis together with adaptive AI systems and human-controlled audits supported by strict ethical protection.

2.5 Summary

An automated solution for handwritten assignment evaluation becomes possible through the combination of OCR and NLP technologies. JobChecker implements established tools Tesseract and GPT-4 to create a practical united solution which caters to educational institutions.

3 System Architecture and Implementation

3.1 Overview

The architecture of JobChecker is built to be both modular and scalable, aimed at supporting the automated evaluation of handwritten student work. The system brings together elements from computer vision, natural language processing (NLP), cloud infrastructure, and full-stack development to deliver a robust, accessible platform across both web and mobile environments. It adopts a multi-tier structure consisting of a frontend interface (web and mobile), a backend server that manages orchestration and logic execution, an AI processing pipeline, and a cloud-based system for data storage and retrieval. This layered design promotes a clear separation of responsibilities, making the system easier to maintain and allowing individual components to scale as needed.

The data flow begins when a student uploads an image of their handwritten assignment using the mobile app. This image is securely stored in the cloud and placed in a processing queue. The backend retrieves the image, runs it through an OCR module to extract the text, and then sends that text to GPT-4 for evaluation. Once the AI generates feedback, the results are returned to the frontend, where they are made available to both students and teachers. Throughout this process, the system ensures data security, persistence, and efficiency by using asynchronous processing and optimized communication protocols.

3.2 Frontend Design

JobChecker has two different user interfaces for different users: the web interface for teachers and admins and the mobile app for students and teachers. The web interface is built using ReactJS, an efficient component-based JavaScript framework for building

dynamic and interactive one-page applications [7]. The reactive nature of React's virtual DOM makes it well-adapted for dynamic update and real-time displaying of feedback and moderation. Teachers are able to log in, construct assignments, track student submissions, read AI-produced feedback and grade.

The application is developed on top of SwiftUI, Apple's new declarative framework for creating user-friendly iOS applications. The interface is created to provide easy teacher-student interaction, including submission through the camera or file upload, viewing of the feedback, creating assignments, and grading the submissions. The two interfaces are based on RESTful APIs to communicate with the backend easily.

The UI/UX design on both platforms was first designed using Adobe Photoshop for the draft concepts and Figma for interactive prototype and design validation. Through these, rapid iteration, design consistency, and communication within the team were achieved. The features of feedback notification, image upload guidelines, and progress indicators are included to further enhance usability.

3.3 Backend and API Layer

The backend of JobChecker is powered by Java Spring Boot, a technology known for its ease of configuration, strong dependency injection, and scalability. The backend serves as the primary hub, coordinating API endpoints, managing business logic, handling user authentication, and facilitating communication with the OCR and AI modules.

To enable secure interactions between the frontend and backend, JobChecker uses RESTful API endpoints. Role-based access control (RBAC) is used to ensure that only authorized users, such as teachers or students, have access to specific features depending on their rights. Token-based authentication using OAuth 2.0 improves session security.

When a student uploads an image, it is initially saved in a Firebase Cloud Storage bucket. The backend then initiates downstream processing tasks via asynchronous job queues. This design enables job execution in parallel, reducing latency and improving system responsiveness, particularly during peak demand.

3.4 AI Processing Pipeline

The AI pipeline is made up of two sequential components: an OCR module that converts handwritten images into machine-readable text and an NLP module that analyzes and evaluates the retrieved data.

The system employs Tesseract OCR with preprocessing techniques such as grayscale conversion, noise filtering, binarization, and skew correction. These techniques are crucial for increasing text recognition accuracy, particularly when working with unstructured or

badly scanned material. Tesseract's capability for bespoke training data and open-source extensibility make it appropriate for the variety of handwriting styles found in high school-level work [8].

NLP Layer: GPT-4 evaluates the extracted text and provides comments based on coherence, grammatical accuracy, and relevance to the query. GPT-4 uses prompt templates to drive its evaluation logic, simulating rubric-based grading while keeping flexibility for various question formats.

To ensure system efficiency, GPT-4 queries are rate-limited and processed asynchronously [9]. The output is prepared and directed back to the user-facing frontend, where teachers can update or approve the feedback before it is ready for student review.

3.5 Data Management and Security

Data management in JobChecker is powered by PostgreSQL, a robust open-source relational database system known for its reliability, ACID compliance, and ability to handle complex queries. The database schema includes tables for user accounts, submission metadata, OCR results, AI-generated feedback, and moderation logs. To ensure efficient data retrieval, indexing and normalization techniques are applied, optimizing query performance.

