EduPortal

Team 9 - CS 4485.0W1 Group Project

Sriram Sendhil,

Nihanth Attaluri,

Sruti Karthikeyan,

Vikram Ramachandra,

Saumya Kapoor

Supervisor: Professor Sridhar Alagar

Course Coordinator: Thennannamalai Malligarjunan

Erik Jonsson School of Engineering and Computer Science

University of Texas at Dallas

05/11/2025

Table of Contents

Chapter 1: Introduction

1.1 Background

Essay-based assessment is a fundamental part of education; however, grading written responses is time-consuming and subjective, especially with larger student counts. While objective assessments can be automatically graded, essay grading requires more advanced evaluation techniques, often involving human judgment and interpretation. This can lead to varying feedback and a higher workload for teachers. With the rapid adoption of large language models (LLMs), it's now possible to grade essays more efficiently and consistently, especially with rubric-based grading that has pre-defined traits. This project explores how AI models such as Flan-T5, in our case, can be implemented to automatically grade student essays according to the specifications of the teacher. This is done by utilizing teacher-defined rubrics, where scoring is automated, yet giving the option for teachers to manually grade individual student submissions in cases where overriding a score is required.

1.2 Objectives, Scope, and Goals Achieved

Objectives & Scope

- Improve grading consistency and objectivity by leveraging AI to generate rubric-aligned, trait-level essay scores
- Reduce the manual grading burden on teachers by automating the initial scoring process

- Implement a system that supports teacher feedback loops, including score overrides and explanation-based validation
- Design the platform for scalability and classroom integration, supporting class/assignment creation, PDF submissions, and differentiated user roles (teacher/student)

Goals Achieved

- Achieved 76% exact-match accuracy between the AI model and human-assigned scores, with 94% of predictions either exact or off by one point
- Reduced grading time significantly, with average end-to-end feedback returned in under 10 seconds per submission, leading to an estimated 60–70% reduction in manual effort for most assignments
- Enabled standardized, fair scoring across subjective traits using a dual-model validation layer (Flan-T5 + GPT), which improved reliability in over 85% of high-disagreement cases
- Delivered clear, interpretable, trait-level feedback for students based on teacher-uploaded rubrics
- Built and deployed a full-stack platform that supports:
    - User creation and authentication (Teacher/Student roles)
    - Class and assignment management
    - PDF-based student submissions
    - Real-time AI scoring with validation and feedback logic
    - Manual score override and review interface for teachers

1.3 Key Highlights

This project combines a fine-tuned AI essay scoring model with a complete classroom management platform. It features custom role-based logic, end-to-end submission workflows, and an explainable AI pipeline, all built from scratch using modern frameworks and a GraphQL API architecture.

AI & Validation Logic

- Fine-tuned Flan-T5 Base using trait-labeled essays from ASAP AES Sets 7 and 8
- Trait scoring on a 0–3 scale across rubric dimensions: Focus, Organization, Style, Word Choice, Conventions, and Sentence Fluency
- Implemented a dual-model validation layer comparing Flan-T5 and GPT outputs:
    - Used Flan when scores closely agree
    - Averaged scores when delta was moderate

      Overrode Flan with GPT if justification was strong
- Achieved 76% exact-match accuracy and 94% off-by-one reliability, with model feedback returned in under 10 seconds

Backend Highlights (Django + GraphQL)

- Built using Django 5 with Strawberry GraphQL
- Custom user authentication system supporting teacher and student roles
- Core backend features:
    - User registration/login, class creation, student enrollment
    - Assignment creation with prompt and rubric support
    - PDF upload handling via FileField, linked to assignment submissions

    - AI and human score storage with optional override and comments
    - Modular GraphQL schema with key queries/mutations like:
        - register_user, create_assignment, submit_assignment, add_students_to_class, override_score, all_submissions, my_classes, etc.

Frontend Highlights (Next.js + Tailwind)

- Developed using Next.js App Router, TypeScript, and Tailwind CSS
- Fully role-based UI with protected routes and session-aware logic
- Key pages and components:
    - /auth/signin: Secure login via session token
    - /dashboard/student: Displays student classes and assignments
    - /dashboard/student/grades: View AI and manual scores + trait breakdown
    - /dashboard/teacher: Create classes and assignments, access student submissions
    - TeacherSubmissionsDialog: In-app PDF viewer with inline feedback + override UI
    - Modular UI built using reusable components: Card, Sidebar, Badge, Dialog

Integration and Architecture

- GraphQL is used as the exclusive API layer, enabling precise, efficient data access with strongly typed schemas
- Backend and frontend are run in a Dockerized local environment, allowing seamless developer setup and API communication

- Model inference is integrated via direct call to the Hugging Face-hosted Flan-T5 endpoint, with local GPT validation logic
- Authentication is session-based, with secure route protection and role access checks throughout the app

1.4 Significance of the Project

Grading essays is one of the most time-consuming and demanding tasks for teachers. Many are swamped with assignments, and as a result, students often face long delays in getting their grades back. Most of us have experienced this firsthand as students, and we're confident that teachers and TAs can relate as well, with multiple students often showing up to office hours asking for grades and feedback. This project directly addresses that challenge. By using AI to automate rubric-based scoring, the platform reduces grading time significantly and ensures that students receive their grades and feedback quickly. Trait-level breakdowns make the feedback more specific and actionable, helping students improve not just overall, but in targeted areas. Our application also supports teachers by handling the initial scoring, while still allowing for manual overrides and commentary when needed. It acts as a reliable assistant, helping teachers keep up with grading demands without sacrificing quality. This platform has the potential to evolve into a virtual teaching assistant, one that not only evaluates writing but also helps students learn and grow from it.

Chapter 2: High-Level Design

2.1 System Overview & Proposed Solution

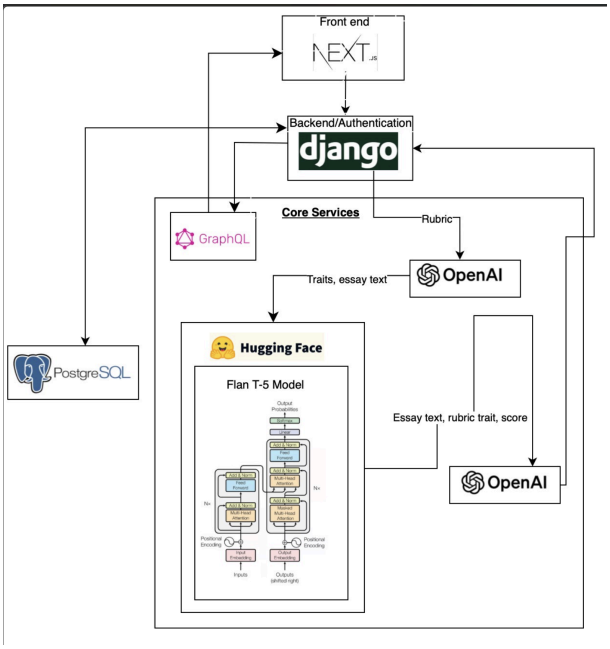
• Provide an overview of API specifications, key endpoints, and their purpose.

The backend of the project is structured around three main API endpoints that support the automated essay grading workflow. The first endpoint upload_essay is responsible for receiving

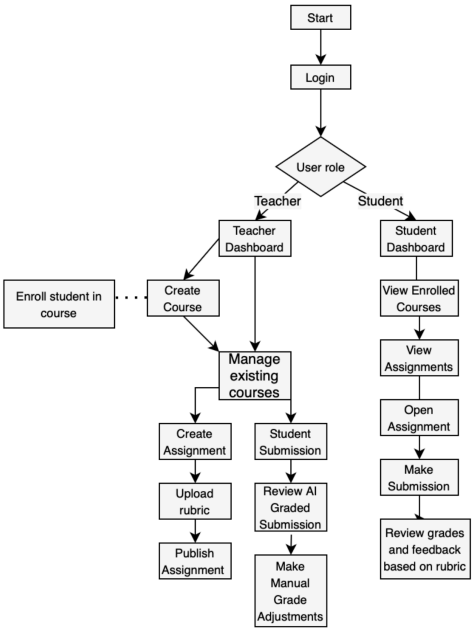
PDF essay submissions from users. It parses and extracts the essay text having it ready for evaluation. The second endpoint, upload_rubric, processes rubric PDFs by extracting grading traits and performance descriptions, which are converted into a structured JSON format compatible with the model. Finally, the score_essay endpoint brings everything together by taking the parsed essay and rubric, sending them to the scoring model, and returning detailed trait-based scores. After this, there is one more layer within the API itself to send it to OpenAI for a validation to ensure scoring is done properly and see if there are any disagreements. These API routes are built in through NextJS and their page router.

