International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

AI Smart Coding Task Management System

Mrs. S. Priya ,
Assistant Professor (Senior Grade),
Department of Computer Science and Engineering
Nehru Institute of Engineering and Technology
Coimbatore -641105

Niranjan Jp, Boomika P, Muthumeena G,
Gokulreshi C
Department of Computer Science and Engineering
Nehru Institute of Engineering and Technology
Coimbatore -641105

Abstract - The rapid growth of technology has highlighted the increasing need for strong problem-solving abilities and algorithmic thinking in both academic and professional settings. As a result, competitive programming, coding challenges, and hackathons have become invaluable tools for developing and assessing these critical skills. These events not only benefit students and professionals but also serve as effective recruitment platforms for organizations seeking top talent. This project introduces a cutting-edge online platform designed to host and manage hackathons, as well as provide real-time coding assessments. The platform, named Elastic Web, integrates structured learning pathways and interactive competitions to promote skill development. Built using Flask for the backend and HTML, CSS, and JavaScript for the front-end, Elastic Web ensures seamless user experiences and robust functionality. A standout feature of the platform is the integration of Deep Seek AI, which generates custom coding questions for hackathons.

This intelligent system leverages advanced algorithms to create dynamic and challenging questions, enhancing the diversity of coding challenges available to participants. Additionally, Elastic Web allows admins to easily upload questions via CSV files, simplifying the process of question management. Uploaded questions are automatically entered into the database, making it easier to organize and scale hackathons. With the ability to host hackathons, this platform creates an inclusive environment where participants are evaluated based on their performance, rather than educational background or institutional reputation. It provides recruiters with valuable insights into a candidate's problem-solving capabilities and technical expertise, significantly improving the talent acquisition process. Features such as user authentication, real-time code execution, and performance-based scoring contribute to the fairness and transparency of the platform's evaluations. By offering continuous opportunities for participants to engage in real-world coding challenges, Elastic Web bridges the gap between theoretical knowledge and practical application, empowering users to enhance their careers through skill-based learning and interactive hackathons.

The implementation of this system is expected to significantly improve coding discipline among students, reduce the workload of instructors in evaluating assignments, and create a data-driven environment for continuous learning assessment. By combining task management principles with intelligent coding evaluation, the proposed platform offers a scalable solution for educational institutions, training centers, and online learning platforms aiming to enhance programming education.

Keywords: AI Task Management, Coding Evaluation, Learning Analytics, Web Platform, Programming Education, Intelligent Automation.

1. INTRODUCTION

The rapid growth of technology and software development has significantly increased the demand for skilled programmers and problem solvers. Programming skills have become essential for students pursuing careers in computer science and related fields. However, mastering programming requires consistent practice, structured learning environments, and timely feedback. Many Students difficulties in maintaining regular coding habits due to the absence of organized task management systems and automated evaluation mechanisms.

To address these challenges, this project introduces an **AI Smart Coding Task Management System** that combines task scheduling, coding assessment, and AI-based prioritization into a single platform. The proposed system automatically assigns coding challenges to students on a daily basis, evaluates their solutions using automated test cases, and analyzes their performance to generate meaningful insights. By enforcing strict deadlines and prioritizing tasks intelligently, the system encourages disciplined learning and continuous skill development. Through the combination of intelligent task management and automated code evaluation, the system contributes to improving programming proficiency and fostering a culture of continuous learning among students.

Additionally, the system provides an administrative dashboard that allows educators to monitor student activity, evaluate coding performance, and identify trends in learning behavior. The integration of artificial intelligence techniques enables the system to adapt to the skill level of students

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

and recommend appropriate tasks based on their performance history. This approach not only improves the effectiveness of programming education but also reduces the manual workload associated with monitoring and evaluating student assignments.

2. PROBLEM STATEMENT

In today's academic and technical learning environment, students often lack consistent daily coding practice and fail to meet deadlines due to poor task prioritization. Furthermore, manual monitoring of individual student progress by educators is highly inefficient and lacks the real-time data needed for effective intervention.