For managing large media files, such as assignment image uploads, the system uses Firebase Cloud Storage. By separating the storage of media files from structured data, JobChecker improves both read/write efficiency and fault tolerance, ensuring smooth and reliable operation even when handling large volumes of content.

Security is embedded at multiple layers of the system. All communication is encrypted using SSL/TLS. Sensitive data, such as user credentials and assignment content, is encrypted at rest using AES-256. Authentication tokens are securely stored and refreshed using industry-standard practices. Audit logging mechanisms track access and modifications to sensitive content, supporting both security and transparency in evaluation.

The system complies with data privacy norms akin to the GDPR and FERPA frameworks, including user consent mechanisms and data deletion requests.

3.6 Performance Optimization

Performance is a central concern given the compute-intensive nature of OCR and LLM processing. Redis, an in-memory key-value store, is used to cache frequently accessed data such as user session states and recently generated feedback. This reduces redundant database queries and accelerates system responsiveness.

To handle spikes in usage, asynchronous processing queues (e.g., via Spring Boot's @Async annotation or a message broker like RabbitMQ) allow OCR and GPT-4 jobs to be processed in the background. This architecture decouples time-consuming tasks from user interactions, enhancing overall responsiveness.

Additionally, the system uses lazy loading, pagination, and image compression techniques to reduce frontend load times and bandwidth usage. Benchmarking tests under simulated load conditions help fine-tune thread pools, connection pools, and memory usage to support reliable scaling.

4 Implementation

4.1 Development Tools

JobChecker was developed utilizing an extensive range of tools to guarantee performance, scalability, and maintainability. The frontend was developed using ReactJS for dynamic and responsive web interfaces, and SwiftUI was employed for the iOS mobile version to ensure a flawless native experience.

The backend was developed using Java 22 and the Spring Boot framework, chosen for its modular structure, security protocols, and enterprise-level scalability. The architecture is layered, comprising controller, service, and repository layers, which improves testability, maintainability, and code organization. The backend manages business logic, data storage, and external integrations. Spring Security utilizing JWT guarantees secure authentication and role-based access for students, educators, and administrators. To ensure modularity and scalability, we split the functionality across multiple services:

- **AuthService**: Manages registration, login, JWT issuance.
- **AssignmentService**: Handles assignment creation and retrieval.
- **SubmissionService**: Processes and stores student submissions.
- **FeedbackService**: Integrates with OCR (Tesseract) and AI (OpenAI API) for feedback generation.
- **SectionService**: Organizes courses, sections, and user associations.

The database layer uses PostgreSQL, interfaced via Spring Data JPA, ensuring relational integrity and efficient query abstraction. Data Transfer Objects (DTOs) create an abstraction layer between the API and the database schema, enabling future extension.

Development and collaboration included tools such as Git, GitHub, Visual Studio Code, IntelliJ IDEA, and Postman, allowing version control, testing, and API integration.

4.2 Integration Process

Integrating different aspects of JobChecker was a core aspect of the development process. The OCR engine (Tesseract) was connected to the backend to process scanned handwritten assignments. The extracted text underwent preprocessing — binarization, reduction of noise, and correction for skew — to enhance the accuracy of recognition.

The cleaned text is forwarded to the GPT-4 model through the OpenAI API for analysis of grammar, coherence, and relevance to the assignment question. The resulting feedback was returned to the frontend for instant viewing by the student.

RESTful APIs facilitate communication between frontend and backend, enabling smooth data exchange. Web and mobile clients interact with the backend to submit assignments, receive feedback, and execute course-related actions.

To ensure smooth integration, continuous testing, global exception handling, and custom response entities were implemented. Additionally, asynchronous processing — using Java 22 Virtual Threads — was applied to AI and OCR operations to prevent blocking and improve performance.

4.3 Algorithms and Workflow

JobChecker operates through a series of interconnected algorithms that work in harmony to evaluate student assignments and generate feedback. The OCR engine preprocesses the scanned images before extracting the text. Preprocessing steps, such as binarization, noise reduction, and skew correction, enhance the accuracy of text extraction, even in cases of poor handwriting or low-quality scans.

Once the text is extracted, the GPT-4 model is employed to evaluate the content. The model analyzes the text for grammar, clarity, structure, and its relevance to the assignment prompt. It adapts dynamically to the content, providing personalized feedback based on the specific nuances of each response. This approach ensures a deeper analysis compared to traditional rule-based grading systems.