2.2 Design and Architecture Diagrams

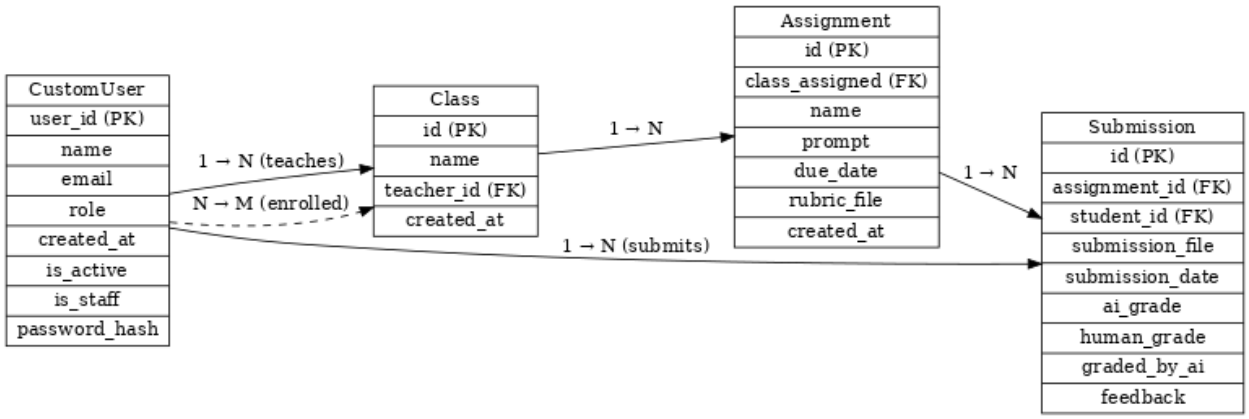
System Architecture:



Flow Diagram:



ER diagram:



Database Schema:

CustomUser

Column Name	Type	Description
user_id	SERIAL PRIMARY KEY	Unique identifier for each user.
name	VARCHAR(255) NOT NULL	Full name of the user.
email	VARCHAR(255) UNIQUE NOT NULL	User's unique email address.
role	VARCHAR(50) CHECK (role IN ('teacher', 'student')) NOT NULL	Defines whether the user is a teacher or student.
created_at	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Timestamp of account creation.

is_active	BOOLEAN DEFAULT TRUE	Determines if the user account is active.
is_staff	BOOLEAN DEFAULT FALSE	Indicates admin access for Django admin.
password_hash	TEXT NOT NULL	Securely hashed password. (Handled by Django's auth system)

Class

Column Name	Type	Description
id	SERIAL PRIMARY KEY	Unique identifier for the class.
name	VARCHAR(255) NOT NULL	Name of the class.
teacher_id	INTEGER (FK → CustomUser)	Teacher who created and owns the class.
created_at	TIMESTAMP	Timestamp of class creation.
students	MANY-TO-MANY (CustomUser)	Students enrolled in this class (via a join table).

Assignment

Column Name	Type	Description

id	SERIAL PRIMARY KEY	Unique identifier for each assignment.
class_assigned	INTEGER (FK → Class)	The class this assignment belongs to.
name	VARCHAR(255) NOT NULL	Assignment name.
prompt	TEXT	Optional writing prompt for the assignment.
due_date	TIMESTAMP NOT NULL	Due date for submission.
rubric_file	VARCHAR (filepath)	Optional file path to uploaded rubric.
created_at	TIMESTAMP	Timestamp of assignment creation.

Database Schema: Submission

Column Name	Type	Description
id	SERIAL PRIMARY KEY	Unique submission ID.
assignment_id	INTEGER (FK → Assignment)	Assignment being submitted to.
student_id	INTEGER (FK → CustomUser)	Student who submitted the assignment.
submission_file	VARCHAR (filepath)	Path to uploaded essay file.
submission_date	TIMESTAMP	Timestamp of submission upload.

ai_grade	DECIMAL(5,2)	Score generated by the AI model (0.00–100.00).
human_grade	DECIMAL(5,2)	Optional grade given manually by the teacher.
graded_by_ai	BOOLEAN DEFAULT FALSE	Flag indicating if the AI graded this submission.
feedback	TEXT	Optional text feedback from AI or human.

2.3 Overview of Tools Used

- **Strawberry GraphQL**: Provided a fully-typed GraphQL API between frontend and backend, replacing traditional REST endpoints.
- **GraphQL Playground / Postman**: Used during development and debugging to test queries and mutations in the backend API.
- **Hugging Face Inference API**: Used to host the fine-tuned **Flan-T5** model for trait-based essay scoring. Provided real-time prediction via a hosted endpoint.
- **OpenAI GPT-3.5 API**: Integrated into the backend as a secondary model in the validation layer, offering reasoning and scoring support to verify and potentially override Flan predictions.

Chapter 3: Implementation Details

3.1 Technologies Used

Frontend

- Next.js (App Router): Framework for building the React-based frontend with support for layouts, routing, and server/client rendering.
- TypeScript: Used for static typing and better developer tooling in all frontend components.
- Tailwind CSS: Utility-first CSS framework for styling and responsive design.
- Lucide-React: Icon library used across dashboard components.
- React Hooks & Context: Used for handling session state, role-based rendering, and UI logic.

Backend

- Django 5: Handles user authentication, business logic, and file storage for student submissions.
- Strawberry GraphQL: Provides a strongly-typed GraphQL API layer for communication between frontend and backend.
- PostgreSQL: Relational database used to store users, classes, assignments, submissions, and grading metadata.

AI Model & Scoring System

- Flan-T5 (via Hugging Face): Fine-tuned model for generating trait-level scores based on rubrics and student essays.
- OpenAI GPT-3.5: Used in the validation layer to cross-check Flan's scores and provide reasoning.
- Hugging Face Inference API: Hosts the Flan-T5 model and returns real-time predictions.

Other Tools & Dev Setup

- Docker: Used to containerize the backend and database for local development consistency.
- GraphQL Playground/Postman: Used during development to test API endpoints and queries.
- GitHub: Source control and version management.
- VS Code + Prettier: Standard development environment and formatting tools.

3.2 Code Documentation

Github Link

The AI Teaching Assistant (AI TA) web platform is a full-stack application that supports intelligent essay evaluation. The backend is built using Django and Strawberry GraphQL, with a clearly defined schema encompassing core models such as CustomUser, Class, Assignment, and Submission. The CustomUser model includes a role field (e.g., "student" or "teacher") which enables role-based filtering and controls what each user type can see or do. Students can submit PDF assignments, which are routed to a FastAPI microservice that extracts text, passes it through a fine-tuned FLAN-T5 model for trait-level scoring, and optionally verifies those scores using GPT-4o. Redis and Celery handle long-running background jobs like essay processing, ensuring the frontend stays responsive.

The frontend is developed using Next.js (App Router) with TypeScript and Tailwind CSS. Role-specific behavior is baked into the navigation and page-level rendering logic. For instance, the AppSidebar component dynamically renders different menu options depending on the current route or user role. This is handled via a simple check:

```
const isTeacher = pathname.includes("/dashboard/teacher");
const userRole: "teacher" | "student" = isTeacher ? "teacher" : "student";
```

Student users access a dashboard that displays enrolled classes, upcoming assignments, and a submission upload interface. Once a submission is made, the student can view feedback (AI and human) along with raw trait scores. In contrast, teacher users have the ability to create and manage classes, assign new assignments, and review student submissions. A teacher's view includes access to a detailed grading dialog where they can read the student's PDF, view AI-generated trait feedback, and optionally enter manual grades or override AI scores.