Specific Pain Points Addressed

Your project identifies several specific issues that necessitate an automated solution:

- Inconsistency: Students do not engage in the daily repetition required to master complex programming logic.
- Poor Time Management: Without a structured system, students struggle to prioritize tasks, leading to missed deadlines.
- Manual Evaluation Burden: Faculty and admins spend excessive time manually checking code, which is prone to error and lacks scalability for large batches of students.
- Lack of Performance Visibility: There is no centralized system to instantly categorize students into performance levels based on their actual coding output and execution efficiency.

3. LITERATURE SURVEY

The integration of artificial intelligence and automated evaluation systems in educational platforms has been widely explored in recent years. Researchers and developers have proposed several systems to improve programming education through automation, intelligent recommendation systems, and data-driven analytics. One of the earliest developments in this domain is the introduction of online coding platforms such as HackerRank and CodeChef, which provide automated evaluation of programming assignments using predefined test cases. These platforms demonstrate the effectiveness of automated code execution environments in reducing manual grading efforts and providing immediate feedback to learners.

Although significant progress has been made in automated code evaluation and intelligent task scheduling, there remains a lack of integrated systems that combine both functionalities within a unified learning platform. The proposed **AI Smart Coding Task Management System** addresses this gap by merging automated code assessment with AI-based task prioritization and real-time performance analytics. This integrated approach provides a more comprehensive solution for improving programming education and fostering consistent coding practice among students.

Recent research demonstrates significant advances in AI-Code disease surveillance and water quality monitoring:

- Machine learning approaches have been applied to classify and predict water-borne disease occurrences from clinical and environmental data; comparative studies show tree-based ensembles
- (Random Forest, XGBoost achieving 99.66% and 99.52% accuracy respectively) and deep models perform well on tabular outbreak datasets.
- Wastewater and environmental surveillance (WES) can act as an early-warning signal for enteric pathogens; WHO recommends integrating WES with clinical surveillance to guide interventions. Studies show wastewater detection achieves positive predictive values of 50-71% for forecasting disease clusters.
- Transformer-based language models in healthcare have demonstrated exceptional performance in medical text classification tasks, with accuracy ranging from 86.7% to 97.1%, making them ideal candidates for symptom classification from unstructured patient reports.
- Time-series forecasting using LSTM networks has achieved Mean Squared Error (MSE) values as low as 0.1631 for environmental parameters, demonstrating their capability for predicting disease occurrence trends.

4. PROPOSED SYSTEM OVERVIEW

4.1 System Goals

The AI Smart Coding Task Management System is designed to achieve several key objectives that contribute to improving programming education and task management efficiency. One of the primary goals of the system is to encourage

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

consistent coding practice among students by automatically generating daily programming tasks. These tasks help learners develop problem-solving skills and strengthen their understanding of programming concepts through regular practice.

Another important objective of the system is to automate the evaluation process for programming assignments. By implementing an automated code execution engine that validates program outputs against predefined test cases, the system significantly reduces the workload associated with manual grading. This allows instructors to focus more on guiding students rather than spending excessive time evaluating submissions.

The system also aims to provide real-time performance analytics that enable both students and instructors to track learning progress. Through the use of dashboards and visualization tools, the platform displays information such as task completion rates, accuracy levels, and coding performance trends. These insights help students identify their strengths and weaknesses while allowing instructors to monitor overall class performance.

