COMP/SOFT4902
Graduation Design Project

FEYZİYE SCHOOLS FOUNDATION
IŞIK UNIVERSITY

Project Title
LLM-Based Essay Grading Assistant System (Gradia)

Ece Büşra Civelek/21SOFT1032, Cansu Zeynep Kars/21SOFT1013, Orhan Murat Tuncer/21SOFT1028

REQUIREMENTS ANALYSIS DOCUMENT

1. Introduction

The LLM-Based Essay Grading Assistant System (Gradia) is a solution designed to address the challenges of manual essay grading in educational institutions. With the increasing demand for timely and accurate feedback on student essays, teachers often face significant workloads, leading to delays and inconsistencies in grading. Existing automated grading systems, such as ETS's e-rater and Turnitin's Revision Assistant, have made strides in automating the grading process. However, they often lack the flexibility for teachers to review and adjust the grades manually. Additionally, these systems don't provide comprehensive feedback on grammar, content, and structure, limiting their effectiveness in supporting student learning [1, 2].

Recent advancements in Large Language Model (LLM) technologies, such as GPT-3 and BERT, have demonstrated the potential for more accurate essay grading. Studies have shown that these models can evaluate essays based on grammar, coherence, and content, achieving grading accuracy comparable to humans [3, 4]. Gradia builds on these advancements, integrating LLM technologies into a user-friendly system that automates grading while allowing teachers to provide meaningful feedback. The system also includes a blackboard system for any educational platform, enabling teachers to manage classes, assignments, and announcements, while students can view their grades and submit essays.

1.1. Purpose of the System

The primary purpose of Gradia is to upgrade the essay grading process by leveraging Large Language Model (LLM) technologies. Unlike automated grading systems, which often focus on grammar and plagiarism detection, Gradia aims to provide holistic feedback on grammar, content, and structure, ensuring that students receive thorough evaluations of their work.

Key Goals of Gradia Include:

- Automating Grading: To reduce the workload on teachers by automating the grading process using LLM, while maintaining the flexibility for teachers to review and adjust grades manually.
- Providing Detailed Feedback: Offer students detailed and actionable feedback on their essays, helping them improve their skills over time.

- Enhancing Communication: Facilitate better communication between teachers and students through features like class announcements.
- Improving Efficiency: Streamline the grading process to ensure timely feedback, enabling students to learn from their mistakes and improve their performance.

By combining AI-based grading with a user-friendly interface, Gradia seeks to address the limitations of existing systems and provide a more effective solution for essay evaluation in educational settings.

1.2. Scope of the System

Within Scope:

- Teachers and students can register and log in to the system securely.
- Essay submission by students through the system, either by uploading files or typing directly into the platform. The system validates file formats and checks for basic plagiarism.
- AI-based essay grading using LLM models according to the rubrics. The LLM evaluates essays based on grammar, coherence, and content, according to the rubrics provided by the teacher, and returns feedback in JSON format.
- Teachers can review AI-generated grades and make manual adjustments to the grades, including overriding grades and providing additional feedback.
- Teachers can create classes on the dashboard and add relevant students via CSV upload or manually by entering student numbers.
- Grading history and feedback viewing for students and teachers, including timestamps for submissions, grading, and feedback.
- Teachers can create(or let the system create), modify, and delete grading rubrics, which must include a total score and a breakdown of points for each criterion.
- Teachers can post announcements for their classes, which can include text, links, or attachments. Students can view these announcements on their dashboards.
- Users can reset passwords and log out of the system. Password reset involves email verification via the Email API.

Out of Scope:

- Offline functionality (the system requires an Internet connection).
- Integration with external Learning Management Systems (LMS).
- Support for languages other than English.
- Real-time collaboration between students and teachers is not supported (e.g., live editing or chat).
- While the system checks for basic plagiarism, it does not include advanced plagiarism detection features like Turnitin.

1.3. Objectives and Success Criteria of the Project

Objectives:

- Develop a functional web-based essay grading system using LLM models. The system will use on-premise servers.
- Provide accurate and consistent essay grading with minimal human intervention. The system will achieve 90% accuracy compared to human grading.
- Allow teachers to manually review and adjust AI-generated grades, ensuring fairness and accuracy.
- Ensure a user-friendly interface for teachers and students, including quick tutorials with clear instructions for all major functionalities.
- Enable teachers to post class announcements to facilitate better communication.

- Provide tools for teachers to create and manage classes, assignments, rubrics, and announcements. Teachers can upload student lists via CSV and customize rubrics with point breakdowns.

Success Criteria:

- The system successfully grades essays with an accuracy rate of at least 90% compared to human grading.
- Teachers can review and adjust grades with ease.
- Students receive timely feedback on their submissions within at least a week from the end of the deadline.
- Teachers can post announcements effectively.
- The system is fully functional and deployed for use by the end of the project timeline.

1.4. Definitions, Acronyms, and Abbreviations

- LLM: Large Language Model
- AI: Artificial Intelligence
- UC: Use Case
- RAD: Requirements Analysis Document
- GUI: Graphical User Interface
- API: Application Programming Interface
- Rubric: A set of criteria for grading essays, including a total score and a breakdown of points for each criterion.
- Essay: A written piece of work submitted by a student for grading.
- Announcement: A message posted by a teacher for a specific class, which can include text, links, or attachments.

1.5. Overview

This document outlines the requirements for the LLM-Based Essay Grading Assistant System (Gradia). It includes a detailed description of the system's functionality, use cases, non-functional requirements, system models, and project risks. The document is organized into the following sections:

- Introduction: Provides an overview of the system, its purpose, scope, objectives, and success criteria.
- Current System: Describes the existing systems and the problems it addresses.
- Proposed System: Details the functionality, use cases, and non-functional requirements of the new system.
- System Models: Includes use case diagrams, class diagrams, sequence diagrams, and user interface mock-ups.
- Other Analysis Elements: Covers risks, alternatives, and the project plan.
- Glossary: Defines key terms and acronyms used in the document.
- References: Lists of all the documents and resources referenced in the RAD.

2. Current System

Existing automated grading systems, such as ETS's e-rater and Turnitin's Revision Assistant, have been widely used in educational settings. However, these systems focus on grammar and plagiarism detection, with limited capabilities for evaluating the depth of content or providing detailed feedback on essay structure. For instance, while Turnitin excels at identifying plagiarism and grammatical errors, it falls behind in assessing the quality of arguments, logical coherence, or creativity

in student essays [2]. Similarly, tools like Grammarly offer grammar and style suggestions but do not provide holistic grading or feedback tailored to specific assignment rubrics. As a result, students often receive feedback that lacks actionable insights, leaving students without clear guidance on how to improve themselves. These limitations highlight the need for a more advanced and flexible grading system that can provide comprehensive evaluations and support student learning.

Recent advancements in LLMs, such as GPT-3 and BERT, have demonstrated the potential to address these gaps. Studies have shown that LLMs can effectively evaluate essays based on grammar, coherence, and content, achieving accuracy comparable to human grading [3, 4]. For example, LLMs can analyze the logical flow of arguments, assess the relevance of supporting evidence, and provide nuanced feedback on essay structure. These capabilities go beyond the surface-level analysis offered by existing systems, enabling more accurate and meaningful evaluations. However, despite their potential, LLMs have yet to be fully integrated into user-friendly, classroom-ready systems that allow teachers to review and adjust AI-generated grades while providing actionable feedback to students. This gap presents an opportunity for Gradia to leverage the latest advancements in LLM technology to create a more effective and flexible grading solution. Gradia aims to provide detailed and actionable feedback and empower teachers and students to achieve better educational outcomes.

3. Proposed System

3.1. Overview

The LLM-Based Essay Grading Assistant System (Gradia) is designed to transform the essay grading process in educational settings by leveraging LLMs. The system aims to automate grading while providing detailed, actionable feedback on grammar, content, and structure, addressing the limitations of existing tools. Gradia integrates seamlessly into classroom workflows, offering a user-friendly interface for teachers and students to manage assignments, grades, and feedback.

The primary stakeholders of the system include:

- Teachers: They can create classes, assignments, and rubrics; review and adjust AI-generated grades; and post announcements for their students.
- Students: Can submit essays, view grades and feedback, and access class announcements.

Gradia's core functionality includes AI-based essay grading, manual grade adjustment by teachers, rubric management, and classroom communication tools. By combining the strengths of LLMs with teacher oversight, Gradia ensures accurate, consistent, and fair grading while reducing educators' workloads.

3.2. Requirements

The following requirements define the high-level functionality of Gradia:

- Automated Essay Grading: The system uses LLMs to grade essays based on predefined rubrics, evaluating grammar, coherence, and content.
- Teacher Oversight: Teachers can review and adjust AI-generated grades to ensure fairness and accuracy.
- Detailed Feedback: Students receive comprehensive feedback on their essays, helping them improve their skills.
- Classroom Management: Teachers can create classes, add students, and post announcements, while students can view their assignments and grades.
- User Authentication: Secure login and registration for teachers and students, with password reset functionality.
- Grading History: Both teachers and students can view past grades and feedback for submitted essays.

- Rubric Customization: Teachers can create, modify, and delete grading rubrics tailored to specific assignments. Teachers also have the option to let LLM create rubrics itself.
- Notifications: Students are notified when their essays are graded from the dashboard.

3.2.1. Use Cases

Table 1

ID and Name:	**UC-1: User Registration & Authentication**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System
Description:	This use case represents when a user registers or logs into the system. The system verifies user credentials and grants access accordingly. During registration, the system sends an email confirmation via the Email API to verify the user's email address.		
Trigger:	User attempts to log in or register.		
Preconditions:	PRE-1: User has access to the web application. PRE-2: User must have valid credentials or complete the registration process. PRE-3: User's email is not already registered (for registration).		
Postconditions:	POST-1: User is authenticated and granted access. POST-2: User's account details are updated if newly registered. POST-3: The system displays the user's dashboard (e.g., links to submit essays, view grades, etc.).		
Normal Flow:	**Normal Flow: 1.0 Register & Authenticate** 1. The system displays login and registration options. 2. User selects "Log In" or "Sign Up." 3. The system prompts the user to enter credentials. 4. User provides valid credentials. 5. The system verifies the credentials (or creates a new account if registering). 6. System grants access and redirects to the dashboard.		
Alternative Flows:	ALT-1: Incorrect credentials: System displays an error message. ALT-2: Registering an email already exists: System prompts the user to log in. ALT-3: User forgets their password: System provides a password reset option.		
Exceptions:	**1.0.E1: Email is already in use:** • The system notifies the user that the email is already registered. • User is prompted to either log in or reset their password.		
Priority:	High		

Table 2

ID and Name:	**UC-2: Student Dashboard**

Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Student	Secondary Actors:	System
Description:	This use case represents the student's ability to view their enrolled classes, upcoming assignments, grades, and feedback they received from their essays.		
Trigger:	Student logs in to the system.		
Preconditions:	PRE-1: Student is registered and logged in. PRE-2: Student is enrolled in at least one class.		
Postconditions:	POST-1: Student views their dashboard with classes, assignments, and grades. POST-2: Student can navigate to submit essays or view feedback.		
Normal Flow:	**Normal Flow: 2.0 Student Dashboard** 1. Student logs in. 2. The system displays the students' dashboards with classes, assignments, feedback, and grades. 3. Students select a class to view details (e.g., assignments, feedbacks, announcements).		
Alternative Flows:	ALT-1: No classes or assignments exist: The system displays a message indicating no data is available. ALT-2: Student is not enrolled in any classes: The system displays a message.		
Exceptions:	**2.0.E1: System fails to retrieve data because of server issues:** • The system displays an error message and suggests retrying later.		
Priority:	High		

Table 3

ID and Name:	**UC-3: Teacher Dashboard**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Teacher	Secondary Actors:	System
Description:	This use case represents the teacher's ability to view their created classes, add students, create assignments, and manage rubrics.		
Trigger:	Teacher logs in to the system.		
Preconditions:	PRE-1: Teacher is registered and logged in. PRE-2: Teacher has created at least one class.		
Postconditions:	POST-1: Teacher views their dashboard with classes, assignments, and rubrics. POST-2: Teacher can navigate to create assignments or manage rubrics.		
Normal Flow:	**Normal Flow: 3.0 Teacher Dashboard** 1. Teacher logs in. 2. The system displays the teacher's dashboard with classes and assignments that they created. 3. The teacher selects to plus button to create a class and add students. 4. Can create assignments, feedbacks and announcements inside the class.		
Alternative Flows:	ALT-1: No classes exist: The system prompts the teacher to create a new class.		

Exceptions:	**3.0.E1: System fails to retrieve teacher data:** • The system displays an error message.
Priority:	High

Table 4

ID and Name:	**UC-4: Create Class**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Teacher	Secondary Actors:	System
Description:	This use case represents the teacher's ability to create and add students to a new class. Students can be added via CSV upload or manually by entering student numbers.		
Trigger:	Teacher selects "Create Class" button, which has a logo "+" from the dashboard.		
Preconditions:	PRE-1: Teacher is logged in. PRE-2: Teacher has permission to create classes.		
Postconditions:	POST-1: A new class is created and saved in the system. Post-2: Students are added to the class.		
Normal Flow:	**Normal Flow: 4.0 Create Class** 1. Teacher selects "+" button to create class. 2. Teacher enters the class details (e.g., class name, description). 3. Teacher adds students to the class by uploading a CSV file or manually entering student numbers. 4. The system saves the class and notifies the students.		
Alternative Flows:	ALT-1: Invalid class details: The system displays an error message and prompts the teacher to re-enter details. ALT-2: CSV file upload fails: The system displays an error message and prompts the teacher to re-upload the file.		
Exceptions:	**4.0.E1: System fails to save the class:** • The system displays an error message.		
Priority:	Medium		

Table 5

ID and Name:	**UC-5: Create Assignment**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Teacher	Secondary Actors:	System
Description:	This use case represents the teacher's ability to create essay assignments for their classes, specifying deadlines and rubrics. The rubric must include a total score of hundred and a breakdown of points for each criterion.		
Trigger:	Teacher selects "Create Assignment" button, which has a "+" symbol, for a specific class.		
Preconditions:	PRE-1: Teacher is logged in. PRE-2: Teacher has created at least one class.		
Postconditions:	POST-1: A new assignment is created and saved in the system. POST-2: Students in the class are notified of the new assignment.		
Normal Flow:	**Normal Flow: 5.0 Create Assignment** 1. Teacher selects "Create Assignment" button in a specific class.		

	2. Teacher enters assignment details (e.g., title, descriptions, deadline). 3. The teacher selects or creates a rubric for grading, including total points and criteria breakdown. 4. System saves the assignment and notifies students in the class via dashboard.
Alternative Flows:	ALT-1: Invalid assignment details: System displays an error message and prompts the teacher to re-enter details. ALT-2: Invalid deadlines: The system displays an error message and prompts the teacher to enter a valid deadline.
Exceptions:	**5.0.E1: Rubric not found:** • The teacher selects a rubric that no longer exists. • System displays an error.
Priority:	High

Table 6

ID and Name:	**UC-6 Submit an Essay**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Student	Secondary Actors:	System
Description:	This use case represents a student submitting an essay for grading before the deadline.		
Trigger:	The student selects an assignment from their dashboard and tries to upload an essay through the system.		
Preconditions:	PRE-1: Student is logged into the system. PRE-2: Student has an active assignment with a deadline.		
Postconditions:	POST-1: Essay is submitted and stored in the database. POST-2: Essay is sent for grading.		
Normal Flow:	**Normal Flow: 6.0 Submit an Essay** 1. Student selects an assignment from their dashboard. 2. System prompts the user to upload a file. 3. Student uploads the essay file or types the essay. 4. The system validates the format and checks for plagiarism. 5. System confirms submission and sends the essay for grading.		
Alternative Flows:	ALT-1: Invalid file format: System displays an error and asks for re-upload. ALT-2: Essay exceeds word limit: System rejects the submission and prompts the student to shorten the essay. ALT-3: Submission after deadline: The system rejects the submission and displays an error message.		
Exceptions:	**6.0.E1: File storage failure** • The system fails to save the uploaded essay. • Displays a message indicating the error. • Logs the storage error for technical support.		
Priority:	High		

Table 7

ID and Name:	**UC-7 AI-Based Essay Grading**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	System	Secondary Actors:	LLM Model
Description:	This use case represents the system grading an essay using the LLM model. The system parses the essay, evaluates it against the rubric, and returns a JSON response with the grade and feedback.		
Trigger:	An essay is submitted and sent for grading.		
Preconditions:	PRE-1: An essay is submitted. PRE-2: The grading rubric is set. PRE-3: LLM model is available and functioning.		
Postconditions:	POST-1: A grade is generated for the essay. POST-2: Feedback is stored in the database. POST-3: The student is notified of the grade via dashboard.		
Normal Flow:	**Normal Flow: 7.0 AI-Based Grading** 1. System retrieves the submitted essay. 2. System applies the grading rubric. 3. System processes the essay through the LLM model. 4. LLM Model analyzes grammar, content, and structure. 5. The system generates a score and feedback. 6. The system saves the grade and notifies the student.		
Alternative Flows:	ALT-1: LLM model fails to generate feedback: System notifies and queues the essay for manual grading.		
Exceptions:	**7.0.E1: Essay content unreadable** • LLM cannot parse the essay (e.g., corrupted file, non-text content). • The system leaves the essay for manual grading.		
Priority:	High		

Table 8

ID and Name:	**UC-8 Teacher Review & Manual Adjustment**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Teacher	Secondary Actors:	System
Description:	This use case represents a teacher reviewing AI-generated grades and making adjustments. The teacher can override the AI's grade and provide additional feedback.		
Trigger:	The teacher selects an essay for review.		
Preconditions:	PRE-1: AI has graded an essay.		
	PRE-2: Teacher has permission to review and adjust grades.		
Postconditions:	POST-1: Grade is updated if modified. POST-2: Adjusted feedback is stored.		

Normal Flow:	**Normal Flow: 8.0 Review & Adjust Grade** 1. The system displays a list of graded essays. 2. The teacher selects an essay for review. 3. System shows AI-generated feedback and score. 4. The teacher edits the score or feedback if necessary. 5. The system updates the grade and notifies the student via dashboard.
Alternative Flows:	ALT-1: Teacher does not modify the grade: No changes are made. ALT-2: Teacher attempts to adjust a grade outside the allowed range: System displays an error message and prompts the teacher to enter a valid grade. ALT-3: Invalid feedback: The system displays an error message and prompts the teacher to enter valid feedback.
Exceptions:	**8.0.E2: Invalid session during adjustment:** • Teacher's session expires mid-edit. • System discards unsaved changes and redirects to the login page. • Logs the incident for audit purposes.
Priority:	Medium

Table 9

ID and Name:	**UC-9 View Grading History**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System
Description:	This use case represents a user viewing past grades and feedback for submitted essays. Users can view a list of all graded essays and access detailed feedback for each essay.		
Trigger:	User navigates to the grading history page.		
Preconditions:	PRE-1: At least one graded essay exists. PRE-2: User has permission to view grading history.		
Postconditions:	POST-1: User sees past grades and feedback (e.g., essay title, grade, feedback, submission date). POST-2: User can view detailed feedback for a specific essay (e.g., grammar, content, structure).		
Normal Flow:	**Normal Flow: 9.0 View Grading History** 1. The system displays the grading history page for each class. 2. User selects a class. 3. User selects a past essay from the list of that class. 4. The system shows the grade, feedback, and teacher comments.		
Alternative Flows:	ALT-1: No graded essays exist: System displays an empty history message. ALT-2: No permission: The system displays an error message.		
Exceptions:	**9.0.E1: Data retrieval timeout:** • System fails to load grading history due to a slow database response • Logs the timeout event for performance tuning.		
Priority:	Low		

Table 10

ID and Name:	**UC-10 Create and Manage Grading Rubric**

Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Teacher	Secondary Actors:	System
Description:	This use case represents a teacher creating, modifying, or deleting a grading rubric to evaluate essays. The rubric must include a total score and a breakdown of points for each criterion.		
Trigger:	Teacher selects the "+" button in the assignment's rubric section.		
Preconditions:	PRE-1: Teacher is logged in. PRE-2: Teacher has permission to create or modify rubrics.		
Postconditions:	POST-1: A new rubric is created, modified, or deleted. POST-2: The rubric is stored in the database with a unique ID.		
Normal Flow:	**Normal Flow: 10.0 Manage Grading Rubric** 1. The system displays the grading rubric management page under an assignment. 2. System prompts the teacher to enter rubric criteria (e.g., grammar, content, coherence). 3. The teacher clicks "Create" and saves the rubric. 4. The system stores the rubric and confirms the update.		
Alternative Flows:	ALT-1: Teacher attempts to delete rubric: System asks for confirmation before deletion. ALT-2: Teacher attempts to create rubric with invalid criteria: System displays an error message and prompts the teacher to create the criteria. ALT-3: Incomplete criteria: The system displays an error message and prompts the teacher to complete the criteria.		
Exceptions:	**10.0.E1: Rubric linked to active assignment:** • The teacher attempts to delete a rubric used in an active assignment. • System blocks deletion. **10.0.E2: Duplicate rubric name:** • The teacher tries to save a rubric with a duplicate name. • System highlights the name field and prompts that the name already exists.		
Priority:	Medium		

Table 11

ID and Name:	**UC-11 Reset Password**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System
Description:	This use case allows users to reset their password if they forget it. The system sends a password reset link via the Email API.		
Trigger:	User selects "Forgot Password."		
Preconditions:	PRE-1: User's email is registered in the system.		
Postconditions:	POST-1: User's password is updated in the database, and a confirmation message is displayed on the screen. Post-2: User can log in with the new password.		