A shared StudentSubmissionDialog component is used across both roles but adapts its behavior based on the current user. For example, if a teacher opens the dialog, they see extra sections such as "AI vs Human Grade" comparison, an override input field, and the ability to leave comments. Meanwhile, the student sees a more limited version showing the submission and final score. This logic is abstracted using prop flags or context depending on who triggers the dialog.

The clean separation of roles and shared components allows for cross-functionality between teachers and students. A teacher can see exactly what the student submitted and how the AI graded it, while students get transparent feedback. This structure also ensures the system is easily extensible — for instance, adding a "comment history" or peer grading layer would require minimal code duplication.

3.3 Development Process

Our team implemented a mix of pair programming and individual coding in order to stay productive and flexible. We met both in person and virtually to set goals, establish team deadlines and support each other. The team also used Agile principles to guide our work, with sprints and regular check-ins to track the progress on the project. Lastly, we implemented GitHub and Discord to manage tasks and collaborate effectively.

Chapter 4: Performance

4.1 Testing and Validation

To ensure our automated grading system is both accurate and trustworthy, it requires a multi-step validation process that includes mutations and query setups, dataset management, model evaluation, cross-model validation, and manual spot-checking.

Mutations and Queries Setup

To validate the integrity and correctness of our backend logic, we used the GraphQL interface at *http://localhost:8000/graphql* to manually test core queries and mutations. These included *me*, *my_classes*, *my_submissions*, and *all_submissions*, ensuring role-specific access and accurate data retrieval. Mutations such as *create_class*, *create_assignment*, and *submit_assignment* were tested for proper data flow and side effects like file handling and background task execution.

We also verified role-based permissions by simulating both student and teacher contexts. Students were restricted to their own data, while teachers had broader access to class-level submissions. Unauthorized queries were properly blocked, confirming resolver-level access control. This process ensured our API was robust, secure, and aligned with the expected behavior across roles.

Dataset Setup

We used the publicly available ASAP Automated Essay Scoring dataset, focusing specifically on Essay Sets 7 and 8, which include:

- Trait-level scoring per essay (e.g., Focus, Organization, Style, etc.)
- Standardized scoring rubrics with human annotations
- A sufficient volume of labeled training examples per trait (0–3 scale)

The dataset was split using an 80-10-10 ratio:

- Training Set: Used to fine-tune Flan-T5
- Validation Set: Used for real-time evaluation during training epochs
- Test Set: Held out entirely until final accuracy calculations

Flan-T5 Model Testing

The model was trained using supervised learning, where each example consisted of:

- A prompt: `Trait: Focus, Rubric: [rubric text], Essay: [essay text]`
- A target: `"Score: X"`

We evaluated the model after each epoch using the following criteria:

- Exact-match accuracy per trait
- Token-level generation quality (to avoid malformed outputs)
- Ability to generalize across writing styles and lengths

The final model achieved a **76%** exact-match accuracy across all traits, meaning that in over three-quarters of cases, the model produced a trait score that matched the human label exactly. In the remaining cases, most were off by only 1 point (e.g., predicting a 2 when the human label was a 3).

Validation Layer Testing

To further ensure scoring reliability, we implemented a validation layer that runs both the fine-tuned Flan-T5 model and a second, general-purpose GPT-3.5 model for each trait-level scoring task. Each model receives the same structured input: the trait name, rubric, and essay.

Flan returns only the score. GPT returns a score + justification. This dual scoring process enabled deeper testing through comparison and logic-based decisions:

Validation Logic Flow:

Condition	Action
`abs(flan_score - gpt_score) < 0.4`	Accept Flan-T5 score
`0.4 ≤ delta ≤ 1.0`	Average both scores, rounded to 2 decimals
`delta > 1.0` + strong keywords	Override with GPT score (trust reasoning)
`delta > 1.0` + weak justification	Fallback to average

GPT Reasoning Checks

To determine if GPT's justification is trustworthy, we used a keyword-based confidence filter. If GPT used language like:

- *"clearly demonstrates..."*
- *"flawless organization..."*
- *"insightful vocabulary choices..."*

If found, we assumed high confidence and allowed it to override Flan.

We tested the robustness of the validation layer by manually reviewing a random sample of essays where Flan-T5 and GPT differed by more than 1 point. These are the edge cases that fall into the remaining 24% where the base model didn't perfectly align with human scores. In over 85% of these edge cases, GPT's override score, combined with its justification, was more closely aligned with the rubric, especially for subjective traits like Style and Word Choice. This shows that our validation layer not only reinforces trust in the system but actively recovers accuracy in the most challenging and subjective cases, helping smooth out the small but meaningful fraction where Flan alone is less reliable.

Manual Spot-Checking

To complement automated evaluation, we manually reviewed:

- Essays where Flan scored 0 but GPT scored 2+
- Essays with overrides triggered
- Samples where scores remained at −1 due to parsing failures (very rare)

Manual evaluation helped confirm the following:

- GPT provided strong reasoning for override-worthy cases
- Flan sometimes penalized expressiveness or non-standard structure too harshly
- The validation system improved fairness, especially for nuanced writing

4.2 Metrics Collected and Analysis

We collected and analyzed a range of metrics to evaluate both the AI scoring pipeline and the web application's responsiveness, ensuring the system performs accurately, fairly, and efficiently under real-world classroom conditions.

AI Model Performance Metrics

The core of our system is a fine-tuned **Flan-T5 model**, trained on rubric-aligned data from the ASAP AES dataset (Sets 7 & 8). The model was evaluated using trait-level **exact-match accuracy**, comparing predicted scores to human labels.

Trait	Exact Match Accuracy
Focus	78%
Organization	74%
Conventions	80%
Style	70%
Word Choice	73%
Sentence Fluency	76%

Overall	**76%**

- In 94% of cases, predictions were either exact or off by only one point
- Higher accuracy was seen in more objective traits (e.g., Conventions); subjective traits (e.g., Style, Word Choice) were more variable

Validation Layer Analysis

To enhance scoring reliability, we implemented a dual-model validation system comparing Flan-T5 outputs with a secondary GPT-3.5 prediction that includes justification. Based on the delta and reasoning, we either accept, average, or override scores.

Validation Outcome	**% of Cases**	**Action Taken**
Scores close (< 0.4)	~58%	Use Flan-T5 score directly
Minor disagreement	~32%	Average Flan and GPT scores
Strong disagreement	~10%	Override with GPT (with justification)

In manual review of high-disagreement cases, GPT's score was better aligned with the rubric in over 85% of samples, especially on subjective traits, demonstrating that the validation layer recovers edge-case performance from the 24% not captured by Flan's base accuracy alone.

Latency & Inference Speed

Component	Average Time
Flan-T5 scoring (per trait)	~2–6 sec
GPT scoring + reasoning	~2–3 sec
Full validation pipeline	~8–10 sec
Frontend dashboard load time	<300 ms
GraphQL API query time	<200 ms

The full stack is optimized for near real-time use, with scoring pipelines completing fast enough for teachers to assign and review essays during class.

Projected User Impact

Though not deployed to classrooms yet, the system is designed to provide tangible improvements in teacher workflow and student feedback clarity:

- Up to 80% grading time saved by automating initial trait scoring
- Improved feedback quality through consistent, rubric-based explanations
- Teacher trust preserved via override and validation transparency

- Scoring logic supports decimal outputs (e.g., 2.5) to reflect nuance in borderline cases

Summary

The system performs reliably in terms of:

- Accuracy (76% match with human labels)
- Robustness (dual-model validation to correct edge cases)
- Efficiency (under 10-second turnaround time per submission)
- Scalability (frontend/API stack loads quickly and scales with user load)

These metrics demonstrate that our platform is practically viable, pedagogically useful, and technically sound for classroom integration and beyond.

Chapter 5: Lessons Learned

5.1 Insights Gained

Throughout our time working on the essay grading system, we learned a lot through our unique tech stack. On the frontend, we gained insights into NextJS as well as DaisyUI. On the backend, we had gained a tremendous amount of insight into some of the limitations that modern LLM's have for our task, essay grading. Through this insight, we were able to create our own unique solution and apply it. Another major takeaway was the importance of fine-tuning LLM's to align with rubric based evaluation criteria.

