1. Topic (ACRID Framework)

- **A (Approve)**:
    - The topic of **Automated Programming Evaluation using MERN** is approved because it addresses a significant need in educational institutions—automating the evaluation of programming assignments to reduce manual effort.
- **C (Create)**:
    - The topic focuses on creating a novel, automated evaluation tool using the MERN stack (MongoDB, Express.js, React.js, and Node.js). This approach is well-chosen for its modernity and scalability.
- **R (Recreate)**:
    - While the paper's topic is innovative, it's not entirely new. There are similar tools in existence. However, recreating this using the MERN stack offers modern technology advantages, including seamless integration and real-time feedback.
- **I (Improve)**:
    - Improvement is needed in terms of explaining the advantages of using the MERN stack over other frameworks (like Django, Spring, etc.) and addressing its limitations (e.g., handling heavy computational loads).
- **D (Disprove)**:
    - The paper could challenge the notion that traditional educational management systems (like Moodle) are enough for programming evaluations. It could argue that these systems lack real-time, multi-language support, which is the key contribution of this tool.

---

2. Abstract

- **Overview**:
    - The abstract gives a brief overview of the MERN-based tool for automated programming evaluation but should include more details about the problem it solves and the innovations introduced.
- **Informative**:
    - It could be more informative by summarizing the key benefits and limitations of the system (e.g., supporting multiple languages, real-time feedback, scalability concerns).
- **Standalone**:
    - The abstract can stand alone as a summary, but it should do a better job of emphasizing the system's real-world applications and significance in educational technology.
- **Structure**:
    - The structure is clear but could be improved by mentioning specific technologies used, challenges overcome, and results.

- **Brief Contribution**:
    - The contribution of the paper is mentioned, but it needs to emphasize the uniqueness of the MERN-based approach in this context.

---

3. Introduction

- **What?**:
    - The introduction explains what the system does (automated evaluation of programming assignments) but lacks a strong justification of why this approach was chosen over others.
- **Why?**:
    - The "why" is somewhat addressed but could be clearer. Why is an automated system essential? Why use the MERN stack, specifically? More depth in justifying the need for this tool would help.
- **How?**:
    - The "how" is described through the MERN stack components, but more technical details on the evaluation mechanism (such as how test cases are run and graded) would strengthen the introduction.
- **Contribution**:
    - The contribution is the development of an automated tool that allows for quick grading and real-time feedback, but this should be contrasted with existing tools.
- **Novelty Itemized**:
    - Novelty is highlighted, but it could go further by itemizing specific features, such as multi-language support and real-time evaluation.
- **Summary of Chapters**:
    - The introduction does not provide a clear roadmap of the chapters to follow, which would be helpful for guiding the reader.

---

4. Literature Review (Previous Work)

- **Why?**:
    - The literature review should explain why this tool was developed by identifying gaps in existing solutions. The review does not currently give enough insight into what problems remain unsolved by other tools (e.g., lack of real-time feedback, limited language support).
- **Harmonizing Approach**:
    - The paper lacks a detailed harmonizing approach, where it shows how the proposed system bridges the gap between existing methodologies.
- **Find/Create a Hole**:

    - The paper could more explicitly identify a gap (e.g., "Current tools lack real-time assessment capabilities and multi-language support") and then create a hole where this tool fits in.
- **Look for Debate**:
    - No significant debate is addressed. The paper could discuss how some educators might prefer manual grading for accuracy, and then counter this argument with the benefits of automation.
- **Consistent in Reference**:
    - The references provided are somewhat consistent but could be more up-to-date with recent technologies or methodologies in automated grading.

---

5. Methodology (How?)

- **Algorithm**:
    - The methodology explains that the system uses test-case based evaluation, but does not go into depth about the algorithm used to manage and evaluate the test cases (e.g., what happens in case of partial correctness?).
- **Feature Selection & Accuracy**:
    - There's no mention of how features (like code structure, syntax, and logic) are selected for evaluation. The accuracy of the system (compared to manual grading) is not discussed.
- **Research Design**:
    - The research design outlines how the system uses the MERN stack. However, the design lacks sufficient technical detail on how data (assignments) flows from one component to another.
- **Model Figures/Diagram**:
    - A detailed architectural diagram showing the flow from student submission to final grading would clarify the process and strengthen the methodology.
- **Data Selection**:
    - The selection of programming assignments as data is appropriate, but it does not cover how diverse the data (different programming languages, difficulty levels) needs to be to thoroughly test the system.
- **Collection Procedure**:
    - Data collection (from students submitting assignments) is explained well, but the system's robustness in handling simultaneous submissions is not discussed.

---

6. Preliminary Data (Result Section)

- **What?**:

    - The system successfully evaluates basic test cases, but the results are minimal. More data on performance metrics, speed, and error-handling should be provided.
- **Evidence of Importance**:
    - The preliminary results show that the tool works, but no evidence is presented to demonstrate that this is an improvement over manual grading or other tools (e.g., time saved, accuracy rate).
- **Informed Methodology**:
    - The methodology seems logical, but it would be strengthened by explaining why certain technical choices (e.g., MongoDB for data storage) were made.
- **Preliminary Findings**:
    - The preliminary findings are promising, but the absence of data on scalability and long-term use is a limitation.