Normal Flow:	**Normal Flow: 11.0 Reset Password** 1. User clicks "Forgot Password." 2. The system prompts the user to enter their email. 3. User provides the correct answer. 4. The system allows the user to set a new password. 5. System updates the password.
Alternative Flows:	ALT-1: User cancels the process: System returns to the login page. ALT-2: Email not registered: The system displays an error message and prompts the user to register.
Exceptions:	**11.0.E1: Email service unavailable:** • Password reset email fails to send due to Email API downtime. • System queues the request.
Priority:	Medium

Table 12

ID and Name:	**UC-12 Log out**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System
Description:	This use case allows users to log out of the system.		
Trigger:	User selects "Log Out."		
Preconditions:	PRE-1: User is logged in.		
Postconditions:	POST-1: User is logged out and redirected to the login page.		
Normal Flow:	**Normal Flow: 12.0 Log Out** 1. User clicks "Log Out." 2. System logs the user out. 3. The system redirects the user to the login page.		
Alternative Flows:	ALT-1: Session expires: System automatically logs the user out.		
Exceptions:	**12.0.E1: Session termination failure:** • System fails to invalidate the session token. • Logs the error and redirects the user to the login page.		
Priority:	Low		

Table 13

ID and Name:	**UC-13: Create Class Announcement**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Teacher	Secondary Actors:	System
Description:	This use case represents the teacher's ability to post announcements for their classes.		
Trigger:	Teacher selects "Create Announcement" for a specific class.		
Preconditions:	PRE-1: Teacher is logged in. PRE-2: Teacher has created at least one class.		

Postconditions:	POST-1: Announcement is saved and displayed to students in the class. POST-2: Students are notified of the new announcement.
Normal Flow:	**Normal Flow: 13.0 Create Class Announcement** 1. Teacher selects "Create Announcement" for a specific class. 2. The teacher writes the announcement (e.g., text, links, or attachments). 3. The system saves the announcement and notifies students in the class in their dashboard.
Alternative Flows:	ALT-1: No classes or assignments exist: The system displays a message indicating no data is available/prompts the teacher to create a class first.
Exceptions:	**13.0.E1: Attachment size exceeds limit:** • Teacher uploads a file larger than 10MB. • System shows message indicating the error. **13.0.E2: Invalid link format:** • Teacher includes a broken/malformed URL. • System shows message indicating the error.
Priority:	Medium

Table 14

ID and Name:	**UC-14: View Class Announcements**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	Student	Secondary Actors:	System
Description:	This use case represents the student's ability to view announcements posted by their teachers.		
Trigger:	Student selects a class from their dashboard.		
Preconditions:	PRE-1: Student is logged in. PRE-2: Student is enrolled in at least one class.		
Postconditions:	POST-1: Student views a list of announcements for the selected class.		
Normal Flow:	**Normal Flow: 14.0 View Class Announcements** 1. Student selects a class from their dashboard or from the right side of the dashboard with instant notifications. 2. The system displays a list of announcements for the selected class. 3. Student selects an announcement to view its details.		
Alternative Flows:	ALT-1: No announcements exist: The system displays a message indicating no announcements are available.		
Exceptions:	**14.0.E1: Corrupted announcement data:** • System retrieves incomplete or garbled announcement text. • Logs the database corruption for admin review.		
Priority:	Medium		

Table 15

ID and Name:	**UC-15 Notify Student of Grade**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	03/03/25
Primary Actor:	System	Secondary Actors:	Student
Description:	This use case notifies the student when their essay has been graded.		
Trigger:	Essay is graded by the system.		

Preconditions:	PRE-1: Essay has been submitted and graded.
Postconditions:	POST-1: Student receives a notification via in-dashboard.
Normal Flow:	**Normal Flow: 15.0 Notify Student of Grade** 1. The system generates a grade and feedback. 2. System sends a notification to the student. 3. Student views the notification from dashboard and accesses their grade.
Alternative Flows:	ALT-1: Notification fails to send: System retries or logs the error.
Exceptions:	**15.0.E1: Notification system outage:** • System retries every 5 minutes and logs the failure.
Priority:	Medium

Table 16

ID and Name:	**UC-16 Edit Profile Picture**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	11/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System
Description:	This use case allows users to upload or change their profile picture.		
Trigger:	User selects the "Edit Image" icon on the top right corner of the picture profile.		
Preconditions:	PRE-1: User is logged in.		
Postconditions:	POST-1: The new profile picture is saved and displayed in the user's profile.		
Normal Flow:	**Normal Flow: 16.0 Edit Profile Picture** 1. User navigates to their profile settings. 2. User selects "Edit Image" icon on the top right corner of the picture profile. 3. User uploads a new image file. 4. The system validates the file format and size. 5. The system saves the new profile picture and updates the user's profile.		
Alternative Flows:	ALT-1: Invalid file format: The system displays an error message and prompts the user to upload a valid image file. ALT-2: File size exceeds limit: The system displays an error message and prompts the user to upload a smaller file.		
Exceptions:	**16.0.E1: Database update failure:** • System cannot save the new profile picture. • Reverts to the previous image and displays a proper error message.		
Priority:	Low		

Table 17

ID and Name:	**UC-17 Change Email Address**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	11/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System

Description:	This use case allows users to change their registered email address. The system sends a confirmation email to the new address for verification.
Trigger:	User selects “Edit Email” icon on the top right corner of the mail address label.
Preconditions:	PRE-1: User is logged in.
Postconditions:	POST-1: The new email address is saved and verified. POST-2: The user receives a confirmation email at the new address.
Normal Flow:	**Normal Flow: 17.0 Change Email Address** 1. User navigates to their profile settings. 2. User selects “Edit Email” icon. 3. User enters the new email address. 4. The system sends a confirmation email to the new address. 5. User clicks the confirmation link in the email. 6. The system verifies the new email address and updates the user’s profile.
Alternative Flows:	ALT-1: Invalid email format: The system displays an error message and prompts the user to enter a valid email address. ALT-2: Email already registered: The system displays an error message and prompts the user to use a different email address. ALT-3: Confirmation email fails to send: The system retries sending the email or logs the error.
Exceptions:	**17.0.E1: Confirmation link expiration:** • User clicks a confirmation link older than 24 hours. • System prompts a message indicating to error.
Priority:	Medium

Table 18

ID and Name:	**UC-18 View Profile**		
Created By:	Ece Büşra Civelek, Cansu Zeynep Kars, Orhan Murat Tuncer	Date Created:	11/03/25
Primary Actor:	User (Teacher or Student)	Secondary Actors:	System
Description:	This use case allows users to view their profile information, including their name, email, profile picture, course titles, and specifically for students, the instructor names for the corresponding classes.		
Trigger:	User navigates to their profile page.		
Preconditions:	PRE-1: User is logged in.		
Postconditions:	POST-1: User views their profile information.		
Normal Flow:	**Normal Flow: 18.0 View Profile** 1. User navigates to their profile page. 2. The system displays the user’s profile information.		
Alternative Flows:	ALT-1: No profile information available: The system displays a message indicating no data is available.		
Exceptions:	**18.0.E1: Profile data retrieval error:** • System fails to load the user’s enrolled classes. • Displays partial profile data and logs the missing fields.		
Priority:	Low		

3.2.2. Scenarios

Below are scenarios for some high-priority use cases:

- Scenario 1: Student Submits an Essay (UC-6)
  - Sarah, a college student, logs into Gradia to submit an essay for her English Literature class. She navigates to her dashboard and sees an assignment titled "Analysis of Shakespeare's Sonnets" with a deadline in two days. She clicks on the assignment and is prompted to upload her essay file. Sarah uploads a PDF file containing her essay. The system validates the file format and checks for basic plagiarism. Once the file is accepted, Sarah can see the status of her assignment as "Submitted". She feels relieved knowing her work is in the system and looks forward to receiving feedback.
- Alternative Scenario 1 (UC-6): Invalid File Format
  - Sarah tries to upload her essay in a .TXT file, but the system displays an error message*: "Invalid file format. Please upload a PDF or DOCX file.".* Sarah quickly converts her essay to a PDF and re-uploads it. This time, the system accepts the file, and Sarah receives the confirmation message.
- Alternative Scenario 2 (UC-6): Submission After Deadline
  - Sarah realizes she missed the deadline for the assignment. She tries to upload her essay, but the system doesn't let her because she realizes the deadline for this assignment has passed so submissions are no longer allowed. Disappointed, Sarah makes a note to submit her next assignment on time.
- Scenario 2: AI Grades an Essay (UC-7)
  - After Sarah submits her essay, the system automatically sends it for grading. The AI-powered grading module retrieves the essay and applies the grading rubric created by her teacher, Prof. Johnson. The rubric evaluates the essay on grammar, coherence, and content. The LLM analyzes Sarah's essay, identifying strengths such as her clear thesis statement and weaknesses like repetitive sentence structures. Based on the rubric, the system assigns a score of 85/100 and generates detailed feedback: *"Your argument is well-supported but try varying your sentence structure to improve readability."* The system saves the grade and feedback in the database and notifies Sarah: *"Your essay has been graded. Please check your grades and feedback."*
- Alternative Scenario 1 (UC-7): AI Cannot Process the Essay
  - The system attempts to grade Sarah's essay but encounters an unexpected error. The system notifies Prof. Johnson: "*The essay could not be graded automatically. Please grade it manually."* Prof. Johnson reviews the essay, assigns a grade of 90/100, and provides personalized feedback. Sarah is notified of her updated grade and feedback.
- Scenario 3: Teacher Reviews and Adjusts Grade (UC-8)
  - Prof. Johnson logs into Gradia to review the AI-generated grades for his class. He navigates to the "Assignments" section and selects Sarah's essay from the list of students, titled "Analysis of Shakespeare's Sonnets." The system displays the AI-generated grade (85/100) and feedback. Prof. Johnson reads the essay and feels that Sarah's analysis of the sonnet's themes deserves a higher score. He adjusts the grade to 90/100 and adds a comment: *"Excellent analysis of themes but try to vary sentence structure for better flow."* The system updates the grade and notifies Sarah. Sarah clicks on the notification and is thrilled to see her improved grade and appreciates the detailed feedback.
- Alternative Scenario 1 (UC-8): Teacher Does Not Modify the Grade
  - Prof. Johnson reviews Sarah's essay but agrees with the AI-generated grade of 85/100. He decides not to make any adjustments. The system retains the original grade and feedback, and Sarah is notified as usual.
- Scenario 4: Teacher Creates a Class (UC-4)

- Prof. Johnson needs to create a new class for the upcoming semester. He logs into Gradia and selects the "Create Class" button in shape of "+". He enters the class details: Class Name: "English Literature 101," Description: "Introduction to English Literature." He uploads a CSV file containing the names and email addresses of his students. The system validates the CSV file and adds the students to the class. Sarah checks her dashboard and sees the new class listed.

- Alternative Scenario 1 (UC-4): Invalid CSV File
  - Prof. Johnson uploads a CSV file with missing email addresses. The system displays a message indicating the error. He corrects the file and re-uploads it. This time, the system successfully adds the students to the class.
- Scenario 5: Teacher Creates an Assignment (UC-5)
  - Prof. Johnson wants to create a new assignment for his English Literature class. He logs into Gradia and navigates to the "Create Assignment" button as "+". He enters the assignment details: Title: "Analysis of Modern Poetry," Description: "Write a 1000-word essay analyzing themes in modern poetry," Due Date: "2025-03-15." He enters the file format as PDF. He selects a pre-existing rubric that evaluates essays on grammar (30 points), coherence (30 points), and content (40 points). The system saves the assignment and notifies the students in the class. Sarah sees the assignment notification in her dashboard and starts planning her essay.
- Alternative Scenario 1: Invalid Assignment Details
  - Prof. Johnson forgets to enter a title for the assignment. When he tries to save it, the system displays an error message: *"Please enter a title for the assignment."* He adds the title "Analysis of Modern Poetry" and successfully saves the assignment.
- Scenario 6: Teacher Creates Grading Rubric (UC-10)
  - Prof. Johnson wants to create a new rubric for grading essays. He logs into Gradia and navigates to the "Create Assignment" section where rubrics can be managed. He selects "Create New Rubric" button and enters the criteria: Grammar (30 points), Coherence (30 points), Content (40 points). He also declares the due date, file format, and max file number. The system saves the rubric and confirms: *"Rubric created successfully."* Prof. Johnson can now use this rubric to grade essays in his class.
- Alternative Scenario 1 (UC-10): Incomplete Rubric Criteria
  - Prof. Johnson forgets to specify the "Content" criterion he declared. When he tries to save the rubric, the system doesn't let's him. He adds the missing criterion and successfully saves the rubric.
- Scenario 7: Student Views Grading History (UC-9)
  - Sarah wants to track her progress in Prof. Johnson's class. She logs into Gradia and navigates to the "Grades" section. The system displays a list of all her grades for each class. She selects "English Literature 101" from the list and can see all graded essays. Sarah selects her most recent essay, "Analysis of Shakespeare's Sonnets," to view the detailed feedback and the grade. The system shows her grade (90/100), Prof. Johnson's comments, and the AI-generated feedback. Sarah notices a recurring suggestion to vary her sentence structure and decides to focus on this in her next essay.
- Alternative Scenario 1 (UC-9): No Graded Essays
  - Sarah has just started the semester and hasn't submitted any essays yet. When she navigates to the "Grades" section, and sees it is empty. Sarah understands that she needs to submit an assignment to see her grades.
- Scenario 8: Teacher Posts a Class Announcement (UC-13)
  - Prof. Johnson needs to remind his students about an upcoming deadline. He logs into Gradia and navigates to the "Announcement" section for his English Literature 101

class and selects the "+" button to create an announcement. He writes the announcement: *"Reminder: The deadline for the 'Analysis of Modern Poetry' assignment is two days away. Please submit your essays on time."* He adds a link to additional resources and posts the announcement. The system notifies all students in the class, and Sarah sees the announcement on her dashboard.

- Scenario 9: Student Views Class Announcements (UC-14):
    - Sarah logs into Gradia and navigates to her dashboard. She selects her English Literature 101 class and sees a list of announcements posted by Prof. Johnson. She clicks on the most recent announcement: *"Reminder: The deadline for the 'Analysis of Modern Poetry' assignment is two days away."* Sarah reads the announcement and finalizes her essay before the deadline.
- Alternative Scenario 1 (UC-13): No Announcements
    - Sarah's class doesn't have any announcements yet. When she navigates to the announcements section, the system displays a message: *"No announcements available."* Sarah checks back later and sees a new announcement from Prof. Johnson.
- Scenario 10: Student Resets Password (UC-11)
    - John, another student in Prof. Johnson's class, forgets his password and can't log into Gradia. He clicks the "Forgot Password" link on the login page. The system prompts him to enter his email. The system sends him an email with a reset link. He chooses a strong password and confirms it. The system updates his password. John logs in and continues working on his assignment.
- Scenario 11: Student Receives Grade Notification (UC-15)
    - After Sarah submits her essay, the system grades it and generates feedback. Sarah receives a notification on her dashboard. She clicks on the notification and sees her grade (90/100) along with detailed feedback from Prof. Johnson and the AI. Sarah is pleased with her grade and takes note of the feedback to improve her next essay.

3.3. Nonfunctional Requirements

- Usability:
    - The system must have an intuitive interface that is easy for teachers and students to navigate.
    - The system should provide clear instructions and tooltips for all major functionalities.
- Reliability:
    - The system must handle errors gracefully, providing meaningful error messages and recovery options.
- Performance:
    - The system must grade essays within 5 minutes of submission to ensure timely feedback.

    - Teachers must be able to finalize grades within 1 week after the assignment deadline, ensuring timely feedback for students.
- Supportability:
    - The system must include a quick tutorial page with steps to proceed.
- Implementation:
    - The system will rely on on-premises servers or a local hosting solution.
- External Interface (External interface is not only UI.)
    - The system must support multiple browsers (e.g., Chrome, Firefox, Safari) and devices (e.g., desktop, mobile).
- Legal:
    - The system must ensure that all user data is encrypted and stored securely.
- Social Impact:
    - Gradia reduces the workload on teachers by automating the essay grading process, allowing them to focus more on teaching and less on administrative tasks.
    - The system addresses the need for timely and consistent feedback for students, helping them improve their writing skills and academic performance.
    - By automating a task traditionally performed manually (essay grading), Gradia may reduce the need for additional grading staff in educational institutions. However, it also creates opportunities for teachers to focus on higher-value tasks, such as personalized instruction and mentoring.
    - The system does not directly create new jobs but enhances the efficiency of existing roles in educational settings.
- Environmental Impact:
    - Gradia reduces the need for paper-based submissions and manual grading processes, contributing to environmental sustainability by minimizing paper waste.
    - The system's reliance on on-premises servers (instead of cloud infrastructure) may reduce energy consumption associated with large-scale data centers, depending on the local hosting setup.
- Other Constraints:
    - No Administrator Role: The system will be designed to operate without dedicated administrators, with teachers and students managing their accounts and activities.
    - Regulatory Compliance: The system must adhere to educational and data privacy regulations, ensuring student data is handled securely and ethically.
    - Ethical Considerations: The system must ensure that AI-generated grades are fair and unbiased, with teachers having the final authority to review and adjust grades as needed.