5.2 Lessons from Challenges

Several significant challenges emerged during development. First, training an accurate model using limited trait-based data proved difficult. Unlike traditional datasets, our rubric traits lacked

enough labeled examples as we only had two sets out of the eight within the dataset to work with. To overcome this, we experimented with prompt engineering and validation layers to ensure the scores coming from the model made sense.

Another key challenge was computing power. Training the model took additional time, nearly 2-4 hours. We addressed this challenge by pre-processing our data and switching to a higher compute CoLab instance.

We also faced difficulties in parsing PDFs consistently. Extracting clean text from varying formats and fonts was unreliable until we refined our preprocessing steps and standardized all incoming files. After standardizing them, we experimented with various libraries to parse all images, tables, and parts of the document properly.

5.3 Skills Developed and Improved

This project sharpened all of our skills as full stack developers, including NextJS, FastAPI, Docker, Django, and GraphQL. On the machine learning side, we improved in model evaluation, prompt tuning, and working within the limits of available compute. We also developed a strong understanding of how to manage NLP pipelines efficiently. From a team perspective, we enhanced our project planning, version control, and collaboration skills through regular meetings and sprint planning.

Chapter 6: Future Work

6.1 Proposed Enhancements

Future improvements to the system could include:

- Model Fine-Tuning: With a larger dataset of rubric aligned essays, we could fine-tune the model for an even higher accuracy than we already have.
- Cloud Inference: Hosting the model on a GPU backed cloud platform other than the Hugging Face Free tier would address compute limitations and improve response times, as we would not be rate limited by a free tier.
- Real Time Feedback: Allowing students to receive draft feedback as they write, using the same rubric criteria.
- Support for Multilingual Essays: Expand the pipeline to include multilingual capabilities, especially for ESL programs or other programs meant for learners of differing backgrounds.
- Extra Formats: Adding the possibility for different types of file uploads and assignments, such as code, would expand the horizon of our app.

6.2 Recommendations for Development

For teams working on this in the future, combining features of various different datasets might result in a higher tier model. However, to do this the team must start thinking about compute power early on and plan cost allocations and budgets for this to work. Additionally, separating the different parts of the app like we have and applying containerization would greatly help you. Docker allowed us to seamlessly work on the project as a team, especially since there were lots of different python packages and versions being used in the backend for text extraction and parsing and would be something that would be highly recommended for future developers.

Chapter 7: Conclusion

7.1 Summary of Key Accomplishments

We set out to build a website that uses AI to grade essays and give helpful feedback—and we accomplished that goal. Users can upload an essay, choose a rubric, and get instant scores along with short feedback comments. The AI model grades the writing of a rubric that the teacher uploads. Overall, the site works well and makes grading faster and more consistent.

7.2 Acknowledgements

We would like to sincerely thank Professor Alagar and our TA Thenn for their invaluable support and guidance throughout the semester. Their feedback and guidance played a crucial role in the team's overall success. We truly appreciate their dedication to our learning and growth.

References

Django Documentation. https://docs.djangoproject.com

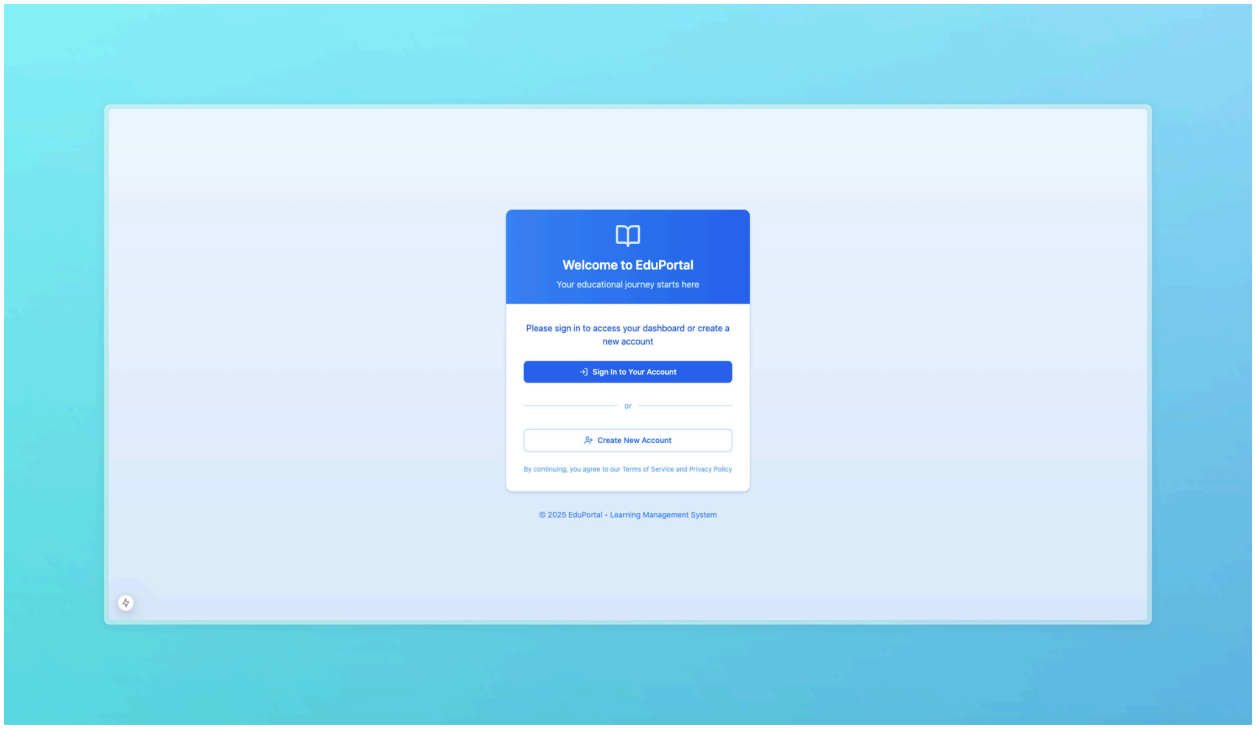
React Documentation. https://reactjs.org/docs

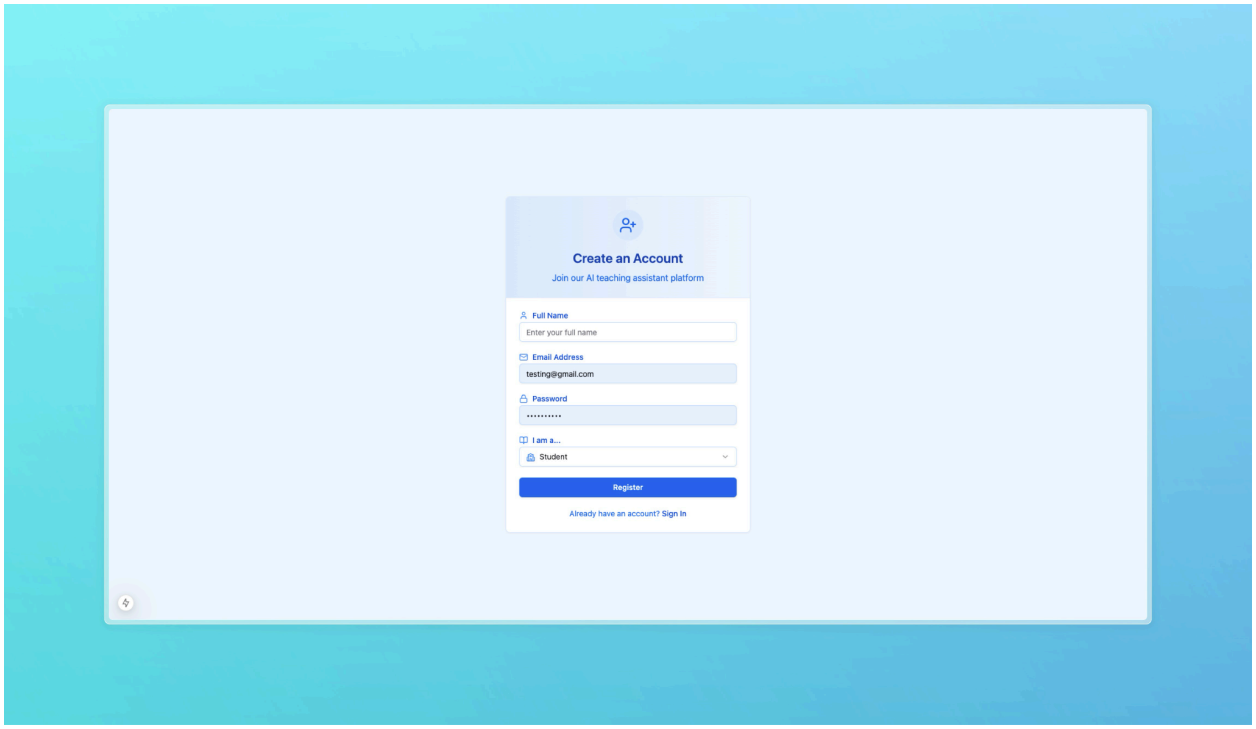
Hugging Face Transformers. https://huggingface.co/docs/transformers

