A close-up of a logo

AI-generated content may be incorrect.

**M. S. Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**Department of Computer Science and Engineering**

**CSP:Project Work Credit:12**

**Project Synopsis**

**Title: Cloud based project management tool with Evaluation System**

**Project Guide Name: BRUNDA G**

|  |  |
| --- | --- |
| **USN** | **NAME** |
| **1MS21CS040** | **Charishma Reddy Mallem** |
| **1MS21CS056** | **Jyothi Yadav** |
| **1MS21CS063** | **Kavyasri R** |

**Guide Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Guide Comment:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expert 1 Comment \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expert 2 Comment**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expert 1 Signature: Expert 2 Signature:**

**Guide Signature:**

**Table Of Contents**

|  |  |  |
| --- | --- | --- |
| **Sl no.** | **Content** | **Page No.** |
| 1 | Problem Statement | 1 |
| 2 | Relevance of topic | 1 |
| 3 | Objectives | 1 |
| 4 | Methodology | 2 |
| 5 | Hardware and Software tools | 3 |
| 6 | Testing Technologies | 3 |
| 7 | Contributions | 4 |
| 8 | References | 4 |
| 9 | Conclusion | 6 |

**1. Problem statement:**

Managing academic projects involves tracking implementation progress, evaluating documentation, assessing presentations, and conducting project fair evaluations. Manual, paper-based assessments are time-consuming and error-prone, leading to inconsistencies, delays, and miscommunication. Additionally, local hosting limits accessibility, scalability, and efficient data management for multiple users. A cloud-based infrastructure ensures seamless access, standardized evaluations, and efficient project tracking.

**2. Relevance of Topic**

The topic was chosen due to the growing need for a streamlined system to manage academic projects efficiently. Currently, project management involves tracking implementation progress, evaluating documentation, and scoring presentations, which are often done manually. This manual process is time-consuming, and prone to errors, especially when evaluations are paper-based. As the volume of projects grows, maintaining consistency in tracking and evaluation becomes increasingly difficult, leading to delays in feedback and miscommunication between students, faculty, and panel members.

Additionally, traditional local hosting methods limit accessibility and scalability, making it hard to manage multiple users who need real-time access to the system. In academic environments where students, faculty, and panel members are often working remotely or on different schedules, these limitations become a significant barrier. A cloud-based system is necessary to overcome these challenges by providing seamless, real-time access, scalable infrastructure, and the ability to handle large volumes of data. By automating processes such as implementation score evaluation (based on sprints and story points), documentation checks, and structured presentation scoring, this project will significantly reduce manual workload and improve the overall efficiency and accuracy of academic project management.

**3. Objectives:**

1. **Student-Friendly Report Submission Platform:** Provide an intuitive interface for students to track project updates and submit documents related to the project.
2. **Automate the evaluation of progress and work done (Implementation)**: Automate project implementation evaluation by developing a system that integrates Git to track progress using sprints, story points, and predefined metrics, reducing manual effort and enhancing accuracy.
3. **Automate Document Evaluation**: Implement an automatic evaluation feature that reviews and scores project documentation based on predefined standards.
4. **Embed and Streamline Panel Scoring and encourage peer evaluation**: Provide a dedicated panel member page for evaluating and scoring presentations in a systematic, paperless manner, allowing for efficient and error-free assessments.
5. **Configure and Set-up College Cloud Infrastructure for Cost-Effective Storage Solutions**: Leverage cloud-based infrastructure to ensure that the system is accessible to all users in real-time, with the ability to scale as the number of users or projects increases.

**4. Methodology**

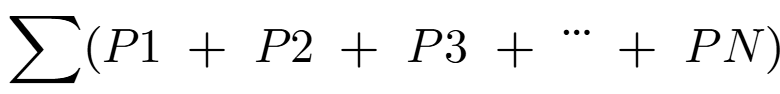
This system automates the evaluation of academic projects by integrating cloud computing, AI-based documentation assessment, sprint-based implementation tracking, and structured presentation scoring. The methodology follows an agile-based development approach to ensure scalability, accuracy, and efficiency in project evaluation.

**A.**      **Automated evaluation system**

1. **Panel-Based Presentation Scoring:**

Panel members access a dedicated evaluation portal and evaluate presentations based on clarity, content quality, project demonstration effectiveness, and responses to questions.

**The final Presentation Score** is calculated as the sum of all panel members scores:



P1 :-- Score given by Panel number 1

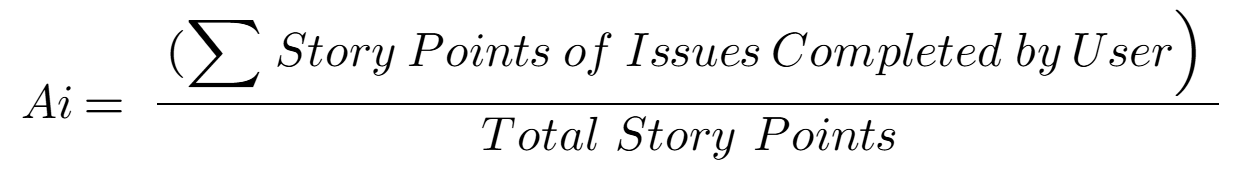
PN :-- Score given by Panel number N

ensuring a fair and structured assessment.

