RIZVI EDUCATION SOCIETY's
Rizvi College of Engineering

- Approved by AICTE
- Recognized by DTE
- Affiliated to University of Mumbai
- Accredited by NAAC
- Department Accredited by NBA
  (Computer, Mechanical & Civil)

Department of Computer Engineering

MAJOR PROJECT SYNOPSIS

AI-Powered Codebase Debugger & Analysis System

Team Members:

1. Nishad Joglekar (221P045)
2. Binish Moosa (232P001)
3. Samar Khan (221P039)
4. Kashif Shaikh (221P060)

Guide: Prof. Shibhuraj Pappu

ABSTRACT

The rapid growth of software systems has introduced immense complexity in understanding, debugging, and maintaining large codebases. This project proposes an AI-powered system that employs multi-agent large language models (LLMs) to analyze, predict, and explain code behavior across multiple languages. By integrating natural language reasoning with traditional static analysis and machine learning, the system reduces debugging time, improves comprehension, and enhances developer productivity.

TABLE OF CONTENTS

1. Introduction
2. Literature Review
3. Working of Project
4. Hardware / Software Details
5. Output Screenshots
6. Report Status
7. Status of Completion
8. Queries

INTRODUCTION

Modern software development involves multiple frameworks, dependencies, and languages. Developers often spend 40–50% of their time understanding and debugging code. Traditional IDEs and static analysis tools provide limited insights.

- Problem Statement: Existing debuggers lack contextual understanding and multi-perspective reasoning across codebases.

- Scope: This project targets polyglot applications, supports cross-language analysis, and employs LLM agents to perform contextual debugging and explanation.

LITERATURE REVIEW

Identifies Research Gaps

HIGH LATENCY: Multi-Agent Frameworks

ABSENCE OF ADAPTIVE INTELLIGENCE: Real Deubuging Tools

LACK OF INTEGRATED SYSTEMS: Static, Visualization, & LLM Reasoning

LIMITED EXPLORATION: Cross-Language Dependency Mapping

Objectives

1. Develop an AI-driven Multi-Agent Deugent Deub ging Framework
2. Construct a Cross-Language Dependency Mapper: Graph Graph-based analysis
3. Enhance Developer Producivity: Reduce debuging duıging time by 40%
4. Integrate a Visual Debugging Dashboard: Interactive frontend

Limitation of Existing Systems

Code Search Engines: Keyword-based retrieval without deeper code code intent compreheration

Single-Agent LLM Assistants: One-dimenisonal assistance without multi-perspective reasning or cross-language dependdency analysis

Traditional Debuggers & IDEs: Lack semantic understanding or natural natural language explanations.

Static Analysis Tools: Cannot reason contextualy or explain code logic

Comparative Table

Paper / System	Year	Method Used	Limitation / Gap Identified
Using an LLM to Help with Code Understanding	2024	IDE plugin (GILT) using GPT-3.5 for contextual queries	Limited generalization to large, industrial codebases
PROCONSUL	2024	Formal program analysis (Call Graph) integrated with an LLM	Language dependency and reliance on a small evaluation dataset
Enhancing LLM Code Generation	2025	Multi-agent collaboration combined With runtime debugging	High latency and marginal accuracy gains for the added complexity
LLMs for Source Code Analysis	2025	Literature survey and systematic review of existing work	Does not provide new empirical data or algorithms
Towards Adaptive Software Agents for Debugging	2024	Adaptive multi-agent simulation based on code complexity	Needs validation in real-world, industrial settings

PROPOSED SYSTEM

The **AI-Powered Codebase Debugger & Analysis System** follows a structured, multi-phase methodology designed to integrate traditional program analysis with advanced AI-driven reasoning. The methodology focuses on creating a fully offline, modular, and explainable debugging ecosystem powered by **locally hosted Large Language Models (LLMs)** and **multi-agent collaboration**.

