

FINAL YEAR (BE) - MAJOR PROJECT PROPOSAL

A.Y. 2025-26

Title of Project

Autonomous Intelligent System for DevOps Automation

Submitted by

Group No.	Name of Students	Roll No.
4	Binayak Bhattacharjee	09
	Shreyash Das	15
	Kapil Dhavale	16
	Sahil Sawant	58

Name of Project Guide:

Dr. T. Rajani Mangala



DEPARTMENT OF ELECTRONICS AND COMPUTER SCIENCE

V.E.S. INSTITUTE OF TECHNOLOGY, CHEMBUR, MUMBAI-74

(An Autonomous College Affiliated to University of Mumbai, Approved by AICTE & Recognized by
Govt. of Maharashtra)

Table of Contents

No.	Chapter	Page No.
1	Introduction	1
2	Literature Review	3
3	Methodology for Implementation of Project	5
4	Project Schedule Plan	9
5	Expected Project Outcomes	10
6	References	12
7	Undertaking	13

List of Figures

No.	Title	Page No.
1	Fig.3.1 Block diagram of the proposed system	6

List of Tables

No.	Title	Page No.
1	Table 3.2: List of Software	8
2	Table 4.1: Plan of Action	9

1.Introduction

1.1 Background Information:

In modern software development, DevOps practices—such as containerization, continuous integration and deployment (CI/CD), automated testing, and proper documentation—are critical for building scalable and reliable systems. However, small teams, student developers, and early-stage projects often struggle to adopt these practices due to limited expertise, time, or resources. As a result, many promising applications face issues like poor maintainability, lack of automation, and slower deployment cycles.

Agentic AI presents a powerful solution to this challenge. It refers to autonomous systems capable of planning, reasoning, and executing multi-step tasks with minimal human intervention. This project aims to build an Agentic DevOps agent that can analyze codebases, identify gaps in DevOps implementation, and autonomously generate configurations, pipelines, and tests. By acting like a junior DevOps engineer, this AI agent can make robust DevOps practices accessible to developers without specialized knowledge—thereby improving software quality, speeding up deployment, and enhancing overall project sustainability.

1.2 Problem Statement:

Setting up and maintaining DevOps pipelines manually is labor-intensive, prone to human error, and often varies in quality from one project to another. Engineers routinely invest substantial effort into authoring Dockerfiles, configuring CI/CD workflows, and scaffolding test suites—tasks that detract from delivering core application features. To streamline development and ensure consistency, there is a pressing need for an autonomous solution capable of analyzing existing repositories and automatically provisioning complete, production-ready DevOps environments.

1.3 Objectives:

1. **Analyze Repositories:** Automatically inspect and parse Node.js codebases to detect missing or outdated DevOps components (e.g., Dockerfiles, CI/CD workflows, test suites) with 95% accuracy in detection.
2. **Generate DevOps Artifacts:** Employ LLM-driven generation to create valid Docker configurations, GitHub Actions pipelines, and unit test scaffolding that execute successfully in at least 90% of automated validation tests.
3. **Enhance Documentation:** Programmatically produce or augment README files with clear installation, usage, and deployment instructions, achieving full documentation coverage across processed repositories.
4. **Integrate with Version Control:** Seamlessly commit generated artifacts to Git, push changes to remote repositories, and open pull requests automatically, ensuring 100% success in integration trials.
5. **Validate End-to-End Workflow:** Demonstrate complete automation from code analysis to artifact generation and repository update through three consecutive error-free end-to-end runs.

2. Literature Review

2.1 Existing Research:

- **LADs: Leveraging LLMs for AI-Driven DevOps (2025)**

This study introduces *LADs*, a comprehensive framework that utilizes Large Language Models (LLMs) to automate cloud configuration and deployment processes. It combines advanced techniques such as Retrieval-Augmented Generation (RAG), Few-Shot Learning, Chain-of-Thought prompting, and feedback-driven prompt refinement. *LADs* is designed to handle challenges in dynamic cloud environments like misconfigurations and heterogeneous deployment settings by learning iteratively from failures and adapting resource usage. The framework is evaluated on robustness, scalability, and cost-effectiveness, with open-source tools available for community adoption.

Impact: *LADs* significantly reduces manual effort in large-scale cloud DevOps, improving reliability and optimizing resource consumption.