The feedback generated by GPT-4 is then returned to the user interface, where students can review it immediately. The frontend communicates with the backend via APIs to fetch and display the feedback.

4.4 Deployment Details

JobChecker's deployment architecture emphasizes scalability, security, and reliability. It uses cloud infrastructure to dynamically allocate resources based on user demand, ensuring high availability and performance even during peak times. Key deployment features

include:

- **Secure data handling**: All sensitive data is encrypted at rest and in transit.
- **JWT-based authentication**: Ensures role-specific access and session security.
- **CI/CD pipelines**: Enable rapid, low-disruption updates through automated testing and deployment.
- **Monitoring & logging**: Real-time tracking of system performance and issues using automated tools.

The application adopts a modular monolithic architecture with microservice-like service separation internally, allowing for scalability and maintainability without the operational complexity of full microservices.

4.5 Web Front End Overview

The front end of the platform is designed around a clean, role-specific user experience, divided into three core sections: the Welcome and Authentication module, the Administration Panel, and the Teacher Panel. This structure leans on a responsive, modular architecture with a clear separation of concerns, so users only interact with tools relevant to their specific roles.

Everything begins on the Welcome page, which features the platform's branding and a brief explanation of its academic mission. From here, users move into the Authentication module, where a unified form-based layout handles both login and signup. Returning users enter their email and password, while new users are guided through entering a name, email, and password—all within the same consistent interface.

Once logged in, administrators access the Administration Panel, a centralized interface for managing the school's people and courses. The panel displays tables listing all registered students, teachers, and subjects. Clicking on a subject reveals its sections, and digging further into a section shows options to assign a teacher or enroll students. Admins can also create new subjects or add sections under existing ones.

Teachers, on the other hand, use a streamlined Teacher Panel that lists only their assigned courses and sections. Viewing a section opens an assignment-focused workspace, where existing assignments are listed and new ones can be created by filling in a short form with title, description, due date, and optional files. Selecting an assignment displays its details along with a list of student submissions, each with an AI-generated grade and feedback. Teachers can review or download these submissions for deeper evaluation.

The entire interface is underpinned by a shared component library that ensures a uniform look and feel—consistent colors, spacing, and typography throughout. The separation of user paths for authentication, admin duties, and teaching not only keeps things tidy and

maintainable, but also ensures each user sees exactly what they need. This modular design sets the stage for scalable future features like student dashboards, real-time alerts, and visual analytics.

The project integrates several foundational technical features, each shaping distinct core aspects of the web application's architecture:

- **Interactivity**: The system is built around an interactive design paradigm that goes beyond basic static content delivery. Users can fluidly move between contextually related pages, initiate client-side actions like clicking buttons, and receive instant visual feedback — all made possible through dynamic manipulation of the DOM and asynchronous data retrieval, eliminating the need for full-page reloads.
- **CRUD Operations and Front-End Dynamism**: A full set of Create, Read, Update, and Delete operations has been embedded within the client-server communication model. These functions are central to how the application updates and displays content in real time, ensuring the interface always mirrors the actual state of the data model and enabling end-to-end dynamic user experiences.
- **Authentication**: The authentication layer validates user identities through secure credential exchange protocols. Upon successful login, the system issues an access token, which acts as a key for validating all subsequent requests. This mechanism blocks unauthorized access and protects confidential user information from exposure.
- **Authorization**: Authorization controls are enforced through a structured role-and-permissions model, tightly linked to the authenticated user profile. Each attempt to access a protected resource is checked against these assigned privileges, ensuring that actions like editing data or accessing admin tools stay within the bounds of security policies. When a user lacks the necessary permissions, a dedicated error page is displayed. [10]

4.6 Mobile (iOS) Application Overview

The developed mobile iOS application comprises several key components, including an onboarding page, a tab bar with the assignments and courses pages for students, a courses page for teachers, and dedicated pages for creating, submitting, and grading the assignments. It also includes login/registration pages to manage user authentication. The iOS application was implemented using the native SwiftUI framework, chosen for its declarative way of building the UI and providing the native UX capabilities.

We adopted the MVVM (Model-View-ViewModel) [11] architectural pattern to manage the application's state and UI interactions efficiently. Given the complexity of the project, MVVM was chosen as it aligns well with SwiftUI's declarative nature, providing a clean

separation between the UI and the business logic.