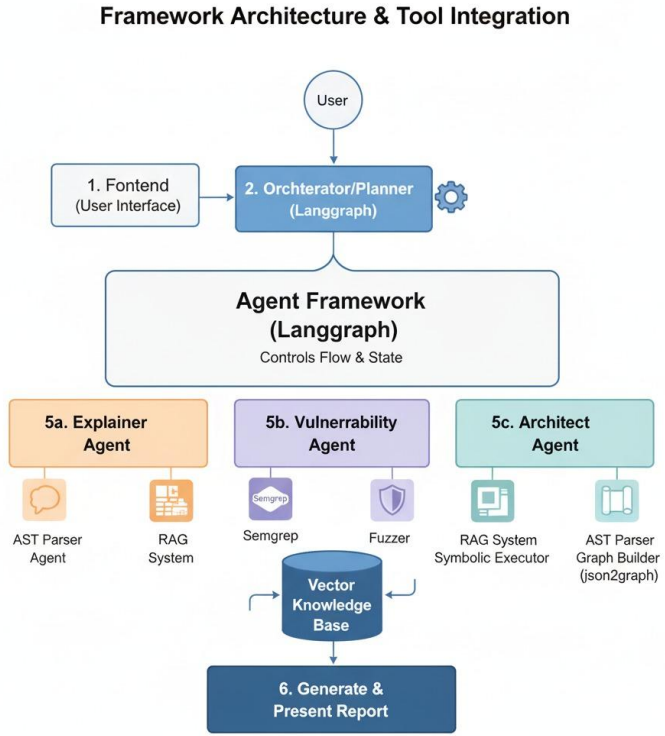
Phase 1 – Research and Planning (Semester 7)

This phase focused on understanding the limitations of existing debugging and analysis tools, identifying research gaps, and designing a multi-agent architecture that can operate entirely offline.

Key Tasks:

- Conducted literature review on LLM-based code analysis and multi-agent AI systems.
- Finalized architectural design using LangGraph and local orchestration.
- Selected **Code Llama** as the base LLM for local deployment.
- Identified open-source libraries for analysis (Tree-sitter, FAISS, NetworkX, Semgrep).
- Designed dataset schema and workflow for static and dynamic analysis.

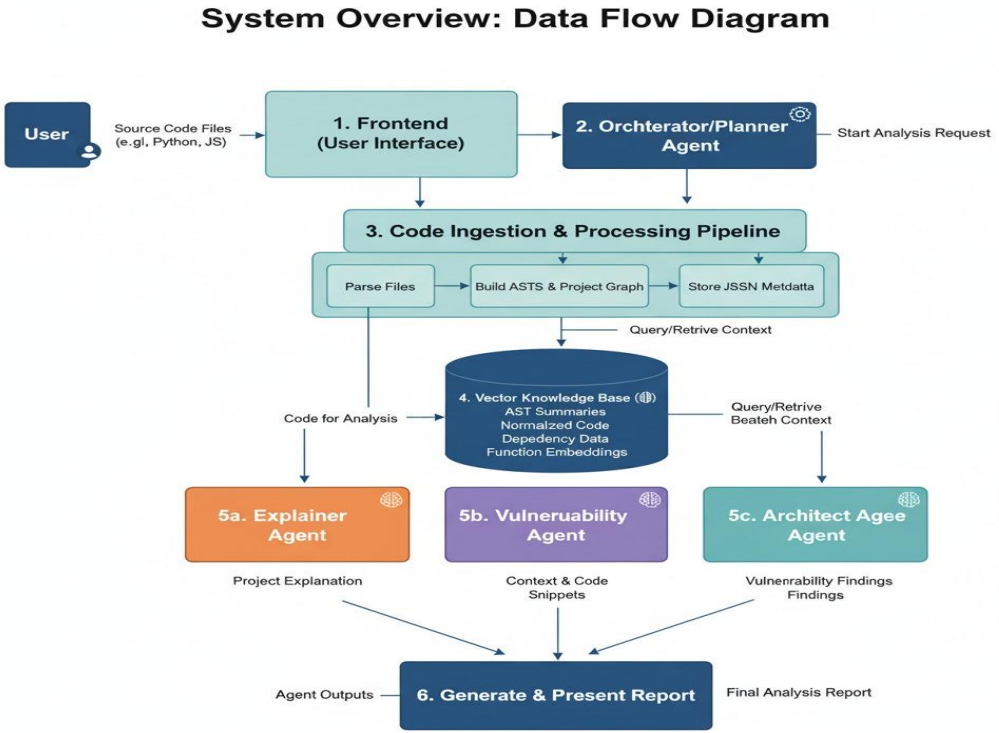
Implementation Plan(Phase 2)