Comparison & Relevance: While *LADs* focuses on optimizing large-scale cloud deployment, our project targets early-stage and small-to-medium Node.js projects by introducing autonomous agents capable of bootstrapping the entire DevOps setup. Unlike *LADs*, which concentrates on runtime optimization, our system focuses on *initial project setup*—analyzing existing repositories to generate Dockerfiles, CI/CD pipelines, tests, and documentation—thereby enabling DevOps adoption from the ground up.

- **Automated DevOps Pipeline Generation for Code Repositories using Large Language Models (arXiv:2312.13225, 2023)**

This research explores the effectiveness of GPT-3.5 and GPT-4 in generating GitHub Actions workflows based on repository content. The authors introduce a GitHub App (built with Probot) that automatically analyzes repositories and inserts generated CI/CD configurations. The system is evaluated using metrics like BLEU scores and exact match scores, highlighting the improved accuracy and context awareness of newer LLMs.

Impact: The study shows strong potential for LLMs in automating DevOps tasks, especially in workflow file creation, thereby reducing manual configuration efforts for developers.

Comparison & Relevance: While this system effectively generates CI/CD workflows, our project expands the scope by employing an agentic AI approach. Our agent not only creates CI workflows but also generates Dockerfiles, scaffolds unit tests (Jest/Mocha), enhances documentation, and integrates changes via Git operations (commits and pull requests). This broader automation pipeline makes our solution more comprehensive for DevOps bootstrapping in Node.js applications.

3. Methodology

3.1 Project Design:

- **Scope & Requirements**
 - Identify required DevOps artifacts: Dockerfile, CI/CD workflow, test scaffolds, README enhancements.
- **Repository Analysis**
 - Parse project manifest (e.g., package.json) and inspect directory structure.
 - Generate a “gap report” of missing or outdated DevOps components.
- **Agentic Orchestration & Artifact Generation**
 - Assemble dynamic prompts containing project metadata, detected gaps, and template snippets.
 - Dispatch specialized sub-agents to produce initial versions of each artifact (Dockerfile, GitHub Actions YAML, test suites, documentation).
- **Automated Validation & Feedback Loop**
 - Run linters (e.g., hadolint, YAML syntax checkers), container builds, and generated test suites.
 - Capture any errors and re-invoke sub-agents for up to two refinement iterations.
- **Documentation Enhancement & GitOps Integration**
 - Auto-generate Install/Usage/Testing/Deployment sections in README.md.
 - Create feature branches, commit changes via Octokit.js or simple-git, and open pull requests summarizing added artifacts.
- **End-to-End Pipeline Execution & Continuous Improvement**
 - Execute the newly generated CI/CD workflow on PR creation.
 - After three consecutive successful runs, mark the agent version as stable and iteratively refine prompts/templates based on logs and feedback.

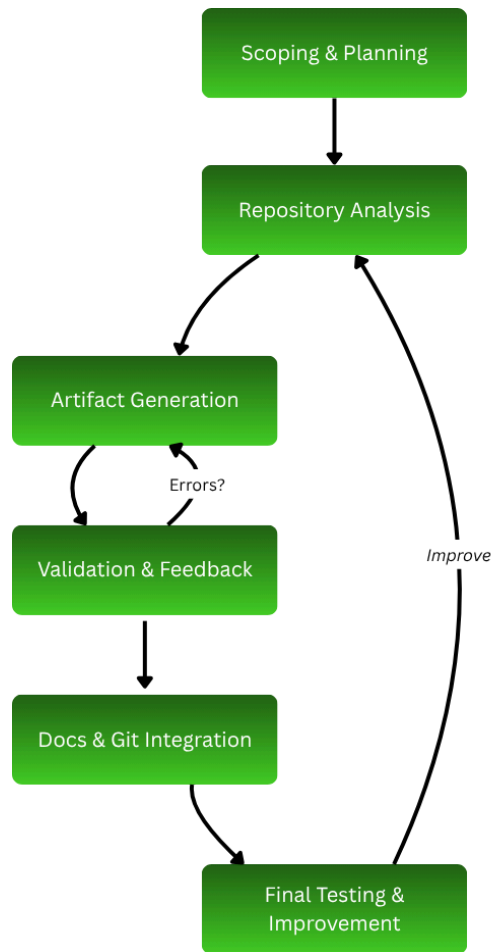


Fig. 3.1: Block diagram of the proposed system

3.2 Method of Analysis:

Stage 1: Scoping & Requirements

1.1 Artifact Scope

- Decide which DevOps components to bootstrap:
 - Container specs (e.g. Dockerfile)
 - CI/CD pipelines (e.g. GitHub Actions)
 - Test scaffolds (unit/integration)
 - Project docs (README.md)

