

Design And Development of an AI Powered Open-Source GitHub Repository Finder

Abhishek Jagtap¹, Aditya Hood², Gunjan Choukade³, Rohit Katore⁴, Prof. Nikhil Deshpande⁵

¹ Department Of Information Technology, Sinhgad College of Engineering, Pune- 41

² Department Of Information Technology, Sinhgad College of Engineering, Pune- 41

³ Department Of Information Technology, Sinhgad College of Engineering, Pune- 41

⁴ Department Of Information Technology, Sinhgad College of Engineering, Pune- 41

Email:adityahood13@gmail.com

Abstract - The rapid progress in Artificial Intelligence (AI) and Natural Language Processing (NLP) has enabled the automation of intelligent decision-making processes across various domains, including software development. This research focuses on the design and implementation of an AI-driven Open Source GitHub Repository Finder (OSGRF), developed to assist developers in identifying open-source projects that align with their technical skills and experience. The proposed system utilizes NLP techniques to extract key skills from a user's resume and employs Sentence-BERT embeddings for semantic matching with relevant GitHub repositories. By ranking repositories according to factors such as similarity, popularity, and difficulty level, and by providing personalized AI-based contribution recommendations, the system ensures a seamless onboarding experience for developers entering the open-source community. The overall architecture integrates multiple layers built using ReactJS, Node.js, MongoDB, and Python, forming a scalable and intelligent framework for efficient open-source exploration.

Key Words : AI Repository Recommendation, GitHub, Open Source Contribution, Natural Language Processing (NLP), Sentence-BERT (SBERT), Semantic Matching, Cosine Similarity, Resume Skill Extraction, Large Language Models (LLMs), Automated Contribution Guidance, Repository

Ranking, Machine Learning, Recommender Systems, GitHub API Integration, Developer Skill Mapping.

1. INTRODUCTION

Open-source software has become a fundamental aspect of modern software engineering, offering developers a collaborative environment to learn, innovate, and contribute to projects with global impact. Despite this, new contributors often struggle to identify repositories that align with their technical skills and experience. Existing GitHub search mechanisms primarily depend on keyword-based matching, which lacks the ability to interpret the semantic relationships between user skills and project requirements.

To address this limitation, the present research proposes an AI-driven system capable of understanding the contextual meaning of both skills and repository descriptions. By integrating Natural Language Processing (NLP) and Large Language Models (LLMs), the system automates repository discovery, provides intelligent contribution recommendations, and performs issue classification with improved accuracy. The motivation behind this work lies in the growing demand for streamlined contributor onboarding and the desire to harness intelligent automation to enhance engagement within the open-source ecosystem.

2. LITERATURE SURVEY

[1] This paper, “*Sequential Recommendations on GitHub Repository*” (Applied Sciences, 2021), proposes a deep learning approach for suggesting repositories based on user activity such as stars, follows, and contributions. Using models like GRU4Rec and SASRec, it predicts future repository interests by learning behavioral patterns. However, it performs poorly in cold-start situations where users lack prior activity and does not leverage resume-based skills or semantic understanding for personalized recommendations.

[2] This study explores the use of artificial intelligence and machine learning to automate various aspects of GitHub repository management, including issue classification, documentation generation, and pull request handling with large language models such as LLaMA. While it enhances repository maintenance and workflow efficiency, it lacks features that help new developers discover relevant repositories to contribute to, as its focus remains on internal automation rather than personalized recommendations.

[3] A machine learning-based system was introduced in 2022 to automatically identify “Good First Issues” (GFIs) in GitHub repositories by analyzing features such as issue content, reporter behavior, and project activity. Using an XGBoost classifier, the model effectively predicts beginner-friendly issues with a strong accuracy score. However, its scope is limited to general issue classification and does not account for individual developer skills or semantic matching. In contrast, the proposed system advances this concept by utilizing Sentence-BERT embeddings for skill-based repository matching, integrating large language models for contribution guidance, and delivering real-time, personalized recommendations with automated pull request generation.