To manage the user session and other global states, we opted for ObservableObject for state management, leveraging EnvironmentObject and UserDefaults for global access across the views. This approach is lightweight, scalable, and integrates seamlessly with SwiftUI. By utilizing ObservableObject and @State properties, the app's UI reacts intuitively to state changes without unnecessary complexity.

For better scalability and maintainability, we implemented a modular architecture. We designed the app's network layer using NetworkService as the core service for handling API requests. Additionally, we separated the functionality into distinct services, including AuthService for managing user authentication, AssignmentService for handling assignments, SubmissionService for managing submission data, SectionService for dealing with course sections, and FeedbackService for managing feedback. This modular approach ensures a clean separation of concerns, making the codebase easier to maintain and extend.

Stack: Xcode, Swift, SwiftUI, Swift Structured Concurrency, MVVM, UserDefaults, URLSession

5 Results and Discussion

5.1 Evaluation Metrics

To assess the performance of JobChecker, we employed a variety of metrics aimed at measuring the effectiveness of its core components: OCR accuracy, the quality of feedback generated by GPT-4, system responsiveness, and user satisfaction. These metrics were selected due to their direct relevance to the project's primary objectives: ensuring accurate text extraction, delivering meaningful feedback, and providing a user-friendly experience.

5.2 OCR Performance

The OCR component, which is essential for transforming handwritten text into machine-readable format, was evaluated based on its accuracy in extracting content from scanned student assignments. Tesseract OCR, integrated into JobChecker, demonstrated significant accuracy in recognizing various handwriting styles, with an average extraction accuracy ranging from 85% to 98%. However, variations in handwriting quality and scan resolution did impact the system's performance. For instance, assignments with clearer handwriting and higher-quality scans yielded higher OCR accuracy, while noisy or poorly scanned documents resulted in more frequent recognition errors.

Despite occasional misinterpretations, the OCR system demonstrated sufficient reliability for the purpose of subsequent analysis by GPT-4. It is noteworthy that OCR errors, although occasionally present, had minimal impact on the quality of the final feedback provided to students due to the contextual understanding capabilities of the GPT-4 model.

5.3 Feedback Quality

Feedback generated by GPT-4 was evaluated based on its relevance, coherence, and educational value. GPT-4 was tasked with providing constructive feedback on student assignments, including corrections for grammatical errors, improvements in clarity, and suggestions for better argumentation or structure. The model proved adept at identifying areas for improvement that are often overlooked in manual grading, such as vague phrasing or incomplete reasoning.

Teachers and students who tested the system reported that the feedback was specific, actionable, and context-aware. GPT-4's dynamic evaluation method, which adapts to the content and nuances of each student's response, proved to be an advantage over traditional, rubric-based systems. The ability of GPT-4 to generate personalized comments, rather than generic feedback, contributed significantly to the learning experience, as students received tailored suggestions aligned with the content they submitted.

5.4 System Usability

JobChecker's usability was assessed through user testing with both students and teachers. Feedback collected from users indicated that the web and mobile interfaces were intuitive and easy to navigate. Students found the process of uploading assignments and receiving feedback to be straightforward, while teachers appreciated the ability to review and moderate the feedback generated by the system.

The user interface, built with ReactJS for the web portal and SwiftUI for the mobile application, received positive reviews for its responsiveness and ease of interaction. Any identified areas for improvement—such as additional customization options for feedback or more granular control for teachers over evaluation—were noted for future development.

5.5 Performance and Scalability

Performance testing was conducted to assess how JobChecker manages large volumes of data and simultaneous users. The system demonstrated robust performance under typical usage scenarios, with minimal delays in text extraction and feedback generation.

Response times remained consistent even as the number of assignments processed increased, indicating that the system is capable of handling moderate to heavy workloads.

JobChecker's architecture was designed to be scalable, supporting future expansion to accommodate larger numbers of users and multilingual content. While the system performed well under current loads, further optimization could improve processing times and ensure scalability for use in larger educational institutions or at the national level.

5.6 Challenges Encountered

Several challenges emerged during the development and deployment of JobChecker. The most notable challenges were related to the handling of diverse handwriting styles, which occasionally resulted in OCR inaccuracies. While preprocessing techniques, such as noise reduction and skew correction, mitigated these issues, the system occasionally struggled with heavily stylized or poorly legible handwriting.

Another challenge was ensuring that GPT-4's feedback remained consistent and relevant, particularly when dealing with ambiguous or incomplete answers. Although GPT-4 was able to generate useful feedback in most cases, the occasional lack of context in the input text led to less accurate or overly generic feedback. Ongoing refinement of the system and additional training for the AI model could address these issues in future iterations.