3.4. System Models

3.4.1. Use Case Model

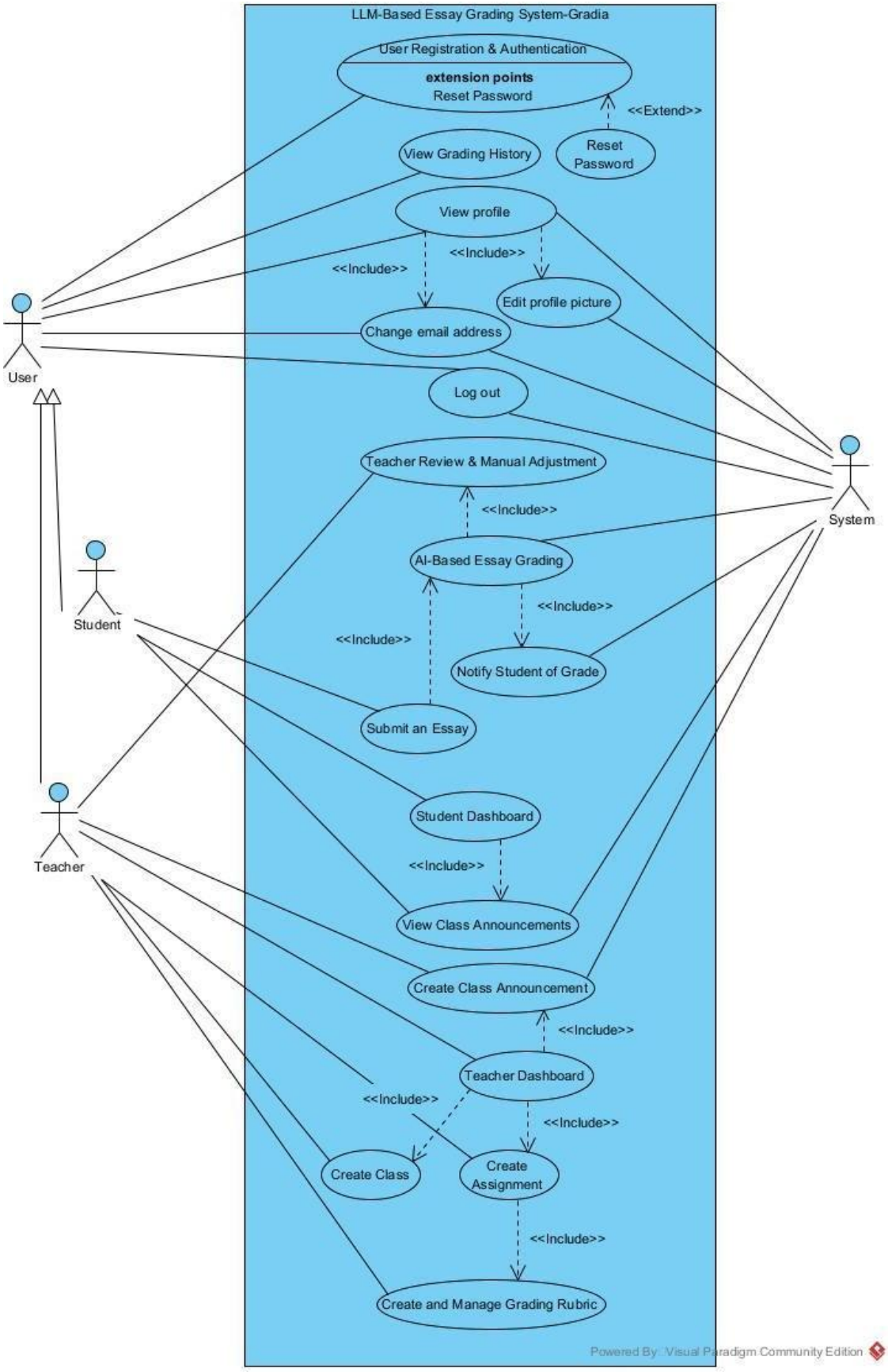

Figure 1

3.4.2. Object Model

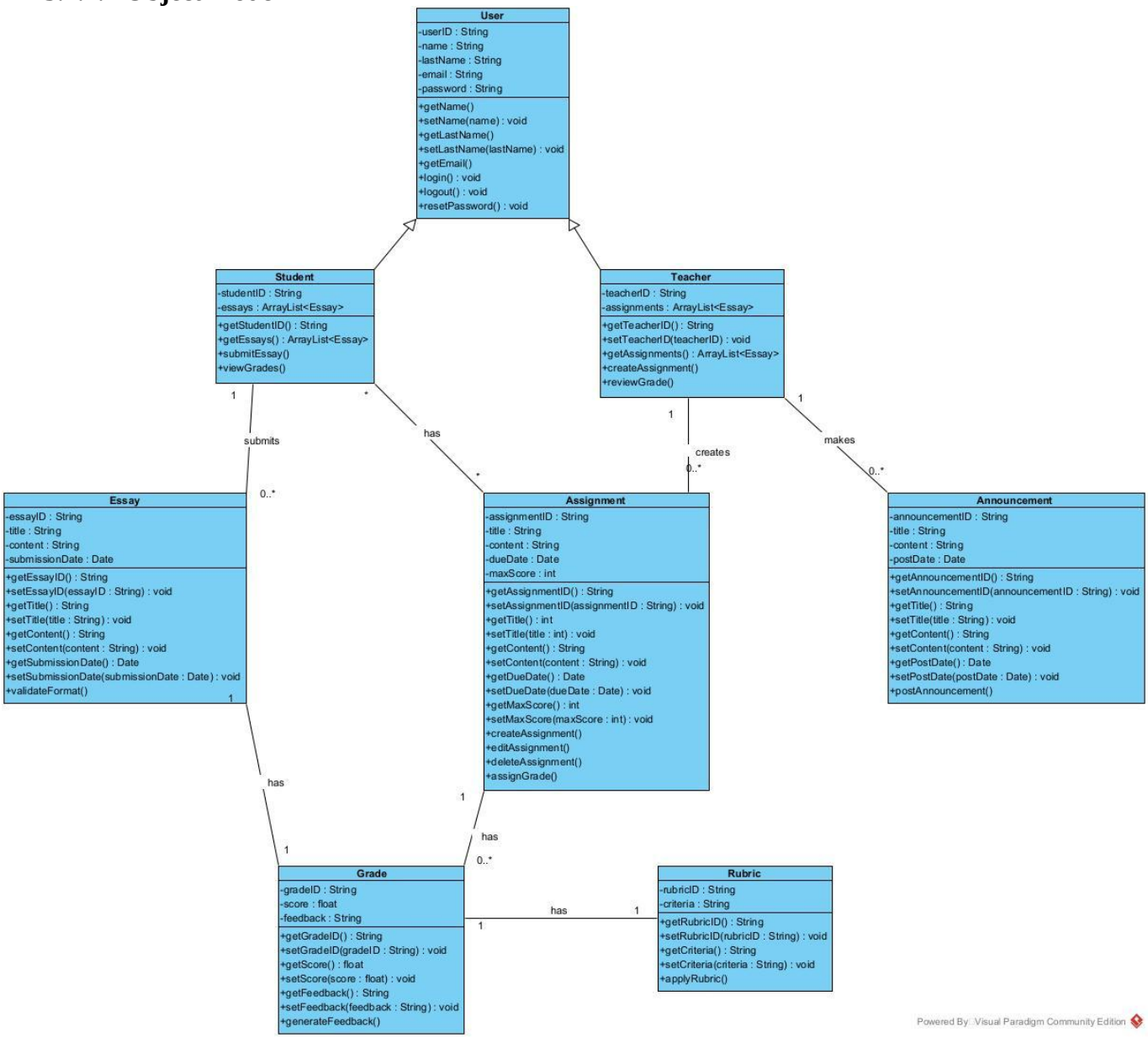

Figure 2

3.4.3. Dynamic Model

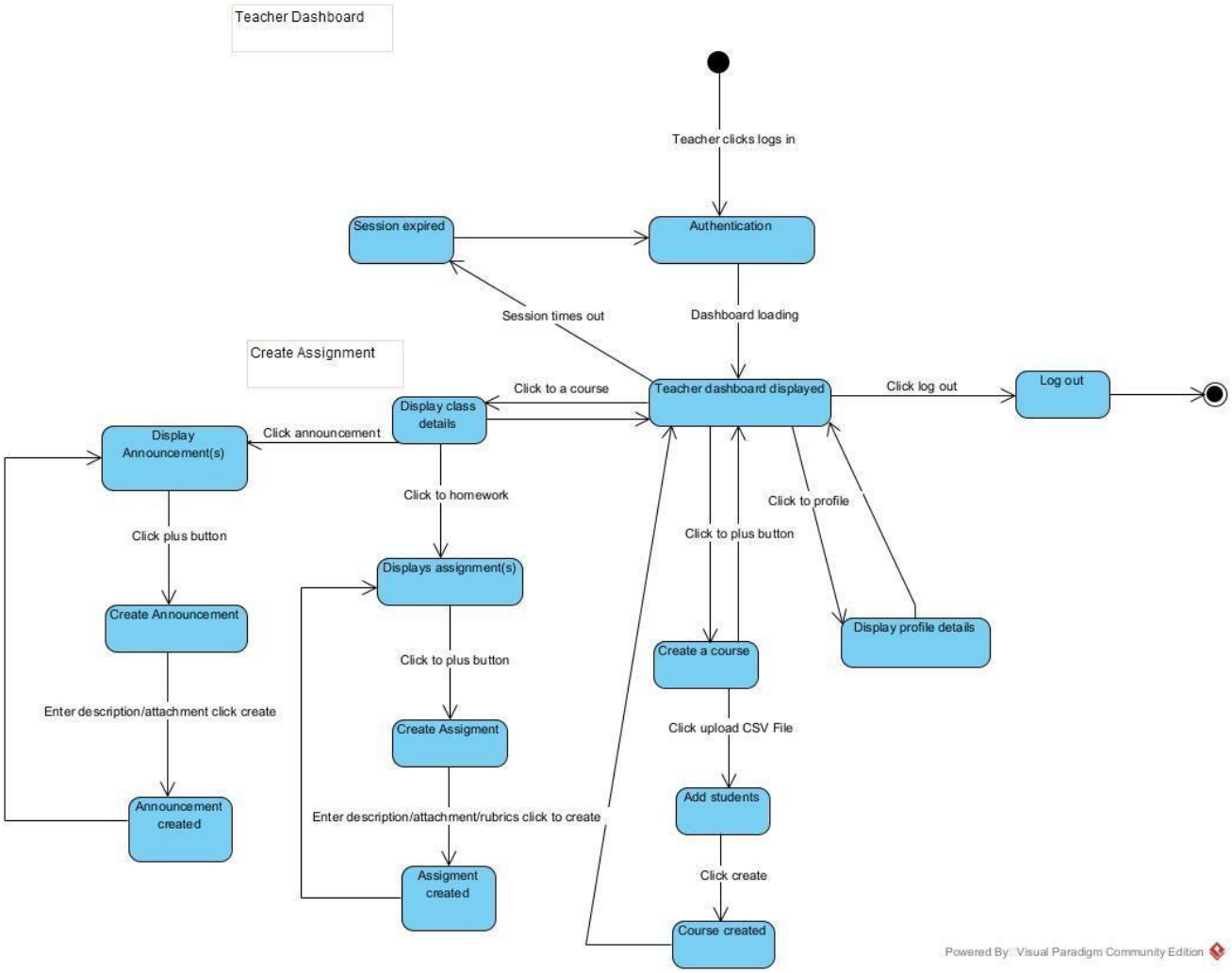

Figure 3

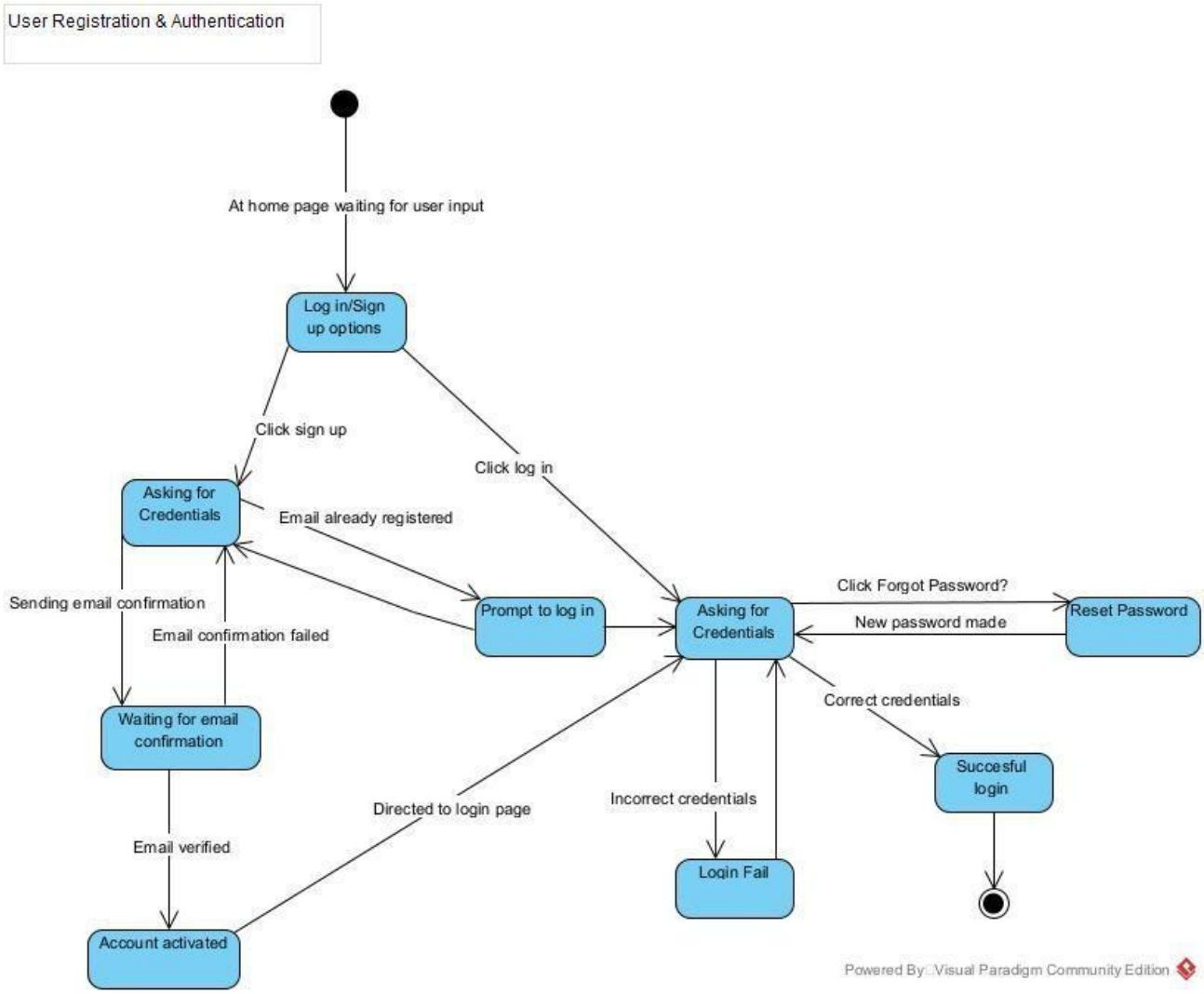
User Registration & Authentication
At home page waiting for user input
Log in/Sign up options
Click sign up
Click log in
Asking for Credentials
Email already registered
Sending email confirmation
Email confirmation failed
Prompt to log in
Asking for Credentials
Click Forgot Password?
New password made
Reset Password
Waiting for email confirmation
Correct credentials
Succesful login
Directed to login page
Incorrect credentials
Email verified
Login Fail
Account activated
Powered By Visual Paradigm Community Edition

Figure 4

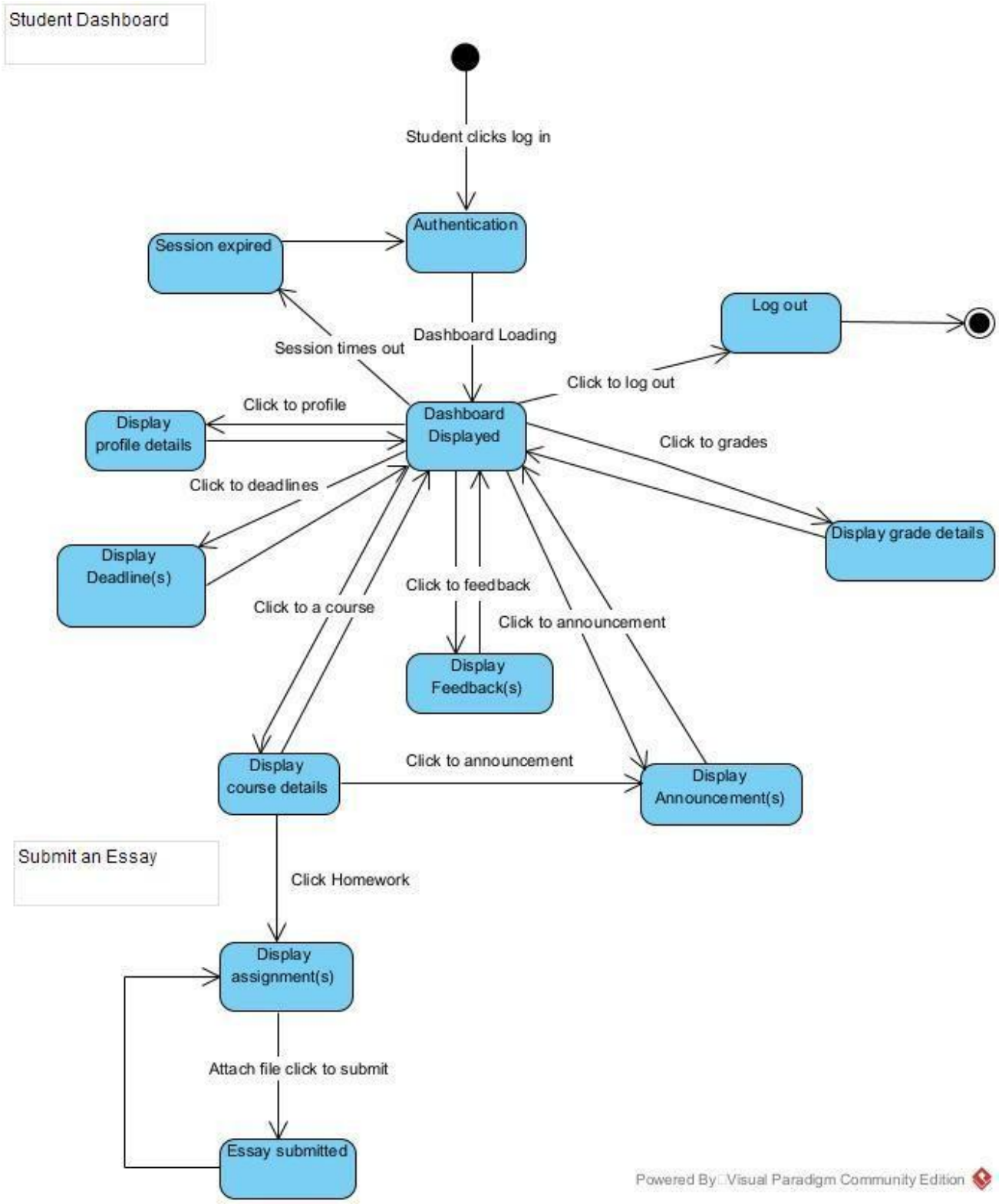
Student Dashboard
Student clicks log in
Authentication
Session expired
Session times out
Dashboard Loading
Log out
Click to log out
Click to profile
Display profile details
Dashboard Displayed
Click to grades
Click to deadlines
Display grade details
Display Deadline(s)
Click to feedback
Click to a course
Click to announcement
Display Feedback(s)
Display course details
Click to announcement
Display Announcement(s)
Submit an Essay
Click Homework
Display assignment(s)
Attach file click to submit
Essay submitted
Powered By Visual Paradigm Community Edition

