Int. J. Sci. R. Tech., 2026 3(5)
A Multidisciplinary peer-reviewed Journal
www.ijsrtjournal.com [ISSN: 2394-7063]

Original Article

Developing A Virtual Leader Using NLP And AI

Dnyaneshwari Rode, Snehal Kamble, Ashwini Kauthale, Sakshi Darandale, S. S. Shinde*, S. R. Patil

Computer Engineering, Sinhgad Institute of Technology, Savitribai Phule Pune University, Pune, India

ABSTRACT

Managing software development teams spread across different locations is a growing challenge. Technical leads spend a significant portion of their time on repetitive yet critical coordination tasks such as scheduling meetings, assigning work, reviewing code, and managing HR processes. This paper introduces Virtual Leader, an autonomous AI agent built on the Pilot AI platform that automates these responsibilities. Using Natural Language Processing (NLP), large language model (LLM) reasoning, and GitHub automation, Virtual Leader reads team conversations, infers required actions, and responds accordingly — from assigning tasks to approving leave requests. Deployed on AWS, the system operates transparently alongside human teams with all consequential actions subject to human approval.

Keywords: Artificial Intelligence, Natural Language Processing, Software Team Management, AI Agent, LLM, Autonomous Systems, GitHub Automation, Task Assignment.

INTRODUCTION

Software teams today often operate remotely across different time zones. In this environment, maintaining team alignment is increasingly difficult. Research shows that coordination activities — such as task assignment, follow-ups, and meeting scheduling — can consume up to 40% of a developer's workday in teams of five or more members [1].

Existing tools address only isolated aspects of this problem. GitHub Copilot assists with code generation, Clockwise optimises meeting scheduling, and Jira manages task tracking. However, none of these tools share context with each other or respond dynamically to the full picture of what a team needs at any given time.

Virtual Leader was developed to address this gap. It is an AI agent embedded within the team's messaging platform that acts as a complete technical lead — reading messages, inferring intent, and taking coordinated action across tasks, meetings, code, HR management, and project health monitoring. Unlike simple chatbots, Virtual Leader proactively monitors team activity and intervenes when required, without waiting to be addressed directly.

LITERATURE REVIEW

The use of AI for automating software development workflows has grown significantly in recent years. Cataldo and Herbsleb [1] demonstrated that coordination breakdowns in distributed teams directly degrade productivity, motivating the need for automated coordination mechanisms. Ziegler et al. [2] evaluated neural code completion tools and found measurable productivity improvements, establishing the viability of AI co-developers in professional settings.

Wang et al. [4] benchmarked large language models on calendar scheduling tasks, revealing both promising capabilities and key limitations in multi-constraint scheduling scenarios. Dominic and Bhatt [5] proposed a skill-graph approach to developer-task matching in distributed agile teams, showing that skill-aware assignment outperforms round-robin or manual delegation.

Zhang et al. [6] explored reinforcement learning for dynamic task assignment in software projects, providing a theoretical basis for adaptive, feedback-driven allocation strategies. The use of spaCy for industrial-strength NLP pipelines [3] has been widely validated in applied settings, making it a strong

Relevant conflicts of interest/financial disclosures: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

foundation for intent classification and entity extraction in conversational AI systems.

RELATED WORK

Several platforms have attempted to address team coordination through automation. GitHub Copilot focuses exclusively on code generation and does not interact with team communication channels. Clockwise and Motion optimise calendar scheduling but lack integration with project management or code review workflows. Jira and Linear offer task tracking but require manual input for every update and do not process conversational messages.

AI-driven project management research has explored LLM-based planning agents, but most prototypes operate in isolation from real communication platforms. No prior system combines NLP-driven intent classification, LLM reasoning, GitHub integration, and HR management within a single unified agent that runs inside a team's existing messaging infrastructure. Virtual Leader uniquely addresses this integration gap.

METHODOLOGY

A. Platform Foundation

Virtual Leader is built on top of Pilot AI, a messaging platform constructed using Django 4.2 for the backend, Django Channels for real-time WebSocket communication, a spaCy NLP pipeline for language understanding, and a plain JavaScript frontend. Every message is automatically analysed before storage.

B. NLP Pipeline

Incoming messages are processed through four sequential stages:

- Intent classification: question, task, meeting, help request, or announcement
- Sentiment scoring: positive or negative polarity
- Named Entity Recognition (NER): extraction of names, dates, and technologies using spaCy
- Keyword identification: extraction of topic-relevant terms

C. LLM Decision Engine

An LLM (GPT-4o or Claude 3.5 Sonnet) receives the classified message alongside recent conversation history, the current state of open tasks and meetings, and a description of available actions. It produces a structured action plan specifying the module to invoke and the ordered steps to execute.

D. Training and Configuration Details

The spaCy en_core_web_sm model is used for NER and dependency parsing. The routing table is configured with keyword-to-module mappings updated through admin settings. LLM prompts are versioned and stored in the database to allow adjustment without code deployment. The system is containerised using Docker and deployed on AWS ECS Fargate.