**2.**   **Automatic Implementation Evaluation (Sprint Tracking):**

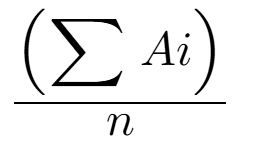
Students submit progress updates based on agile sprint methodology.

The system tracks progress using **story points and issue completion rates**

* Story points represent the difficulty of tasks in the sprint.
* Completion percentage of one sprint 

Ai= Completion percentage of one sprint

* Implementation score

 where i = 1 to n, n = Number of sprints

**5.**  **File Upload & Documentation Evaluation**

* Students upload project documentation (e.g., reports, research papers).
* The backend processes the document by performing **plagiarism detection** (via Turnitin API or custom NLP models), **format checking**, and **NLP-based scoring** to evaluate writing quality.

**The Implementation Score, Documentation Score, and Presentation Score** are combined.

**B.**      **Cloud-Based Infrastructure Setup**

The system is hosted on the **cloud**, allowing:

* Remote accessibility for students, faculty, and panel members.
* Scalability to handle multiple users and projects.
* Centralized data storage in a database.

**5. Hardware and Software Tools**

**1. Hardware Requirements**

* At least 2-3 physical servers (for redundancy, load balancing, and scaling). Depending on your workload, a basic server configuration could include:
* Processor: Intel Xeon or AMD EPYC, with at least 8-12 cores per server.
* RAM: 32GB to 128GB per server, depending on the expected load.
* Storage: SSD storage (512GB to 2TB) for high-speed data access. Consider NVMe SSDs for better performance.
* Networking: High-speed network interfaces (10Gbps Ethernet).
* Networking Equipment: A reliable switch (10/100/1000 Mbps) with sufficient ports to connect the servers.
* Firewall and Router for security and network management.
* Backup Storage: Network Attached Storage (NAS) or a dedicated backup server with high-capacity hard drives (e.g., 4TB to 10TB) for storing backups and archival data.
* Cloud-Based Processing – The system runs on **AWS and Private cloud** reducing reliance on local machines.
* Compute Power – Uses Virtual Machines **(VMs)** or Containers for backend services, with **GPU instances** for AI-based NLP processing.
* Storage – Documents are stored in **Amazon S3/Private CLoud**, while user data and evaluation scores are managed in **PostgreSQL/MySQL** databases.

**2.**       **Software Requirements**

* Backend Development – **Python with Django**, Auth/JWT for authentication
* Frontend Development –React.js, Material-UI/ CSS, Axios/Fetch API.
* AI & NLP Tools – spaCy, OpenAI API (GPT models, PyPDF2/python-docx, and Turnitin API/TF-IDF & Cosine.
* Hypervisor (for private cloud setup):
* VMware ESXi or Proxmox VE for virtualization and managing virtual machines.
* OpenStack for building a scalable private cloud.

**6. Testing Technologies**

To ensure the reliability and accuracy of the system, various testing technologies and methodologies are utilized:

* **Unit Testing** – Conducted using **pytest** (for Python/Django), **Jest** (for React), and **unittest** for backend logic validation.
* **Integration Testing** – Validates the interaction between different components, ensuring seamless communication between the **Django backend, PostgreSQL database, and React frontend**.
* **Functional Testing** – Ensures the automatic documentation evaluation system functions correctly, covering LLM-based relevance checking and format validation.
* **Performance Testing** – Assesses the system's response time and resource usage, particularly for handling large academic documents. Tools like **JMeter** or **Locust** may be used.
* **Security Testing** – Ensures Git integration enforces ticket ID validation and prevents unauthorized access. **OWASP ZAP** or **Bandit** could be used for security checks.
* **User Acceptance Testing (UAT)** – Performed with professors and students to verify that the document evaluation system meets academic needs

**7. Contributions**

This project brings multiple innovations and benefits to academic project management:

* **Enhanced Academic Workflow** – Automates document evaluation, reducing manual workload for professors and improving feedback for students.
* **Objective and Consistent Evaluation** – Uses LLMs and predefined standards to ensure fairness in grading and feedback.
* **Integration with Git for Project Management** – Enforces structured development workflows by requiring ticket IDs before commits.
* **Facilitates Collaboration** – Encourages a structured approach to documentation, aligning student work with academic and industry standards.
* **College Cloud Infrastructure** – Provides a self-hosted solution for efficient storage, computing, and collaboration.

