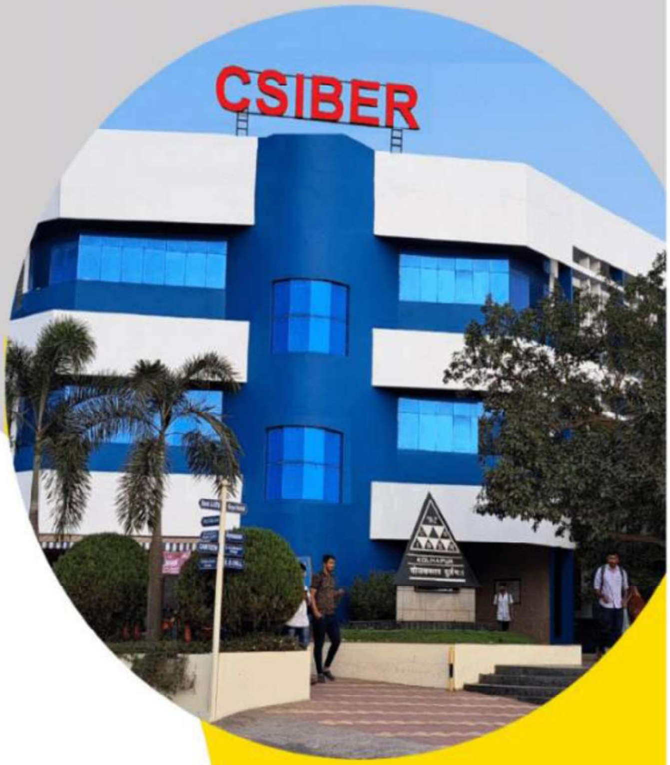
ISSN : 3048-5320 (Online)

CSIBER International Journal - CIJ

Vol. 3, Issue 3, July, 2025

MULTIDISCIPLIANRY JOURNAL



MAKE IN INDIA

Published by : CSIBER Press, Central Library Building, CSIBER Campus, University Road,Kolhapur-416004, Maharashtra, India.

Find the Journal Online at
https://www.siberindia.edu.in/journals
E-mail : cij@siberindia.edu.in

CSIBER International Journal (CIJ)

CONTENTS

 ISSN: 3048-5320 (Online)

Next-Gen Programming Pedagogy through a Personalized, Data-Driven Framework for Adaptive Learning and Evaluation

Dr. P.G.Naik
Professor,
School of Computer Science and Applications,
CSIBER, Kolhapur, MS, India

Dr. R.S.Kamath
Asso. Professor,
School of Computer Science and Applications,
CSIBER, Kolhapur, MS, India

Dr. S.S.Jamsandekar
Asst. Professor,
School of Computer Science and Applications,
CSIBER, Kolhapur, MS, India

Abstract
The current research presents a modern pedagogical framework integrating Personalized Adaptive Learning (PAL) and automated code evaluation to enhance the teaching and assessment of Python programming. The study emphasizes student-centered, technology-driven methodologies that dynamically adjust learning paths based on learner performance, prior knowledge, and engagement. A conceptual and implemented model using Flask and the Piston API enables real-time code evaluation, grade classification, and data visualization. The system features diagnostic entry assessments, adaptive learning modules, and automated grading with feedback. It also categorizes students into performance clusters (for example, 'Needs Attention' to 'Advanced Learner') using visual analytics. The approach promotes efficient, inclusive, and scalable learning, offering a comprehensive solution for outcome-based education in programming courses

Keywords: Adaptive Learning, Code Evaluation, Flask Application, Learning Path, Personalized Learning, Student Clustering

Introduction

Modern Teaching Pedagogy refers to contemporary approaches and methods used in education that focus on active, student-centered learning, integration of technology, collaboration, critical thinking, and real-world application. It moves beyond traditional lecture-based models to engage learners more deeply and effectively.

Key Features of Modern Pedagogy

Learner-Centered Approach

- Focus on individual learning styles, needs, and interests
- Promotes autonomy, self-direction, and motivation

Technology Integration

- Use of digital tools (for example, smartboards, LMS, simulations, AI, VR)
- Enables blended and online learning environments

Collaborative Learning

- Emphasis on group work, peer feedback, and social learning
- Develops communication and teamwork skills

Inquiry-Based and Experiential Learning

- Students explore, ask questions, and learn by doing
- Encourages curiosity and critical thinking

Outcome-Based Education (OBE)

- Focus on learning outcomes aligned with skills and competencies
- Learning activities and assessments are designed to meet these outcomes

Formative Assessment and Feedback

- Continuous evaluation and feedback to guide improvement
- Promotes mastery rather than performance-based grades

Flexible and Inclusive Practices

- Accommodates diverse learners, including those with special needs
- Promotes equity and cultural responsiveness

Examples of Modern Pedagogical Models

- *Flipped Classroom*

Students learn content at home (for example, via videos) and engage in problem-solving or discussion in class.

- *Project-Based Learning (PBL)*
  Students work on real-world projects over time.
- *Gamification*
  Use of game elements to enhance motivation and engagement.
- Personalized Adaptive Learning
  Content adjusts in real time based on learner performance.

Personalized Adaptive Learning

Personalized Adaptive Learning is an educational approach that uses technology and data-driven strategies to tailor learning experiences to the individual needs, pace, strengths, and weaknesses of each learner. It moves away from the traditional 'one-size-fits-all' model by dynamically adjusting content, assessments, and feedback based on the learner's performance and engagement.

Key Features of Personalized Adaptive Learning

The prominent features of personalized adaptive learning are depicted in Fig. 1.

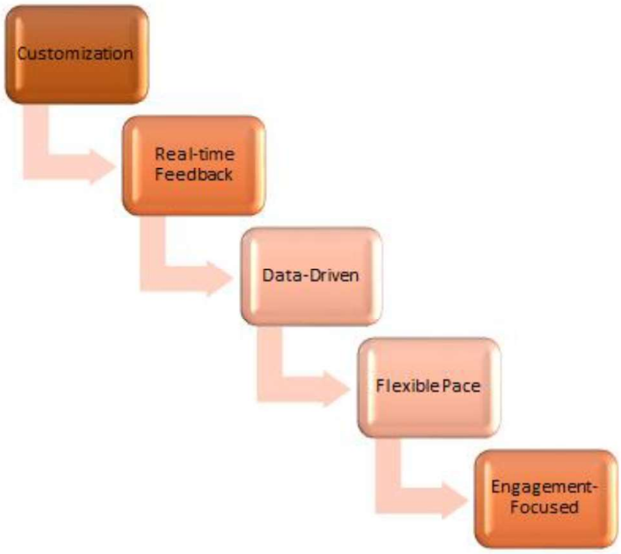


Fig. 1. Key Features of Personalized Adaptive Learning

Customization

Learning content and activities are tailored based on student profiles.

Real-time Feedback

Learners receive instant responses that help guide improvement.