Figure 5

3.4.4. User Interface—Navigational Paths and Screen Mock-ups

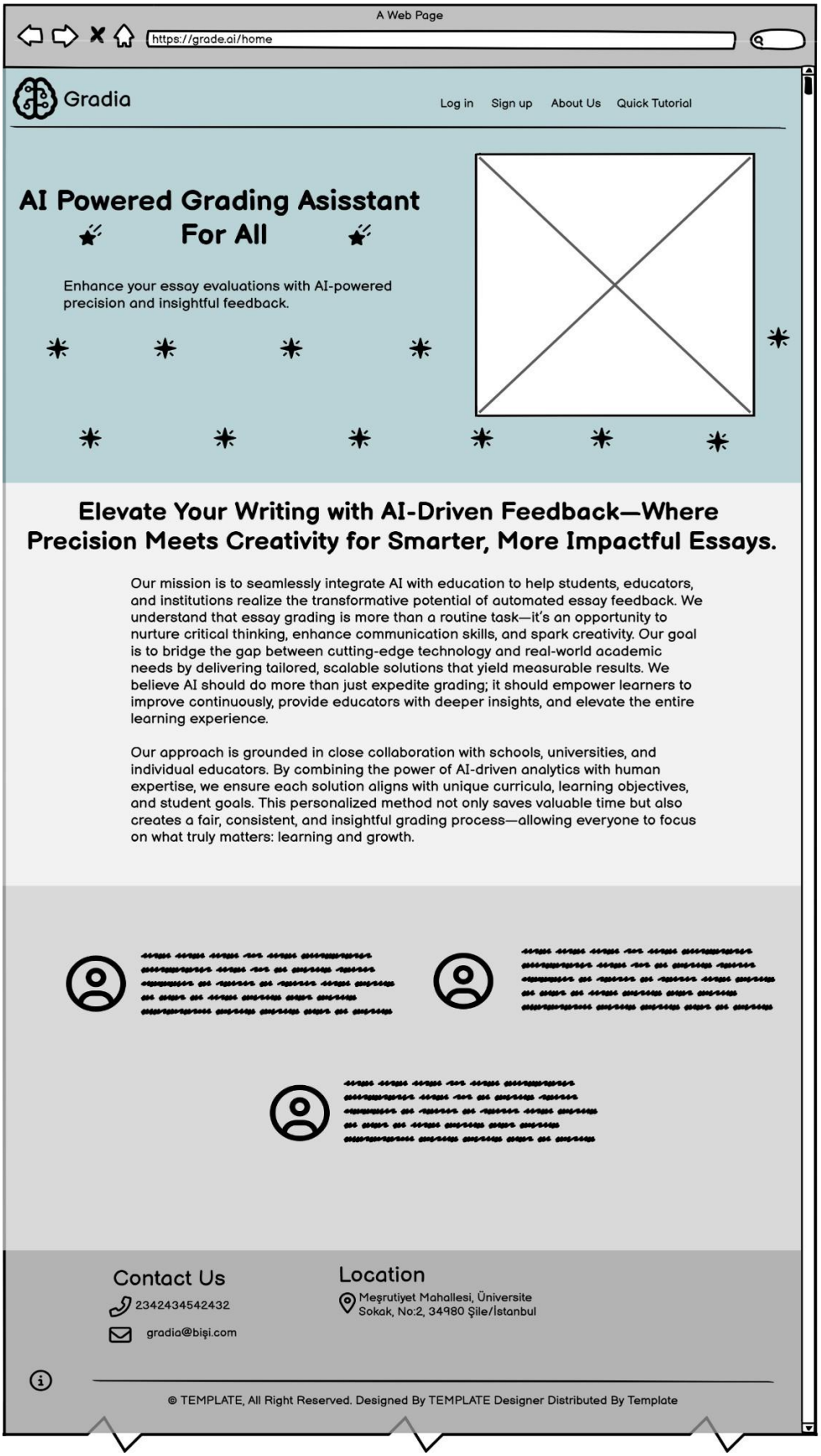

Figure 6

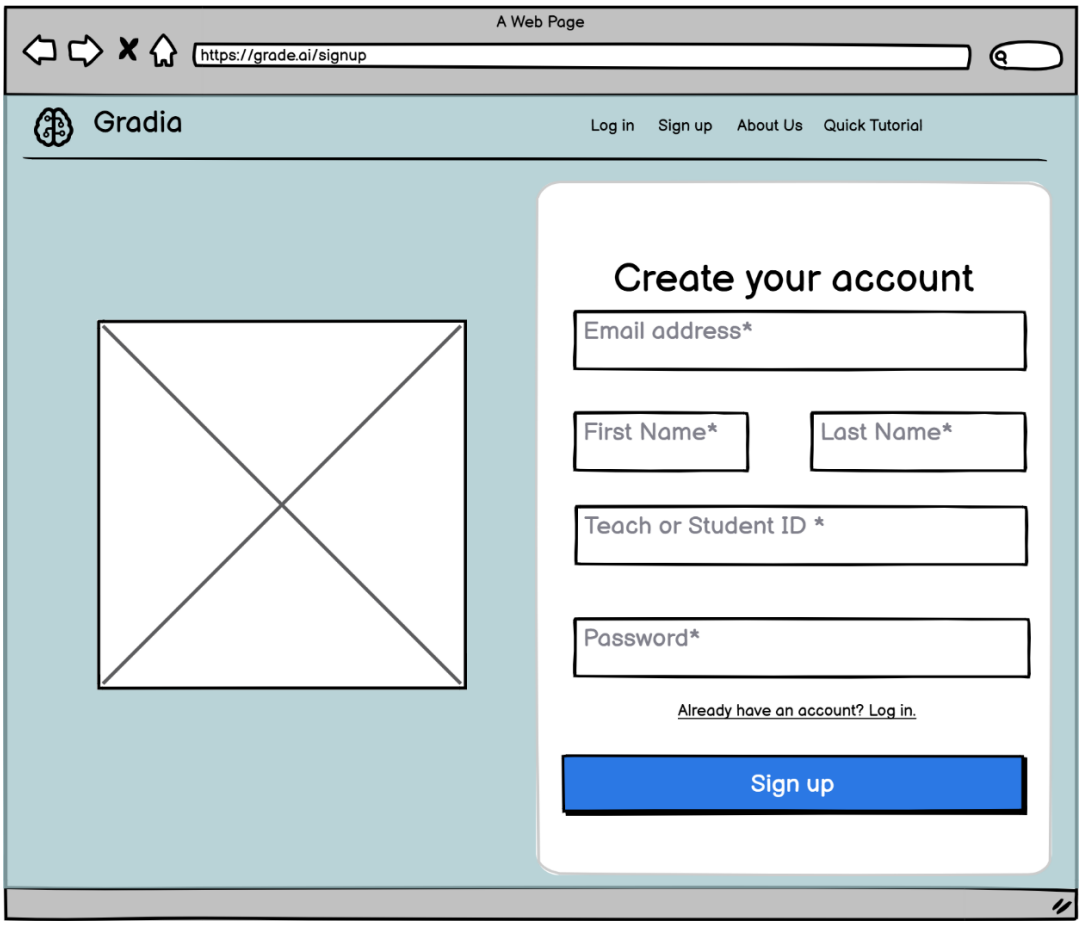

Figure 7

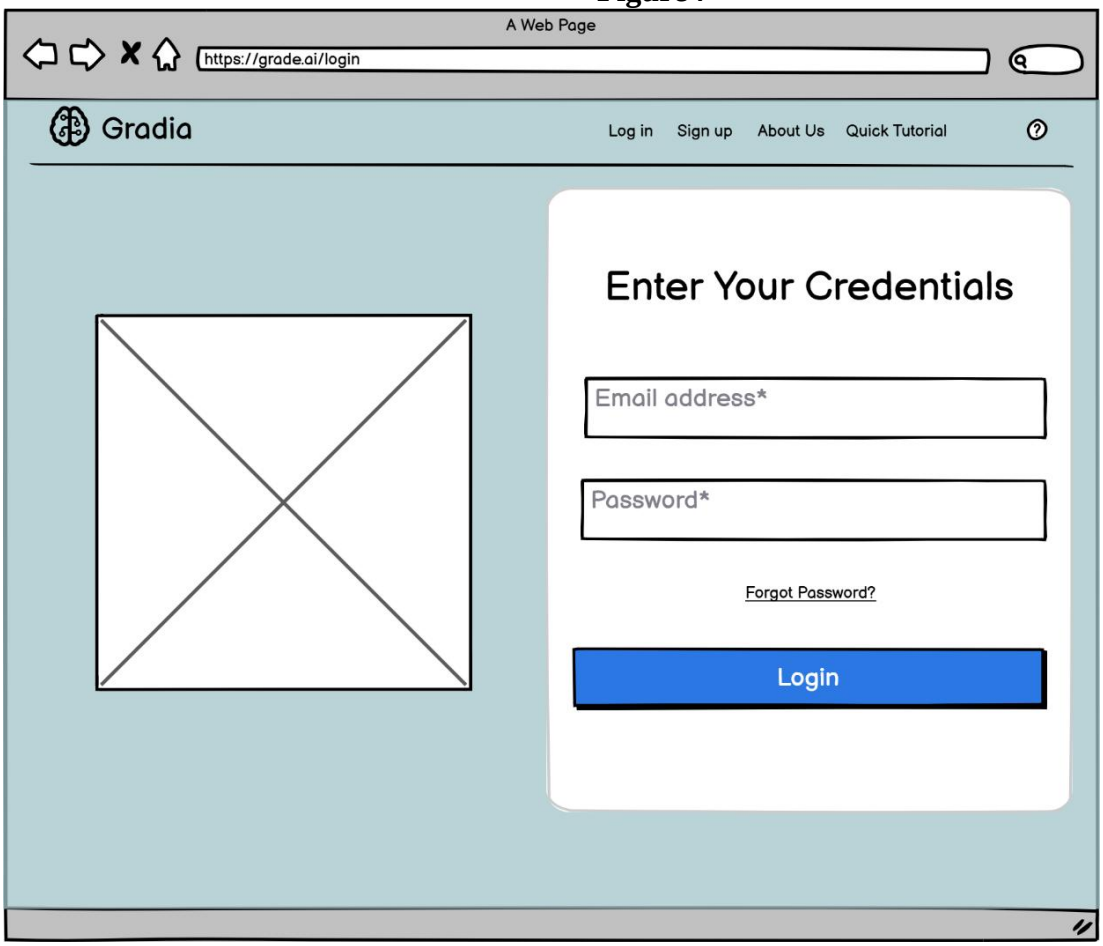

Figure 8

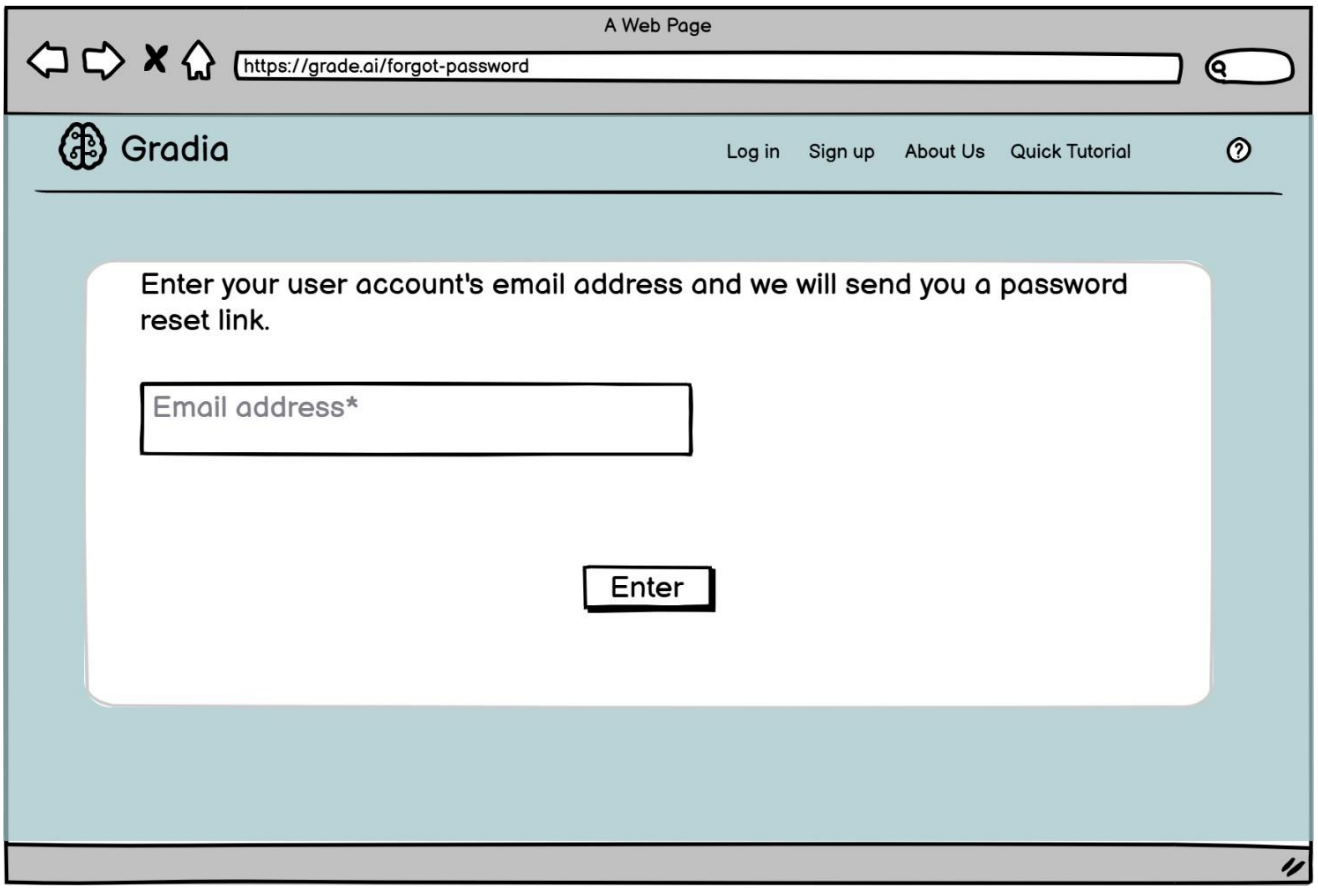

Figure 9

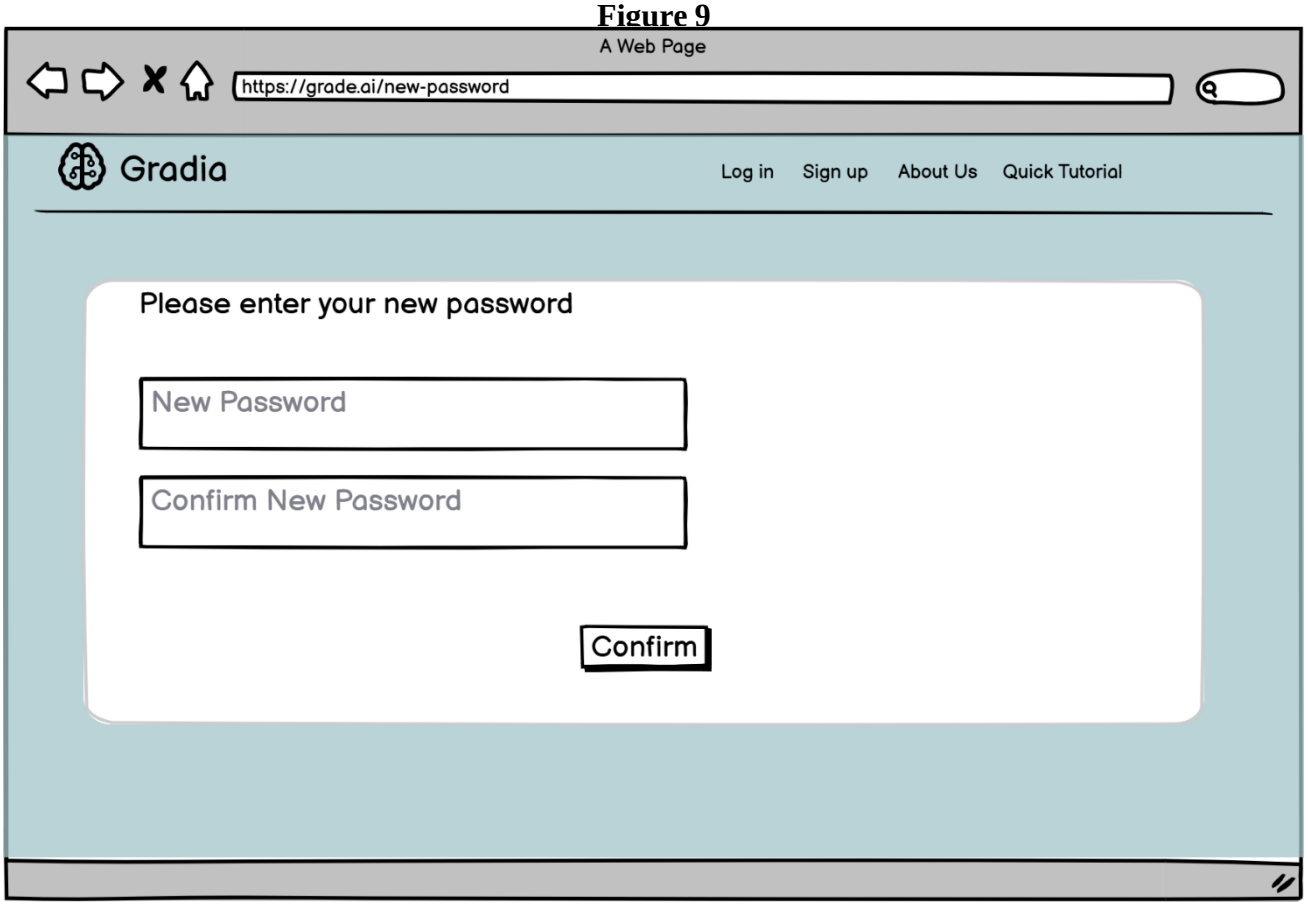

Figure 10

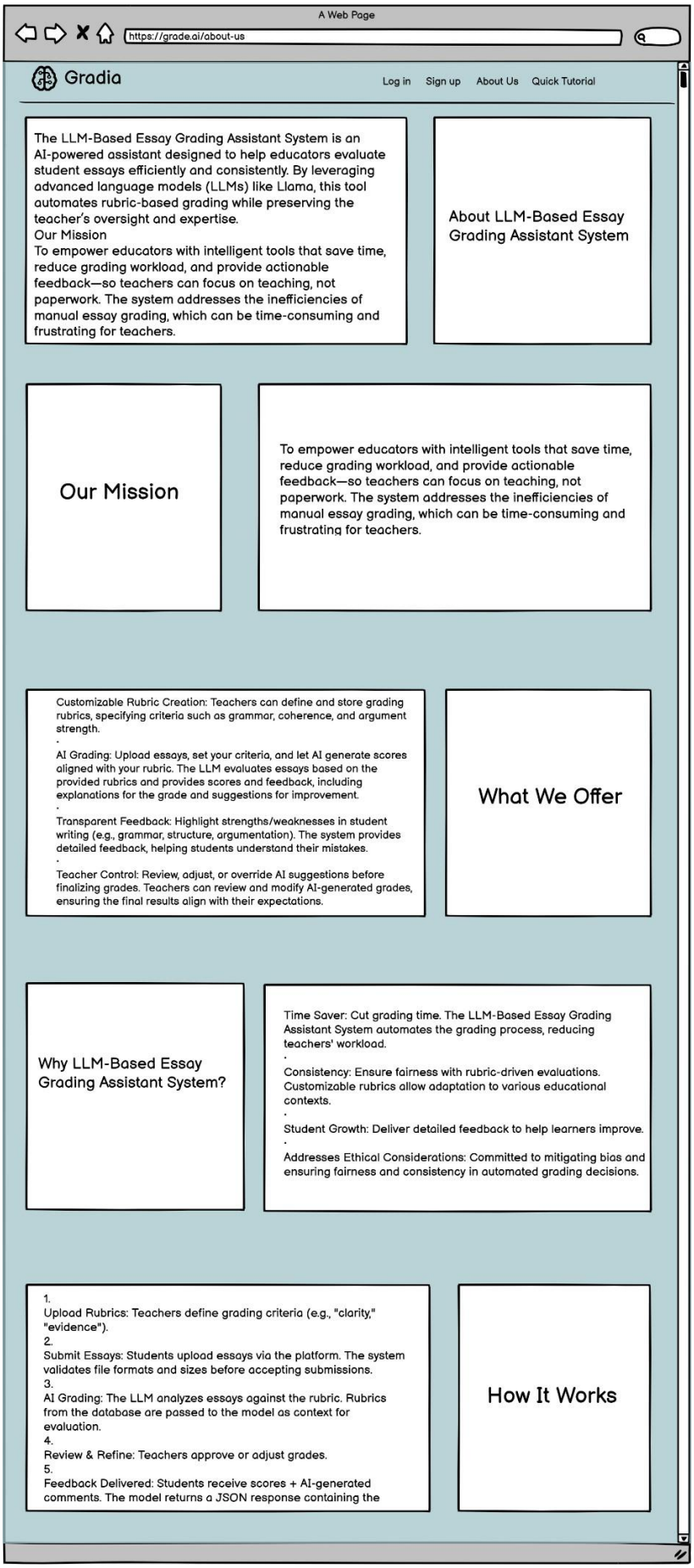
A Web Page
https://grade.ai/about-us
Gradia
Log in
Sign up
About Us
Quick Tutorial
The LLM-Based Essay Grading Assistant System is an AI-powered assistant designed to help educators evaluate student essays efficiently and consistently. By leveraging advanced language models (LLMs) like Llama, this tool automates rubric-based grading while preserving the teacher's oversight and expertise.
Our Mission
To empower educators with intelligent tools that save time, reduce grading workload, and provide actionable feedback—so teachers can focus on teaching, not paperwork. The system addresses the inefficiencies of manual essay grading, which can be time-consuming and frustrating for teachers.
About LLM-Based Essay Grading Assistant System
Our Mission
To empower educators with intelligent tools that save time, reduce grading workload, and provide actionable feedback—so teachers can focus on teaching, not paperwork. The system addresses the inefficiencies of manual essay grading, which can be time-consuming and frustrating for teachers.
Customizable Rubric Creation: Teachers can define and store grading rubrics, specifying criteria such as grammar, coherence, and argument strength.
AI Grading: Upload essays, set your criteria, and let AI generate scores aligned with your rubric. The LLM evaluates essays based on the provided rubrics and provides scores and feedback, including explanations for the grade and suggestions for improvement.
Transparent Feedback: Highlight strengths/weaknesses in student writing (e.g., grammar, structure, argumentation). The system provides detailed feedback, helping students understand their mistakes.
Teacher Control: Review, adjust, or override AI suggestions before finalizing grades. Teachers can review and modify AI-generated grades, ensuring the final results align with their expectations.
What We Offer
Why LLM-Based Essay Grading Assistant System?
Time Saver: Cut grading time. The LLM-Based Essay Grading Assistant System automates the grading process, reducing teachers' workload.
Consistency: Ensure fairness with rubric-driven evaluations. Customizable rubrics allow adaptation to various educational contexts.
Student Growth: Deliver detailed feedback to help learners improve.
Addresses Ethical Considerations: Committed to mitigating bias and ensuring fairness and consistency in automated grading decisions.
1.
Upload Rubrics: Teachers define grading criteria (e.g., "clarity," "evidence").
2.
Submit Essays: Students upload essays via the platform. The system validates file formats and sizes before accepting submissions.
3.
AI Grading: The LLM analyzes essays against the rubric. Rubrics from the database are passed to the model as context for evaluation.
4.
Review & Refine: Teachers approve or adjust grades.
5.
Feedback Delivered: Students receive scores + AI-generated comments. The model returns a JSON response containing the
How It Works