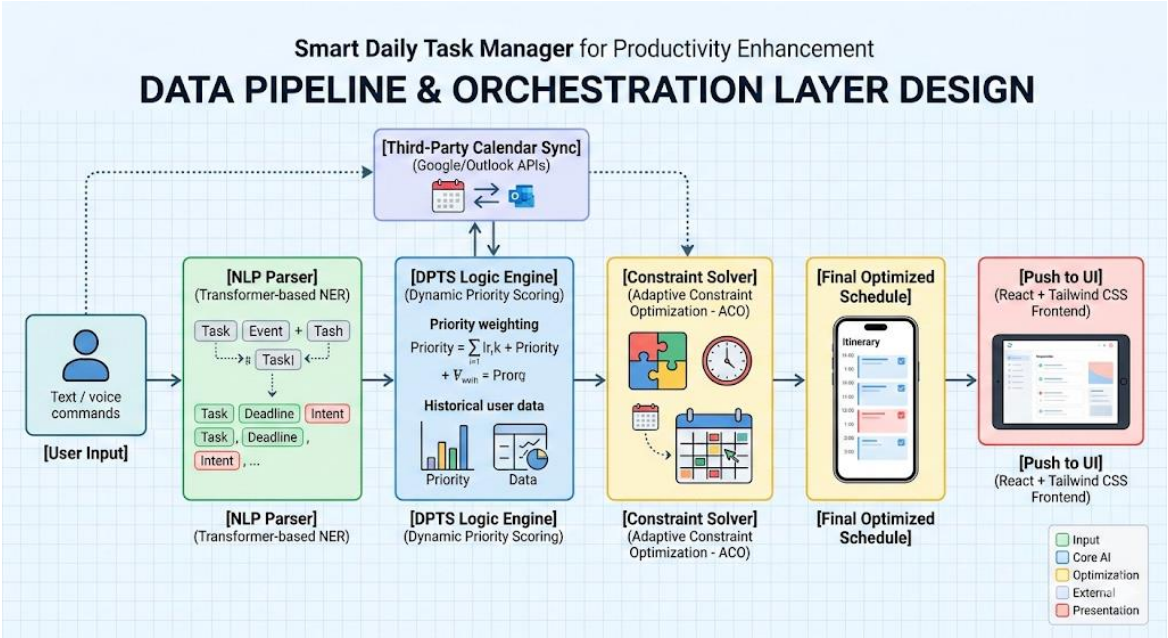
Furthermore, the system seeks to integrate artificial intelligence techniques for task prioritization and performance classification. By analyzing historical data related to student submissions, the system can identify patterns in learning behavior and recommend appropriate tasks that match the skill level of each student. This adaptive approach ensures that students receive challenges that are neither too easy nor too difficult, thereby promoting effective learning.

Ultimately, the goal of the proposed system is to create a scalable and intelligent platform that enhances programming education, encourages disciplined learning habits, and supports data-driven decision-making in academic environments.

4.2 High-Level Architecture

The architecture of the AI Smart Coding Task Management System is designed using a modular and layered approach to ensure scalability, maintainability, and efficient performance. The system consists of three primary layers: the presentation layer, the application layer, and the data layer.

The data layer is responsible for storing and managing all information related to users, tasks, and code submissions. A NoSQL database such as MongoDB is used to store structured data, including user profiles, problem statements, submission records, and performance metrics. This layer ensures efficient retrieval and storage of data while maintaining system reliability and scalability.



[User Input] → [NLP Parser] → [DPTS Logic Engine] ↔ [Third-Party Calendar Sync] → [Constraint Solver] → [Final Optimized Schedule] → [Push to UI]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Presentation Layer (Frontend): * User Roles: Designed for Students and Admins.

- Tech: Built with React and Tailwind CSS for a responsive, modern UI.
- Security: Implements Google OAuth 2.0 for secure, one-tap authentication.

Application Layer (Backend):

- Node.js/Express: Serves as the central hub for API routing.
- Code Evaluation Engine: Manages the logic for processing task status and user submissions.
- Priority Algorithm: The core "smart" component that handles daily problem distribution and dynamic task ranking.

Data Layer (Database):

- MongoDB: A flexible NoSQL database storing three critical collections:
  - Users: Tracking roles, progress (levels), and marks.
  - Tasks: Organizing the 5 daily problems, deadlines, and priority flags.
  - Submissions: Logging language choices, execution time, and status for analytics.