Data-Driven

Learner data is continuously analyzed to adjust the path or difficulty level.

Flexible Pace

Students progress at their own speed, focusing more on areas where they struggle.

Engagement-Focused

Interactive and relevant content improves learner motivation.

Defining the Learning Path

A learning path is a structured sequence of content, activities, and assessments designed to help learners achieve specific goals or competencies. In contrast to the traditional learning path, in adaptive learning this path is not fixed—it evolves based on the learner's progress, choices, and needs.

 ISSN: 3048-5320 (Online)

Components of a Learning Path

Different components of learning path are shown below and outlined in Fig. 2.

- Entry-level assessment to gauge existing knowledge
- Modules or topics arranged based on difficulty or relevance
- Branching paths depending on learner performance
- Milestones and checkpoints to track progress
- End goals that align with course or program outcomes

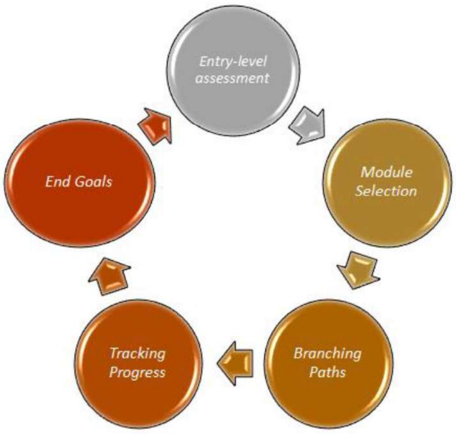


Fig. 2. Different Components of Learning Path

Need for Personalized Adaptive Learning and Defined Learning Paths

Addresses Learner Diversity

Students have varied backgrounds, skills, and paces of learning.

Enhances Learning Efficiency

Time is spent on areas needing improvement rather than repeating known content.

Improves Learning Outcomes

Better alignment of learning with individual needs results in higher achievement.

Supports Lifelong Learning

Encourages self-directed and continuous learning.

Scales with Technology

Easily implemented in digital platforms, enabling education at scale without compromising quality.

Application of Adaptive Learning for a course on Python Programming

This section presents a simple example of how a learning path is defined in a Personalized Adaptive Learning system, using the topic 'Introduction to Python Programming' for beginners.

Goal to be Achieved

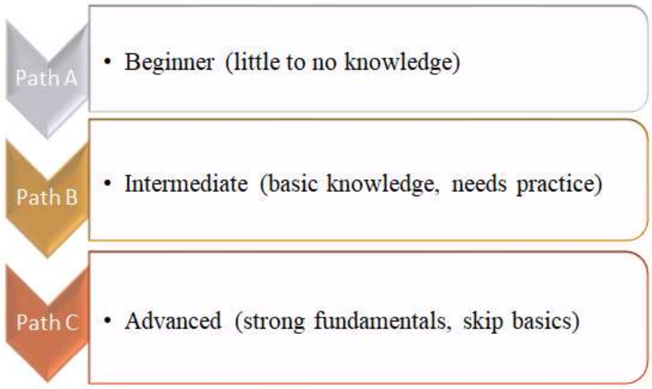
Learner should be able to write basic Python programs involving variables, control structures, functions, and file handling.

Step 1: Initial Diagnostic Test

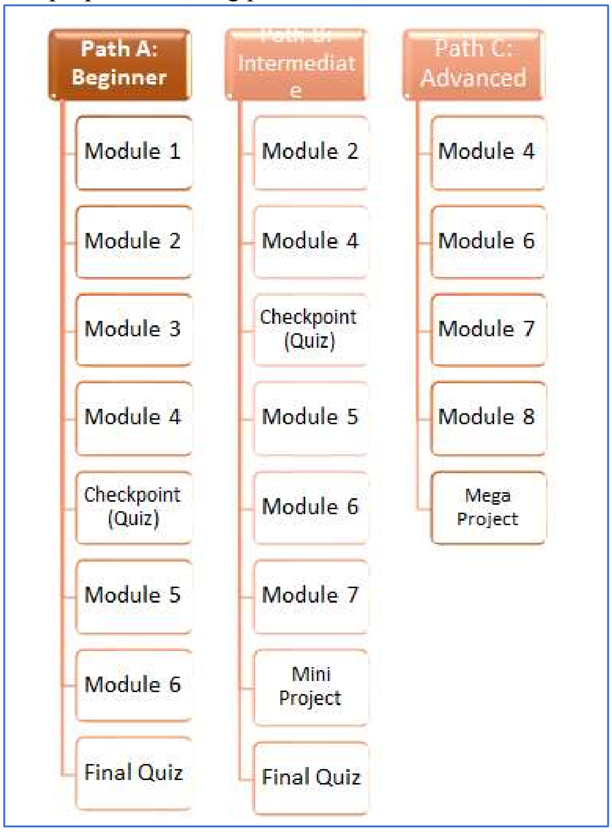
The system assesses prior knowledge through a quiz

Based on answers, the system places learners into one of three paths:

 ISSN: 3048-5320 (Online)



The proposed learning paths for students of different learning abilities is depicted in Fig. 3(a)-3(b).



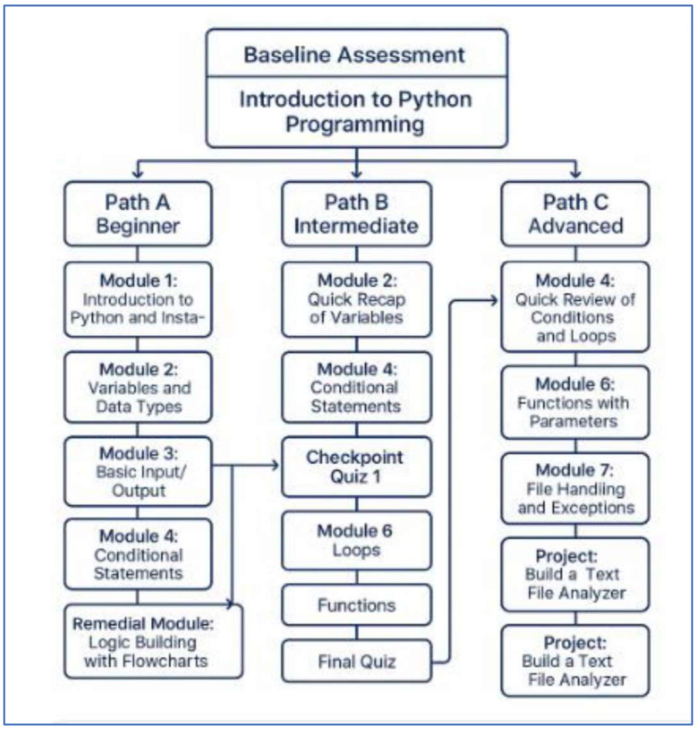


Fig. 3(a)-3(b). Learning Paths for Students of Different Learning Abilities

Description of the modules for different learning paths are outlined below:

Path A: Beginner
Module 1: Introduction to Python and Installation
Module 2: Variables and Data Types
Module 3: Basic Input/Output
Module 4: Conditional Statements
Checkpoint Quiz 1
Module 5: Loops
Module 6: Functions
Final Quiz
If the learner struggles with Module 4, the system inserts a remedial module: 'Logic Building with Flowcharts'.