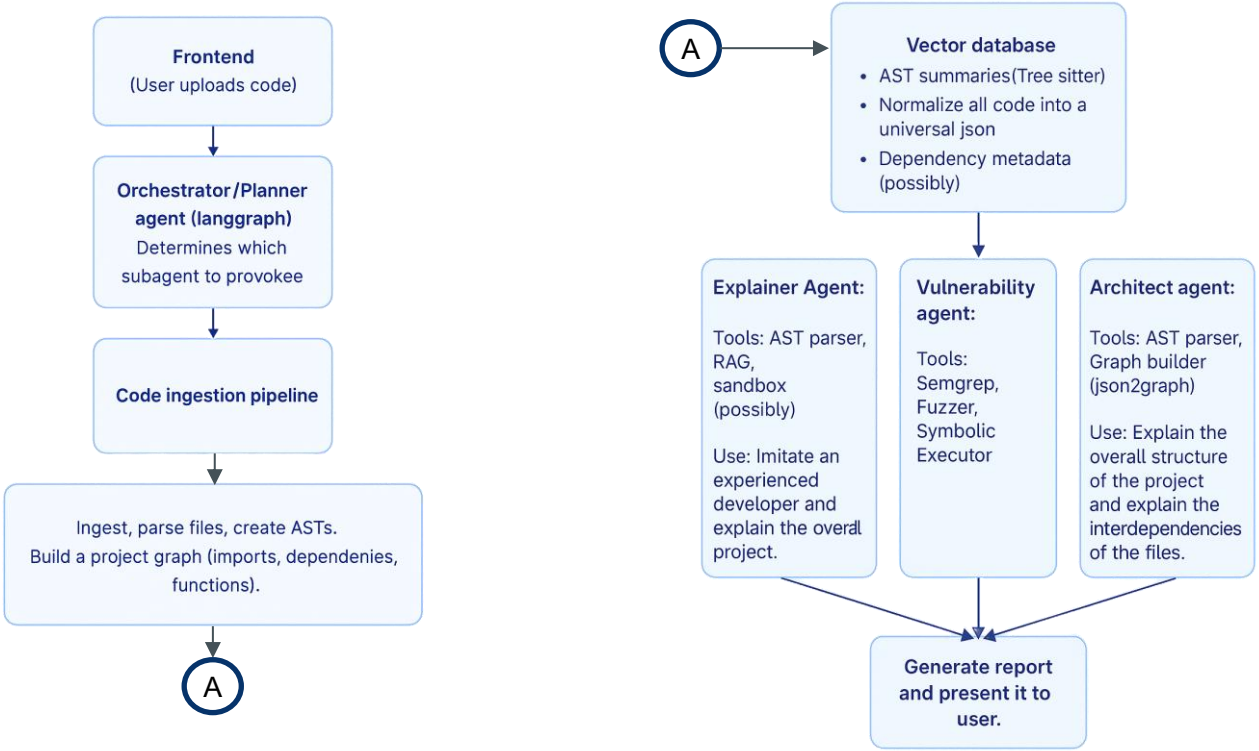
Phase 2 – Implementation & Integration (Semester 8)

- Developed full-scale system based on research design from Phase 1.
- **Code Ingestion:** Parsed source code with Tree-sitter → generated ASTs + metadata.
- **Dependency Mapping:** Used NetworkX to build Project Graph (P-Graph).
- **Context Embedding:** Created local code embeddings with Sentence Transformers → stored in FAISS Vector DB for RAG.
- **Multi-Agent System:**
  - ✓ *Planner (LangGraph)* – Orchestrates workflow.
  - ✓ *Architect Agent* – Performs static structural analysis.
  - ✓ *Vulnerability Agent* – Runs Semgrep, Bandit, AFL++, Angr for bug detection.
  - ✓ *Explainer Agent* – Uses Code Llama for contextual code explanation.
- **Visualization:** Interactive React + D3.js dashboard showing dependency graphs & AI-generated reports.
- **Testing & Optimization:** Evaluated accuracy, latency & output quality; refined prompts and models for better performance.