[4] This paper presents a machine learning-based method for identifying “Good First Issues” (GFIs) in GitHub repositories using features like issue content, reporter activity, and project engagement. An XGBoost classifier predicts beginner-friendly issues effectively but lacks personalization or semantic understanding of developer skills. Building on this, the proposed system enhances the approach by applying Sentence-BERT embeddings for skill-based repository matching, integrating LLMs for contribution guidance, and offering real-time, user-focused recommendations with automated pull request generation.

3. OVERVIEW

The AI-Powered Open Source GitHub Repository Finder (OSGRF) is designed to intelligently bridge the gap between developers and the open-source community. It utilizes data-driven insights and AI-based automation to streamline the process of discovering open-source projects that align with a developer’s skill set and interests. By simplifying project discovery and contribution matching, OSGRF promotes a culture of collaboration, knowledge sharing, and continuous learning within the global developer ecosystem.

1. Objective:

The primary objective of the Open Source GitHub Repository Finder (OSGRF) is to develop a seamless and intelligent recommendation system that connects developers with open-source repositories matching their genuine skills and interests. Through the automation of skill extraction, repository discovery, and AI-driven contribution guidance, the system aims to minimize the challenges faced by newcomers when initiating open-source contributions. Ultimately, OSGRF encourages skill-oriented community growth and fosters a more inclusive and efficient open-source collaboration environment.

2. Core Components:

The Open Source GitHub Repository Finder (OSGRF) consists of five key components that work together to provide intelligent and personalized recommendations:

- **Resume Analysis Engine:** Uses NLP techniques to extract technical skills from a developer’s resume.
- **GitHub Integration Module:** Connects to the GitHub API to fetch and update repository data.
- **Semantic Similarity Engine:** Applies Sentence-BERT embeddings to match user skills with repository descriptions.
- **AI-Powered Guide Generator:** Provides context about repositories and step-by-step contribution suggestions.
- **User Interface Module:** Displays ranked recommendations and allows users to explore results through interactive filters.

Together, these components create a multi-layered system capable of delivering real-time intelligent suggestions and adaptive learning based on user behavior

and profile data.

3. Working Principle:

The system operates through the integration of **Natural Language Processing (NLP)**, **semantic similarity scoring**, and **AI-based content generation**. After a user logs in via GitHub and uploads their resume, the system extracts relevant technical skills and converts them into **vector embeddings**. These embeddings are then matched with open-source repositories retrieved from GitHub. Using **cosine similarity**, the system ranks repositories based on **similarity**, **popularity**, and **recency**. Finally, an **LLM-powered module** generates concise summaries and contribution guidelines, helping users quickly understand project contexts and initiate meaningful contributions.

4. Accessibility and Innovation:

The system features an intuitive web interface that enables developers from diverse backgrounds to explore open-source projects without extensive manual searching or technical analysis. Its key innovation lies in combining NLP-based skill extraction with LLM-generated contribution guidance, forming a fully automated recommendation pipeline. By accommodating various skill levels and offering filtered task suggestions, OSGRF effectively reduces entry barriers and promotes greater participation among students and developers worldwide.

5. Social Impact:

Beyond its technical capabilities, the OSGRF project fosters community-driven innovation within the open-source ecosystem. It promotes inclusivity by guiding new developers toward suitable mentorship opportunities and collaborative projects. Through intelligent matchmaking, the system facilitates skill sharing, accelerates project development, and enhances knowledge exchange among contributors. Over time, such interactions can significantly boost global participation in open-source initiatives, leading to a more connected, collaborative, and sustainable developer community.

4. METHODOLOGY

The proposed OSGRF system is developed using an integrated approach that combines Natural Language Processing (NLP), semantic embedding methods, and a hybrid ranking mechanism to generate accurate and personalized GitHub repository recommendations. The

methodology focuses on effectively interpreting user input, understanding contextual relationships, and delivering meaningful results.

The process begins with the acquisition of user input, which can be provided either as manually entered skill sets or through uploaded resumes. In the case of resumes, NLP techniques are utilized to identify and extract relevant technical information such as programming languages, development frameworks, libraries, and tools. The extracted data is then preprocessed through cleaning and normalization steps, where irrelevant terms, redundancies, and noise are removed. This ensures that the input remains consistent, structured, and suitable for further computational analysis.