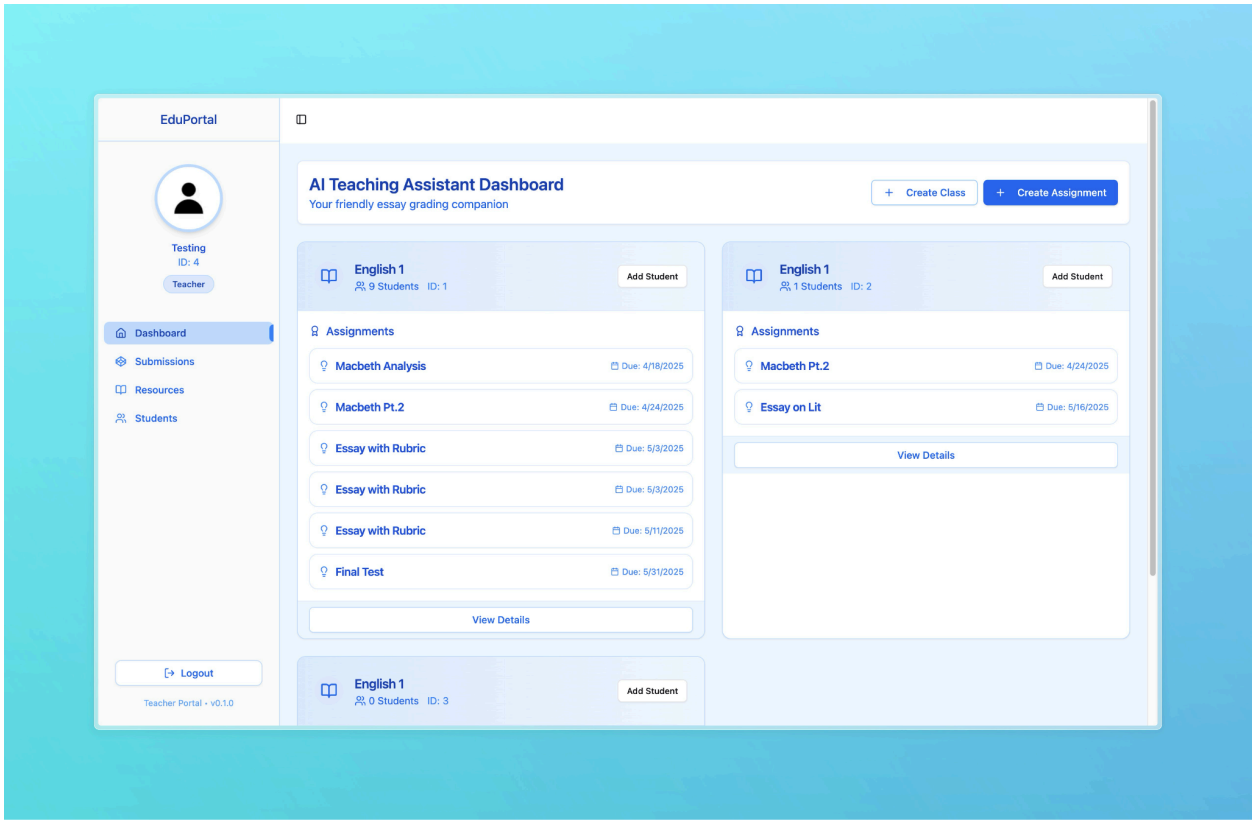
Docker Documentation. https://docs.docker.com