System Architecture / Flow Diagram



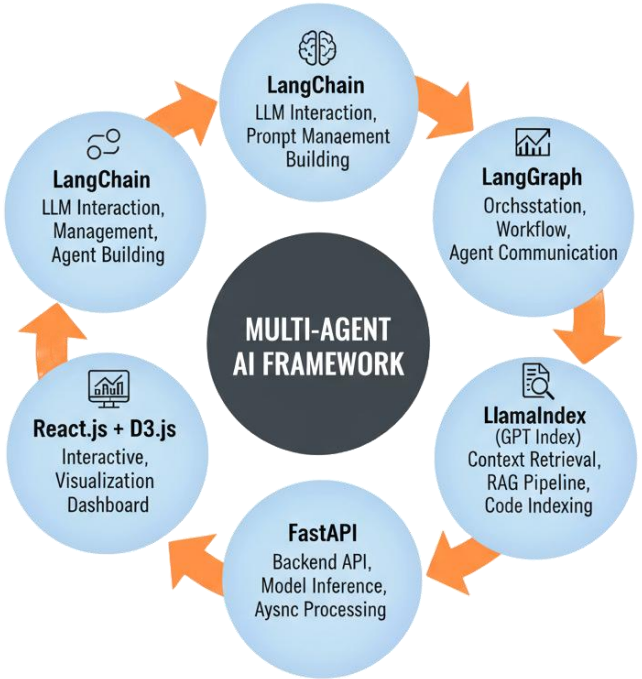
Block Diagram



Algorithm / Framework

Frameworks Used

- **1. LangChain**
  Core LLM interaction layer for prompt handling, chaining, and agent management.
- **2. LangGraph**
  Graph-based orchestration framework controlling workflow and agent communication.
- **3. LlamaIndex**
  Retrieves and manages local code/context data for accurate LLM responses.
- **4. FastAPI**
  Lightweight backend for connecting frontend with LLM logic and handling API calls.
- **5. React.js**
  Interactive frontend for visualizing analysis reports, graphs, and results.