To represent the semantic meaning of both user skills and repository descriptions, the system employs a transformer-based embedding model, Sentence-BERT (all-MiniLM-L6-v2). This model transforms textual information into dense vector representations that capture contextual and semantic relationships between words and phrases. By applying the same embedding technique to both user input and repository data, the system maintains uniformity in representation, which is essential for accurate comparison.

The similarity between user profiles and repositories is determined using cosine similarity, a widely used metric for measuring the angular distance between two vectors in a high-dimensional space. This technique enables the system to identify repositories that are semantically related to the user's interests, even in cases where exact keyword matches are absent. As a result, the recommendation process becomes more intelligent and context-aware rather than relying solely on traditional keyword-based matching.

In order to improve the reliability and usefulness of the recommendations, a filtering stage is incorporated. This stage eliminates repositories that do not meet certain quality criteria, such as inactive projects, mismatched programming languages, or lack of contribution opportunities. By applying these constraints, the system ensures that only relevant and valuable repositories proceed to the ranking phase.

The final recommendation list is generated using a hybrid ranking strategy that considers multiple factors. These include semantic similarity scores, repository popularity indicated by star count, frequency of recent

updates, and compatibility with the user's preferred programming languages. This multi-dimensional ranking approach enhances the practical relevance of the recommendations by balancing both technical accuracy and real-world usability.

The system ultimately produces a ranked list of GitHub repositories along with key details such as repository description, primary programming language, and popularity indicators. Additionally, the system provides AI-generated summaries and contribution guidance, enabling users to better understand the selected repositories and actively participate in open-source development.

5. SYSTEM ARCHITECTURE

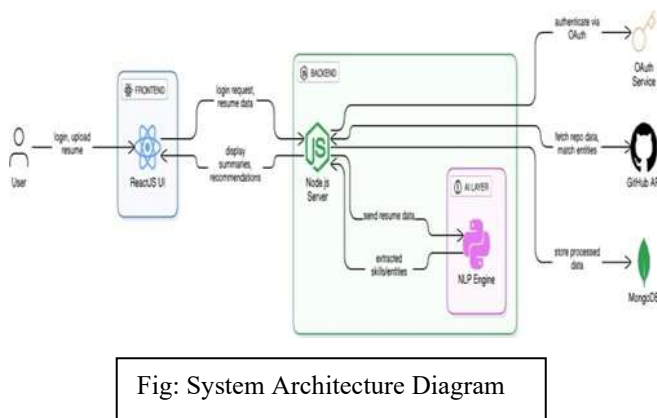


Fig: System Architecture Diagram

The OSGRF system is designed using a modular and layered architectural approach to ensure scalability, efficient data handling, and smooth coordination among different components. The architecture is structured into multiple layers, including the frontend, backend, AI/NLP processing module, external service integration, and database layer, all working together to deliver personalized repository recommendations.

At the user interaction level, the frontend layer is developed using React.js, providing a responsive and intuitive interface. This layer allows users to authenticate, upload resumes, or manually enter their technical skills. It is responsible for capturing user input and transmitting it to the backend via API calls. Additionally, the frontend presents the final output in a structured format, including recommended repositories, brief summaries, and contribution suggestions, thereby enhancing user experience and accessibility.

The backend layer, implemented using Node.js, serves as the core processing unit of the system. It handles client requests, manages authentication mechanisms

using OAuth protocols, and orchestrates communication between different modules. The backend also interacts with external services such as the GitHub API to retrieve repository-related information, including descriptions, programming languages, and activity metrics. Furthermore, it ensures proper data flow between the frontend, AI module, and database.

An essential part of the architecture is the AI/NLP processing module, which enables intelligent and context-aware recommendations. This module applies Natural Language Processing techniques to extract meaningful information from user resumes and skill inputs. The extracted data is then transformed into structured representations and converted into semantic embeddings using advanced language models. These embeddings facilitate similarity comparison with repository data, allowing the system to recommend repositories based on contextual relevance rather than exact keyword matches.

The architecture also incorporates external integrations to enhance functionality and security. OAuth-based authentication is used to securely verify user identity and enable controlled access to GitHub-related data. The GitHub API is utilized to fetch real-time repository information, ensuring that recommendations are based on the most recent and relevant data available in the GitHub ecosystem.