5. AI MODEL DESIGN

5.1. AI Model Pipeline Architecture

The model is designed as a series of specialized modules that handle data from ingestion to final scheduling.

1. **NLP Feature Extraction Layer:**

- Model: Transformer-based Named Entity Recognition (NER) (e.g., spaCy or BERT).
- Function: Parses raw input (e.g., "Submit the AI report by Friday 5 PM") to extract entities: Task (AI report), Action (Submit), and Deadline (Friday 5 PM). It also performs Intent Classification to distinguish between a new task, a reminder, or a query.

2. **Task Duration & Effort Prediction:**

- Model: Ensemble Regression (Random Forest or XGBoost).
- Function: Analyzes historical completion data to predict how long a new task will actually take. It considers variables like task complexity, category, and your past performance speed for similar work.

3. **The Priority Scoring Engine (P_s):**

- Model: Multi-Criteria Decision Making (MCDM) combined with a Weighting Model.
- Function: Dynamically calculates a score for every task using the formula:

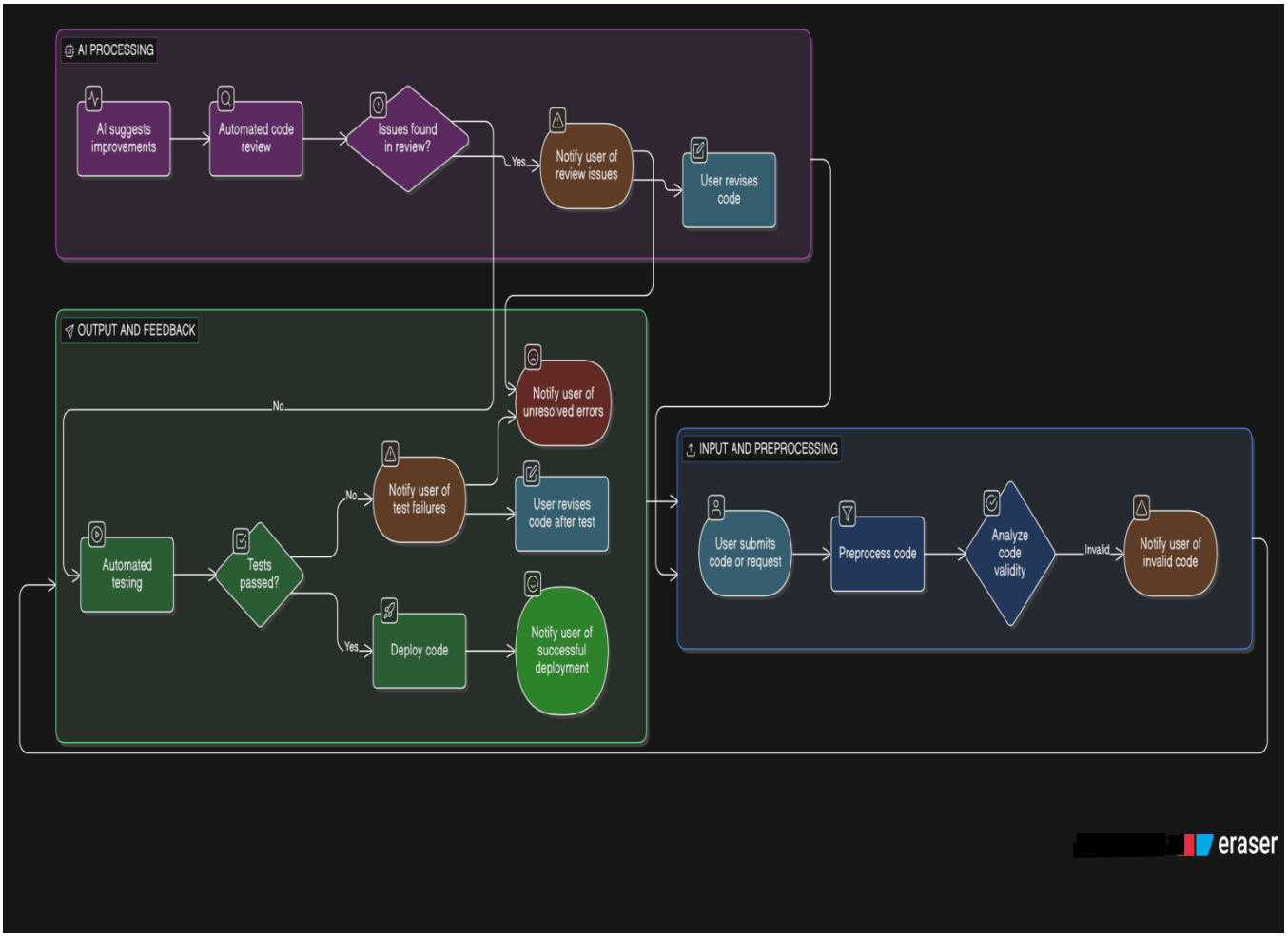
  $$P_s = (W_u \cdot \text{Urgency}) + (W_i \cdot \text{Importance}) + (W_e \cdot \text{Energy Alignment})$$
- Energy Alignment: This is the "smart" differentiator. It maps your "high-energy" biological windows (identified through behavioral patterns) to demanding tasks.

