Auto-Grading System

Student Name & Enrollment No.:

Ena Luhadia Jain (0827CS221089)

Training Program Name:

AIML With GenAI

Submission Date:

01-09-2025

Abstract

This project is a web-based Auto-Grading System that can automatically check and score student answers. Usually, teachers spend a lot of time reading answers and giving marks, which can be slow and sometimes inconsistent. This system solves that problem by using Python and machine learning techniques to compare student answers with correct answers stored in a database.

The app uses TF-IDF and cosine similarity to find out how close a student's answer is to the correct one. Based on this similarity, it gives a score out of 10 and short feedback like "Excellent!Very close to the correct answer. ", "Good attempt,but missing some details." or "Needs improvement."

The purpose of this project is to make grading faster, fair, and more efficient. It also gives students instant feedback so they can improve. This system can be expanded in the future to include more subjects, handle more answers at once, or use advanced AI models for even smarter grading.

Project Details

Objective

- Make grading of answers automatic and quick.
- Give students immediate feedback.
- Reduce teachers' workload and make scoring consistent.
- Help students identify mistakes and improve their learning.
- Provide a scalable solution for grading large classes efficiently.
- Create a user-friendly system that can be accessed online anytime.

Methodology

- **Tools Used:** Python, Flask, VS Code, GitHub, Render (for online hosting)
- **Libraries:** scikit-learn (for similarity calculation), Flask, HTML/CSS/JavaScript
- **Steps Taken:**

    1. Created a database of questions and correct answers.
    2. Built a web interface to type in a question and a student answer.
    3. Used **TF-IDF** to turn text into numbers and **cosine similarity** to compare answers.
    4. Generated a score (0–10) and feedback automatically.
    5. Deployed the app online using Render.

Implementation

Steps followed:

1. **Setup Environment**
    - Installed Python and Flask.
    - Installed required libraries: flask, scikit-learn, numpy.
2. **Created Flask Backend (app.py)**
    - Defined routes for the homepage (/) and grading (/grade).
    - Used Flask to handle requests from the frontend.
3. **Prepared QA Bank**
    - Stored subjects, questions, and correct answers in a Python dictionary.
    - This acts as the knowledge base for grading.

4. **Frontend Development (index.html)**
    - ○ Designed a clean UI with HTML, CSS, and JavaScript.
    - ○ Added subject selection, automatic question display, and a "Next Question" button.
    - ○ Students enter answers and submit them for grading.
5. **Integrated Backend + Frontend**
    - ○ Used fetch (AJAX) in JavaScript to send student answers to Flask.
    - ○ Flask compares the submitted answer with the correct answer from QA Bank.
6. **Answer Evaluation Logic**
    - ○ Converted answers into vectors using TF-IDF (from scikit-learn).
    - ○ Calculated cosine similarity between the student's answer and the correct answer.
    - ○ Converted similarity value to a score (0–10).
7. **Displayed Results**
    - ○ Send back the score and feedback to the frontend.
    - ○ Results shown instantly on the web page.