The database layer is implemented using MongoDB, which provides a flexible and scalable NoSQL storage solution. It stores user-related data, extracted skill sets, embedding vectors, and repository metadata. This layer supports efficient data retrieval and ensures consistency, making it suitable for handling large volumes of data and multiple concurrent users.

In summary, the system architecture facilitates a streamlined flow of information from user input to final recommendation output. By integrating user interface components, server-side processing, intelligent analysis, and external data sources, the architecture ensures high performance, maintainability, and scalability. This structured design makes the OSGRF system a reliable and efficient platform for recommending relevant GitHub repositories.

6. IMPLEMENTATION

The GitCompass platform was developed as a part of the OSGRF system to provide intelligent GitHub repository recommendations based on user skills. The system utilizes semantic matching and ranking techniques to ensure that the suggested repositories are relevant and aligned with the user's technical expertise.

The implementation primarily focuses on three key aspects: skill processing, repository matching, and result generation. User inputs are analyzed and structured, followed by semantic comparison with repository data to identify suitable matches. Finally, the system generates a ranked list of repositories, ensuring that the recommendations are both meaningful and practically useful for users.

6.1 Skill Input and Processing

The system allows users to enter their skills manually or upload a resume. For resume input, Natural Language Processing (NLP) techniques are applied to extract relevant technical skills. spaCy's Named Entity Recognition (NER) is used to identify programming languages, frameworks, and tools. The extracted skills are then cleaned and normalized to remove duplicates and irrelevant terms.

6.2 Embedding Generation

The processed skills are converted into numerical vectors using Sentence-BERT (all-MiniLM-L6-v2). Each user profile is represented as a 384-dimensional embedding capturing semantic meaning. Repository descriptions are also transformed into embeddings, enabling semantic comparison instead of keyword-based matching.

6.3 Repository Processing

Repository data is collected and processed to extract key attributes such as description, language, stars, and topics. The textual data is cleaned and converted into embeddings using the same model, ensuring consistency for comparison.

6.4 Similarity Computation

Cosine similarity is used to measure the similarity between user skill embeddings and repository embeddings. Higher similarity scores indicate better alignment, and such repositories are considered more relevant.

6.5 Hybrid Ranking Mechanism

The system applies a hybrid ranking approach by combining semantic similarity, repository popularity (stars), recency, and language matching. Additional weight is given when the repository language matches user skills, improving recommendation relevance and usability.