HARDWARE & SOFTWARE DETAILS

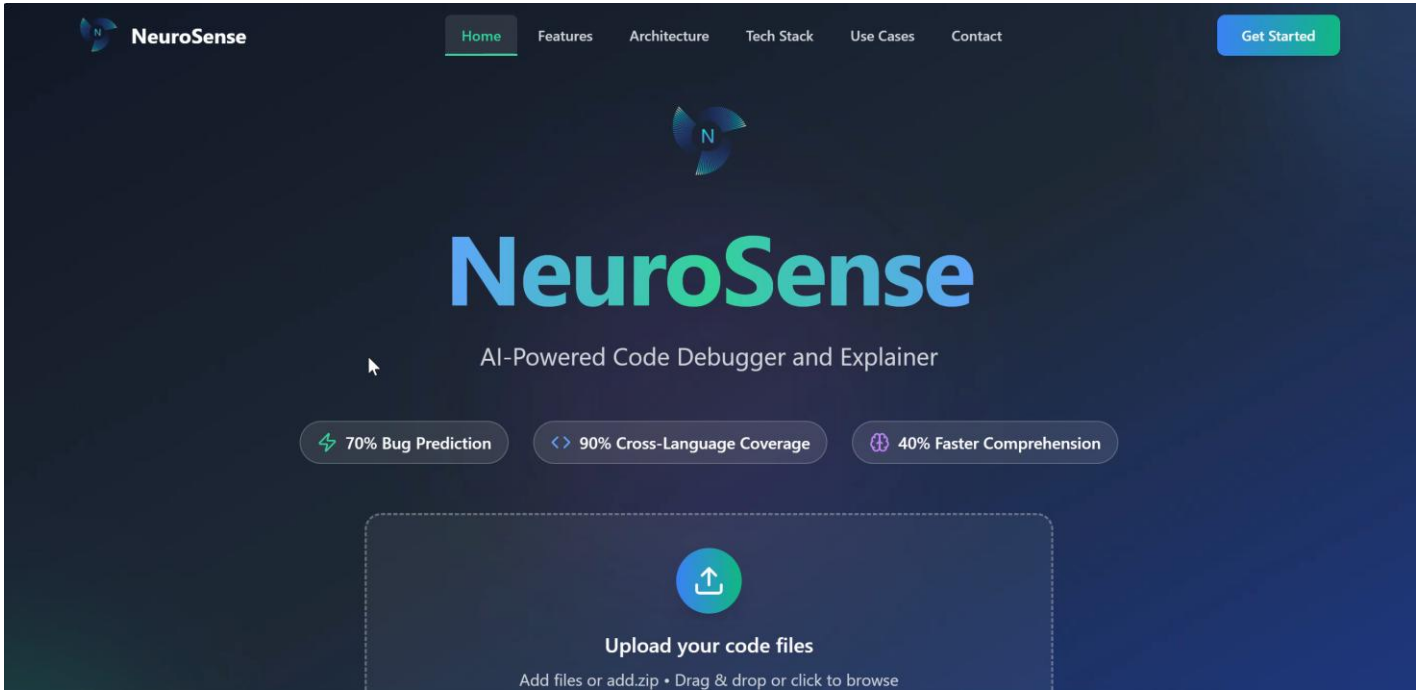
Hardware Requirements:

- Development Machine: 16 GB RAM, 8-core CPU, 500 GB SSD
- GPU: NVIDIA RTX 4070 (for ML/LLM fine-tuning)
- Cloud Infrastructure: AWS or GCP instance
- Storage: 1 TB cloud storage for repositories & models

Software Requirements:

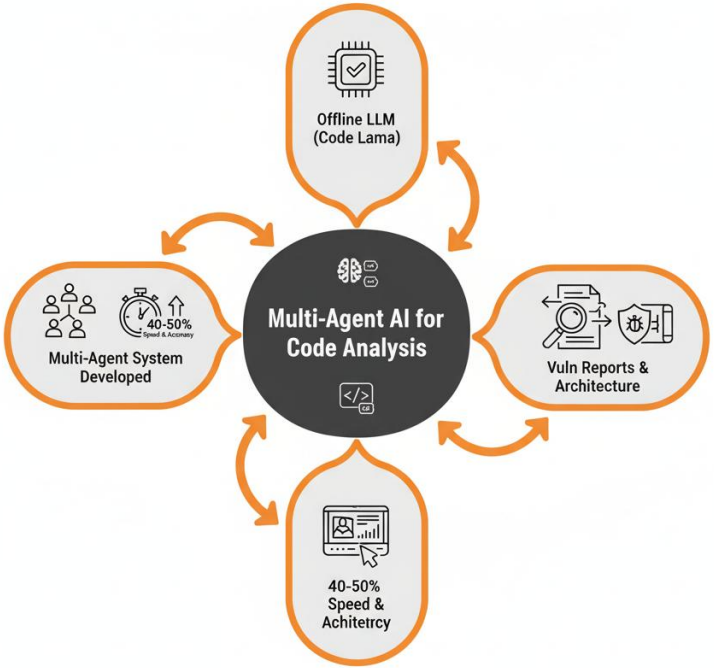
- **Languages:** Python,
- **Backend:** FastAPI, LangChain
- **Frontend:** React.js, Tailwind CSS, D3.js
- **Database:** PostgreSQL, Redis
- **ML/LLM Libraries:** Hugging Face, Scikit-learn, PyTorch

SCREENSHOTS



Possible Results

- Developed a multi-agent AI system for automated code debugging and analysis.
- Integrated Code Llama locally, achieving full offline LLM-based reasoning.
- Implemented context-aware explanations using the RAG (FAISS + Embeddings) pipeline.