5.7 Comparison to Existing Solutions

Compared to traditional manual grading and existing automated grading solutions, JobChecker offers a more scalable and efficient approach. Traditional manual grading is time-consuming and prone to inconsistencies, while most existing automated systems rely on rigid rules or multiple-choice questions. In contrast, JobChecker's combination of OCR and GPT-4 allows for more flexible, context-aware evaluations of open-ended written responses.

While other systems, such as ETS's Criterion or Pearson's MyLab, focus primarily on typed responses, JobChecker stands out for its ability to handle handwritten input. This unique feature positions JobChecker as a potential game-changer in educational settings where handwritten assignments remain prevalent.

5.8 Future Work

Several opportunities exist for future enhancements to JobChecker. Key areas for improvement include the enhancement of OCR accuracy for more diverse handwriting styles, the introduction of additional NLP-based features such as sentiment analysis or deeper content evaluation, and the incorporation of more customizable feedback options for teachers.

Furthermore, scaling JobChecker to handle larger volumes of students and assignments could involve leveraging cloud-based infrastructure and improving load balancing techniques. Integrating advanced machine learning models, including fine-tuning GPT-4 or exploring alternative models for specific educational tasks, could also enhance the system's overall performance.

5.9 Team Roles

The development of JobChecker was the result of collaborative work among team members, each specializing in distinct areas to ensure a cohesive and well-rounded system.

- **Backend Development** was led by Timur Issamadiyev, who was responsible for implementing the server-side logic, designing the API endpoints, managing database interactions, and integrating the OCR and GPT-4 modules into a scalable backend architecture.
- **Frontend Web Development** was handled by Taikozha Turdyakyn, who developed the teacher and admin-facing web interface using modern frameworks. Their focus included interface responsiveness, user authentication, and seamless communication with backend services.
- **iOS Development** was carried out by Dias Murzagaliyev, who built the mobile application targeted at student and teacher users. The app emphasized intuitive interaction, authorization, viewing assignments, grade viewing, and easy submission of assignments on the student side, as well as additional creating assignments, generating the AI feedback and grading features on the teacher side.
- **UI/UX Design and Documentation** were managed by Alua Meiirbekova, who created the system's design prototypes using Photoshop and Figma. Alua also led the structuring and writing of project documentation, ensuring clarity and coherence across technical and non-technical materials.

6 Conclusion and Future Work

6.1 Conclusion

JobChecker represents a significant advancement in the automation of grading handwritten assignments. By leveraging Optical Character Recognition (OCR) for text extraction and GPT-4 for feedback generation, the system provides an innovative solution to the common challenges of manual grading, including time constraints, subjectivity, and inconsistencies in evaluation. The system has been designed to offer real-time, constructive feedback, empowering both students and teachers while promoting more efficient educational workflows.

Through the integration of OCR and GPT-4, JobChecker demonstrated the ability to handle diverse handwritten inputs with a high degree of accuracy and provide meaningful feedback. Although challenges related to OCR accuracy and the need for continuous refinement of AI-based feedback remain, the results from user testing and evaluation highlight the system's potential to revolutionize grading practices, particularly in educational environments where handwritten assignments are still prevalent.

The success of this project also highlights the importance of incorporating advanced AI technologies into education to reduce the burden on educators while improving the quality of learning for students. The system's ability to scale and adapt to multiple subjects, languages, and educational settings further reinforces its potential as a valuable tool for the future of digital assessment.

6.2 Contributions

The development of JobChecker has contributed to both the academic and practical aspects of AI-driven educational tools. It has demonstrated how integrating OCR and natural language processing can address existing gaps in automated grading systems, particularly for handwritten assignments. The project has provided insights into the challenges of working with handwritten text and the complexities of applying AI models to such tasks.

Additionally, the system has offered valuable experience in full-stack development, AI model integration, and user-centric design, allowing the development team to deepen their understanding of how to create scalable and effective solutions in the educational technology domain.

6.3 Limitations

While JobChecker represents a promising approach, it is not without limitations. One of the primary challenges is the OCR system's sensitivity to handwriting quality. Although Tesseract OCR performs well on standard handwritten content, it can struggle with cursive, messy, or highly stylized handwriting. As such, the accuracy of the text extraction is still dependent on the quality of the scanned documents.