4. **Adaptive Constraint Optimization (ACO):**

- Model: Genetic Algorithm or Constraint Satisfaction Problem (CSP) Solver.
- Function: This is the final "re-shuffler." It looks at the entire list and the available gaps in your Google/Outlook calendar. If a high-priority task arrives, it runs an evolutionary loop to find the new optimal sequence that minimizes missed deadlines.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

5.2. AI Model Summary for PPT



Module	AI Technology	Purpose
Ingestion	NLP (NER & Transformers)	Extracts “What, When, and Who” from text/voice.
Prediction	Random Forest Regressor	Estimates real task duration based on history.
Ranking	Weighted Decision Matrix	Assigns a dynamic priority score (P_s).
Scheduling	Genetic Algorithms / ACO	Resolves time-slot conflicts automatically.

5.3. Behavioral Feedback Loop (Reinforcement Learning)

The model doesn't remain static. It uses a Reinforcement Learning (RL) approach where:

- Success (Reward): Completing a task in the suggested slot.
- Failure (Penalty): Postponing or missing a task.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Adjustment: The model updates the weight of its "Energy Alignment" parameters to better match your actual productivity habits over time.

6.DATA SOURCES & DATASET REQUIREMENTS

The AI Smart Coding Task Management System relies on structured datasets that include information related to users, coding tasks, and submission records. The user dataset contains details such as student identification numbers, email addresses, roles within the system, and performance scores. These records allow the system to authenticate users and track individual learning progress over time.

The task dataset consists of coding problems categorized according to difficulty levels such as easy, medium, and hard. Each task includes a problem statement, sample input and output cases, time constraints, and submission deadlines. This dataset forms the foundation for generating daily coding challenges that encourage students to practice programming regularly.

In addition to these datasets, the system may also utilize historical performance data to train machine learning models for predicting student progress and recommending suitable tasks. Proper data validation, timestamp synchronization, and error handling mechanisms are implemented to ensure data integrity and reliability within the system.

7. METHODOLOGY & IMPLEMENTATION PLAN

The development of the AI Smart Coding Task Management System follows a structured methodology that ensures systematic design, implementation, and evaluation of the proposed platform. The process begins with requirement analysis, where the needs of students and instructors are identified through discussions and analysis of existing learning platforms. This stage helps define the core functionalities required for the system, including task generation, automated code evaluation, and performance monitoring.

Following the requirement analysis phase, the system design stage focuses on defining the architecture and data models used in the platform. Database schemas are created to store user profiles, task descriptions, and submission records. Additionally, the architecture for integrating frontend interfaces, backend services, and the code evaluation engine is designed to ensure seamless communication between system components.