**8. References**

1. Dinh Ho Tong Minh, "Persistent Scatterers and Distributed Scatterers (PSDS) InSAR and ComSAR Algorithms," *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Athens, Greece, July 2024. [Online]. Available:<https://www.2024.ieeeigarss.org/tutorials.php>.
2. M. Royal, "Self-Hosting Guide," GitHub Repository, 2025. [Online]. Available:<https://github.com/mikeroyal/Self-Hosting-Guide>.
3. OpsLevel Documentation, "Git Integrations," 2025. [Online]. Available:<https://docs.opslevel.com/docs/git-integration>.
4. Gitpod, "Always Ready-to-Code," 2025. [Online]. Available:<https://www.gitpod.io/>.
5. Checkmarx Documentation, "GitLab Self-Hosted Integration," 2025. [Online]. Available:<https://docs.checkmarx.com/en/34965-68680-gitlab-self-hosted.html>.
6. Microsoft Fabric Documentation, "Get Started with Git Integration," 2025. [Online]. Available:<https://learn.microsoft.com/en-us/fabric/cicd/git-integration/git-get-started>.
7. Seqera Labs Documentation, "Git Integration Overview," 2025. [Online]. Available:<https://docs.seqera.io/platform/23.1/git/overview>.
8. Databricks Documentation, "Git Integration for Databricks Git Folders," 2025. [Online]. Available:<https://docs.databricks.com/aws/en/repos/>.
9. M. Royal, "Self-Hosting Guide," GitHub Repository, 2025. [Online]. Available:<https://github.com/mikeroyal/Self-Hosting-Guide>.
10. OpsLevel Documentation, "Git Integrations," 2025. [Online]. Available:<https://docs.opslevel.com/docs/git-integration>.
11. M. Armbrust et al., "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50-58, Apr. 2010.
12. L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50-55, Jan. 2009.
13. P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Gaithersburg, MD, Special Publication 800-145, Sep. 2011.
14. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, Jun. 2009.
15. M. D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," *IEEE Internet Computing*, vol. 13, no. 5, pp. 10-13, Sep. 2009.
16. B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems," in *Proceedings of the 5th International Joint Conference on INC, IMS and IDC*, Seoul, South Korea, 2009, pp. 44-51.
17. S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud Computing—The Business Perspective," *Decision Support Systems*, vol. 51, no. 1, pp. 176-189, Apr. 2011.
18. H. Wachsmuth, "Perspectives on Large Language Models for Relevance Judgment," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Taipei, Taiwan, 2023, pp. 3223-3227.
19. Z. Abbasiantaeb, C. Meng, L. Azzopardi, and M. Aliannejadi, "Can We Use Large Language Models to Fill Relevance Judgment Holes?," *arXiv preprint arXiv:2405.05600*, May 2024.
20. J. M. S. de Souza, A. S. da Silva, and E. S. de Moura, "Evaluating the Effectiveness of Large Language Models in Abstract Screening for Systematic Reviews," *BMC Medical Research Methodology*, vol. 24, no. 1, pp. 1-12, Dec. 2024.

**9. Conclusion**

In conclusion, represents a significant leap forward in the integration of automated evaluation within academic project management. By harnessing the power of modern technologies such as Large Language Models (LLMs) for content relevance checking, and integrating robust feedback mechanisms based on input from seasoned professors, the project ensures that academic documentation is evaluated both objectively and holistically. This dual-approach methodology not only streamlines the evaluation process but also aligns it with the high standards expected in academic environments.

The first pillar of the project focuses on leveraging LLMs to assess the relevance and quality of content. This system analyzes submitted documents against a backdrop of vast academic resources and best practices, ensuring that the submissions are not only factually accurate but also contextually appropriate. In doing so, it minimizes the subjective variance typically encountered in manual evaluations, promoting fairness and consistency across assessments. This automation not only reduces the administrative burden on educators but also accelerates the feedback loop, enabling students to quickly understand and improve upon their work.

Parallel to the automated content analysis, the project incorporates a mechanism for setting and adhering to strict documentation standards. By gathering insights from experienced professors, tailors its evaluation criteria to the specific needs and expectations of academic institutions. This collaborative input ensures that the evaluation process is both rigorous and relevant, addressing key elements such as structure, formatting, and comprehensive coverage of required topics. Such an approach guarantees that the documentation not only meets baseline academic requirements but also encourages a higher level of analytical and critical thinking.

Additionally, the integration of Git with enforced ticket identification for repository commits adds another layer of discipline and traceability to project management. This measure ensures that every change in the documentation or codebase is accountable and can be easily traced back to a specific task or issue, fostering transparency and efficient collaboration among team members. It also aligns the project with industry best practices, bridging the gap between academic research and professional software development methodologies.