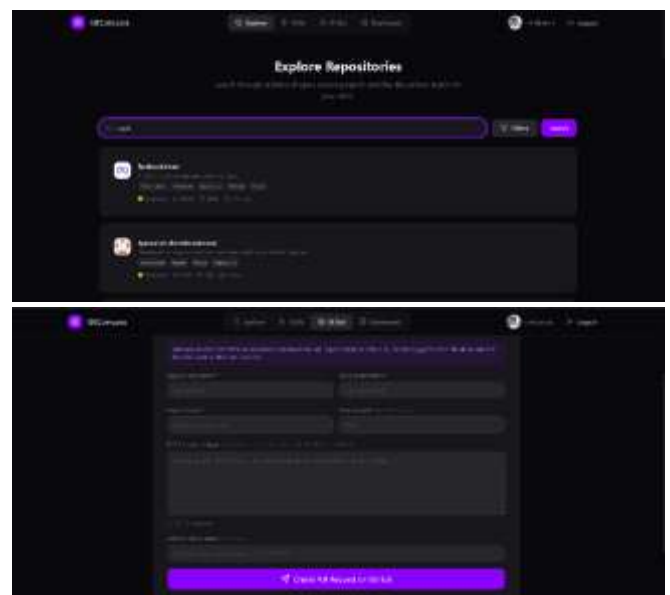
6.6 Alternative Model Implementation

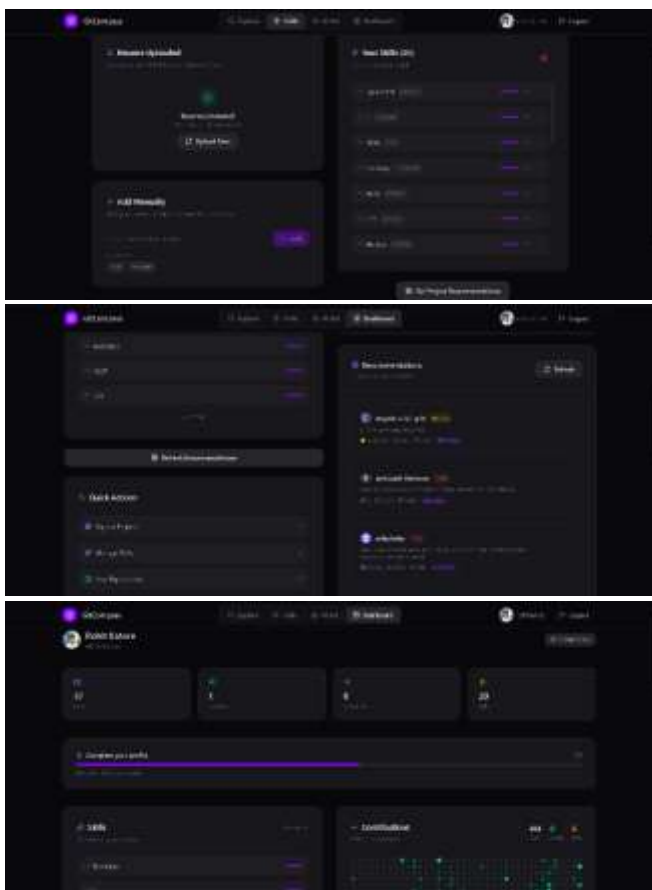
To evaluate the effectiveness of the proposed system, three additional baseline models were implemented. The KeyBERT model extracts important keywords from user skills and matches them with repository content. The TF-IDF model uses a bag-of-words approach with cosine similarity for comparison. The traditional model performs direct matching based on programming languages and tags. These models are used for performance comparison with the proposed hybrid approach.

6.7 Output Generation

The system produces a ranked list of repositories based on the computed scores. Each recommendation includes key details such as repository name, description, programming language, and popularity metrics. This structured output enables users to easily identify relevant repositories and start contributing efficiently.

7. RESULTS AND EVALUATION





The GitCompass system was implemented and evaluated to measure its effectiveness in recommending relevant GitHub repositories. The evaluation focuses on recommendation accuracy, system performance, and comparison with baseline models.

7.1 System Implementation Results

The system was implemented using Sentence-BERT for embedding generation and cosine similarity for matching.

- Skills were successfully extracted from user input and resumes using NLP techniques.
- Repository data was processed and converted into semantic embeddings.
- The hybrid model generated personalized recommendations effectively.
- Repositories were ranked based on similarity, popularity, and recency.

7.2 Model Evaluation

Four models were evaluated for recommendation performance.

- **Hybrid Model (SBERT + Cosine Similarity):** Provides the most accurate results due to semantic understanding.
- **KeyBERT Model:** Performs moderately well but is limited to keyword-based matching.
- **TF-IDF Model:** Lacks contextual understanding,

leading to lower accuracy.

- **Traditional Model:** Relies on exact matching, making it the least effective.

7.3 Performance Analysis

The system was evaluated on key performance factors:

- **Response Time:** Generates recommendations within a few seconds.
- **Recommendation Quality:** Provides relevant results through semantic matching.
- **Scalability:** Supports multiple users and large datasets, with minor dependency on API limits.
- **Processing Efficiency:** Optimized embedding and similarity computation ensure smooth performance.

7.4 Recommendation Quality Analysis

The use of semantic embeddings significantly improves recommendation quality.

- Achieves high recall by retrieving most relevant repositories.
- Maintains high precision by reducing irrelevant results.
- Works effectively across multiple domains such as ML, web, and DevOps.
- Handles complex skill combinations better than keyword-based approaches.

7.5 Comparative Analysis

Feature	Traditional Approach	Proposed GitCompass System
Recommendation Method	Keyword Matching	Semantic Matching (SBERT)
Accuracy	Low	High
Context Understanding	Limited	Strong
Personalization	Low	High
Scalability	Moderate	High

7.6 Overall System Outcome

The system successfully achieves its objective of providing accurate and personalized repository recommendations.