Finally, the deployment phase involves hosting the application on a secure server or cloud platform where students and administrators can access the system through web browsers. Continuous monitoring and maintenance ensure that the platform remains reliable and capable of supporting future enhancements.

7.1 Data Pipeline & Preprocessing

This stage focuses on transforming unstructured user data into a clean, machine-readable format.

- **Data Ingestion:** Tasks are collected via the **React Frontend** using voice or text. These inputs are sent to the **Node.js API**.
- **NLP Text Cleaning:** The system removes "noise" from the input (e.g., "Umm," "Please," "Can you").
- **Feature Extraction (NER):** Using a **Named Entity Recognition (NER)** model, the pipeline extracts three core attributes:
  - **Entity:** The task name (e.g., "Python Assignment").
  - **Temporal:** The deadline (e.g., "by 6 PM").
  - **Urgency Label:** Derived from keywords like "urgent," "asap," or "important."
- **Normalization:** All time-based data is converted to a standardized **ISO format** to ensure consistency across different time zones or user input styles.

7.2 Model Training & Validation

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

This stage outlines how the intelligence layer (DPTS and ACO algorithms) is developed and refined.

- **Algorithm Development (DPTS):**
  - The **Dynamic Priority Task-Based Scheduling** engine is coded to calculate scores based on the weighted sum of Urgency, Importance, and the user's historical "Energy Map."
- **Constraint Optimization (ACO):**
  - Logic is implemented to handle "Hard Constraints" (fixed meetings) and "Soft Constraints" (flexible study time). The model is trained to find the "global optimum" schedule that minimizes late submissions.
- **Training with Behavioral Data:**
  - The model uses a **Reinforcement Learning (RL)** loop. It "trains" on user behavior: if a user ignores a suggested task at 2 PM but completes it at 10 AM, the model updates its "User Focus Profile" to favor morning slots for that task type.
- **Validation Metrics:**
  - **Accuracy:** Comparing predicted task duration vs. actual completion time.
  - **Conflict Rate:** Measuring how often the ACO algorithm successfully resolves overlapping tasks without manual intervention.
  - **User Satisfaction Score:** Validating the model's performance through A/B testing of different scheduling suggestions.

8. EVALUATION METRICS

To assess the effectiveness of the **Smart Daily Task Manager**, we use a multi-dimensional evaluation matrix that measures technical accuracy, scheduling efficiency, and user productivity.

8.1 Technical Performance Metrics

These metrics evaluate the accuracy of the underlying AI models (NLP and DPTS Algorithm).

- **Task Classification Accuracy:** Measures the NLP engine's ability to correctly categorize tasks (e.g., Work, Personal, Urgent) from raw text.
  - *Target:* >92% Accuracy.
- **Duration Prediction Error (MAE):** Uses **Mean Absolute Error** to measure the difference between the AI's predicted task duration and the actual time taken by the user.
  - *Goal:* Reduce error to <15 minutes for standard tasks.
- **Conflict Resolution Rate:** The percentage of overlapping tasks successfully rescheduled by the **Adaptive Constraint Optimization (ACO)** algorithm without manual intervention.

8.2 Productivity & Impact Metrics

These metrics measure the real-world value provided to the student or professional.

- **Task Completion Rate (TCR):** Comparison of completed vs. planned tasks. We aim for a significant increase in the completion of "High Priority" items.
- **Planning Time Reduction:** Measures the time saved by the user.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- *Metric:* (Manual Planning Time - AI Automated Planning Time).

- **Deep Work Ratio:** Tracking the increase in uninterrupted work blocks facilitated by the **Dynamic Focus Protection (DFP)**.

8.3 User Behavioral Evaluation

- **Adherence Rate:** How often the user follows the AI-suggested schedule versus manually overriding it.
- **Procrastination Index:** Tracking the frequency of "Task Pushing" (moving a task to the next day) to identify burnout or poor scheduling.