Figure 11

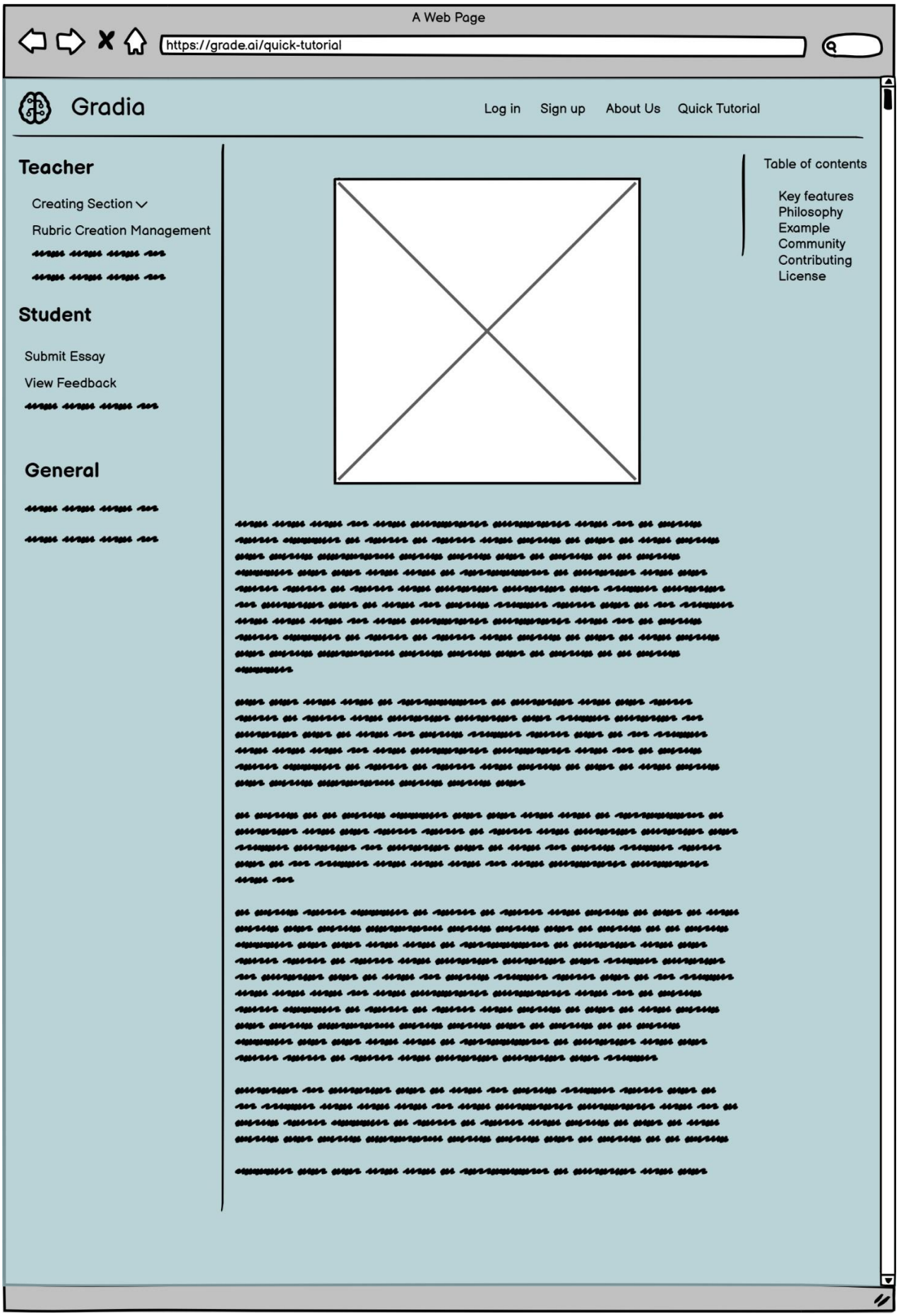
A Web Page
https://grade.ai/quick-tutorial
Gradia
Log in
Sign up
About Us
Quick Tutorial
Teacher
Creating Section
Rubric Creation Management
Student
Submit Essay
View Feedback
General
Table of contents
Key features
Philosophy
Example
Community
Contributing
License