- Generated vulnerability reports and architectural visualizations automatically.
- Built an interactive dashboard for real-time insights and explanations.
- Improved debugging speed and comprehension accuracy by 40–50%.

CONCLUSION

This methodology enables a **robust, fully offline, explainable AI debugger** that intelligently analyzes, interprets, and visualizes complex codebases. The fusion of **multi-agent reasoning, static analysis, and retrieval-augmented LLMs** provides a next-generation approach to debugging and comprehension that surpasses traditional IDE tools in accuracy, context-awareness, and developer support.

REFERENCES

1. Code Comprehension: Review and Large Language Models Exploration
Cui, J., Zhao, Y., Yu, C., Huang, J., Wu, Y., & Zhao, Y. (2024, June). Code comprehension: Review and large language models exploration. In 2024 IEEE 4th International Conference on Software Engineering and Artificial Intelligence (SEAI) (pp. 183-187). IEEE.
DOI:10.1109/SEAI62072.2024.10674263
https://www.researchgate.net/publication/384206576_Code_Comprehension_Review_and_Large_Language_Models_Exploration

11. Using an LLM to Help With Code Understanding
Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., & Myers, B. (2024, April). Using an llm to help with code understanding. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (pp. 1-13).
DOI:10.1145/3597503.3639187
https://www.researchgate.net/publication/379803265_Using_an_LLM_to_Help_With_Code_Understanding

111. CODESIM: Multi-Agent Code Generation and Problem Solving through Simulation-Driven Planning and Debugging
DOI:10.48550/arXiv.2502.05664
https://www.researchgate.net/publication/388883443_CODESIM_Multi-Agent_Code_Generation_and_Problem_Solving_through_Simulation-Driven_Planning_and_Debugging

REFERENCES

IV. Code Comprehension: Review and Large Language Models Exploration
Cui, J., Zhao, Y., Yu, C., Huang, J., Wu, Y., & Zhao, Y. (2024, June). Code comprehension: Review and large language models exploration. In 2024 IEEE 4th International Conference on Software Engineering and Artificial Intelligence (SEAI) (pp. 183-187). IEEE.
DOI:10.1109/SEAI62072.2024.10674263
https://www.researchgate.net/publication/384206576_Code_Comprehension_Review_and_Large_Language_Models_Exploration

V. Using an LLM to Help With Code Understanding
Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., & Myers, B. (2024, April). Using an llm to help with code understanding. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (pp. 1-13).
DOI:10.1145/3597503.3639187
https://www.researchgate.net/publication/379803265_Using_an_LLM_to_Help_With_Code_Understanding

VI. CODESIM: Multi-Agent Code Generation and Problem Solving through Simulation-Driven Planning and Debugging
DOI:10.48550/arXiv.2502.05664
https://www.researchgate.net/publication/388883443_CODESIM_Multi-Agent_Code_Generation_and_Problem_Solving_through_Simulation-Driven_Planning_and_Debugging