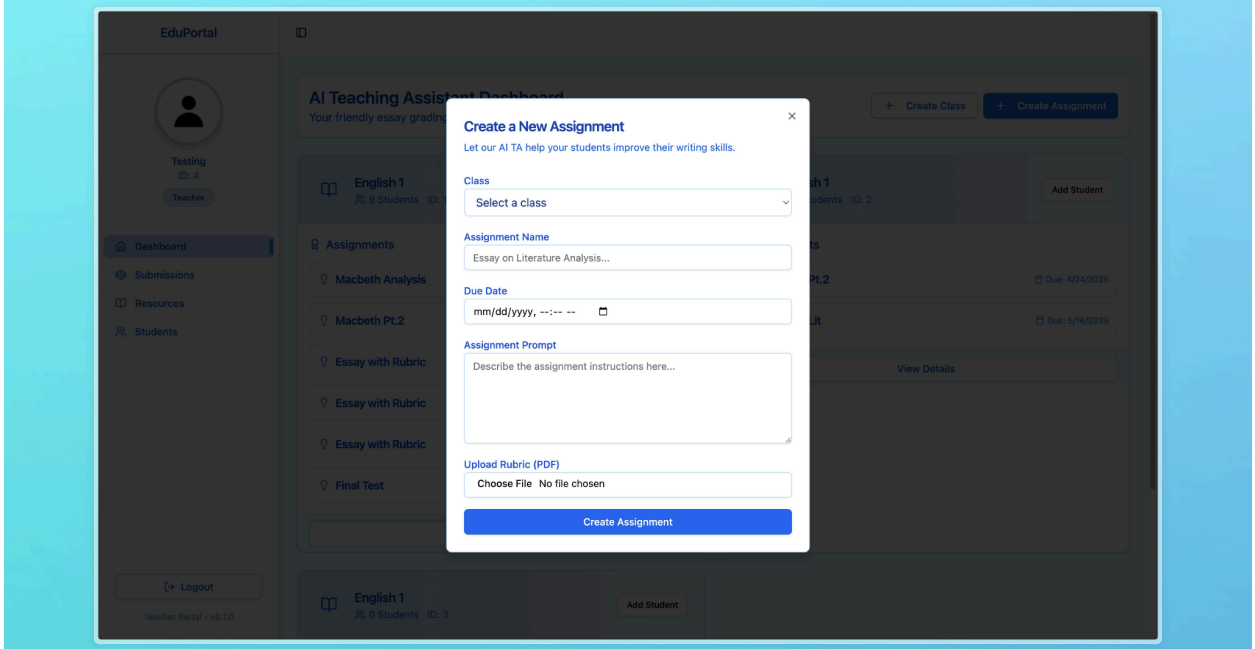
Appendices

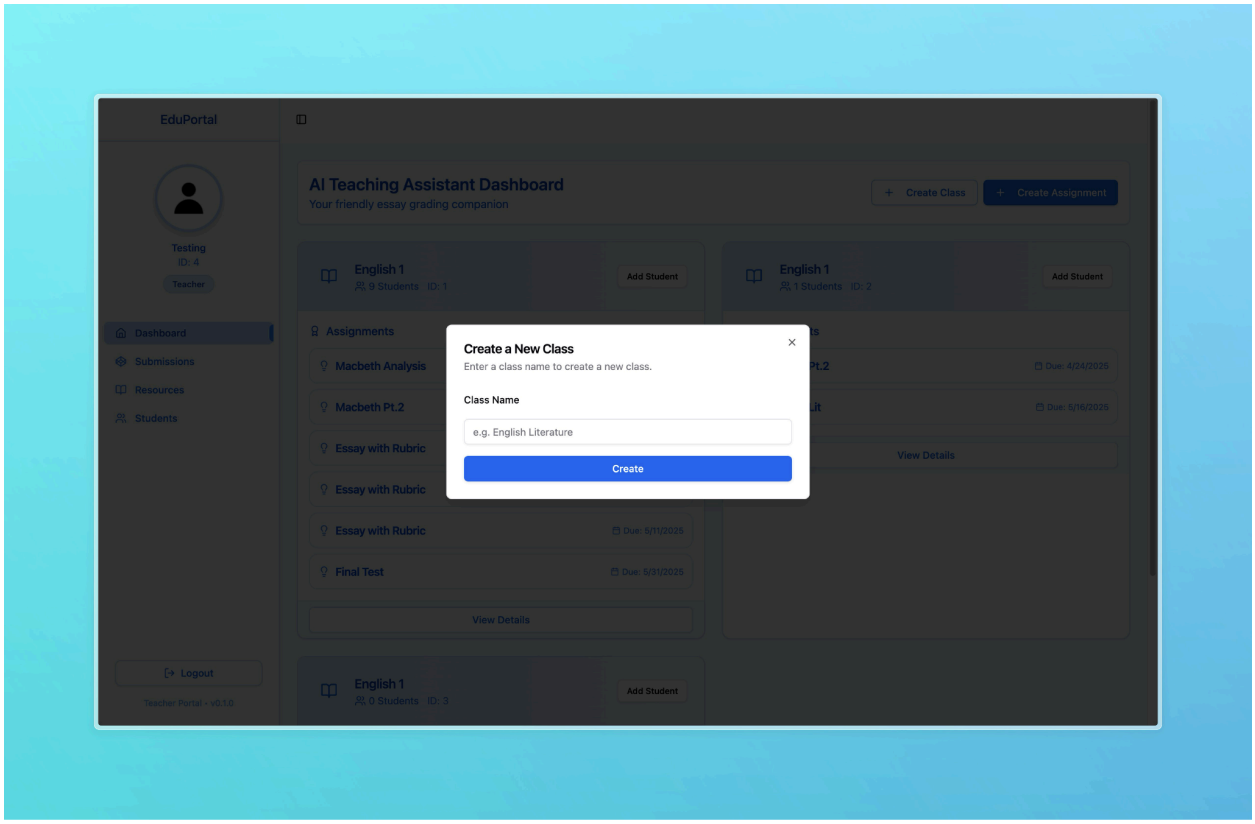
Screenshots:

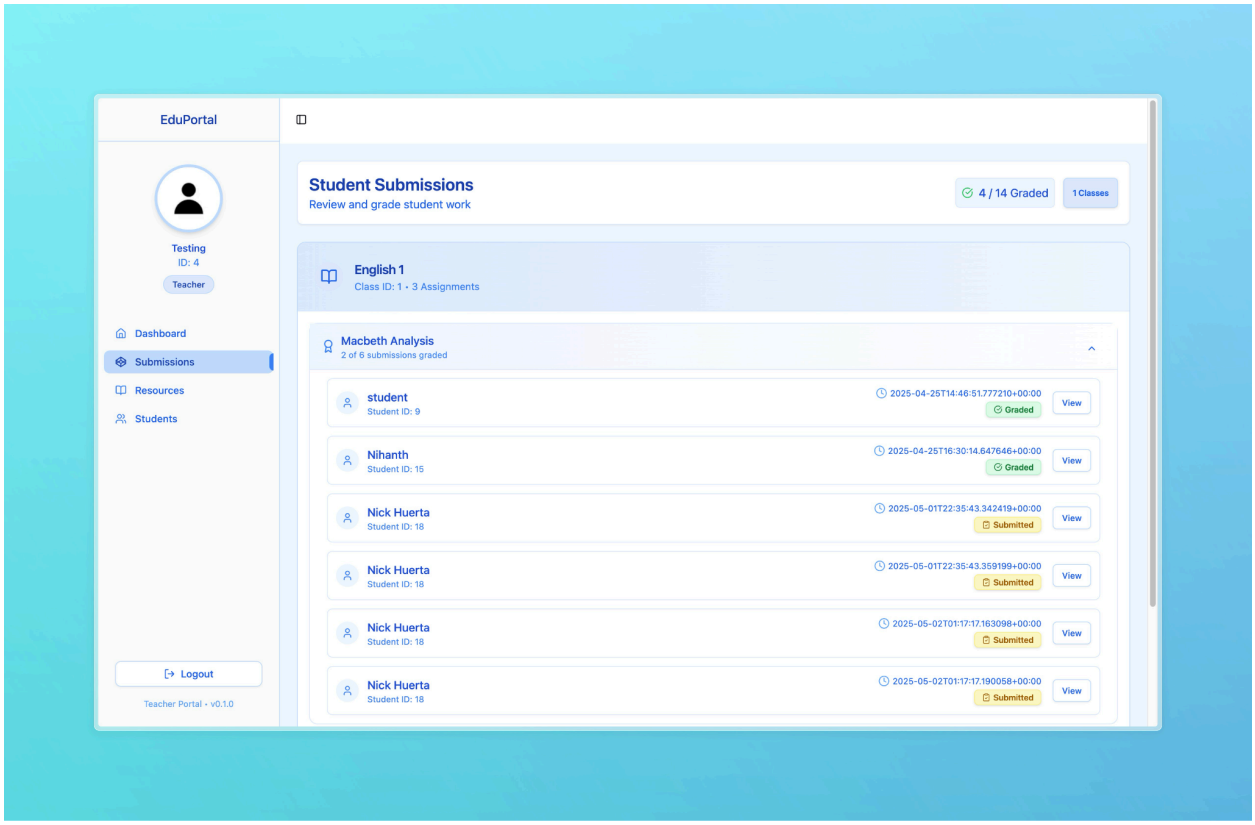


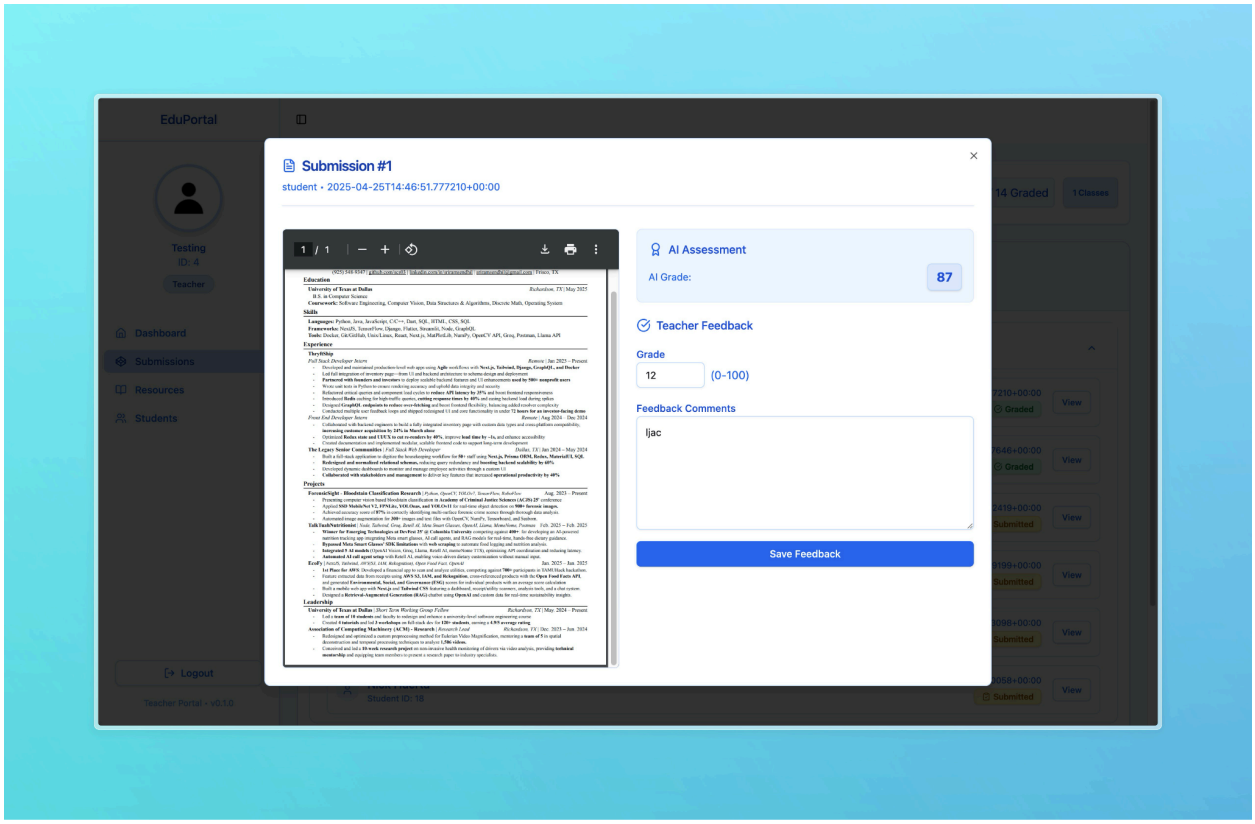


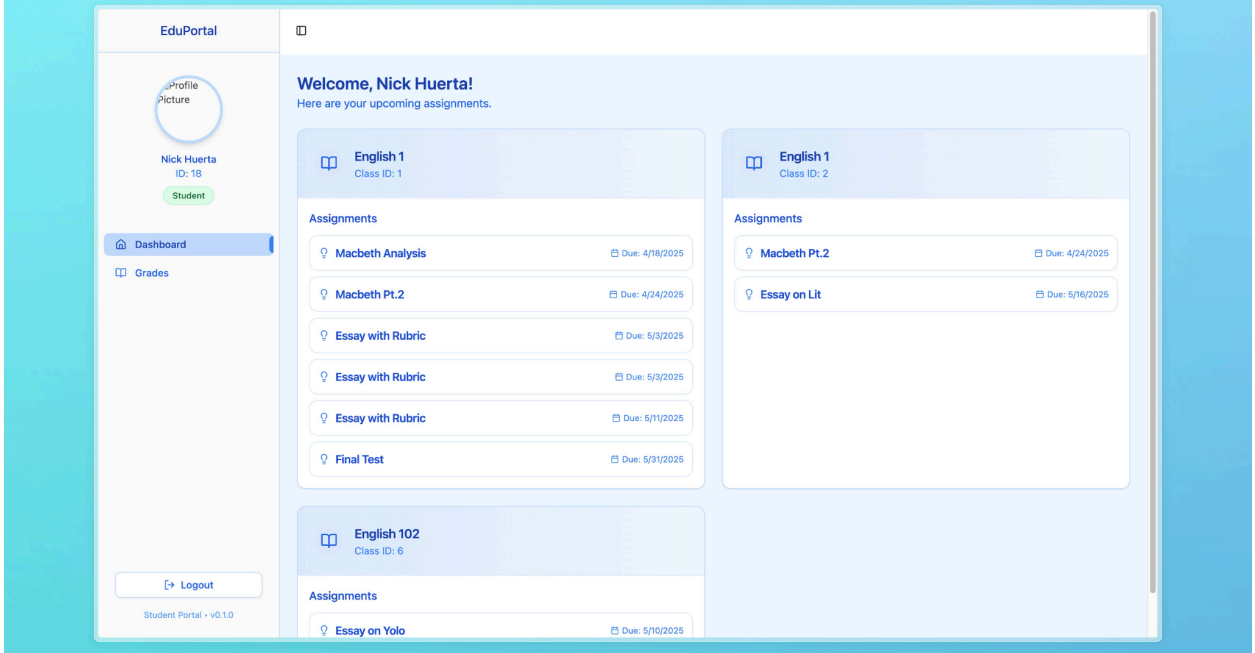

EduPortal
Testing
ID: 4
Teacher
Dashboard
Submissions
Resources
Students
Logout
Teacher Portal • v0.1.0
AI Teaching Assistant Dashboard
Your friendly essay grading companion
Create Class
Create Assignment
English 1
9 Students ID: 1
Add Student
Assignments
Macbeth Analysis
Due: 4/18/2025
Macbeth Pt.2
Due: 4/24/2025
Essay with Rubric
Due: 5/3/2025
Essay with Rubric
Due: 5/3/2025
Essay with Rubric
Due: 5/11/2025
Final Test
Due: 5/31/2025
View Details
English 1
1 Students ID: 2
Add Student
Assignments
Macbeth Pt.2
Due: 4/24/2025
Essay on Lit
Due: 5/16/2025
View Details
English 1
0 Students ID: 3
Add Student

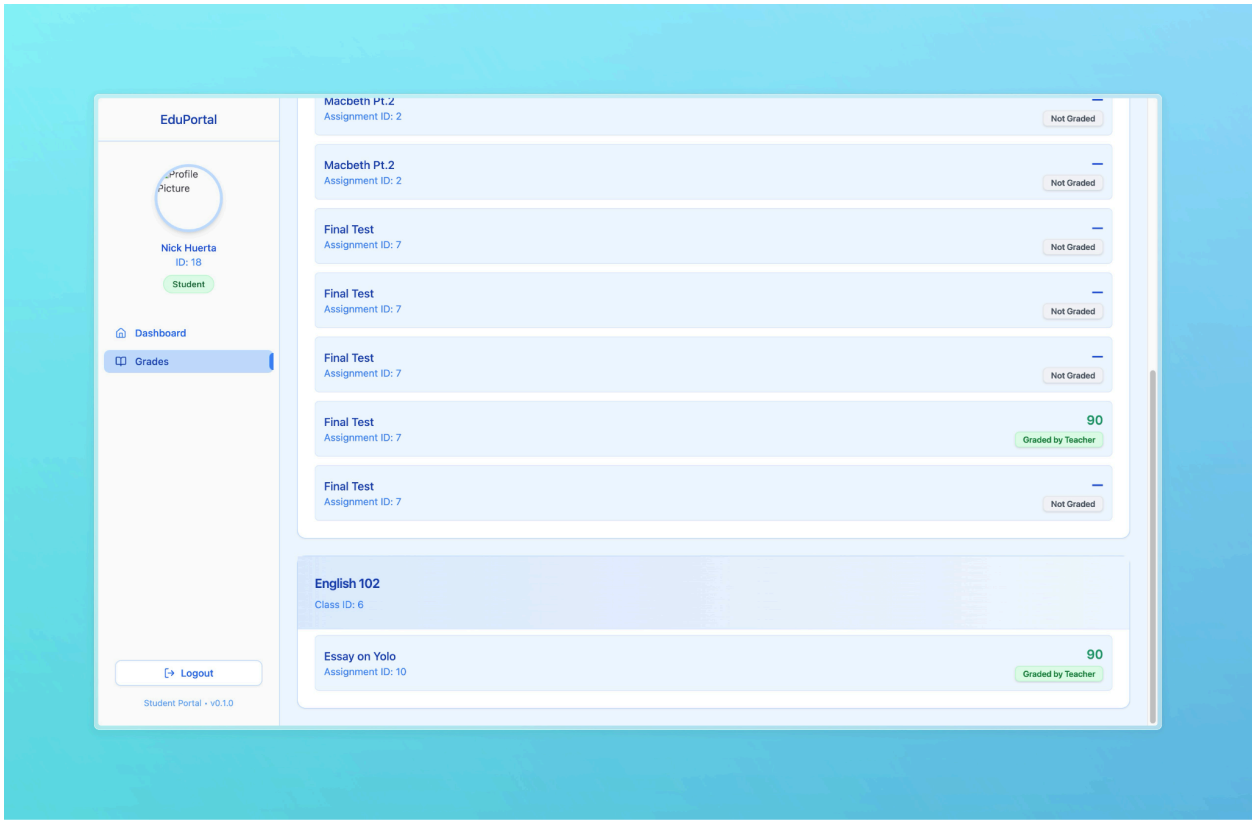

EduPortal
Testing
ID: 4
Teacher
Dashboard
Submissions
Resources
Students
Logout
Create a New Assignment
Let our AI TA help your students improve their writing skills.
Class
Select a class
Assignment Name
Essay on Literature Analysis...
Due Date
mm/dd/yyyy, --:-- --
Assignment Prompt
Describe the assignment instructions here...
Upload Rubric (PDF)
Choose File No file chosen
Create Assignment