Path B: Intermediate
Module 2: Quick Recap of Variables
Module 4: Conditional Statements
Checkpoint Quiz 1
Module 5: Loops
Module 6: Functions
Module 7: File Handling
Mini Project: Simple Calculator
Final Quiz

Path C: Advanced
Module 4: Quick Review of Conditions and Loops
Module 6: Functions with Parameters
Module 7: File Handling and Exceptions
Module 8: Simple Python Projects
Project: Build a Text File Analyzer
If performance drops below threshold in Module 7, learner is redirected to Path B for reinforcement.

 ISSN: 3048-5320 (Online)

The key points of adaptive learning are:

- Learning paths are defined dynamically based on entry behavior and ongoing performance.
- The system adapts both content difficulty and sequence to support mastery.
- Checkpoints and analytics inform decisions on path switching or additional support.

Literature Review

The intersection of personalized adaptive learning (PAL) and automated code evaluation has gained significant traction in programming education, aiming to improve engagement, assessment efficiency, and learning outcomes.

Personalized Adaptive Learning in Programming Education

Personalized Adaptive Learning systems tailor content delivery based on learner profiles, supporting more efficient and engaging educational experiences. A scoping review by González and Cechinel (2024) found that PAL implementations in higher education often rely on initial quizzes to inform content adaptation, with platforms like McGraw-Hill's Connect LearnSmart and Moodle being widely adopted. The study reported improved academic performance in 59% of cases and increased engagement in 36%.

In the specific context of programming education, personalized gamification has been explored as a strategy to enhance motivation. Chittimalli et al. (2023) analyzed 81 studies and concluded that combining personalization with gamification boosts student engagement and cognition. However, they noted that success often depends on the type of gamification used and its alignment with learner preferences.

Automated Code Evaluation Systems

Automated Code Evaluation Systems (ACES) offer scalable, immediate feedback on programming assignments, making them essential in large-scale education settings. Patel et al. (2023) provided a survey categorizing ACES into domains such as education, programming contests, and recruitment. Their findings emphasized the growing importance of integrating AI-driven tasks like bug detection to enhance ACES.

Similarly, Neri and Crescenzio (2023) reviewed various automated grading techniques, highlighting the trade-off between static and dynamic analysis, and the need for rich, formative feedback. They argued that while automation offers efficiency, complementing it with human evaluation can ensure a more holistic assessment approach.

Integration of PAL and Automated Assessment

The fusion of PAL with automated assessment systems is proving to be a powerful approach in programming pedagogy. An example is GitSEED, a Git-based automated assessment platform (Zhou et al., 2024), which provides tailored feedback while reinforcing industry-relevant tools such as version control. The tool enables educators to integrate custom assessment pipelines and monitor student progress through analytics.

Further, using flipped classrooms alongside automatic code evaluation has yielded positive outcomes. Alkhatib et al. (2023) applied Kirkpatrick's evaluation model to assess this strategy in an introductory CS course. Their results showed improved student satisfaction, engagement, and learning outcomes, validating the synergy between adaptive content delivery and automatic assessment.

Conceptual Model Design

The conceptual model of the Flask-based code evaluation system revolves around students submitting their code through a web interface, which is then evaluated against predefined test cases using the Piston API. Each submission includes details like roll number, name, code, marks, and grade, and is stored in a SQLite database via SQLAlchemy. Test cases are read from a file, and the code is executed dynamically to compare actual outputs with expected ones for grading. The system categorizes students based on grades and visualizes the distribution using Plotly charts. It also offers functionality to view all submissions and export the results as a downloadable PDF report, providing a complete solution for automatic code assessment.

Key Entities & Concepts

The key entities of the conceptual model are described below and outlines in Fig. 4.

User Submission

- Represents the code submitted by a student.
- Contains data like rollno, name, code, marks, and grade.

Code Evaluation

- This is the process where student-submitted code is executed using test cases via the Piston API.
- Each test case consists of an input and an expected output.

Test Case

- Defined in testcases.txt.
- Used to verify correctness of submitted code.

Result & Grading

- Based on the number of test cases passed and error-free execution.
- Grade and marks are computed and stored in the database.

Database (SQLite via SQLAlchemy)

- Stores all student submissions for record-keeping and reporting.
- One main table: Submission.

Visualization

- Plotly is used to group and display students by grade.
- Helps identify performance clusters.

Exporting Reports

- Submissions can be exported as PDF using xhtml2pdf.

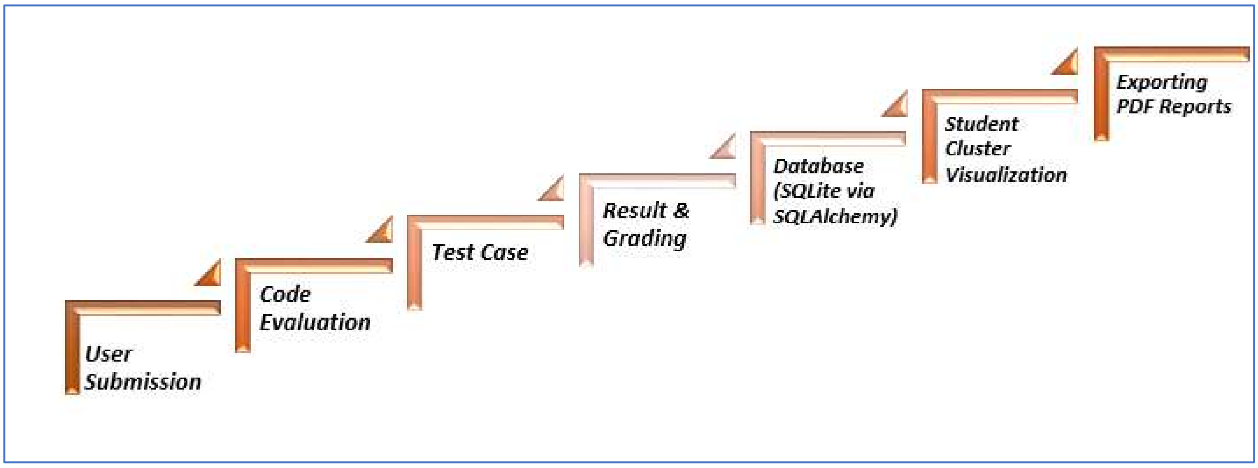


Fig. 4. Key Elements of Conceptual Model

Components and Their Relationships
Flask App (app.py)
Main controller managing all routes and actions.

Templates (Jinja2 HTML)
Views for form display, results, charts, and reports, for example, form.html, result.html, submissions.html.
Static Files
For CSS, JS, images (not detailed but assumed in static/ folder).
Piston API
External code execution engine.
Database (submissions.db)
Stores structured submission data for further operations.