Another limitation lies in the dynamic nature of GPT-4's feedback generation. While the model produces context-aware and relevant feedback in most cases, there are instances where the model's understanding of ambiguous or poorly phrased responses can result in less specific or generalized feedback. Further training of the model to better handle such cases could improve the consistency and usefulness of the feedback.

6.4 Future Work

While JobChecker has demonstrated strong potential in automating the assessment of handwritten student assignments, several areas for future development remain that could enhance its accuracy, versatility, and user impact. A key priority is improving OCR recognition across a broader spectrum of handwriting styles, particularly cursive and less conventional scripts. This may involve integrating more advanced OCR models or developing a custom recognition engine optimized for diverse input quality and handwriting variation.

Expanding the capabilities of the feedback system is another area of focus. Future iterations of JobChecker could incorporate deeper natural language processing functions, including sentiment analysis, contextual content evaluation, and personalized learning recommendations. Such enhancements would provide students with more targeted, meaningful, and pedagogically aligned feedback.

In addition, broadening the system's language support will significantly extend its applicability across different educational contexts. Currently tailored for English-language submissions, JobChecker would benefit from multilingual OCR and NLP capabilities, allowing it to process and evaluate assignments in a variety of languages and scripts.

Integration with established Learning Management Systems (LMS), such as Moodle or Canvas, presents another valuable direction. By embedding JobChecker within existing LMS platforms, educators could benefit from a more unified workflow for assignment distribution, grading, and feedback dissemination, improving both convenience and adoption.

To support increased adoption, future work should also address system scalability and performance. Enhancing the underlying infrastructure, including improvements in processing pipelines, load balancing, and data management, will be essential for maintaining responsiveness and reliability as the user base grows.

Finally, improvements to the user interface can further elevate the user experience. Potential features include enhanced feedback visualization, richer customization options for educators, and tools that allow students to monitor their progress over time. These refinements would reinforce user engagement and promote sustained educational benefit.