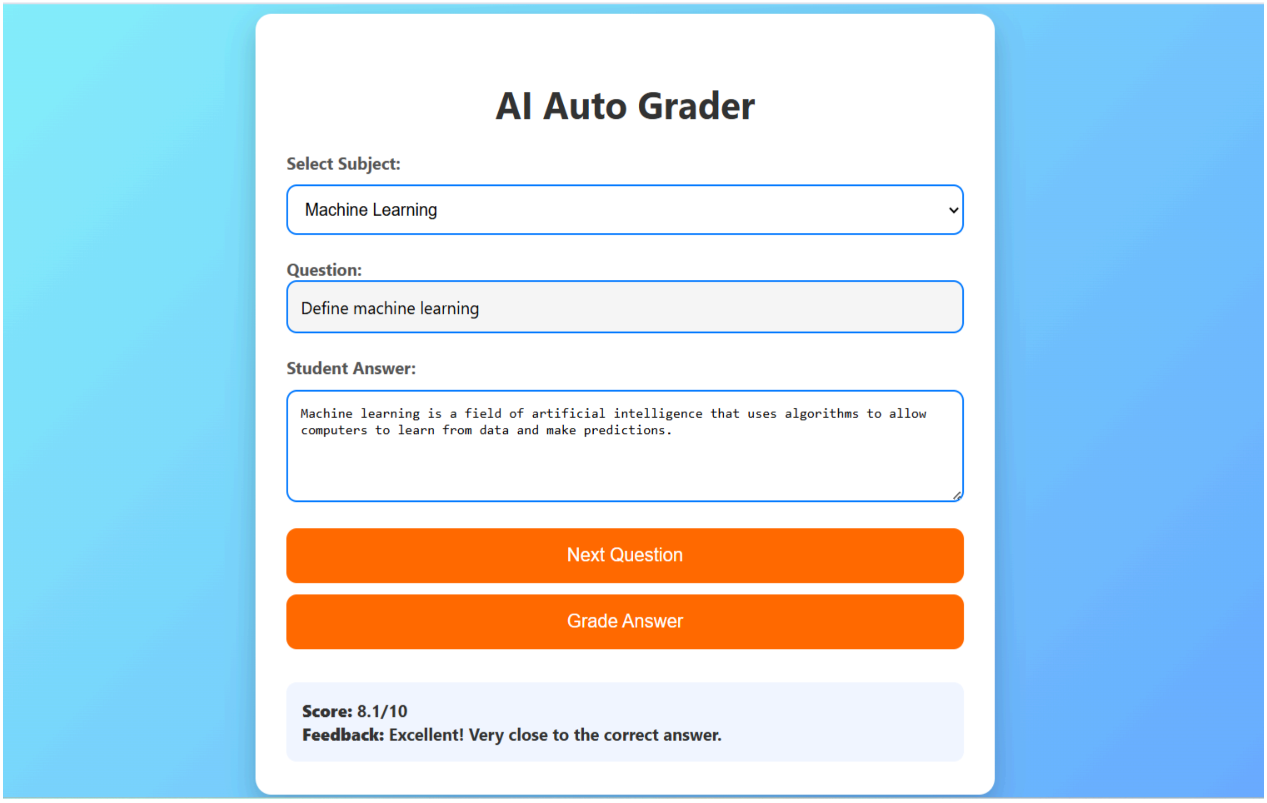
How it works:

1. The teacher/student enters a question and answer on the web page.
2. The app compares the answer with the stored correct answer.
3. It calculates a similarity score and converts it to 0–10.
4. Gives feedback:

    - ○ High similarity → "Excellent!Very close to the correct answer. "
    - ○ Medium similarity → "Good attempt,but missing some details."

- ○ Low similarity → "Needs improvement. Check the key points in the answer."

Screenshot / Output Sample:



Define machine learning

```
Machine learning is a field of artificial intelligence that uses algorithms to allow
computers to learn from data and make predictions.
```

Score: 8.1/10
Feedback: Excellent! Very close to the correct answer.

Results

- Grading is automatic and instant.
- Scores are consistent and based on similarity to correct answers.
- Feedback messages help students know how to improve.
- Fully deployed web app available online.

Conclusion

What I learned:

- How to build a web app using Flask.
- How to use Python and machine learning to compare text answers.
- How to use Git/GitHub for version control.
- How to deploy a web app.

Future improvements:

- Add support for more open-ended answers using smarter AI.
- Allow teachers to upload multiple answers at once from files.
- Make it work for multiple subjects.
- Add a dashboard for teachers to see results.

References

- Flask Framework – used for building the backend of the project.
- HTML, CSS, and JavaScript – for designing the frontend (user interface).
- Python Standard Library – for handling logic and evaluation.
- GitHub – for version control and deployment.
- General online tutorials and documentation