Model Implementation

The model proposed in Section 3. is implemented in Python with SQLite as backend. The application is hosted on the Flask server and is responsive.

Folder Structure

The root folder structure of the web application is depicted in Fig. 4(a) and different views are hosted in 'templates' folder as outlined in Fig. 4(b).

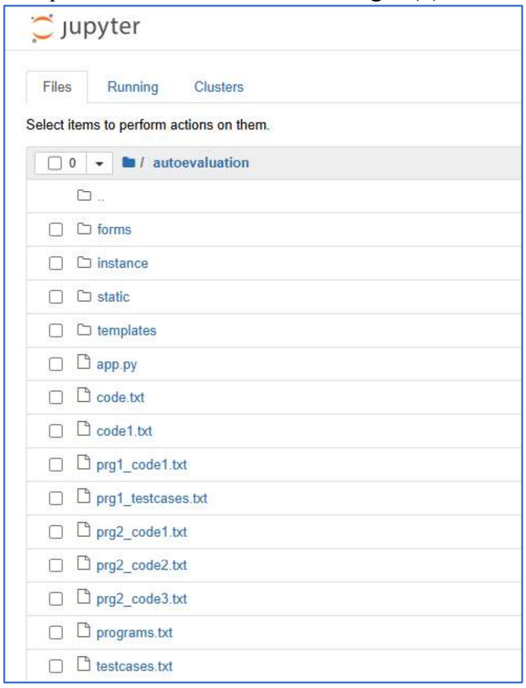


Fig. 4(a). Web Root Folder Structure

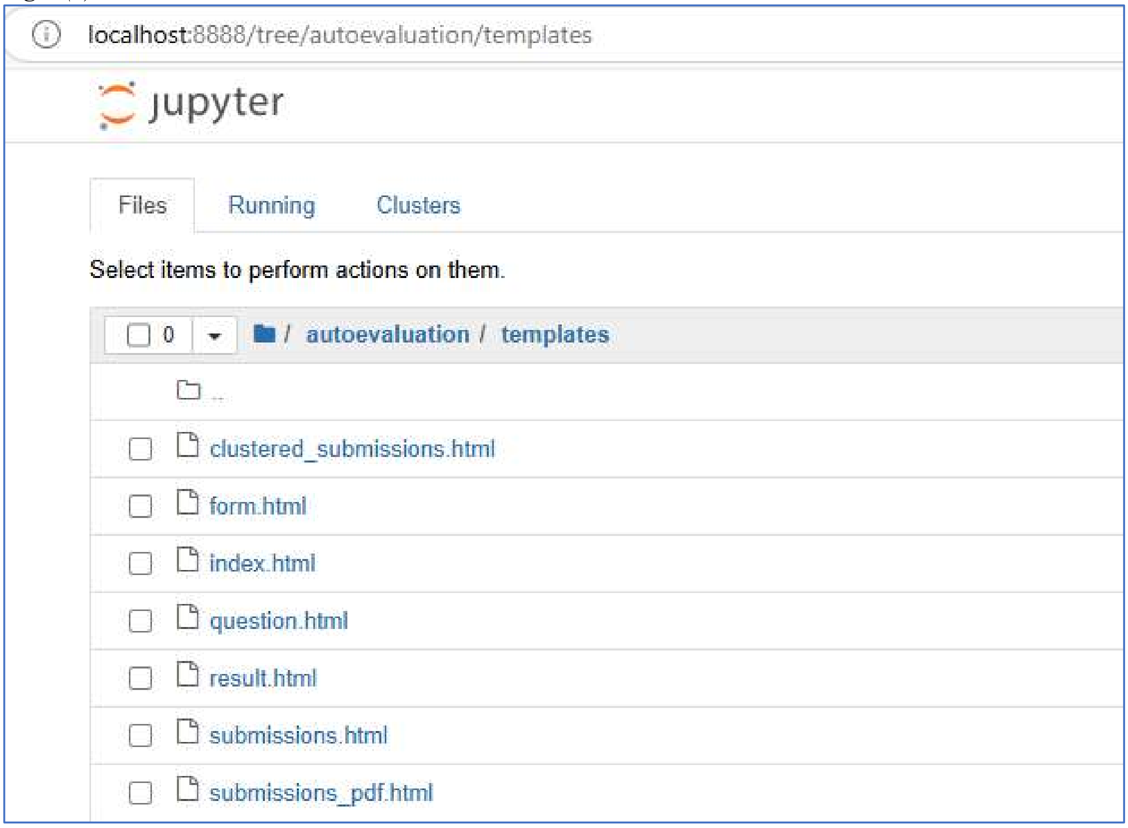
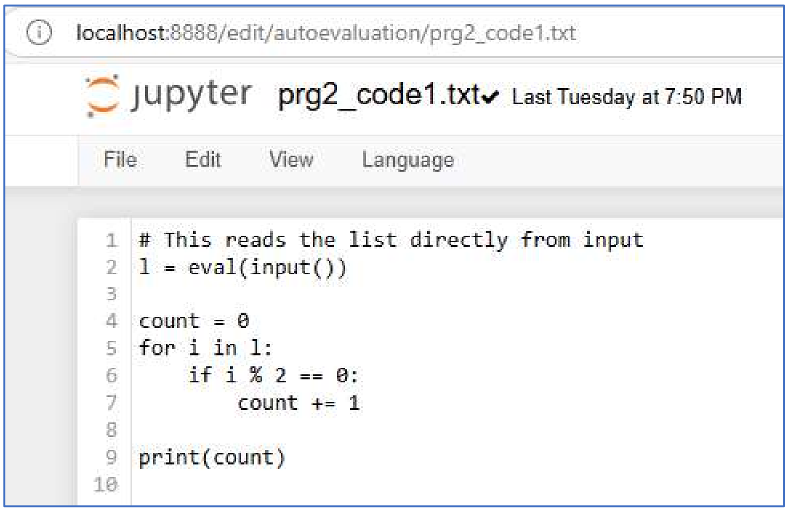


Fig. 4(b) Contents of Templates Folder Hosting Different Views of Web App

Format of Source Code File

Format of source code file to be uploaded by a student should conform to a specific structure as depicted in Fig. 5.