A Use-Case Diagram

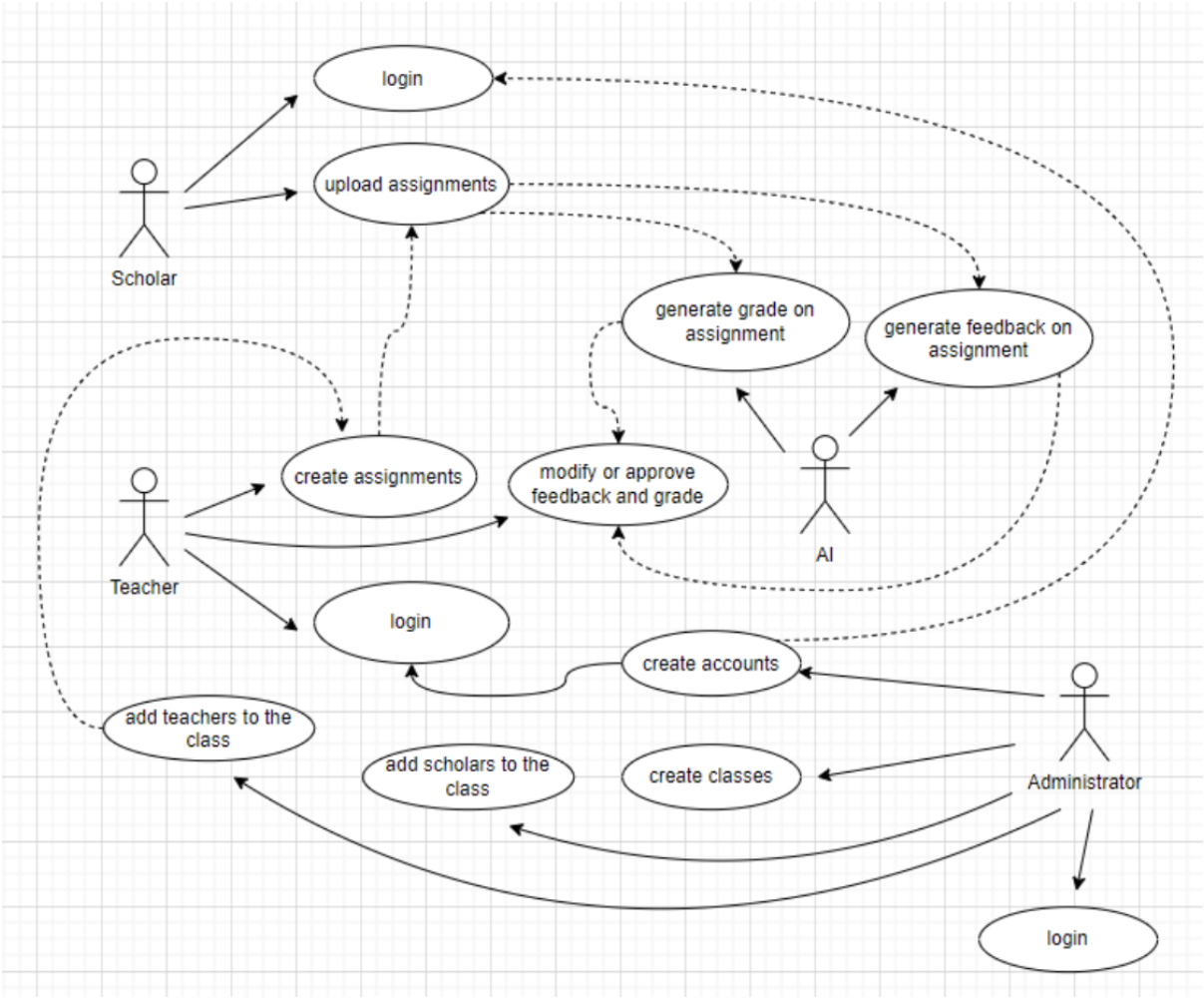

Figure 1: Structured use-case diagram

B Sequence Diagram

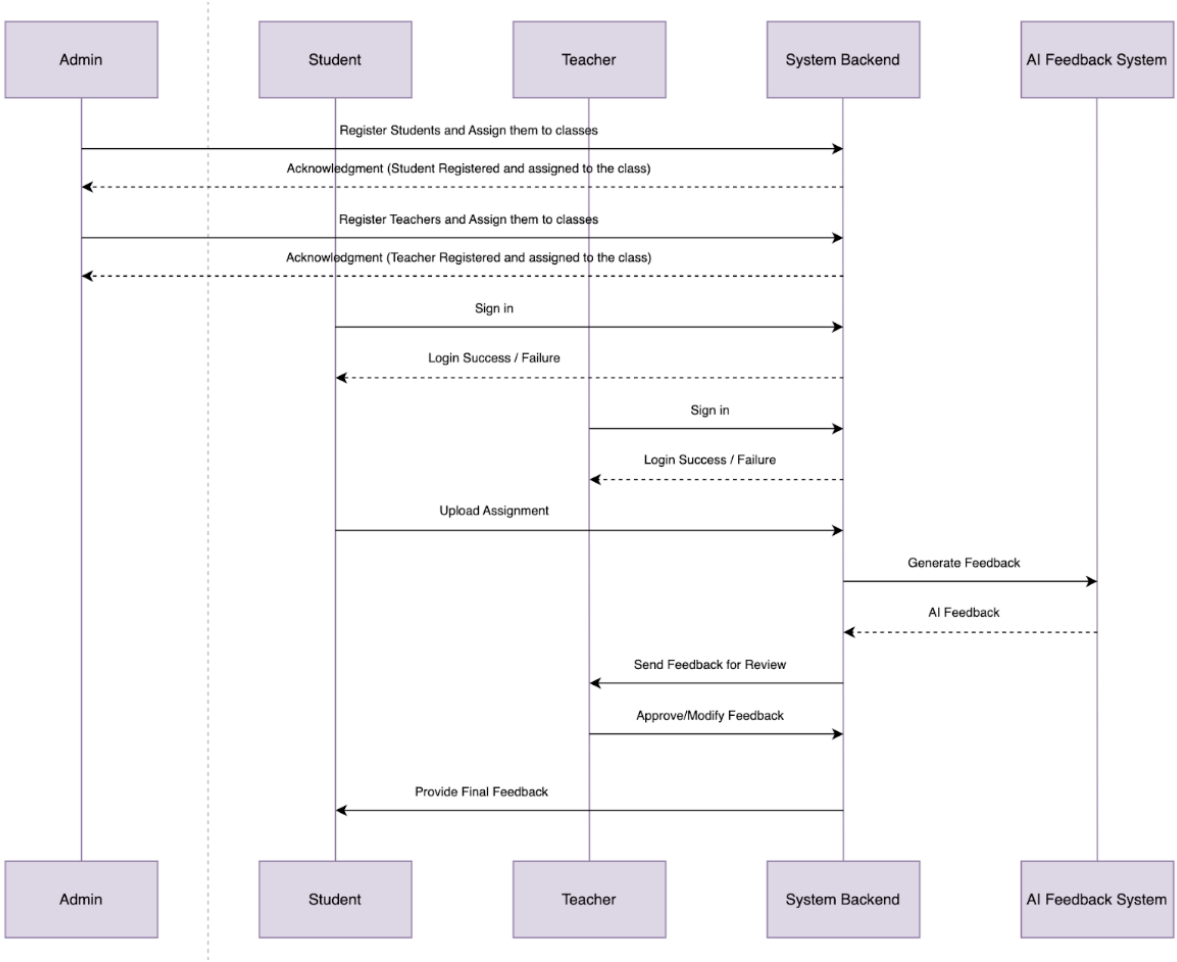

Figure 2: Sequence Diagram

C Architectural Model

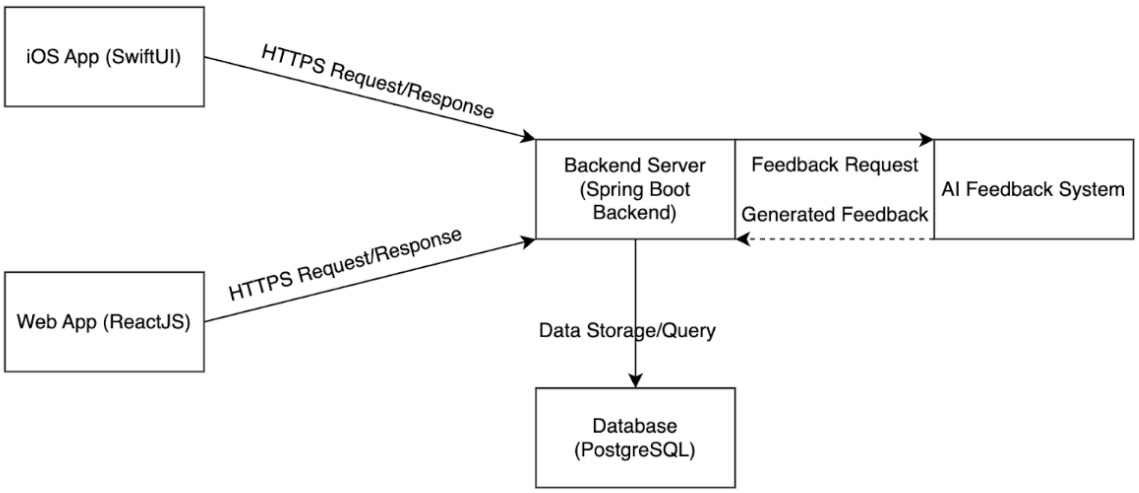

Figure 3: Architectural Model

References

[1] Mohamed Abdellatif Hussein, Hesham Hassan, and Mohammad Nassef. "Automated language essay scoring systems: A literature review". In: *PeerJ Computer Science* 5 (2019), e208.

[2] Steven Burrows, Iryna Gurevych, and Benno Stein. "The eras and trends of automatic short answer grading". In: *International journal of artificial intelligence in education* 25 (2015), pp. 60–117.

[3] Ray Smith. "An overview of the Tesseract OCR engine". In: *Ninth international conference on document analysis and recognition (ICDAR 2007)*. Vol. 2. IEEE. 2007, pp. 629–633.

[4] J OpenAI Achiam et al. "GPT-4 technical report. arXiv". In: *arXiv preprint arXiv:2303.08774* (2023).

[5] Sven Jacobs and Steffen Jaschke. "Evaluating the application of large language models to generate feedback in programming education". In: *2024 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2024, pp. 1–5.

[6] Jamshed Memon et al. "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)". In: *IEEE access* 8 (2020), pp. 142642–142668.

[7] Craig Walls. *Spring in action*. Simon and Schuster, 2022.

[8] Dan Sporici, Elena Cușnir, and Costin-Anton Boiangiu. "Improving the accuracy of Tesseract 4.0 OCR engine using convolution-based preprocessing". In: *Symmetry* 12.5 (2020), p. 715.

[9] Pablo Flores and Guang Rong. "Exploring the Use of GPT-4 in Co-creating Personalized Case Scenarios for Higher Education." In: *Education* (2024).

[10] Kirupa Chinnathambi. *Learning React: a hands-on guide to building web applications using React and Redux*. Addison-Wesley Professional, 2018.

[11] Chris Anderson. "The model-view-viewmodel (mvvm) design pattern". In: *Pro Business Applications with Silverlight 5*. Springer, 2012, pp. 461–499.