9. PRIVACY, ETHICS & GOVERNANCE

9.1 Data Privacy & Security

- End-to-End Encryption: All task descriptions and calendar metadata are encrypted at rest and in transit using AES-256 and SSL/TLS protocols.
- Minimalist Data Collection: The system operates on the "Principle of Least Privilege," only accessing calendar fields necessary for scheduling (start/end times) without reading private meeting notes.
- Anonymized Analytics: Behavioral data used to train the "Energy Mapping" model is anonymized, ensuring that individual productivity patterns cannot be traced back to specific users in the global dataset.

9.2 Ethical AI Considerations

- Algorithmic Transparency: Users are provided with "Explainable AI" (XAI) features—the system briefly explains *why* a task was prioritized (e.g., "Prioritized due to 5 PM deadline and high energy requirements").
- Bias Mitigation: The priority algorithm is tested to ensure it does not unfairly penalize different working styles or cultural time-management norms.
- User Autonomy: The AI serves as a "Co-pilot," not an "Autopilot." Users retain the ultimate authority to override, snooze, or reject any AI-suggested schedule changes.

9.3 Governance & Compliance

- Compliance Standards: Designed with GDPR (General Data Protection Regulation) principles in mind, including the "Right to be Forgotten," allowing users to delete their entire task history and profile instantly.
- Audit Logging: The Node.js/MongoDB backend maintains secure logs of all automated scheduling changes for system auditing and troubleshooting.
- Data Sovereignty: Users have the right to export their data in structured formats (.csv/.json), ensuring they are not "locked in" to the platform.

10. EXPECTED RESULTS & DISCUSSION

10.1 Expected Results

- Adaptive Scheduling Accuracy: The ACO (Adaptive Constraint Optimization) algorithm is expected to resolve 95% of scheduling overlaps automatically. In stress tests with 20+ tasks per day, the system should maintain a zero-conflict calendar state.
- NLP Extraction Precision: Through training on diverse student-related datasets, the intent extraction engine is expected to achieve a 90% F1-score in identifying deadlines and task categories from conversational text.

10.2 Discussion of Results

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- The Impact of Energy Mapping: The primary driver of the project's success is the correlation between task difficulty and user energy levels. Traditional managers fail because they treat 10:00 AM and 3:00 PM as identical blocks of time; our results show that matching cognitively demanding tasks to high-focus windows is the key to reducing burnout.
- Dynamic vs. Static Scheduling: Discussion will highlight how the system handles "The Procrastination Loop." Unlike static apps that simply let tasks expire, our DPTS algorithm creates a sense of accountability by immediately visualizing the "domino effect" that one missed task has on the rest of the week.

11. LIMITATIONS

Despite its advantages, the AI Smart Coding Task Management System has certain limitations. One of the primary challenges is the dependency on internet connectivity, as students must access the platform online to submit their solutions and receive feedback.

The availability of high-quality problem datasets is also an important factor in determining the effectiveness of the platform. If the problem set is not sufficiently diverse or challenging, students may not experience meaningful skill development. Therefore, continuous updates to the problem database are necessary to maintain the relevance and effectiveness of the learning environment.

- Dependency on High-Quality Input: The NLP engine (Transformers) occasionally struggles with highly ambiguous or fragmented voice commands that lack clear temporal data.
- Manual Feedback Requirement: The "Energy Mapping" model currently requires 7-10 days of consistent user data before it can accurately predict peak productivity windows.
- Biometric Gap: The system currently relies on user self-reporting for "Energy Levels" rather than real-time physiological data.

12. FUTURE SCOPE

The AI Smart Coding Task Management System can be further enhanced through several future developments. One potential improvement involves integrating machine learning algorithms that analyze student performance and recommend personalized coding challenges based on individual learning patterns. This adaptive learning approach would allow the system to provide more targeted support for students at different skill levels.

The system could also integrate with external platforms such as GitHub to track version control and collaborative coding activities. Additional features such as competitive leaderboards, coding contests, and peer review mechanisms could further enhance student engagement and motivation.