 ISSN: 3048-5320 (Online)



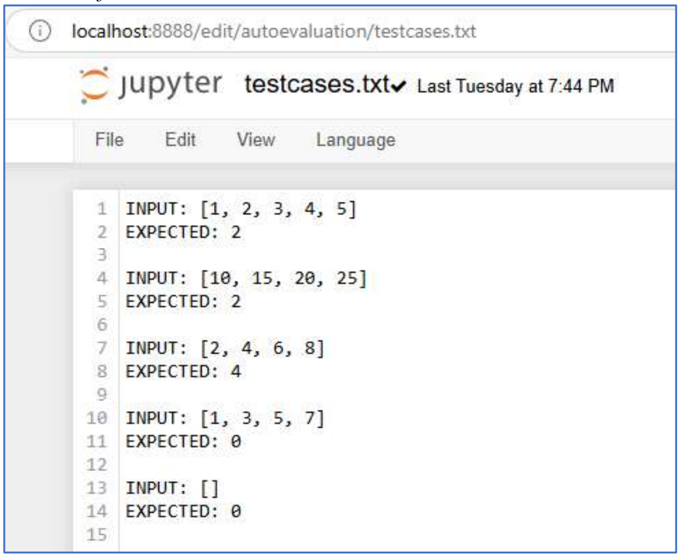

```
# This reads the list directly from input
l = eval(input())

count = 0
for i in l:
    if i % 2 == 0:
        count += 1

print(count)
```

Fig. 5. Format of Source Code File

Format of Test Cases File




```
INPUT: [1, 2, 3, 4, 5]
EXPECTED: 2

INPUT: [10, 15, 20, 25]
EXPECTED: 2

INPUT: [2, 4, 6, 8]
EXPECTED: 4

INPUT: [1, 3, 5, 7]
EXPECTED: 0

INPUT: []
EXPECTED: 0
```

Fig. 6. Format of Test Cases File

Table 1. summarizes the routes defined in the Flask application along with their descriptions.

Table 1. Flask Application Routes

Route	Methods	Description
/	GET	Renders the index.html homepage.
/submit	GET, POST	Handles code submission. Displays the form (GET) and processes code (POST).
/submissions	GET	Displays all student submissions in a table using submissions.html.
/clear_submissions	GET	Deletes all records from the submissions database.
/export_submissions_pdf	GET	Generates a downloadable PDF of submissions using submissions_pdf.html.
/cluster_students	GET	Clusters students by grade and displays a Plotly bar chart with annotations.
/view_question	GET	Displays the programming question via question.html.

 ISSN: 3048-5320 (Online)

Experimental Results

The Flask application is hosted on the machine with IP 192.168.30.31 which accepts the incoming requests across the port 5000 as demonstrated in Fig. 7. The application is executed using the following statement

app.run(host='0.0.0.0', port=5000, debug=True)

and hence enables the application to be accessed from any machine connected to the same network.

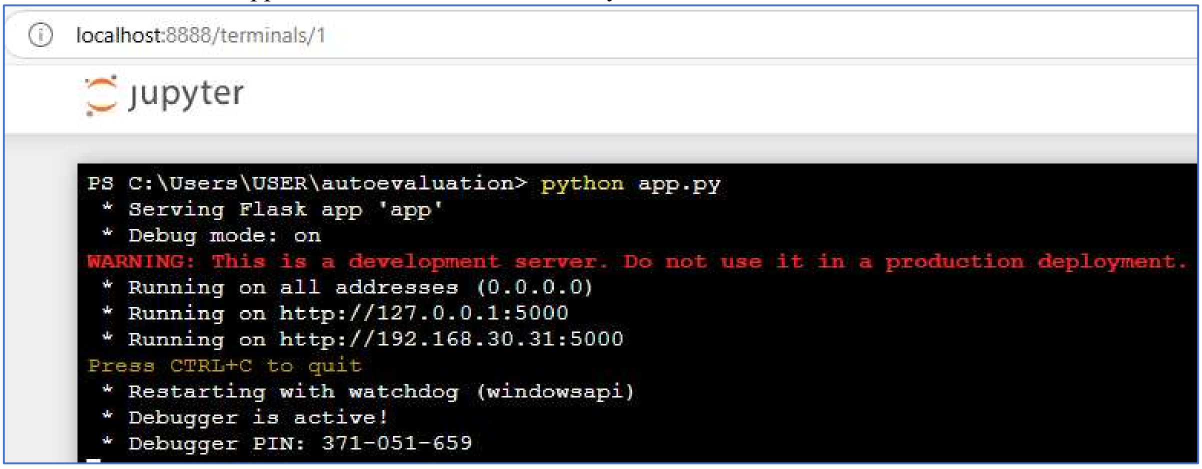


Fig. 7. Execution of Flask Application

Fig 8. Shows the home page of the web application.

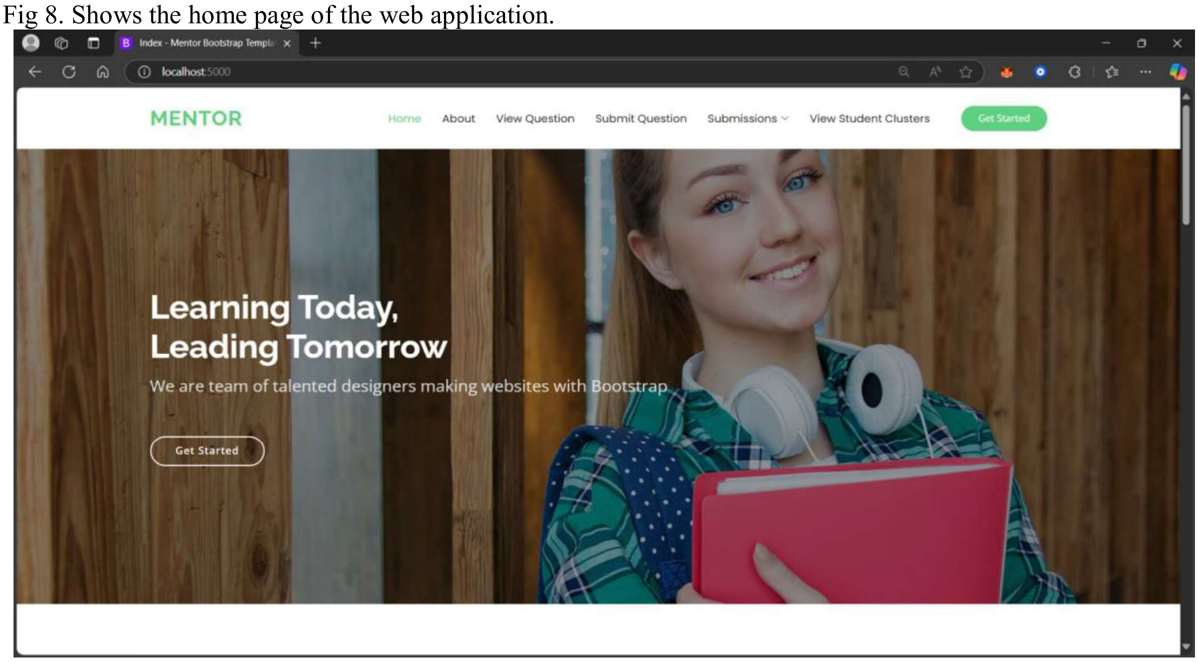


Fig 8. Home page of the web application.

Fig 9(a)-9(c) depict the student interaction with the web application which involves student viewing the question, submitting the source code to the system for evaluation and viewing the results.