---

7. Discussion

- **Unexpected Results**:
    - No unexpected results are mentioned. It would be helpful to discuss any challenges or unexpected findings (e.g., issues with real-time grading at scale).
- **Relevance to the Field**:
    - The tool is relevant, but the discussion does not clearly explain how it fits into the larger field of educational technology. The potential impact (e.g., reducing the burden on educators) could be emphasized more.
- **Answer Research Questions**:
    - The research questions are only partially answered. More focus is needed on how this tool compares to manual grading in terms of efficiency and accuracy.
- **Comparison of Hypothesis**:
    - The discussion could include a more rigorous comparison of the hypothesis (i.e., "Automated grading is faster and as accurate as manual grading") with actual results.
- **Contextualization**:
    - The results need to be better contextualized within the broader scope of automated grading systems. What does this tool offer that others don't?
- **Support For/Against Existing Theory**:
    - The paper lacks a deep engagement with existing theories on automated grading, which would strengthen the argument for its novelty.

---

8. Statement of Limitation

- **Alternative**:
  - The paper does not provide alternatives to the proposed method. For example, integrating AI-based tools or exploring other frameworks beyond MERN could be discussed.
- **Weakness**:
  - The biggest weakness not addressed is scalability. How does the system perform with large class sizes and multiple simultaneous submissions? This limitation is not fully explored.

---

9. Conclusion

- **What?**:
  - The conclusion summarizes the contributions but could go deeper into the implications of the findings. What is the broader impact on educational technology and grading processes?
- **Why?**:
  - The reasons behind using this system (e.g., reducing workload, increasing feedback speed) are somewhat addressed but could be justified with more evidence.
- **Importance**:
  - The importance of the system is noted, but the conclusion lacks a strong emphasis on how this could change the landscape of automated grading tools in education.

---

10. References & Bibliography

- **Cited and Uncited**:
  - The references are cited well but are somewhat limited in scope. More recent works on educational tools, particularly ones utilizing modern stacks like MERN, would provide better context.
- **Affirmed**:
  - The paper could affirm its findings by citing more studies that support the benefits of automated grading, such as increased accuracy and faster feedback times.

➔ To enhance the **literature comparison** in the paper "Automated Programming Evaluation using MERN," here's a detailed comparison with existing research and tools. The paper mainly highlights the need for an automated programming evaluation tool, but it lacks in-depth comparative analysis with other tools or frameworks. Below is a potential literature comparison that could be added to the paper to improve it:

Literature Comparison

Several automated programming evaluation systems have been proposed and implemented, each employing different technologies and approaches. A comparative analysis between these systems and the proposed MERN-based solution would highlight the novelty and advantages of the research.

1. **PROBSOL: A Web-Based Application for Problem-Solving in Introductory Programming**
   PROBSOL is a tool designed to help students practice problem-solving in introductory programming courses. It uses automated assessments and peer learning to improve programming skills(13). However, unlike the MERN-based tool, which focuses on a broader range of programming languages and automated feedback for individual test cases, PROBSOL's primary focus is on the learning process and collaborative problem-solving. The MERN tool's real-time feedback and automated grading for multiple languages provide a more comprehensive assessment system.

2. **Automated Grading Systems for Programming Assignments**
   Hussam Aldriye et al. (2019) proposed a grading system that integrates machine learning and natural language processing (NLP) to evaluate programming assignments(13). This system is more complex than the proposed MERN-based tool, as it involves learning algorithms that analyze code beyond simple correctness and execution. However, the MERN tool offers simplicity, flexibility, and real-time evaluation, which may be more suitable for large-scale use in educational institutions without the need for advanced AI algorithms.

3. **CodeLab: An Interactive Platform for Programming Exercises**
   CodeLab is a widely-used platform for learning programming languages, offering interactive exercises with instant feedback. It supports languages like Java, Python, and C++. Similar to the MERN-based evaluation tool, CodeLab offers real-time feedback but focuses primarily on the educational aspect of learning programming(rp). The MERN tool differentiates itself by integrating multiple roles (admin, faculty, and students), streamlining the evaluation process, and providing full control of assignments to faculty, making it more suitable for institutions.

4. **Moss: A System for Detecting Software Similarity**
   Moss (Measure of Software Similarity) is an advanced system used for detecting plagiarism in programming assignments. While it doesn't evaluate code for correctness, Moss is a critical tool for ensuring academic integrity in programming courses(13). The proposed MERN system, though lacking a plagiarism detection feature, can be integrated with a tool like Moss to enhance its capabilities. This integration would offer both automated evaluation and plagiarism detection within one platform.

5. **CodeOcean: A Cloud-Based Interactive Platform for Learning Programming**
   CodeOcean allows students to write and execute code in a cloud-based environment and receive instant feedback. While CodeOcean provides an easy-to-use interface and cloud accessibility, it lacks the deeper integration with faculty and administrative control that the MERN stack system offers(rp). The MERN system's faculty module allows for a more structured management of programming assignments and test cases, which gives it an edge for institutional use.

6. **Existing Educational Management Systems (EMS)**
   Traditional EMS systems such as Moodle and Blackboard offer assignment submissions, but these systems are not designed for automatic code evaluation. They rely on manual grading by faculty, which can be time-consuming. In contrast, the MERN-based tool automatically tests student code against predefined test cases and grades it in real-time, significantly reducing the grading load on instructors(13).