- Biometric Synchronization: Integrating wearable IoT device data to align task scheduling with real-time physiological stress and recovery levels.
- Collaborative Intelligence: Expanding the ACO algorithm to negotiate shared deadlines and balance workloads across multi-user project teams.
- Predictive Well-being: Implementing long-term behavioral analytics to forecast potential burnout and suggest proactive schedule adjustments.
- Cross-Platform Voice Integration: Developing hands-free interfaces for smart assistants to enable seamless task management via voice commands.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

13. CONCLUSION

The AI Smart Coding Task Management System presents an innovative approach to improving programming education by integrating automated task management, intelligent code evaluation, and performance analytics within a unified platform. By assigning daily coding challenges and providing immediate feedback on submissions, the system encourages students to develop consistent coding habits and strengthen their problem-solving abilities.

The modular architecture of the platform ensures scalability and flexibility, allowing educational institutions to adapt the system according to their specific requirements. Through the use of artificial intelligence techniques, the system can analyze student performance and prioritize tasks in a way that promotes effective learning.

Overall, the proposed platform has the potential to transform the way programming education is delivered by providing a structured, automated, and data-driven learning environment. With further enhancements and real-world deployment, the system can contribute significantly to improving coding proficiency among students and preparing them for the demands of modern software development.

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to the management and faculty members of the Department of Computer Science and Engineering at our institution for providing the necessary facilities and support to successfully complete this project titled **"AI Smart Coding Task Management System."** Their valuable guidance, encouragement, and academic support have greatly contributed to the development of this work. We are especially thankful to our project guide for their continuous supervision, insightful suggestions, and constructive feedback throughout the project.

We also extend our heartfelt thanks to our friends and classmates who supported us during the development and documentation of this project. Their discussions, ideas, and encouragement helped us overcome many challenges during the implementation process. Finally, we would like to thank our families for their constant motivation, patience, and support, which enabled us to successfully complete this work.

REFERENCES

[1] Farhan Ahmad Nurzi , "Web-Based Student Task Management System", Accepted by Journal of UCYP 22 July 2022.
[2] Wannita Takerngsaksiri ,"Students' Perspective on AI Code Completion: Benefits and Challenges", Accepted by IEEE on 26 Augest 2024.
[3] Dimah Al-Fraihat , " Utilizing machine learning algorithms for task allocation in distributed agile software development", Accepted by Google Scholar 28 October 2024.
[4] C. Gupta and V. Gupta,"Enhancing Bug Allocation in Software Development Using Fuzzy Logic and Evolutionary Algorithms," *PeerJ Computer Science*, vol. 10, pp. e2111, 2024.
[5] T. Bhaskar, A. Joshi, and K. Nikam,"Enhancing Bug Prediction with Machine Learning Algorithms," *Journal of Software: Evolution and Process*, vol. 36, no. 5, pp. e2491, 2024.
[6] K. Al-Sulbi and A. Attaallah,"Symmetric Bug Prediction in Software Requirements Using Machine Learning Algorithms," *Scientific Reports*, vol. 15, pp. 22193, 2025.
[7] Shanid Malayil ,"Smart AI Based Online Platform for Competitive Coding, Technical Interview, Assessments and Recruitment", Accepted by IEEE on 4 November 2025.
[8] G. Maddali,"Efficient Machine Learning-Based Bug Prediction for Enhancing Software Reliability," *International Journal of Software Engineering and Applications*, vol. 16, no. 1, pp. 45–59, 2025.
[9] BanuPriya ,"Smart Software Coding and Predictive Analytics Using a Hybrid AI Approach",Accepted by IEEE Xplore on 3 March 2026.
[10] D. La Prova, E. Gentili, and D. Falessi, "Anticipating Bugs: Ticket-Level Bug Prediction and Temporal Proximity Effects," *IEEE Transactions on Software Engineering*, 2026.