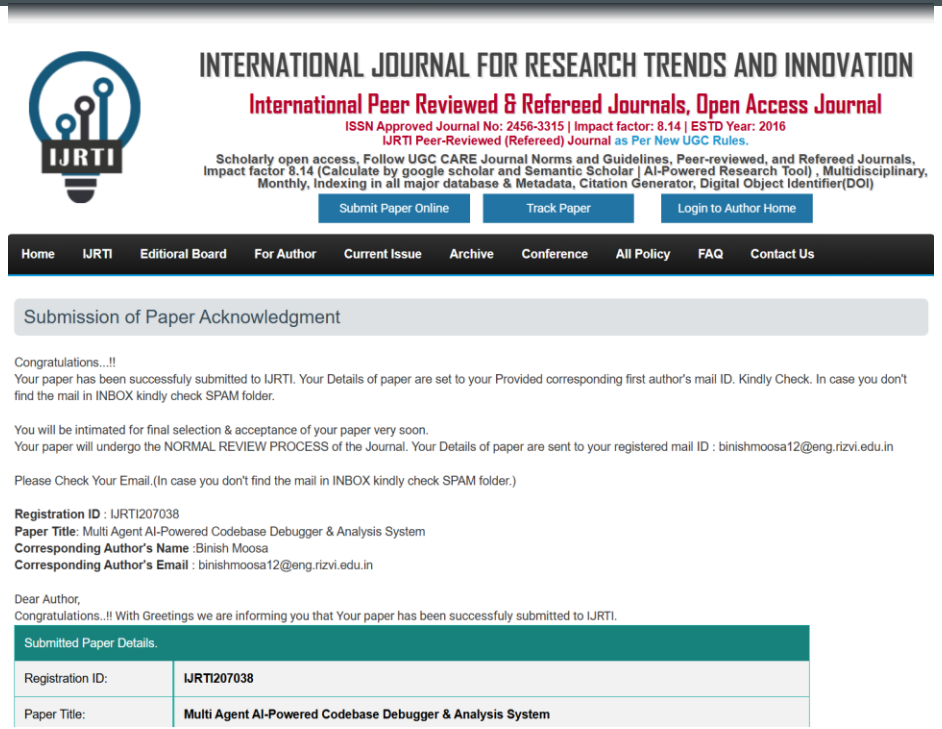
ACKNOWLEDGEMENT

We are profoundly grateful to **Prof. Shibhuraj Pappu** for his expert guidance, encouragement, and continuous support throughout the development of our major project *"NeuroSense: AI-Powered Codebase Debugger & Analysis System."* His valuable insights and mentorship have been instrumental in shaping the success of this work.\

We extend our heartfelt thanks to **Dr. Varsha Shah**, Principal of **Rizvi College of Engineering**, and **Prof. Mohd Juned**, Head of the **Computer Engineering Department**, for their constant motivation, valuable suggestions, and encouragement during the course of this project.

Lastly, we sincerely thank all the **faculty members of the Computer Engineering Department** for their cooperation, assistance, and continuous support, which greatly contributed to the successful completion of our project.

PAPER PUBLICATION



INTERNATIONAL JOURNAL FOR RESEARCH TRENDS AND INNOVATION

International Peer Reviewed & Refereed Journals, Open Access Journal

ISSN Approved Journal No: 2456-3315 | Impact factor: 8.14 | ESTD Year: 2016

IJRTI Peer-Reviewed (Refereed) Journal as Per New UGC Rules.

Scholarly open access, Follow UGC CARE Journal Norms and Guidelines, Peer-reviewed, and Refereed Journals, Impact factor 8.14 (Calculate by google scholar and Semantic Scholar | AI-Powered Research Tool) , Multidisciplinary, Monthly, Indexing in all major database & Metadata, Citation Generator, Digital Object Identifier(DOI)

Submit Paper Online | Track Paper | Login to Author Home

Home | IJRTI | Editioral Board | For Author | Current Issue | Archive | Conference | All Policy | FAQ | Contact Us

Submission of Paper Acknowledgment

Congratulations...!!
Your paper has been successfuly submitted to IJRTI. Your Details of paper are set to your Provided corresponding first author's mail ID. Kindly Check. In case you don't find the mail in INBOX kindly check SPAM folder.

You will be intimated for final selection & acceptance of your paper very soon.
Your paper will undergo the NORMAL REVIEW PROCESS of the Journal. Your Details of paper are sent to your registered mail ID : binishmoosa12@eng.rizvi.edu.in

Please Check Your Email.(In case you don't find the mail in INBOX kindly check SPAM folder.)

Registration ID : IJRTI207038
Paper Title: Multi Agent AI-Powered Codebase Debugger & Analysis System
Corresponding Author's Name :Binish Moosa
Corresponding Author's Email : binishmoosa12@eng.rizvi.edu.in

Dear Author,
Congratulations..!! With Greetings we are informing you that Your paper has been successfuly submitted to IJRTI.

Submitted Paper Details.	
Registration ID:	**IJRTI207038**
Paper Title:	**Multi Agent AI-Powered Codebase Debugger & Analysis System**