Figure 12

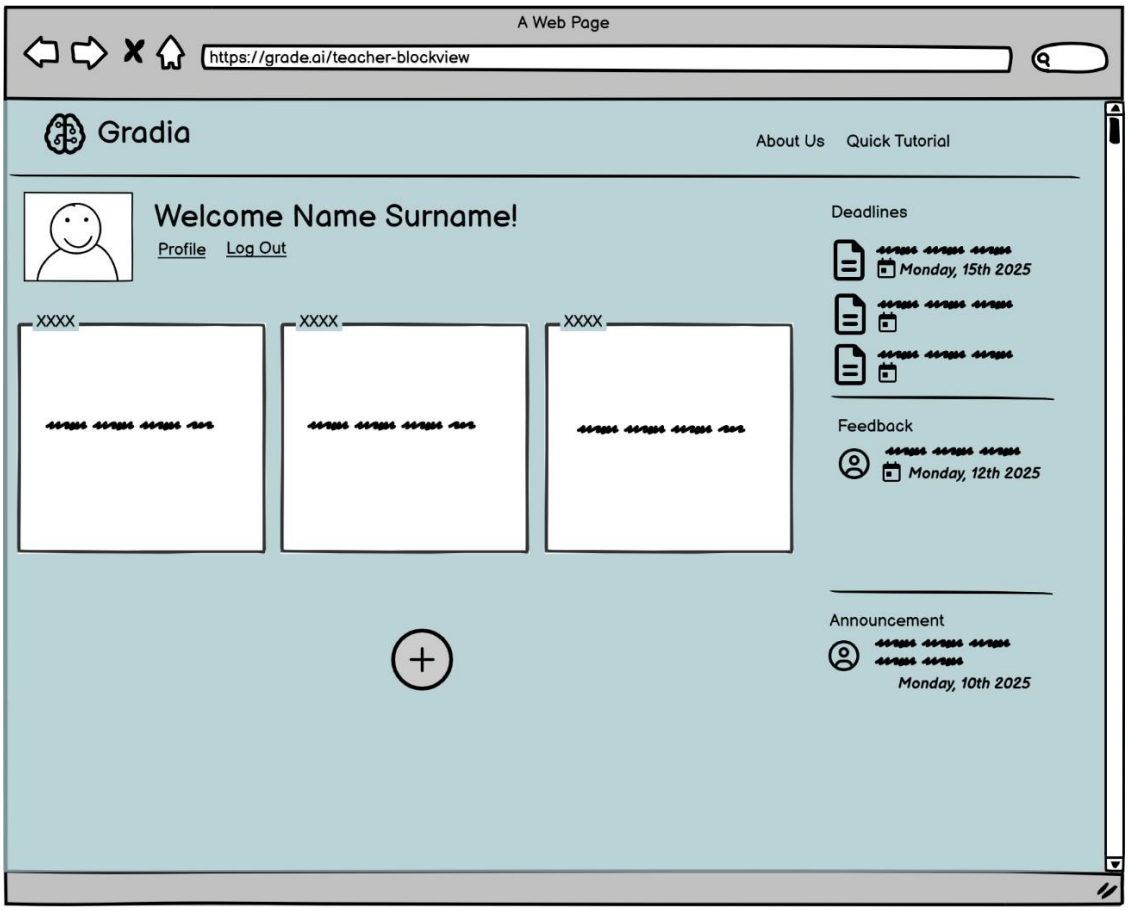

Figure 13

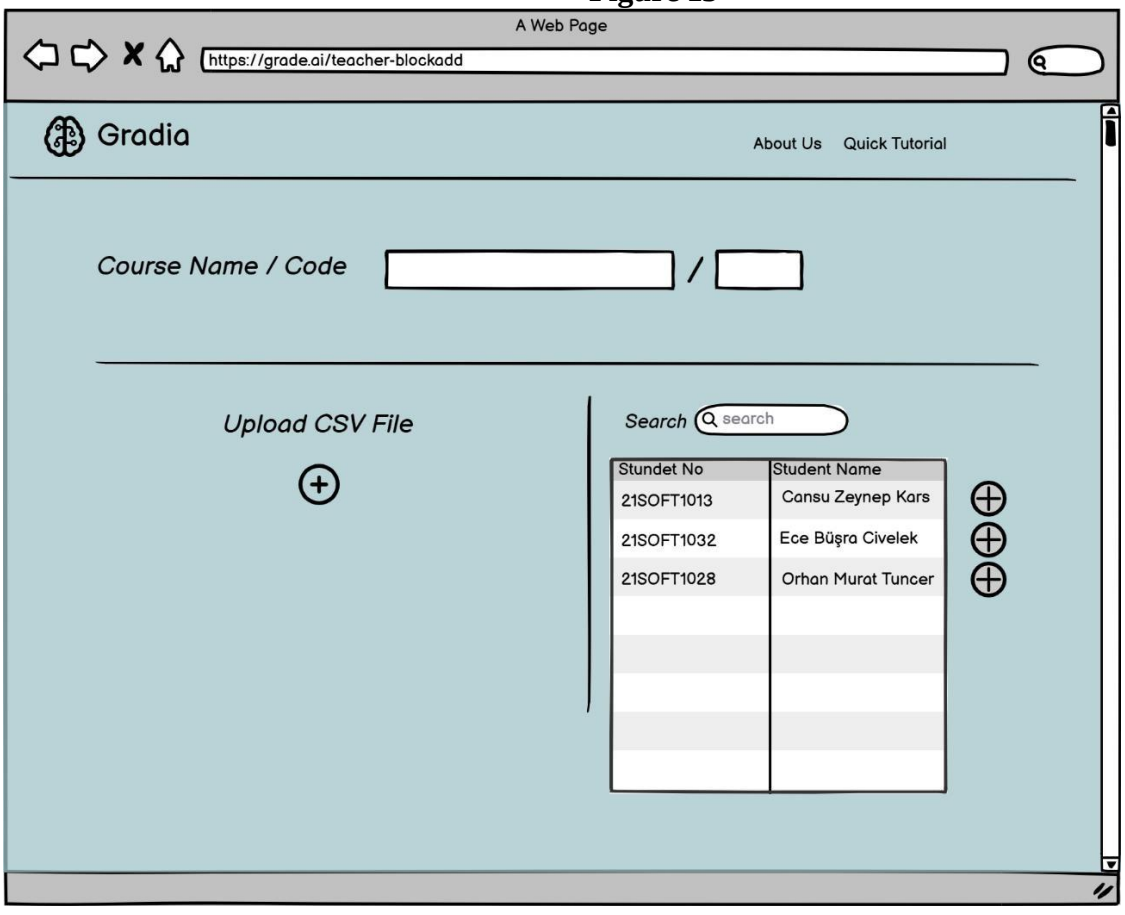

Figure 14

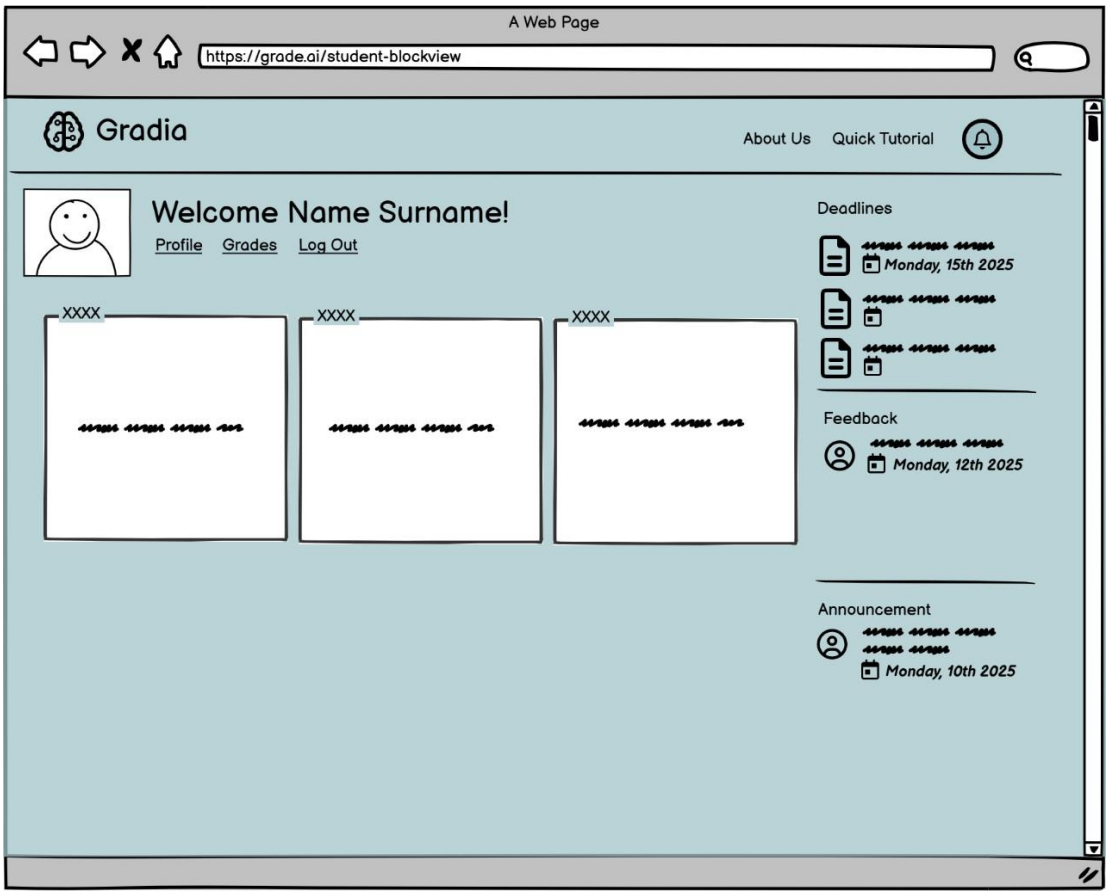

Figure 15

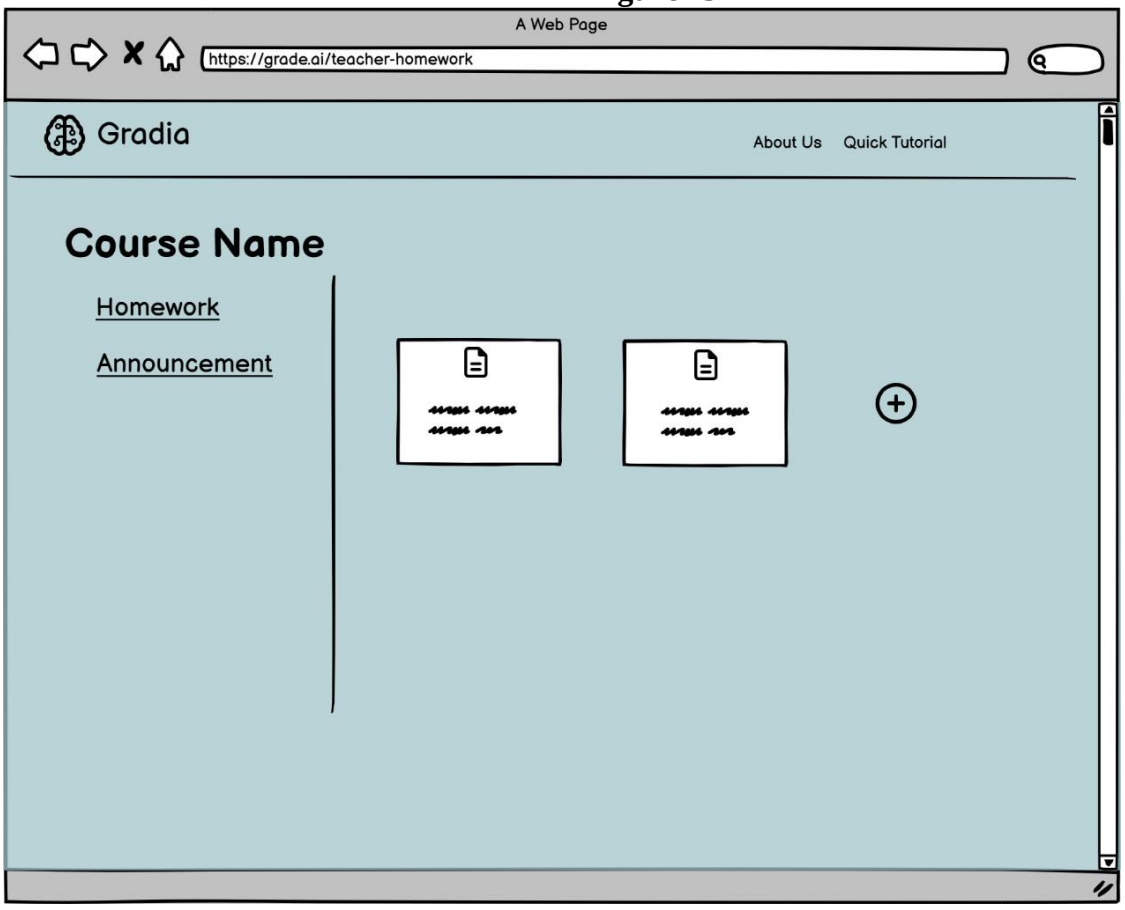

Figure 16

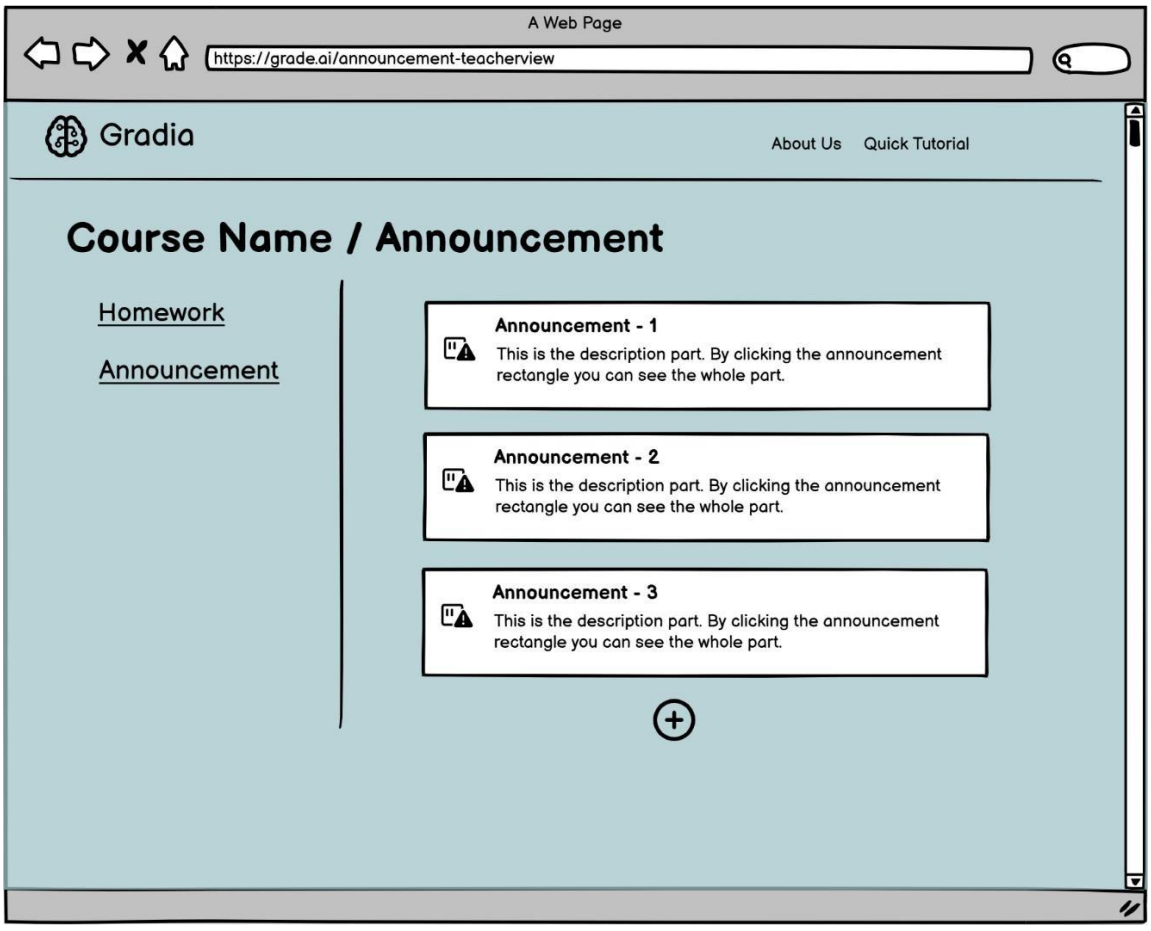

Figure 17

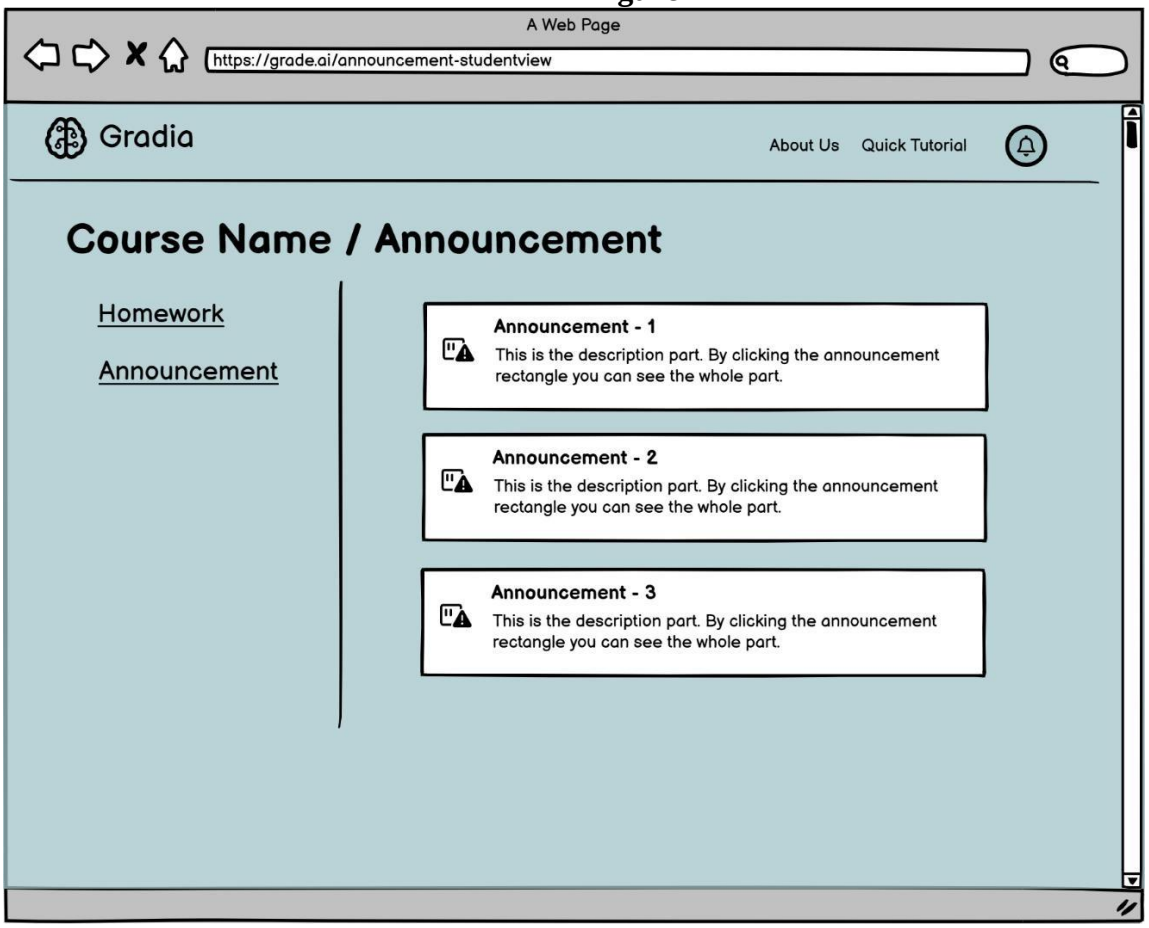

Figure 18

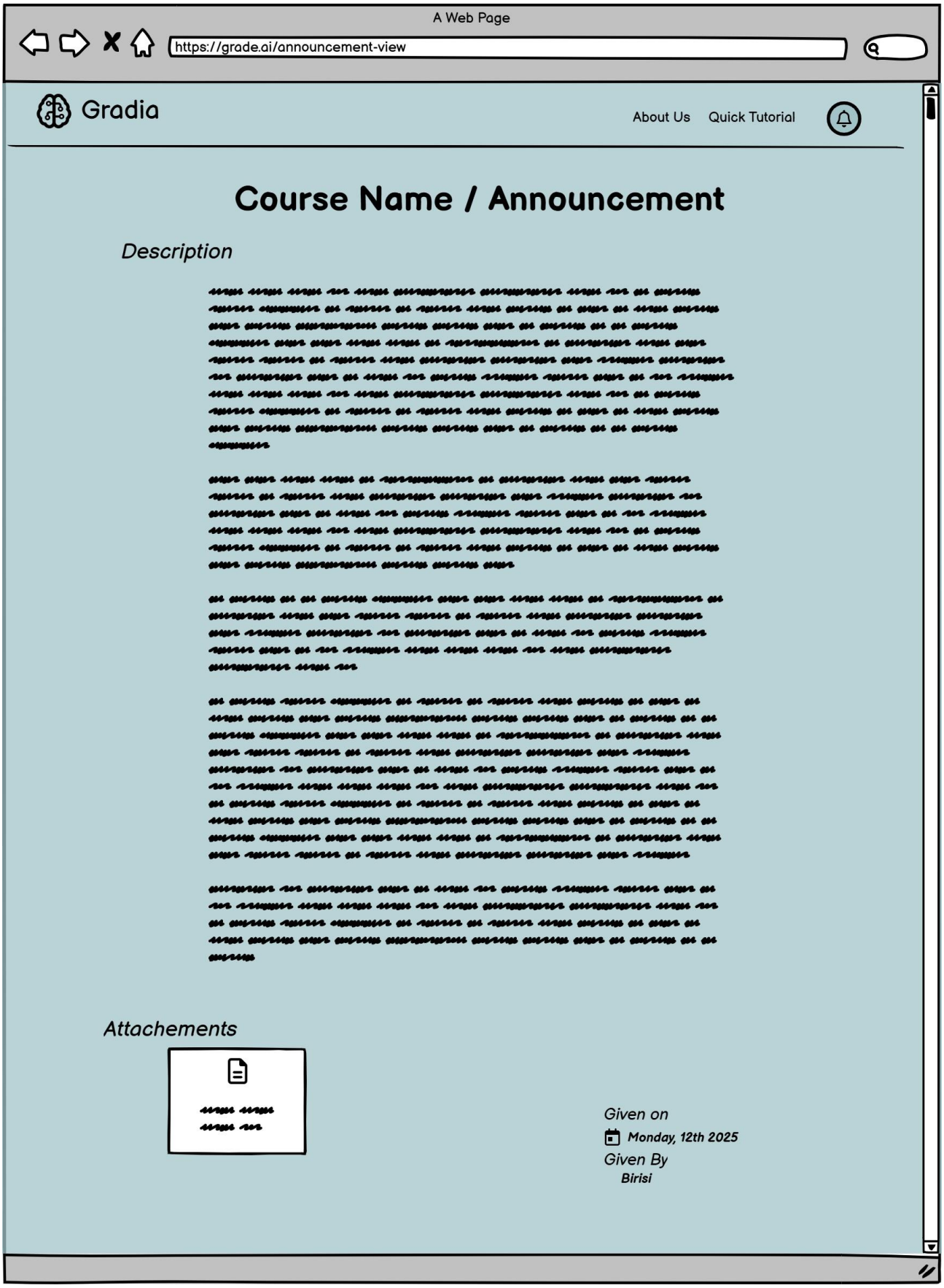
A Web Page
https://grade.ai/announcement-view
Gradia
About Us
Quick Tutorial
Course Name / Announcement
Description
Attachements
Given on
Monday, 12th 2025
Given By
Birisi

Figure 19

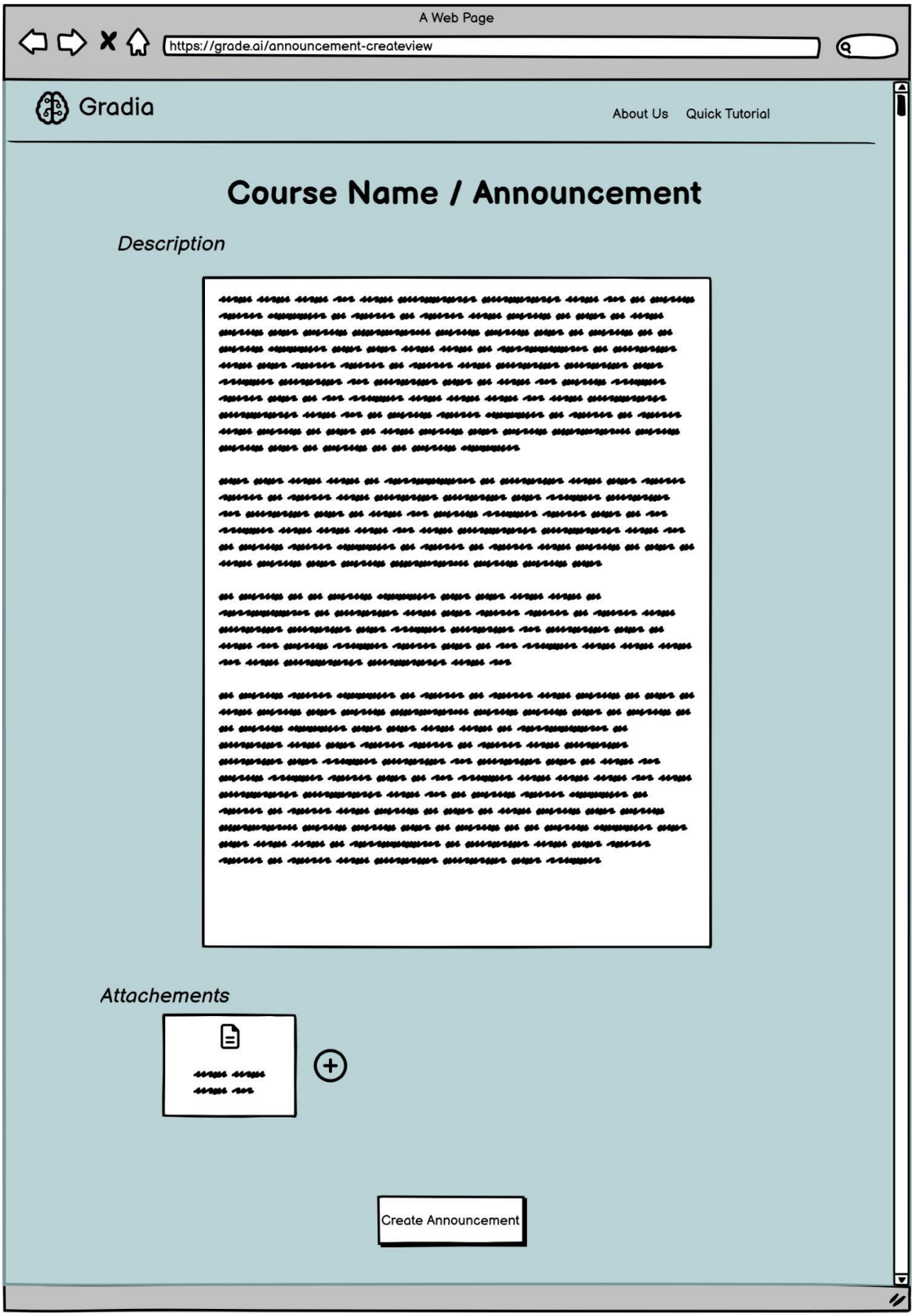
A Web Page
https://grade.ai/announcement-createview
Gradia
About Us
Quick Tutorial
Course Name / Announcement
Description
Attachements
Create Announcement

Figure 20

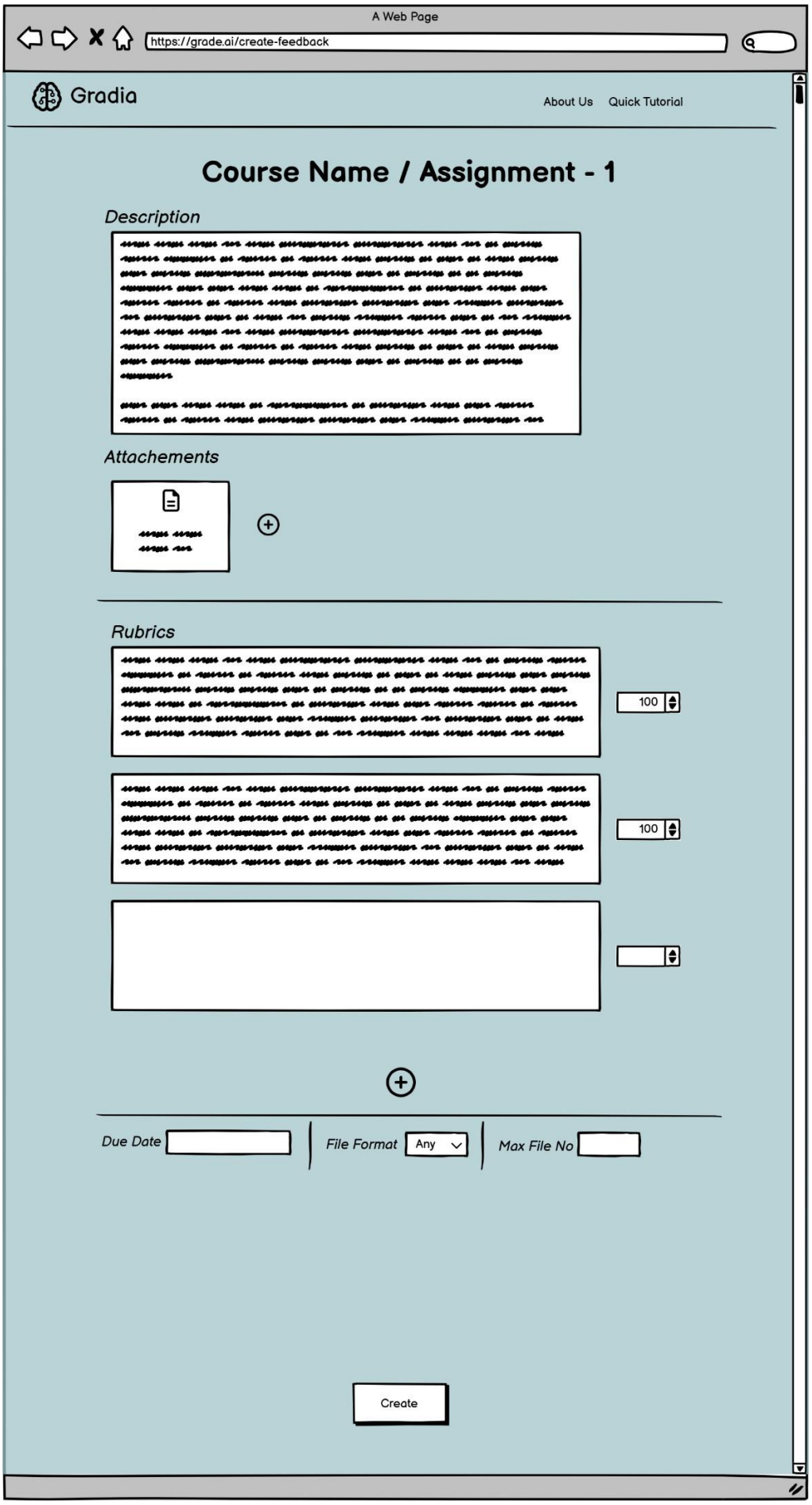
A Web Page
https://grade.ai/create-feedback
Gradia
About Us
Quick Tutorial
Course Name / Assignment - 1
Description
Attachements
Rubrics
100
100
Due Date
File Format
Any
Max File No
Create

Figure 21

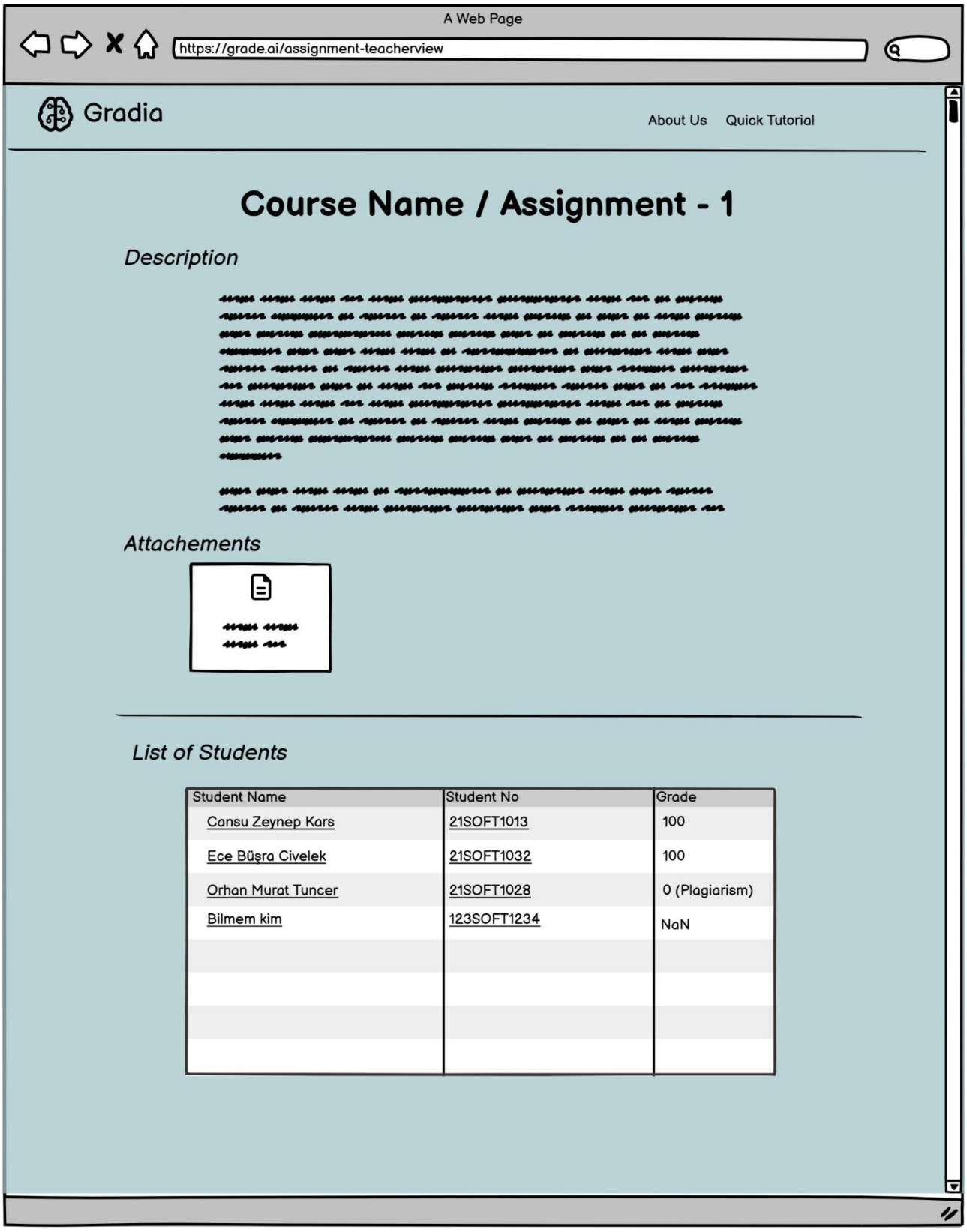
A Web Page
https://grade.ai/assignment-teacherview
Gradia
About Us
Quick Tutorial
Course Name / Assignment - 1
Description
Attachements
List of Students
Student Name
Student No
Grade
Cansu Zeynep Kars
21SOFT1013
100
Ece Büşra Civelek
21SOFT1032
100
Orhan Murat Tuncer
21SOFT1028
0 (Plagiarism)
Bilmem kim
123SOFT1234
NaN