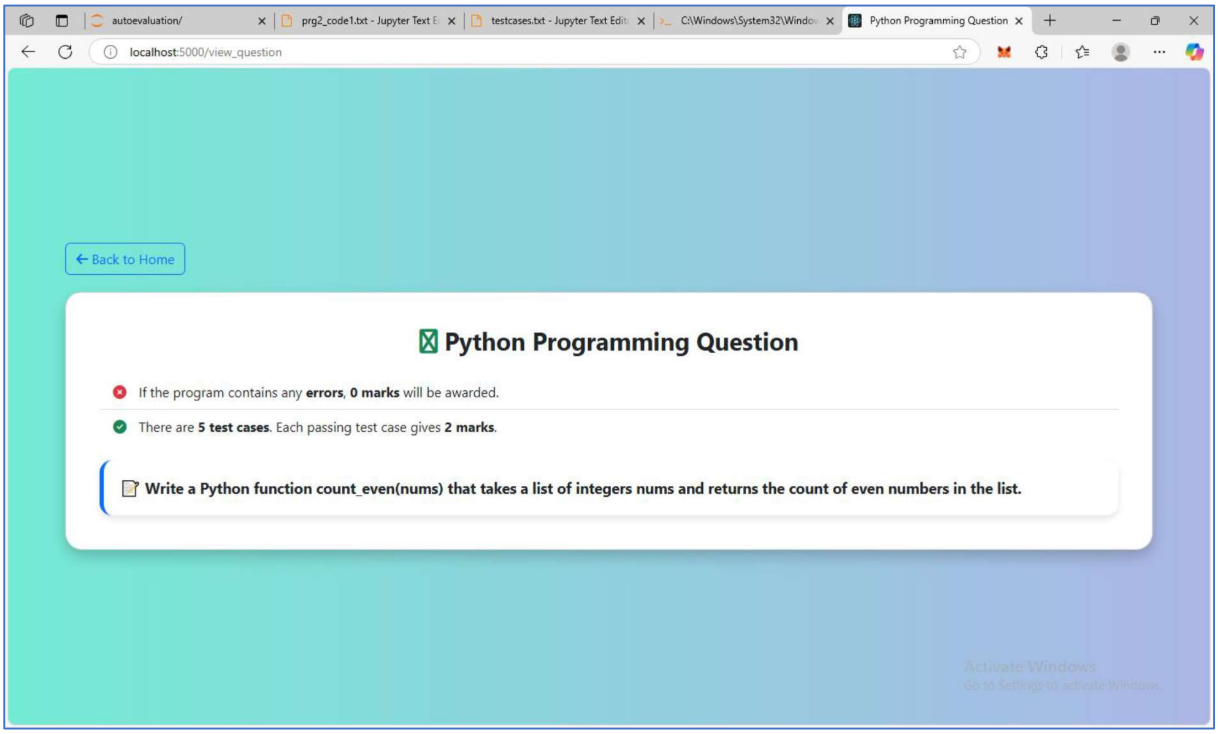


Fig 9(a). Student Viewing Question on Python Programming

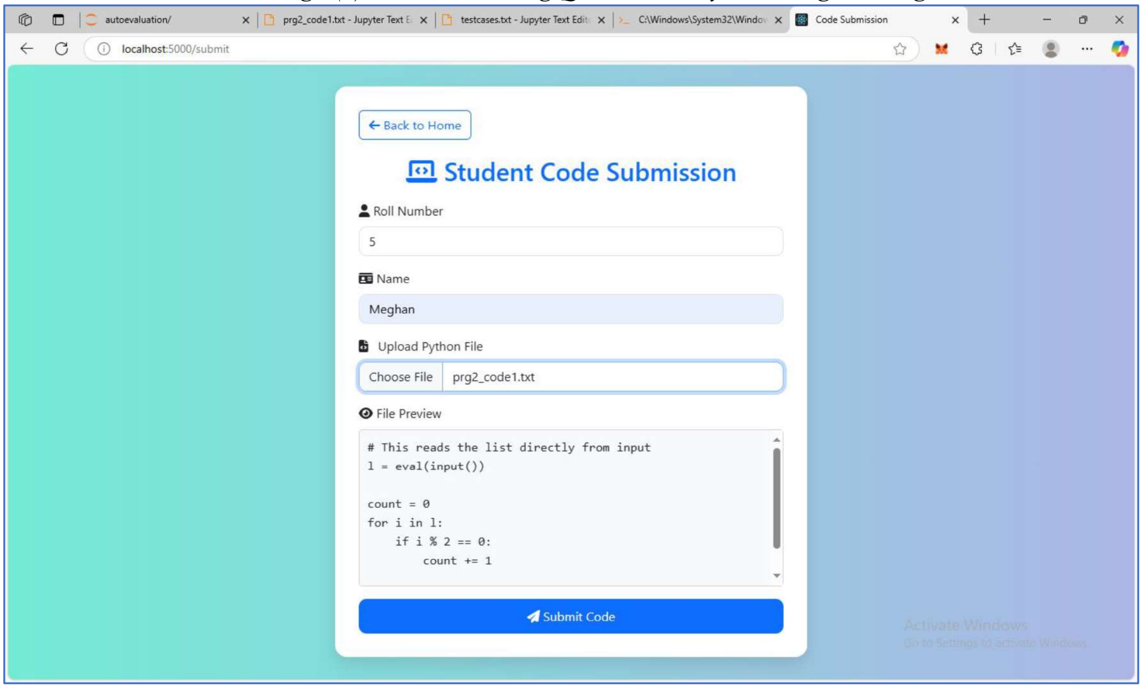


Fig 9(a). Student Submitting Source Code to the System for Evaluation

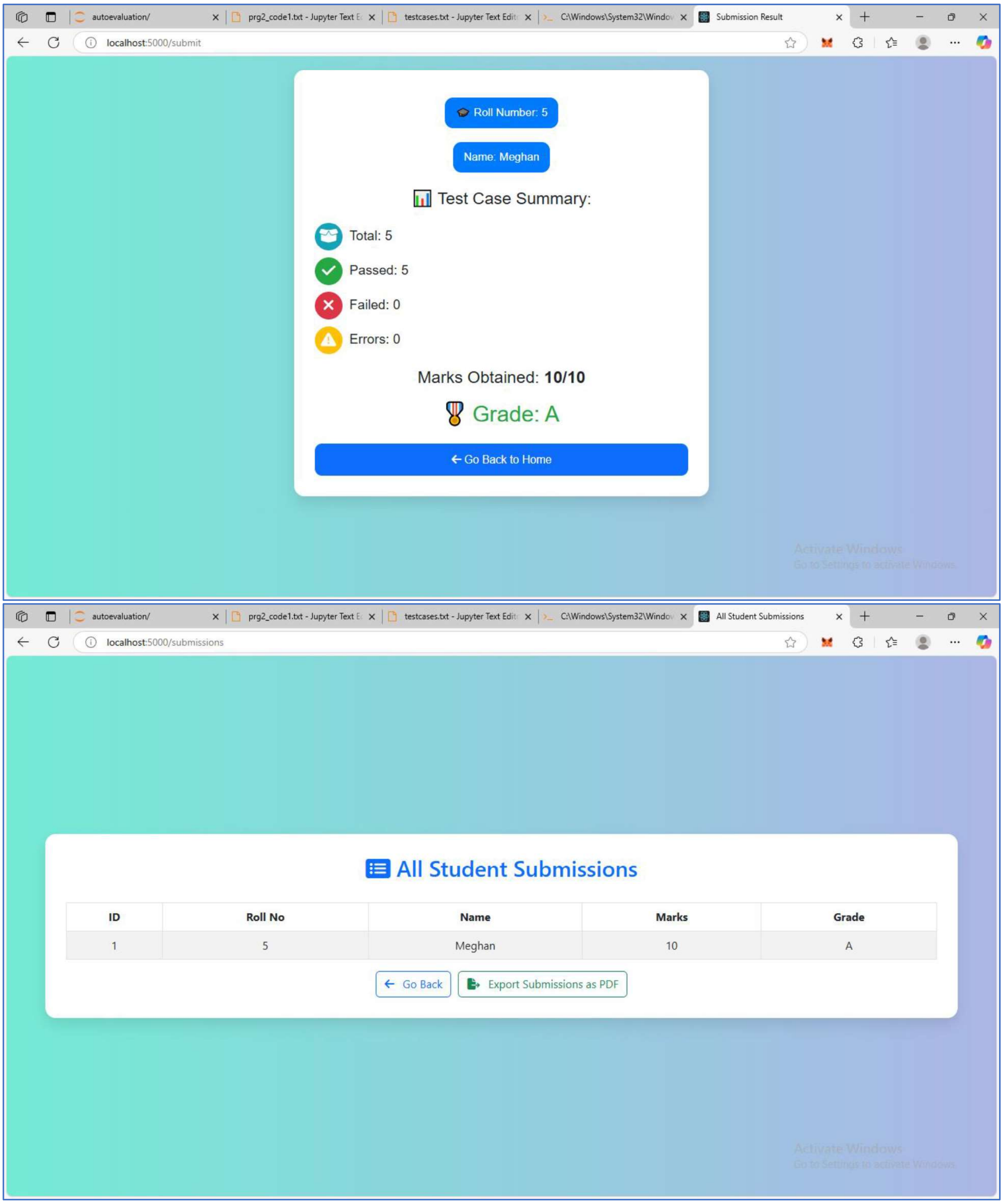
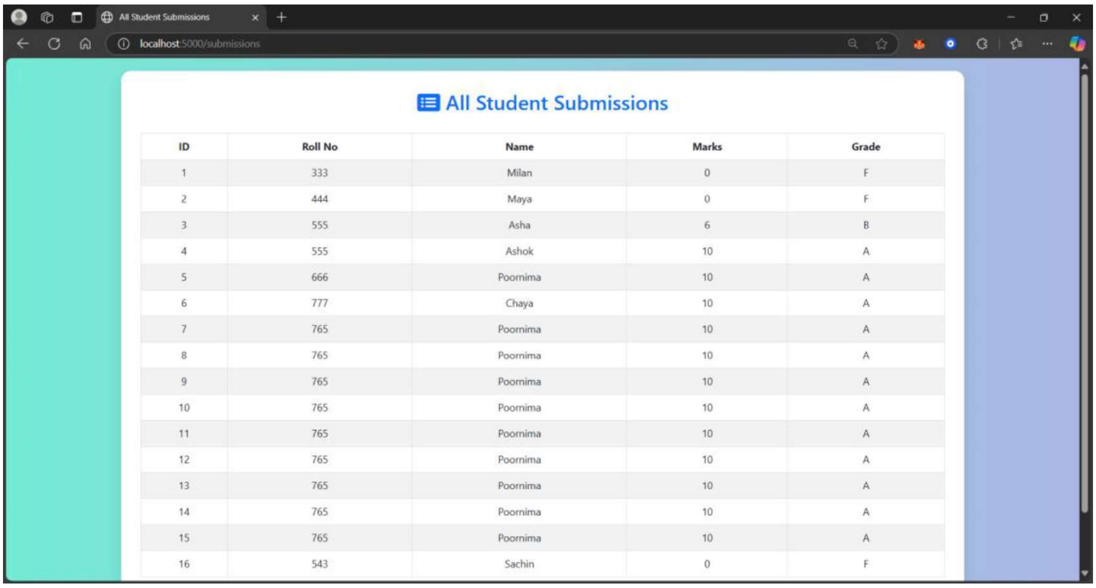


Fig 9(c). Student Viewing Result

The class coordinator can view all submissions made by students along with their marks and grade as shown in Fig. 10.

 ISSN: 3048-5320 (Online)



All Student Submissions

ID	Roll No	Name	Marks	Grade
1	333	Milan	0	F
2	444	Maya	0	F
3	555	Asha	6	B
4	555	Ashok	10	A
5	666	Poornima	10	A
6	777	Chaya	10	A
7	765	Poornima	10	A
8	765	Poornima	10	A
9	765	Poornima	10	A
10	765	Poornima	10	A
11	765	Poornima	10	A
12	765	Poornima	10	A
13	765	Poornima	10	A
14	765	Poornima	10	A
15	765	Poornima	10	A
16	543	Sachin	0	F

Fig. 10. Class Coordinator's View of Bulk Submissions

Fig. 11. shows the student clusters where each student is placed in one of the clusters:

- Needs Attention
- Slow learner
- Moderate Learner
- Advanced Learner

based on the scores as depicted in Table 2.

Category	Description	Sample Criteria
Needs Attention	Faces serious difficulty in understanding concepts and applying them.	Score < 30%
Slow Learner	Struggles with core ideas but shows potential with support and practice.	Score 30% – 50%
Moderate Learner	Understands with effort, performs reasonably well with guidance.	Score 50% – 75%
Advanced Learner	Grasps concepts quickly, solves problems accurately with minimal assistance.	Score > 75%