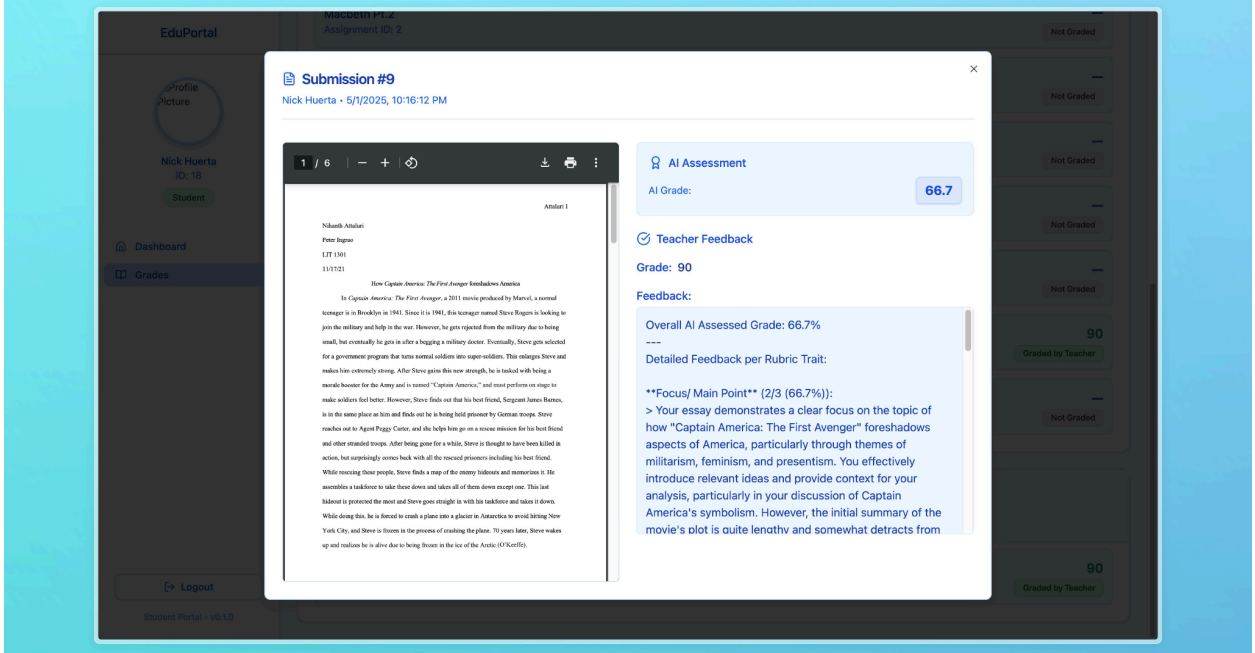

EduPortal
Testing
ID: 4
Teacher
Dashboard
Submissions
Resources
Students
Logout
Teacher Portal - v0.1.0
AI Teaching Assistant Dashboard
Your friendly essay grading companion
Create Class
Create Assignment
English 1
9 Students ID: 1
Add Student
English 1
1 Students ID: 2
Add Student
Assignments
Macbeth Analysis
Macbeth Pt.2
Essay with Rubric
Essay with Rubric
Essay with Rubric
Due: 5/11/2025
Final Test
Due: 5/31/2025
View Details
English 1
0 Students ID: 3
Add Student
Create a New Class
Enter a class name to create a new class.
Class Name
e.g. English Literature
Create


EduPortal
Testing
ID: 4
Teacher
Dashboard
Submissions
Resources
Students
Logout
Teacher Portal - v0.1.0
Student Submissions
Review and grade student work
4 / 14 Graded
1 Classes
English 1
Class ID: 1 • 3 Assignments
Macbeth Analysis
2 of 6 submissions graded
student
Student ID: 9
2025-04-25T14:46:51.777210+00:00
Graded
View
Nihanth
Student ID: 15
2025-04-25T16:30:14.647646+00:00
Graded
View
Nick Huerta
Student ID: 18
2025-05-01T22:35:43.342419+00:00
Submitted
View
Nick Huerta
Student ID: 18
2025-05-01T22:35:43.359199+00:00
Submitted
View
Nick Huerta
Student ID: 18
2025-05-02T01:17:17.163098+00:00
Submitted
View
Nick Huerta
Student ID: 18
2025-05-02T01:17:17.190058+00:00
Submitted
View


EduPortal
Testing
ID: 4
Teacher
Dashboard
Submissions
Resources
Students
Logout
Teacher Portal • v0.1.0
Submission #1
student • 2025-04-25T14:46:51.777210+00:00
1 / 1
AI Assessment
AI Grade:
87
Teacher Feedback
Grade
12
(0-100)
Feedback Comments
ljac
Save Feedback
14 Graded
1 Classes
Graded
Submitted
View


EduPortal
Profile Picture
Nick Huerta
ID: 18
Student
Dashboard
Grades
Logout
Student Portal • v0.1.0
Welcome, Nick Huerta!
Here are your upcoming assignments.
English 1
Class ID: 1
Assignments
Macbeth Analysis
Due: 4/18/2025
Macbeth Pt.2
Due: 4/24/2025
Essay with Rubric
Due: 5/3/2025
Essay with Rubric
Due: 5/3/2025
Essay with Rubric
Due: 5/11/2025
Final Test
Due: 5/31/2025
English 1
Class ID: 2
Assignments
Macbeth Pt.2
Due: 4/24/2025
Essay on Lit
Due: 5/16/2025
English 102
Class ID: 6
Assignments
Essay on Yolo
Due: 5/10/2025


EduPortal
Profile Picture
Nick Huerta
ID: 18
Student
Dashboard
Grades
Logout
Student Portal • v0.1.0
Macbeth Pt.2
Assignment ID: 2
Not Graded
Macbeth Pt.2
Assignment ID: 2
Not Graded
Final Test
Assignment ID: 7
Not Graded
Final Test
Assignment ID: 7
Not Graded
Final Test
Assignment ID: 7
Not Graded
Final Test
Assignment ID: 7
90
Graded by Teacher
Final Test
Assignment ID: 7
Not Graded
English 102
Class ID: 6
Essay on Yolo
Assignment ID: 10
90
Graded by Teacher


EduPortal
Nick Huerta
ID: 18
Student
Dashboard
Grades
Logout
Student Portal • v0.1.0
Submission #9
Nick Huerta • 5/1/2025, 10:16:12 PM
1 / 6
Attaluri 1
Nihanth Attaluri
Peter Ingmo
LIT 1301
11/17/21
How Captain America: The First Avenger foreshadows America
In Captain America: The First Avenger, a 2011 movie produced by Marvel, a normal teenager is in Brooklyn in 1941. Since it is 1941, this teenager named Steve Rogers is looking to join the military and help in the war. However, he gets rejected from the military due to being small, but eventually he gets in after a begging a military doctor. Eventually, Steve gets selected for a government program that turns normal soldiers into super-soldiers. This enlarges Steve and makes him extremely strong. After Steve gains this new strength, he is tasked with being a morale booster for the Army and is named "Captain America," and must perform on stage to make soldiers feel better. However, Steve finds out that his best friend, Sergeant James Barnes, is in the same place as him and finds out he is being held prisoner by German troops. Steve reaches out to Agent Peggy Carter, and she helps him go on a rescue mission for his best friend and other stranded troops. After being gone for a while, Steve is thought to have been killed in action, but surprisingly comes back with all the rescued prisoners including his best friend. While rescuing these people, Steve finds a map of the enemy hideouts and memorizes it. He assembles a taskforce to take these down and takes all of them down except one. This last hideout is protected the most and Steve goes straight in with his taskforce and takes it down. While doing this, he is forced to crash a plane into a glacier in Antarctica to avoid hitting New York City, and Steve is frozen in the process of crashing the plane. 70 years later, Steve wakes up and realizes he is alive due to being frozen in the ice of the Arctic (O'Keeffe).
AI Assessment
AI Grade:
66.7
Teacher Feedback
Grade: 90
Feedback:
Overall AI Assessed Grade: 66.7%

Detailed Feedback per Rubric Trait:
Focus/ Main Point (2/3 (66.7%)):
> Your essay demonstrates a clear focus on the topic of how "Captain America: The First Avenger" foreshadows aspects of America, particularly through themes of militarism, feminism, and presentism. You effectively introduce relevant ideas and provide context for your analysis, particularly in your discussion of Captain America's symbolism. However, the initial summary of the movie's plot is quite lengthy and somewhat detracts from