Figure 22

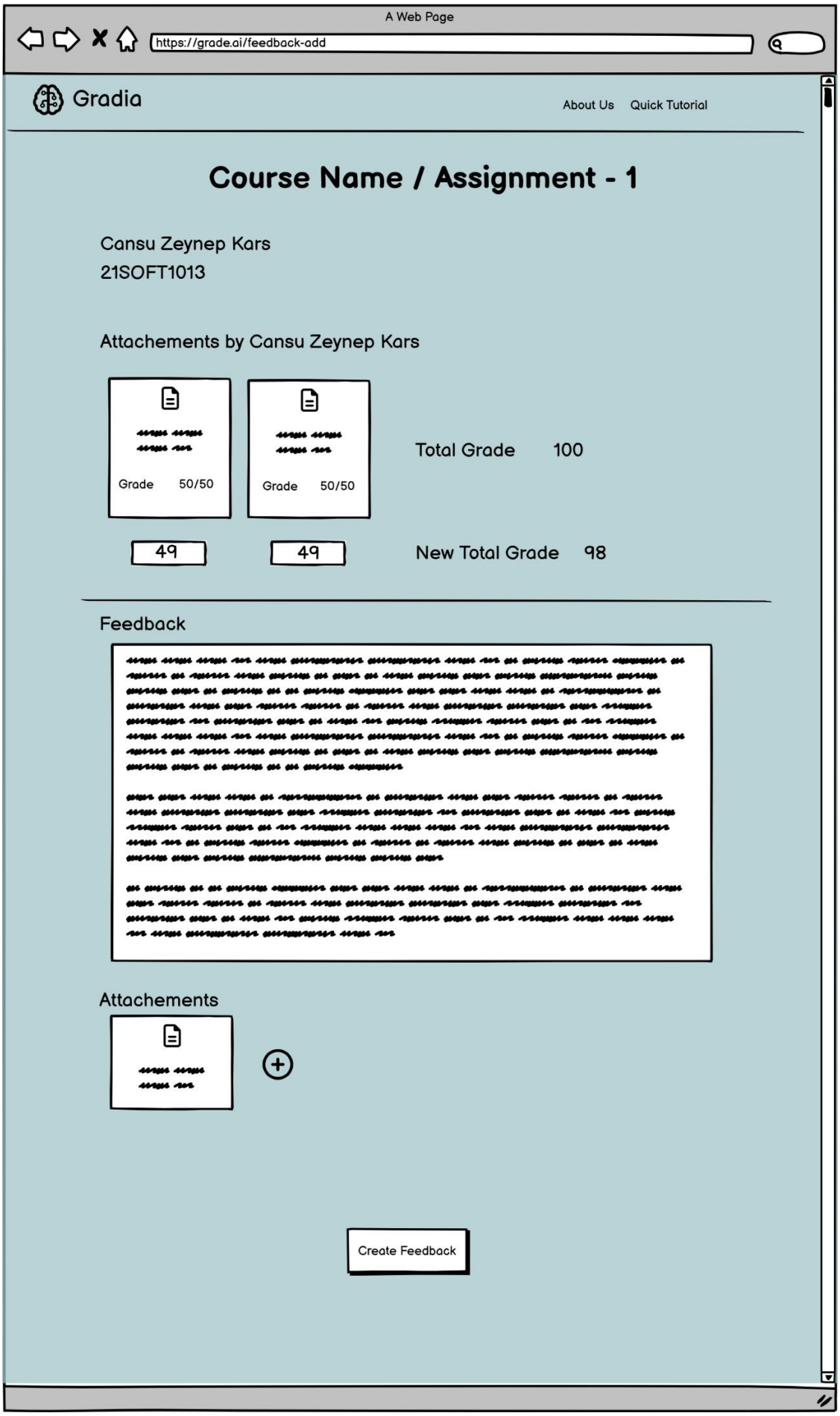
A Web Page
https://grade.ai/feedback-add
Gradia
About Us
Quick Tutorial
Course Name / Assignment - 1
Cansu Zeynep Kars
21SOFT1013
Attachements by Cansu Zeynep Kars
Grade 50/50
Grade 50/50
Total Grade 100
49
49
New Total Grade 98
Feedback
Attachements
Create Feedback

Figure 23

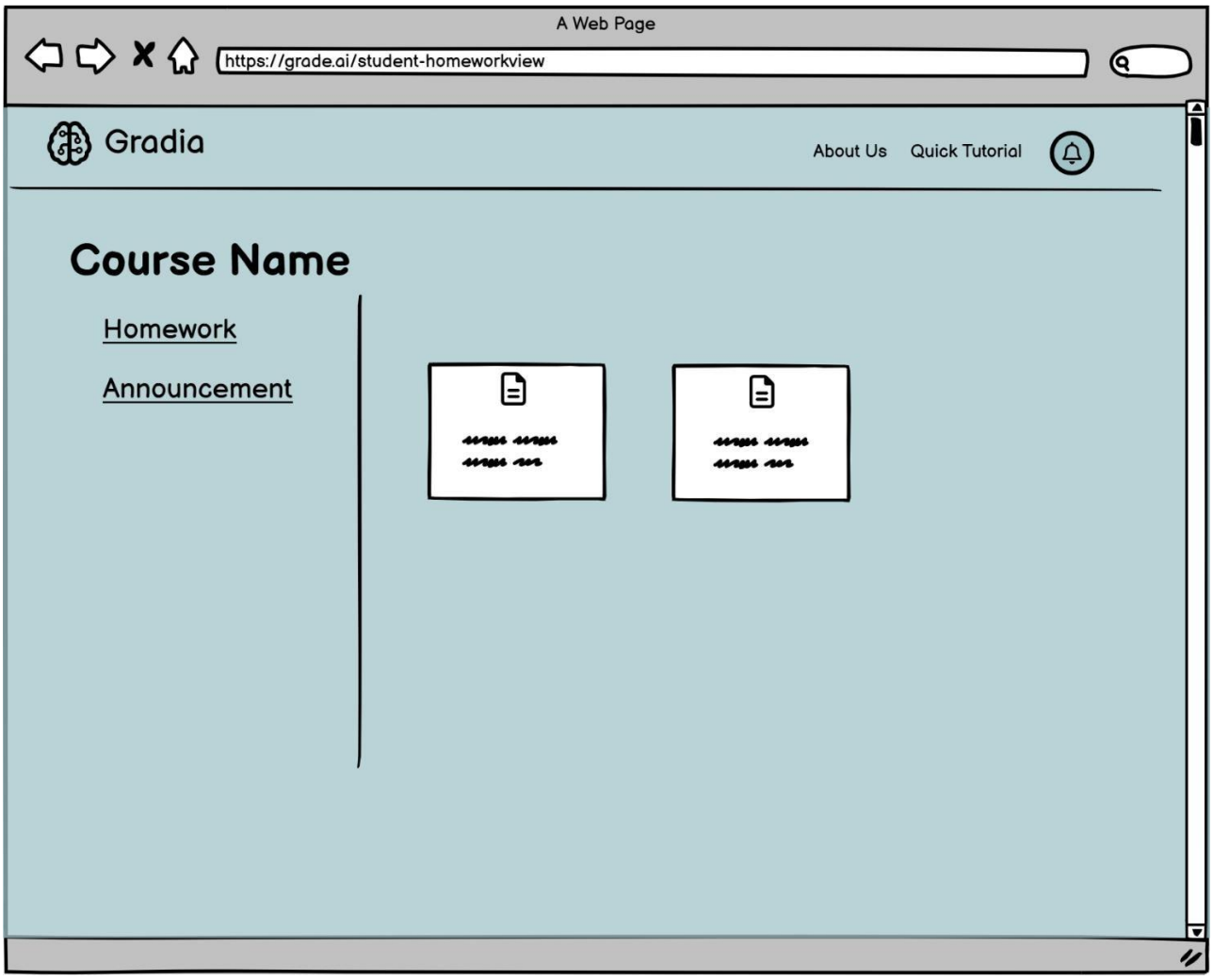
A Web Page
https://grade.ai/student-homeworkview
Gradia
About Us
Quick Tutorial
Course Name
Homework
Announcement

Figure 24

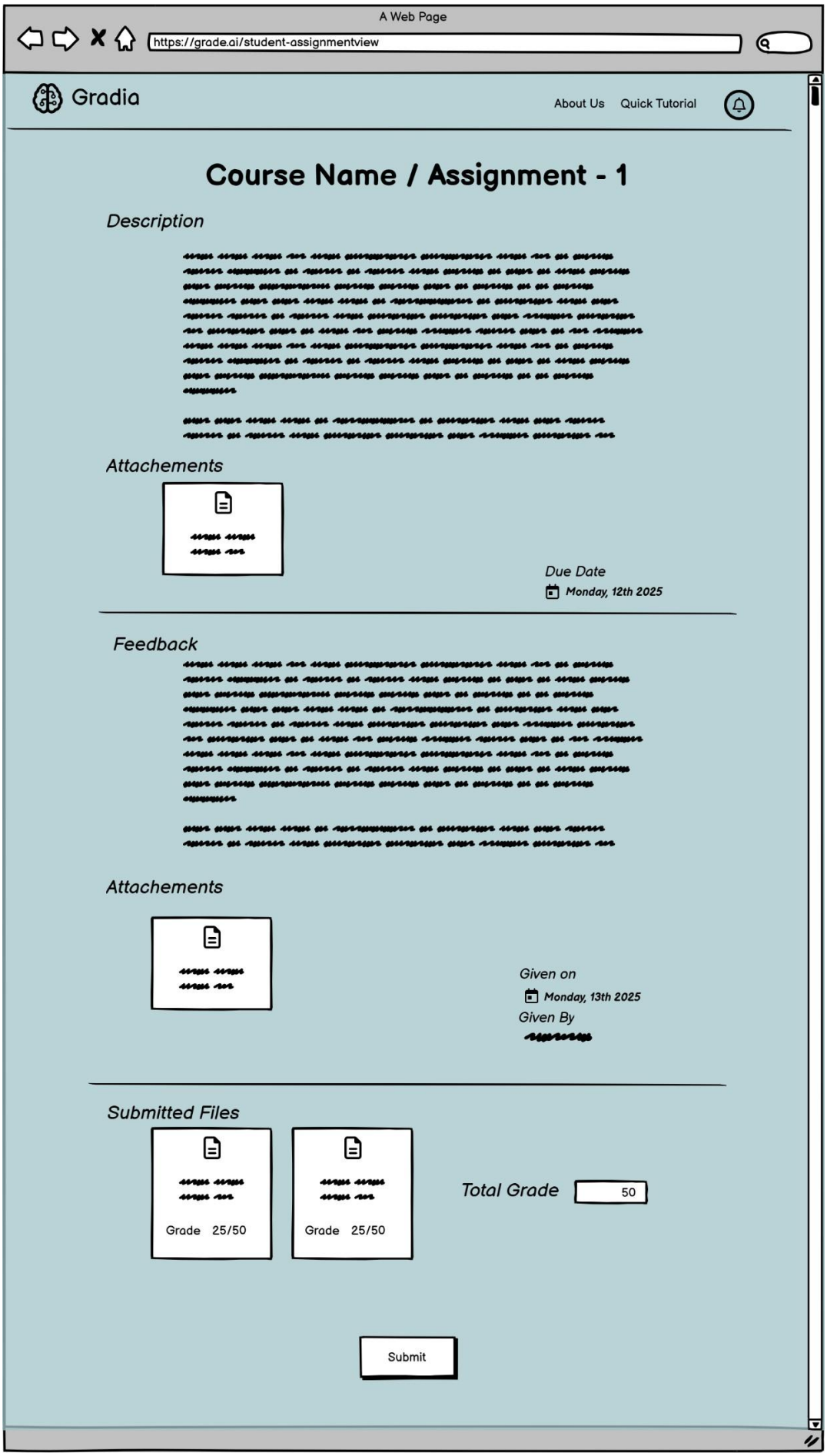
A Web Page
https://grade.ai/student-assignmentview
Gradia
About Us
Quick Tutorial
Course Name / Assignment - 1
Description
Attachements
Due Date
Monday, 12th 2025
Feedback
Attachements
Given on
Monday, 13th 2025
Given By
Submitted Files
Grade 25/50
Grade 25/50
Total Grade
50
Submit

Figure 25

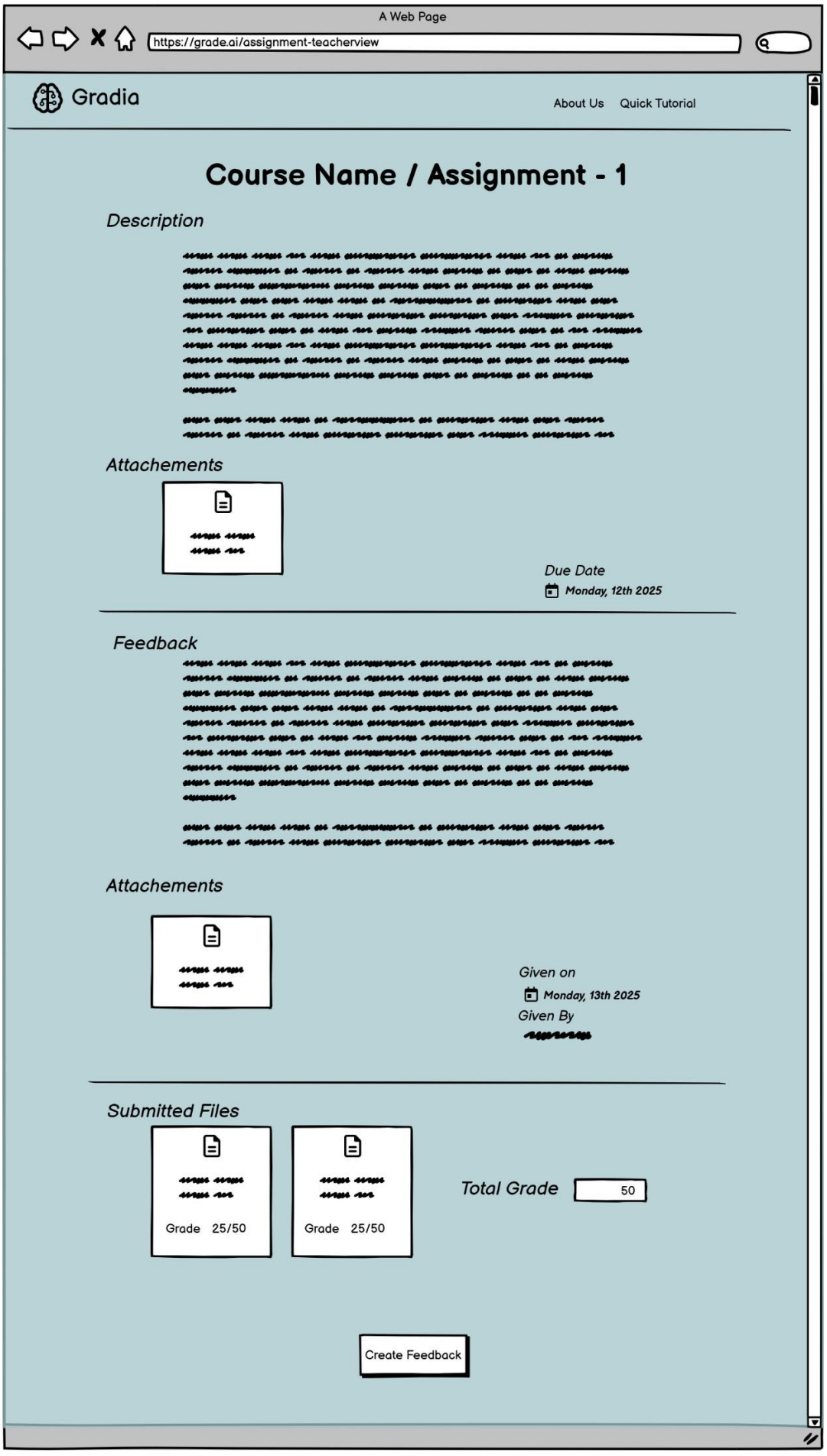
A Web Page
https://grade.ai/assignment-teacherview
Gradia
About Us
Quick Tutorial
Course Name / Assignment - 1
Description
Attachements
Due Date
Monday, 12th 2025
Feedback
Attachements
Given on
Monday, 13th 2025
Given By
Submitted Files
Grade 25/50
Grade 25/50
Total Grade
50
Create Feedback