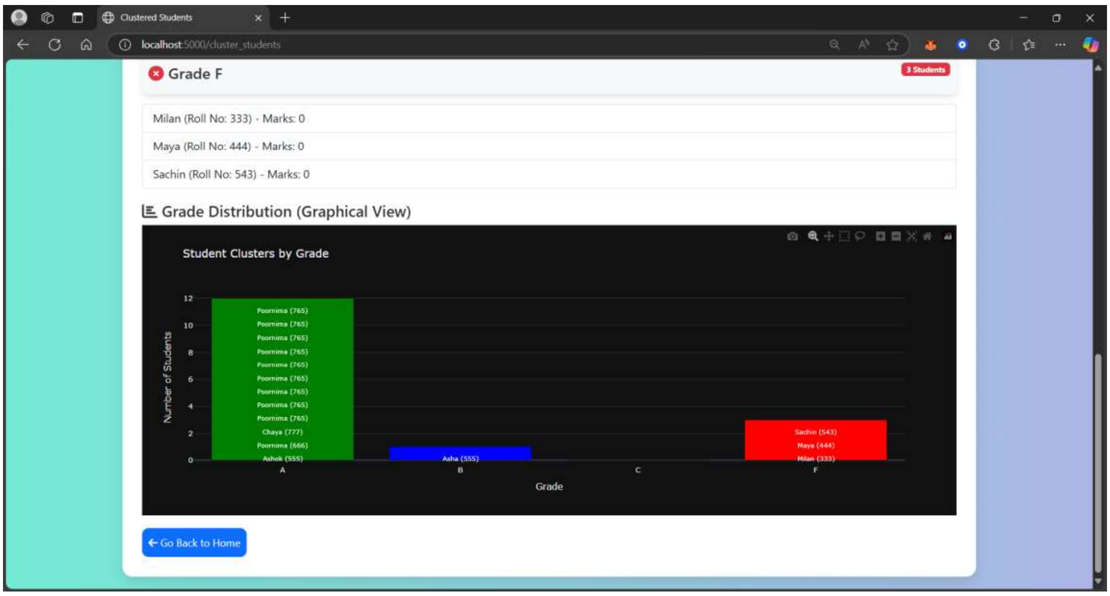


Fig 11. Student Performance Clusters

Classification pyramid for the clusters is shown in Fig 12.

ISSN: 3048-5320 (Online)

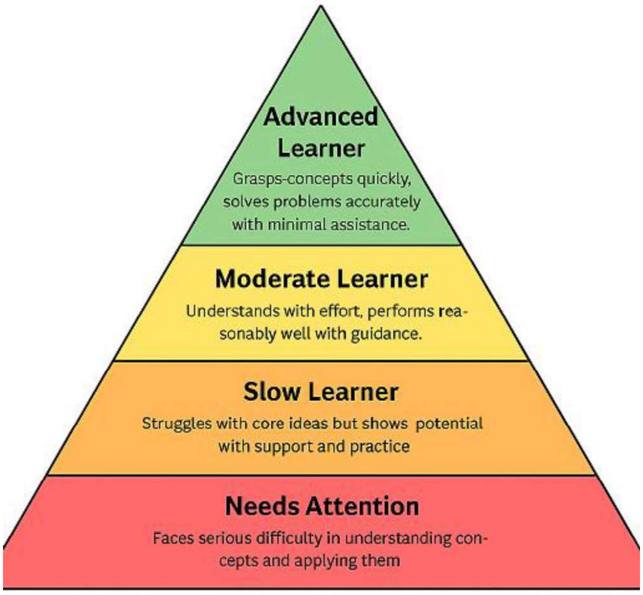


Fig 12. Classification Pyramid for Student Clusters

Conclusion and Scope for Future Work
Conclusion

The proposed system effectively demonstrates how modern teaching pedagogy, particularly Personalized Adaptive Learning, can be integrated with automated code evaluation to enhance programming education. By tailoring learning paths to individual student needs and providing real-time assessment and feedback, the system supports improved learning outcomes, engagement, and instructional efficiency. The use of technologies such as Flask, SQLAlchemy, and the Piston API ensures seamless automation and scalability, while visual analytics aids educators in identifying performance trends and providing targeted support.

Scope for Future Work

Future enhancements could include integrating AI-based code analysis for deeper error diagnostics, expanding the platform to support multiple programming languages, and incorporating gamified elements to boost motivation. Further, adaptive learning could be made more robust by employing machine learning models that predict learner behavior and recommend personalized interventions. Support for multilingual content and integration with Learning Management Systems (LMS) would also broaden the system's applicability across diverse educational environments.

References:

Alkhatib, A., Akour, M., & Al-Marashdeh, I. (2023) 'Evaluating flipped classroom and automatic code evaluation model using Kirkpatrick's framework', *Education and Information Technologies*, [online] Available at: https://doi.org/10.1007/s10639-023-11678-9 [Accessed 2 May 2025].

Chittimalli, P., Saini, R., & Santhosh Kumar, D. (2023) 'Personalized gamification in programming education: A systematic literature review', *arXiv preprint*, arXiv:2309.12362. Available at: https://arxiv.org/abs/2309.12362 [Accessed 2 May 2025].

González, C. and Cechinel, C. (2024) 'Personalized adaptive learning in higher education: A scoping review', *Heliyon*, 10(5), p.e26126. Available at: https://www.sciencedirect.com/science/article/pii/S2405844024156617 [Accessed 2 May 2025].

Neri, F. and Crescenzio, L. (2023) 'Automated assessment of source code in computer science education: A review', *Journal of Learning Analytics and Assessment*, 1(1), pp.1–26. Available at: https://www.mdpi.com/2674-113X/1/1/2 [Accessed 2 May 2025].

Patel, J., Sharma, A., & Shukla, S. (2023) 'Survey on automatic source code evaluation systems', *arXiv preprint*, arXiv:2307.08705. Available at: https://arxiv.org/abs/2307.08705 [Accessed 2 May 2025].

 ISSN: 3048-5320 (Online)

Zhou, Y., Fan, J., & Elbaum, S. (2024) 'GitSEED: Git-Backed Automated Feedback and Analytics for Scalable Programming Assessment', *arXiv preprint*, arXiv:2409.07362. Available at: https://arxiv.org/abs/2409.07362 [Accessed 2 May 2025].

Ihantola, P., Ahadi, A., Karavirta, V. and Hellas, A. (2015) 'Review of recent systems for automatic assessment of programming assignments', *Proceedings of the 2015 ITiCSE on Working Group Reports*, pp. 15–32. Available at: https://doi.org/10.1145/2858796.2858799 [Accessed 2 May 2025].

Kumar, D. and Singh, M. (2021) 'A review of adaptive learning systems in education: Current trends and future directions', *Education and Information Technologies*, 26(2), pp. 1237–1259. Available at: https://doi.org/10.1007/s10639-020-10345-4 [Accessed 2 May 2025].

Keuning, H., Jeuring, J. and Heeren, B. (2018) 'A systematic literature review of automated feedback generation for programming exercises', *ACM Transactions on Computing Education (TOCE)*, 19(1), pp. 1–43. Available at: https://doi.org/10.1145/3231711 [Accessed 2 May 2025].

Brusilovsky, P. and Millán, E. (2007) 'User models for adaptive hypermedia and adaptive educational systems', in *The adaptive web*. Berlin, Heidelberg: Springer, pp. 3–53. Available at: https://doi.org/10.1007/978-3-540-72079-9_1 [Accessed 2 May 2025].

Ihantola, P., Korhonen, A., & Nikula, U. (2010) 'How to study programming on the web? – The overview of reviews', *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pp. 53–62. Available at: https://doi.org/10.1145/1930464.1930472 [Accessed 2 May 2025].