Stage 2: Repository Analysis

2.1 Lightweight Scan

- Parse project manifest (e.g., package.json, requirements.txt) to identify tech stack
- Inspect file tree for existing DevOps artifacts

2.2 Gap Detection

- Flag missing/outdated items against the artifact scope
- Record findings in an internal “gap report”

Stage 3: DevOps Artifact Generation

3.1 Prompt & Context Assembly

- Build dynamic prompts embedding:
 - Repository metadata (language, frameworks, deps)
 - Detected gaps from Stage 2
 - Minimal template snippets

3.2 Agentic Orchestration

- Choose orchestration engine (OpenAI Agents SDK/LangChain)
- Spawn sub-agents for each artifact type:
 - Dockerfile generator
 - CI/CD workflow generator
 - Test scaffold generator

3.3 First-Pass Generation

- Sub-agents produce initial versions of each artifact

Stage 4: Automated Validation

4.1 Sanity Checks

- Lint Dockerfile (hadolint)
- Syntax-lint CI YAML
- Run container build (docker build)
- Execute generated tests (npm test, pytest, etc.)

4.2 Feedback Loop

- Capture errors and stack traces
- Re-prompt sub-agents for up to two refinement iterations

Stage 5: Documentation Enhancement & GitOps Integration

5.1 README Augmentation

- Auto-generate Install, Usage, Testing, Deployment sections from project metadata
- Ensure coverage of all critical commands and environment configs

5.2 Automated Branching & Commits

- Create branch agent/devops-<artifact> for each artifact
- Stage and commit changes with templated messages via Git APIs (Octokit.js or simple-git)

5.3 Pull Request Automation

- Open PRs summarizing gaps and added artifacts
- Optionally trigger the new CI/CD pipeline on the PR for final validation

Stage 6: End-to-End Validation & Continuous Improvement

6.1 Full-Pipeline Runs

- On PR creation, execute the generated workflow end-to-end
- Confirm all steps (build, test, lint) pass without errors

6.2 Iterative Refinement

- After three consecutive successful runs, mark the agent version as stable
- Periodically review logs and user feedback to update prompts and templates

3.3 List of Software: -

Software:

Table 3.2: List of Software

Sr. No.	Name of Software	Version
1	n8n	v1.103.2
2	LangChain	v0.3.27
3	Python	≥3.10
4	Docker	latest 2025
5	GitHub Actions	-

4.Schedule Plan

Timeline:

Table 4.1: Plan of Action

Activity	July	Aug	Sept	Oct	Nov	Dec	Jan	Feb	Mar
Requirements Analysis & Scoping									
Codebase Analysis Module Development									
LLM/Agentic Artifact Generation Engine									
Workflow Orchestration & Integration									
Automated Validation & End-to-End Testing									
Documentation & Iterative Refinement									

5.Expected Project Outcomes

5.1 Results:

Automated DevOps Enablement: The primary result will be a fully autonomous agent that successfully detects DevOps gaps (missing Dockerfiles, CI/CD pipelines, tests, documentation) in diverse code repositories and generates production-grade replacements or additions with high accuracy. This will radically lower the technical and time barriers for non-experts and small teams to adopt industry-standard DevOps practices.

Consistency and Reliability: By standardizing DevOps configurations and consistently applying best practices across projects, the agent will minimize human error and environment drift leading to more maintainable, robust, and secure deployments.

Accelerated Development Cycles: Automated CI/CD setup, test scaffolding, and documentation will reduce project onboarding time and manual repetitive tasks, allowing developers to focus on core feature development and innovation rather than infrastructure scripting.

Measurable Quality Improvements: Successful real-world trials should demonstrate higher build/test pass rates, improved repository documentation quality, and reduced operational incidents in post-adoption repositories, directly supporting project sustainability and scalability.

Demonstrated End-to-End Automation: The project will showcase three fully automated, error-free integrations (from analysis to pull request) across different repositories as a proof of repeatable impact and reliability.