- Improves repository discovery using AI-based techniques
- Reduces manual search effort
- Provides high accuracy and relevance
- Supports better onboarding into open-source development

8.CONCLUSION

The GitCompass system offers an effective and intelligent solution for recommending GitHub repositories based on user skills by integrating Natural Language Processing, semantic embedding techniques, and a hybrid ranking strategy. In contrast to traditional methods that depend mainly on keyword matching, the proposed system focuses on understanding the contextual relationship between user skills and repository content, leading to more accurate and meaningful recommendations.

The use of Sentence-BERT for generating semantic embeddings, along with cosine similarity for measuring relevance, enhances the system's capability to interpret user intent more effectively. Additionally, the hybrid ranking mechanism, which considers multiple factors such as similarity score, repository popularity, and recency of updates, further improves the overall quality and usefulness of the recommendations. The system demonstrates efficient performance with quick response times and consistent behavior across various domains, including machine learning, web development, DevOps, and blockchain.

The evaluation results indicate that the proposed approach performs better than traditional and baseline models in terms of accuracy, relevance, and overall effectiveness. It significantly reduces the manual effort required to search for suitable repositories and simplifies the process of identifying appropriate projects for contribution, especially for beginners.

In conclusion, GitCompass serves as a scalable, reliable, and user-friendly platform that enhances developer productivity and promotes active participation in open-source communities. With future improvements such as enhanced personalization, real-time learning capabilities, and integration with additional platforms, the system has strong potential for practical deployment and continuous advancement.

9.FUTURE WORK

Although the GitCompass system demonstrates strong performance in recommending relevant GitHub repositories, there are several areas where further enhancements can improve its effectiveness and applicability in real-world scenarios.

One key direction for future improvement is the incorporation of advanced personalization techniques. The system can be extended to learn from user interactions such as clicked repositories, starred projects, and contribution history. By leveraging this behavioral data, the recommendation process can become more adaptive and user-centric, leading to improved accuracy over time.

Another potential enhancement involves integrating real-time learning and feedback mechanisms. This would enable the system to continuously refine its recommendations based on evolving user preferences. The adoption of techniques such as reinforcement learning or adaptive ranking models can further optimize the recommendation process and improve overall system performance.

The system can also be expanded by integrating additional development platforms such as GitLab and Bitbucket. This would increase the diversity of available repositories and allow users to explore opportunities beyond GitHub, making the system more comprehensive and versatile.

Further advancements may include the implementation of AI-based contribution assistance features. For example, the system can be enhanced to predict issue difficulty levels, provide step-by-step guidance, generate code explanations, and assist in creating pull requests using Large Language Models. Such features would significantly support beginners and reduce the complexity of contributing to open-source projects.

Scalability can be improved by deploying the system on cloud platforms such as AWS or GCP, along with optimizing data processing pipelines to efficiently handle large-scale datasets and multiple concurrent users. This would ensure consistent performance as the number of users and repositories grows.

Additionally, incorporating multilingual support and improving the user interface design can enhance accessibility for a broader audience, including non-English speakers and users who are new to open-source development.

In the future, the system can also explore integration with development tools and Integrated Development Environments (IDEs), enabling seamless repository recommendations and contribution workflows directly within coding environments.

8. REFERENCES

- [1] J. Kim, J. Wi, and Y. Kim, "Sequential Recommendations on GitHub Repository," *Applied Sciences*, vol. 11, no. 4, p. 1585, 2021. DOI: 10.3390/app11041585.
- [2] L. Barhate, P. Chhajed, and S. Vibhute, "AI Powered GitHub Project Management," *International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS)*, vol. 07, no. 06, pp. 1–6, Jun. 2025. DOI: 10.56726/IRJMETS78747.
- [3] W. Xiao, Y. Sun, D. Lo, and J. Jiang, "RecGFI: Recommending Good First Issues in GitHub OSS Projects," *Proceedings of the IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pp. 1–12, 2022.
- [4] G. Mitrov, B. Stanoev, S. Gievska, G. Mirceva, and E. Zdravevski, "Combining Semantic Matching, Word Embeddings, Transformers, and LLMs for Enhanced Document Ranking: Application in Systematic Reviews," *Big Data and Cognitive Computing*, vol. 8, no. 9, p. 110, 2024. DOI: 10.3390/bdcc8090110.