Figure 26

Figure 27

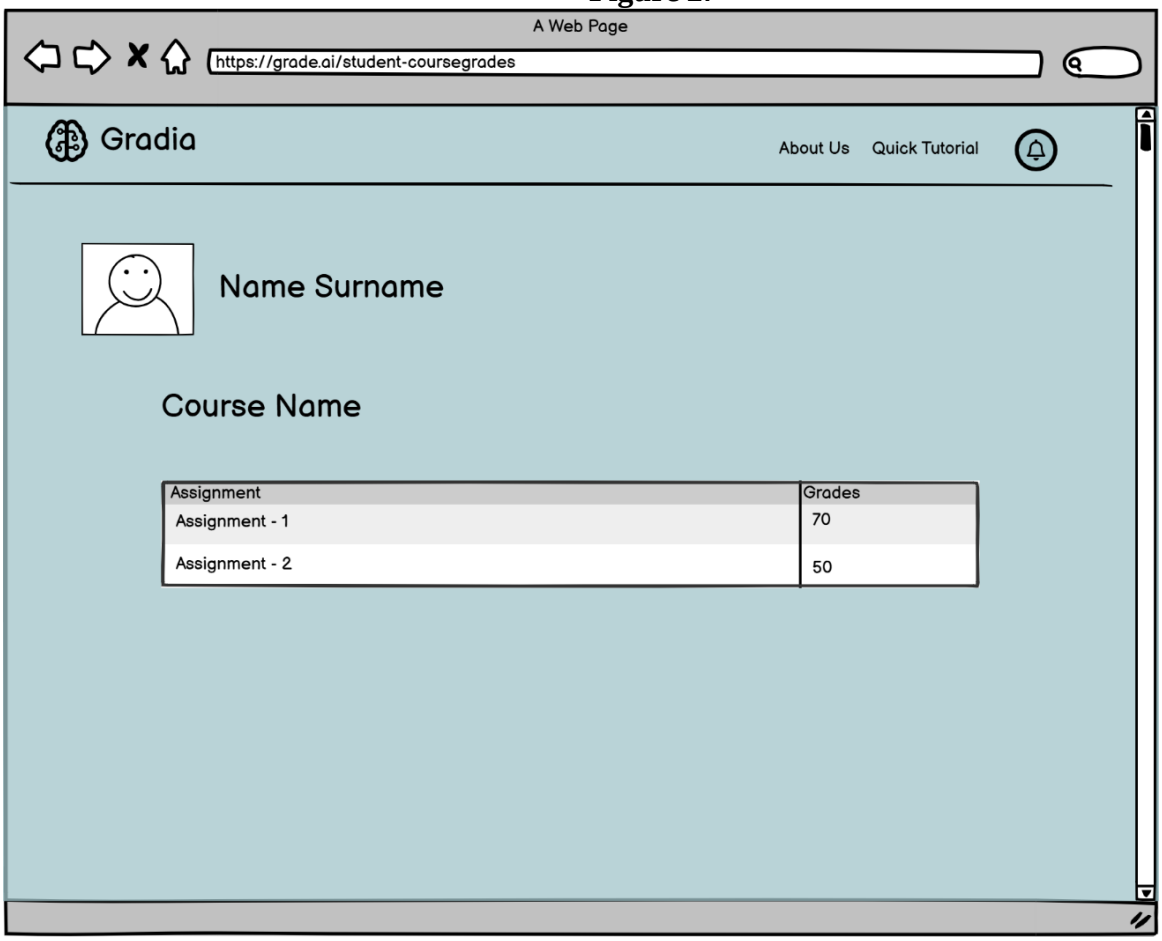

Figure 28

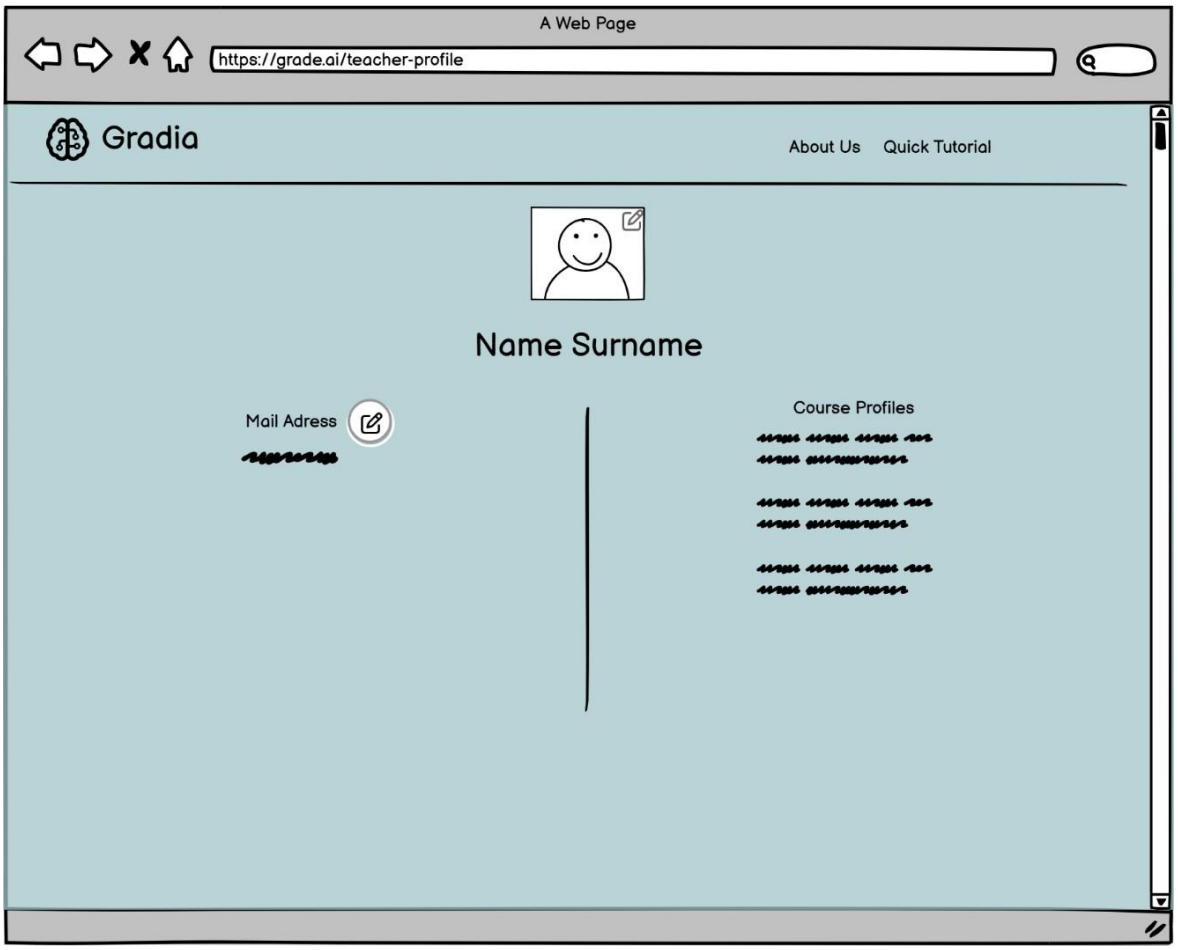

Figure 29

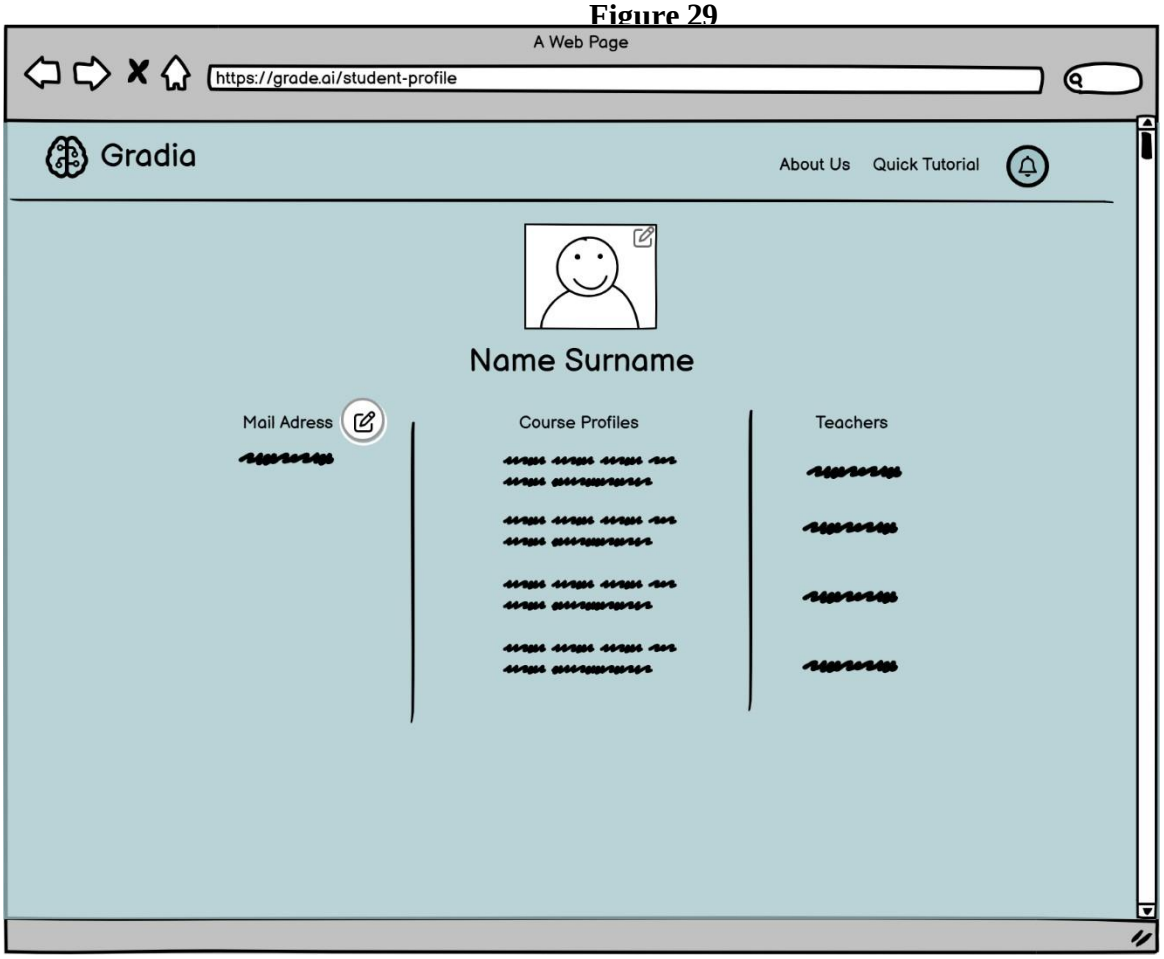

Figure 30

4. Other Analysis Elements

4.1. Risks and Alternatives

Table 19. Risks

	Likelihood	Effect on the project	B Plan Summary
LLM Model Fails to Perform	Medium	The system cannot grade essays accurately, leading to unreliable feedback for students.	Use a pre-trained LLM (e.g., GPT-3) as a fallback or implement manual grading as a temporary solution.
Database Performance Issues	Low	Slow query responses could delay grading and feedback delivery.	Optimize database queries and indexing. Use caching mechanisms to ensure seamless communication.
Frontend-Backend Integration Issues	Medium	Miscommunication between the frontend and backend could cause system crashes or data loss.	Conduct through integration testing and use API documentation to ensure seamless communication.
Data Privacy Breach	Low	Sensitive student data could be exposed, leading to legal and reputational consequences.	Implement robust encryption and comply with KVKK regulations.
Project Timeline Overruns	High	Delays in development could push the project past its deadline.	Prioritize critical tasks and allocate additional resources to high-priority work packages.

4.2. Project Plan

	A	B	C	D	E	F	G	H	I	J	K
1					Feb 17-21	Feb 24-28	Mar 3-14	Mar 17-28	Apr 7-May 16	May 12-23	May 19-30
2	1. Form team, assign supervisor, and discuss project scope				Team Collaboration						
3	2. Finalize project proposal, including Gantt Chart					Team Collaboration					
4	3. Gather and analyze requirements. Prepare RAD document						Team Collaboration				
5	4. Design system architecture and prepare SDD document							Team Collaboration			
6	5. User Interface (UI) and Client-Side Application Development								Team Collaboration		
7	5.1. Registration & Authenticaiton Interface Implementation								Team Collaboration		
8	5.2. Student Interface Implementation(Dashboard, grades, course...)								Team Collaboration		
9	5.3 Teacher Interface Implementation(Dashboard, rubric management...)								Team Collaboration		
10	6. Back-end and API Development								Team Collaboration		
11	6.1. Flask Server Setup and Configuration								Team Collaboration		
12	6.2. API Developments (Authentication, Rubrics, Grades...etc)								Team Collaboration		
13	7. Develop and integrate a database								Team Collaboration		
14	8. Integrate LLM model								Team Collaboration		
15	8.1. LLM Fine-tuning								Team Collaboration		
16	9. Test system, fix bugs, and prepare for demo									Team Collaboration	
17	9.1. Unit Testing of Individual Models									Team Collaboration	
18	9.2. User Acceptance Testing									Team Collaboration	
19	9.3. Performance Testing									Team Collaboration	
20	10. Finalize thesis document and submit source codes										Team Collaboration
21	10.1. Deployment of Web Application on a Local Server										Team Collaboration

Figure 31

Table 20. List of work packages

WP#	Work package title	Members involved
WP1	Project Initiation & Proposal	All members
WP2	Requirements Gathering & Analysis	All members

WP3	System Design	Ece, Cansu (UI); Orhan, Cansu (Backend); All
WP4	Frontend/UI Development	Ece, Cansu
WP5	Backend/API & Database Development	All members
WP6	LLM Integration	All members
WP7	Testing & Bug Fixing	All members
WP8	Finalization & Documentation	All members

Table 21. Work Package 1

WP 1: *Project Initiation & Proposal*			
Start date: *17.02.2025* **End date:** *28.02.2025*			
Leader:	*Ece Büşra Civelek*	**Members involved:**	*Cansu Zeynep Kars, Orhan Murat Tuncer*
Objectives: The objective of this work package is to clearly define the project scope, ensuring all team members understand the system's goals and limitations. Roles and responsibilities will be assigned based on each member's expertise to optimize workflow and efficiency. A project proposal and Gantt chart will be drafted to outline key deliverables, milestones, and timelines for tracking progress.			
Tasks: **Task 1.1 Team Formation & Supervisor Assignment:** Establish team roles and responsibilities, align with the supervisor on project expectations, and finalize communication protocols. **Task 1.2 Draft Project Proposal:** Define project objectives, scope, tools, and success criteria to create a foundational document for stakeholder approval. **Task 1.3 Gantt Chart Creation:** Develop a visual timeline for task distribution, milestones, and deadlines to ensure project alignment and accountability.			
Deliverables **D1.1:** Approved project proposal. **D1.2:** Finalized Gantt chart.			

Table 22. Work Package 2

WP 2: *Requirements Gathering & Analysis*			
Start date: *03.03.2025* **End date:** *14.03.2025*			
Leader:	*Cansu Zeynep Kars*	**Members involved:**	*Ece Büşra Civelek, Orhan Murat Tuncer*
Objectives: The objective of this work package is to gather and document both user and system requirements to ensure the project meets stakeholder expectations. This involves conducting research, analyzing functional and non-functional needs, and refining system specifications. The collected information will be compiled into a structured Requirements Analysis Document (RAD) to serve as a reference throughout the project.			

Tasks:
Task 2.1 Stakeholder Interviews: Conduct interviews with teachers and students to identify pain points in manual grading and expectations for the AI system.
Task 2.2 Functional & Non-Functional Requirements Definition: Translate stakeholder needs into technical requirements (e.g., rubric customization, real-time grading) and system constraints.
Task 2.3 RAD Document Preparation: Formalize requirements into a structured document with use cases, scenarios, and system models for developer reference.

Deliverables **D2.1:** Completed RAD document.

Table 23. Work Package 3

WP 3: *System Design*			
Start date: *17.03.2025* **End date:** *28.03.2025*			
Leader:	*Orhan Murat Tuncer*	**Members involved:**	*Ece Büşra Civelek, Cansu Zeynep Kars*
Objectives: The objective of this work package is to design a scalable and efficient system architecture that defines the overall structure and interactions between system components. This includes developing a well-structured database schema to ensure data integrity, optimize performance, and support all required functionalities. Additionally, APIs will be designed to facilitate seamless communication between the frontend, backend, and external services.			
Tasks: **Task 3.1 UML Diagram Creation:** Develop use case, class, and sequence diagrams to visualize system interactions and data flow between components. **Task 3.2 Database Schema Design:** Define tables for users, rubrics, essays, and grades, ensuring normalization and scalability.			
Deliverables **D3.1:** Software Design Document (SDD). **D3.2:** Database ER diagram.			

Table 24. Work Package 4

WP 4: *Frontend/UI Development*			
Start date: 07.04.2025 **End date:** *16.05.2025*			
Leader:	*Ece Büşra Civelek*	**Members involved:**	*Cansu Zeynep Kars*
Objectives: The objective of this work package is to develop a responsive and user-friendly interface tailored for both teachers and students. The UI will be designed to ensure seamless navigation, accessibility, and compatibility across various devices and screen sizes. Emphasis will be placed on creating an intuitive layout that enhances user experience and efficiently supports key functionalities.			
Tasks: **Task 4.1 Login & Registration Interface:** Implement secure authentication pages using Bootstrap to ensure user-friendly access control. **Task 4.2 Rubric Management Dashboard:** Build an interface for teachers to create, edit, and delete grading rubrics and have an option of LLM generating rubrics with dynamic form inputs. **Task 4.3 Grade & Feedback Display Module:** Design dashboards for students to view grades/feedback and for teachers to review AI-generated scores.			
Deliverables **D4.1:** Functional frontend modules (HTML/CSS/JS).			

Table 25. Work Package 5

WP 5: *Backend/API & Database Development*			
Start date: 07.04.2025 **End date:** *16.05.2025*			
Leader:	*Orhan Murat Tuncer*	**Members involved:**	*Cansu Zeynep Kars, Ece Büşra Civelek*
Objectives: The objective of this work package is to build robust backend logic that supports the system's core functionalities. APIs will be developed to facilitate secure and efficient communication between the client interface and backend services. In addition, a MySQL database will be implemented to provide reliable data storage and optimized performance.			
Tasks: **Task 5.1 Flask Server Setup:** Configure Flask for routing, middleware, and MySQL integration to handle user requests. **Task 5.2 CRUD API Development:** Create APIs for rubric creation, essay submission, and grade retrieval to enable frontend functionality. **Task 5.3 Query Optimization:** Index database tables and optimize SQL queries to reduce latency during high traffic.			
Deliverables **D5.1:** Functional backend APIs. **D5.2:** Populated MySQL database.			

Table 26. Work Package 6

WP 6: *LLM Integration*			
Start date: 07.04.2025 **End date:** *16.05.2025*			
Leader:	*Orhan Murat Tuncer*	**Members involved:**	*Cansu Zeynep Kars, Ece Büşra Civelek*
Objectives: The objective of this work package is to integrate a Large Language Model (LLM) to automate essay grading and generate detailed feedback. The model will be trained to assess grammar, coherence, and content based on predefined grading rubrics. Additionally, the system will ensure that teachers can review and adjust AI-generated grades to maintain accuracy and fairness.			
Tasks: **Task 6.1 LLM Fine-Tuning:** Train the Llama model using rubric-aligned essays to ensure grading aligns with teacher-defined criteria. **Task 6.2 Grading API Development:** Build an API endpoint to send essays/rubrics to the LLM and receive JSON-formatted grades/feedback. **Task 6.3 Feedback Parsing & Display:** Convert LLM output into structured feedback (e.g., grammar score, coherence comments) for frontend rendering.			
Deliverables **D6.1:** Grading API endpoint. **D6.2:** Sample graded essays with feedback.			

Table 27. Work Package 7

WP 7: *Testing & Bug Fixing*
Start date: 12.05.2025 **End date:** *23.05.2025*

Leader:	*Cansu Zeynep Kars*	**Members involved:**	*Orhan Murat Tuncer, Ece Büşra Civelek*
Objectives: The objective of this work package is to integrate a Large Language Model (LLM) to automate essay grading and generate detailed feedback. The model will be trained to assess grammar, coherence, and content based on predefined grading rubrics. Additionally, the system will ensure that teachers can review and adjust AI-generated grades to maintain accuracy and fairness.			
Tasks: **Task 7.1 Unit & Integration Testing:** Test individual modules (e.g., rubric creation) and integrated workflows (e.g., essay submission → grading → feedback display). **Task 7.2 Performance Optimization:** Resolve bottlenecks (e.g., slow API responses) and improve system reliability.			
Deliverables **D7.1:** Test reports and resolved issues.			

Table 28. Work Package 8

WP 7: *Finalization & Documentation*			
Start date: 19.05.2025 **End date:** *30.05.2025*			
Leader:	*Ece Büşra Civelek*	**Members involved:**	*Orhan Murat Tuncer, Cansu Zeynep Kars*
Objectives: The objective of this work package is to finalize the thesis by refining documentation, ensuring clarity, and incorporating all necessary research findings. The source code will be reviewed, tested, and documented to ensure completeness and functionality. Additionally, preparations for the project defense will include creating presentation materials and rehearsing key points to effectively communicate the system's development and outcomes.			
Tasks: **Task 8.1 Thesis Compilation:** Consolidate project documentation (RAD, SDD, test reports) into a cohesive thesis. **Task 8.2 Source Code Submission:** Clean and comment code for readability, then upload to GitHub for evaluation. **Task 8.3 Defense Preparation:** Create a presentation and poster summarizing project goals, implementation, and outcomes.			
Deliverables **D8.1:** Final thesis document. **D8.2:** Deployed system and GitHub repository.			

5. Glossary

- LLM: Large Language Model. A type of artificial intelligence model designed to understand and generate human-like text.

- AI: Artificial Intelligence. The simulation of human intelligence processes by machines, especially computer systems.
- UC: Use Case. A description of a system's behavior as it responds to a request from an external actor.
- RAD: Requirements Analysis Document. A document that outlines the requirements for a system, including functional and non-functional requirements.
- GUI: Graphical User Interface. A type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.
- API: Application Programming Interface. A set of protocols and tools for building software applications.
- Rubric: A set of criteria for grading essays, including a total score and a breakdown of points for each criterion.
- Essay: A written piece of work submitted by a student for grading.
- Announcement: A message posted by a teacher for a specific class, which can include text, links, or attachments.
- KVKK: Kişisel Verileri Koruma Kurumu (Personal Data Protection Authority). The Turkish regulatory body responsible for ensuring data privacy and protection.

6. References

(1) Educational Testing Service. (n.d.). e-rater® automated essay scoring. Retrieved from https://www.ets.org/erater

(2) Turnitin. (n.d.). Revision Assistant. Retrieved from https://www.turnitin.com/products/revision - assistant

(3) OpenAI. (2020). GPT-3: Language models are few-shot learners. Retrieved from https://openai.com/gpt-3

(4) Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805. Retrieved from https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

(5) Kişisel Verileri Koruma Kurumu (KVKK). (n.d.). Turkish Personal Data Protection Law. Retrieved from https://www.kvkk.gov.tr/