5.2 Applications:

Open-Source and Student Projects: Immediate enablement for students, hackathon teams, and contributors lacking DevOps expertise allowing rapid setup of modern workflows and tests regardless of experience level.

Early-Stage Startups: Fast-tracking the implementation of scalable, production-grade DevOps from day one, helping lean teams avoid technical debt and maintain deployment velocity under resource constraints.

Legacy Code Modernization: Streamlined upgrading of older or neglected codebases to current DevOps standards, prime for organizations looking to modernize without major manual retooling.

Education and Training: Serving as an interactive learning tool or “junior DevOps engineer” assistant for developer bootcamps, online courses, and self-learners wanting hands-on, AI-guided DevOps experience in their projects.

Template and Best-Practices Propagation: Organizations can encode customized standards via the agent, using it to propagate and enforce compliance across all new and existing repositories automatically.

6. References

[1] D. Mehta, K. Rawool, S. Gujar, and B. Xu, "Automated DevOps Pipeline Generation for Code Repositories using Large Language Models," arXiv preprint arXiv:2312.13225, Dec. 2023.

This paper investigates the application of GPT-3.5 and GPT-4 for auto-generating GitHub Actions workflows directly from codebases. The authors introduce new evaluation metrics for DevOps pipeline correctness and a GitHub App for practical workflow integration, offering a pioneering technical foundation for AI-driven DevOps.

[2] G. Kambala, "Intelligent Software Agents for Continuous Delivery: Leveraging AI and Machine Learning for Fully Automated DevOps Pipelines," *Iconic Research and Engineering Journals*, vol. 8, no. 1, pp. 662-670, 2024.

This review explores how intelligent software agents, AI, and ML empower the full automation of software delivery pipelines—including testing, integration, deployment, and monitoring—highlighting significant benefits in efficiency, reliability, and software quality.

[3] P. Fitsilis, V. Damasiotis, V. Kyriatzis, and P. Tsoutsas, "DOLLmC: DevOps for Large Language Model Customization," arXiv preprint arXiv:2405.11581, May 2024.

This recent study presents a customizable DevOps framework for LLM-centric applications, emphasizing continuous learning, prompt engineering, and domain adaptation across the entire DevOps lifecycle.

[4] Y. Chen et al., "AIOpsLab: A Holistic Framework to Evaluate AI Agents for Enabling Autonomous Clouds," arXiv preprint arXiv:2501.06706, Jan. 2025.


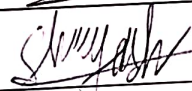

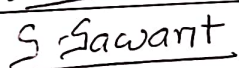
The authors introduce "AgentOps," an agentic paradigm for fully autonomous cloud operations, describing their AIOpsLab framework for benchmarking the effectiveness of next-generation AI agents in multi-task, cross-layer operational settings—a direct inspiration for agentic DevOps research.

[5] IJSRA, "Integrating AI into DevOps pipelines," *Int. J. Sci. Res. Arch.*, vol. 8, no. 2, pp. 52-57, Feb. 2024.

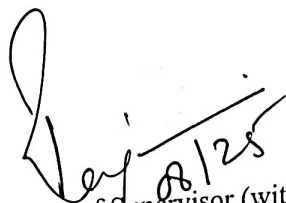
This paper reviews how AI augments CI/CD, automates repetitive DevOps tasks, and enables predictive analytics, providing insights on future trends and challenges.

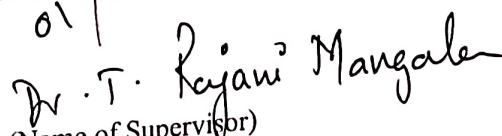
Undertaking by Students

I have adhered to all departmental guidelines and instructions in preparing and submitting this project proposal. Under the guidance of **Dr. T. Rajani Mangala**, I have incorporated all feedback and suggestions received during the drafting process. This proposal complies with all ethical standards, with all sources of information properly cited. I take full responsibility for the content of this proposal and acknowledge that any errors or omissions are my own, committing to corrective action if necessary. I am dedicated to executing the proposed project as outlined and will strive to achieve its objectives to the best of my ability.

Roll No.	Name of Students	Sign
09	Binayak Bhattacharjee	
15	Shreyash Bibhu Das	
16	Kapil Pramod Dhavale	
58	Sahil Santosh Sawant	

Project Guide Comments


Signature of Supervisor (with date)
01/02/25


(Name of Supervisor)