SYSTEM DESIGN

Virtual Leader is registered as a special user (role: ai_lead) within the Pilot AI platform and connects through the same WebSocket infrastructure used by human team members. This design allows it to receive every message in real time, send replies, create threads, and send direct messages without requiring special API endpoints.

Messages are routed to one of five capability modules based on intent classification. The routing is governed by a keyword-trigger table (Table II). Each module exposes a standardised interface accepting a structured context object and returning a list of actions to be executed. All irreversible actions are held in a pending state until a human team member confirms them via a channel reaction.

The production deployment runs three independent Docker services on Amazon ECS Fargate: the WebSocket server, the HTTP API server, and a background worker pool for scheduled tasks such as reminders and health-score calculations. The database is hosted on Amazon RDS (PostgreSQL) and real-time state is managed through Amazon ElastiCache (Redis). Secrets are stored in AWS Secrets Manager.

MODULE ARCHITECTURE

Virtual Leader is composed of five distinct functional modules, each responsible for a specific domain of team coordination. The architecture ensures that each module operates independently while sharing a

common context object passed by the LLM decision engine.

A. Task Manager

When a project requirement is posted, the Task Manager decomposes it into subtasks, estimates effort, identifies required skills, and assigns each task to the most suitable developer based on their GitHub commit history, discussed technologies, and declared skills. Deadline reminders are sent automatically 24 hours before due dates.

B. Meeting Coordinator

This module monitors channels for signals that a meeting is needed — such as unresolved multi-turn discussions or approaching sprint milestones. It queries the Google Calendar API to identify the earliest available slot for all required attendees, dispatches invitations, and post-meeting generates a structured summary including decisions and action items converted into tracked tasks.

C. Code Assistant

On pull request creation, the Code Assistant fetches the diff, evaluates it against project coding standards, posts inline comments, and shares a quality score and recommendation in the project channel. It can also create branches, commit boilerplate code, and update documentation. All AI-generated commits carry a [VL] prefix for transparency.

D. HR Liaison

Developers submit leave requests through natural conversation. The HR Liaison extracts dates, checks leave policy and balance, and approves or escalates the request. It monitors online presence as a proxy for attendance, sends weekly attendance summaries to administrators, and generates sprint-end performance reports labelled explicitly as AI inputs.

E. Health Monitor

Every 15 minutes, a Project Health Score (0–100) is computed from five weighted signals: task completion rate (30%), open blockers (25%), team sentiment (20%), PR review speed (15%), and standup participation (10%). Alerts are dispatched when the score drops sharply or specific risk conditions are detected, such as three or more open blockers or a developer with no commits for five days.

RESULTS

A. Capability Summary

The table below summarises the five core modules, their responsibilities, and primary actions:

Module	Responsibility	Key Actions
Task Manager	Task decomposition and assignment	Skill-based assignment, deadline reminders
Meeting Coord.	Scheduling and agenda management	Auto-scheduling, action item tracking
Code Assistant	Code review and GitHub automation	PR review, branch management, [VL] commits
HR Liaison	Leave and attendance management	Leave approvals, performance reports
Health Monitor	Project risk monitoring	Health scoring, risk alerts, sprint reports

TABLE I: Virtual Leader Capability Modules

B. Intent-to-Module Routing

The routing table below shows how message intent maps to the appropriate module:

Intent	Trigger Keywords	Module
Task	deadline, assign, todo	Task Manager
Meeting	standup, sync, schedule	Meeting Coordinator
Help	error, bug, code	Code Assistant
Status	status, progress	Health Monitor
HR/Leave	leave, absent, HR	HR Liaison

TABLE II: Intent-to-Module Routing

C. Observations

Virtual Leader successfully handles the full spectrum of a technical lead's coordination duties within a unified conversational interface. The attention mechanism of the LLM decision engine accurately routes complex multi-intent messages. Human approval gates have been found to build user trust without materially slowing operations, as most approvals are granted within minutes.

CONCLUSION

This paper presented Virtual Leader, a compact yet comprehensive AI agent designed to automate the full scope of software team coordination. By embedding itself directly into the team's communication platform, it functions as a natural participant rather than an external tool, offering capabilities spanning task assignment, automated meeting scheduling, GitHub-integrated code review, HR management, and continuous project health monitoring.

The goal of Virtual Leader is not to replace human leaders but to eliminate the repetitive coordination overhead that consumes their time, freeing them to focus on architecture, mentorship, and strategy. With transparent, auditable behaviour and human approval gates for all consequential actions, the system provides a trustworthy foundation for AI-assisted software team management.

FUTURE WORK

Future improvements planned for Virtual Leader include:

- Integration with external task management tools such as Jira and Asana
- Replacement of spaCy en_core_web_sm with a fine-tuned sentence transformer for improved nuance handling
- Introduction of a lightweight local LLM for simple routing decisions to reduce cost and latency
- Development of a voice interface for meeting interactions
- Application of reinforcement learning from human feedback (RLHF) to continuously improve task assignment accuracy over time

Broadening of skill profiling to incorporate code review activity, pair programming sessions, and documentation contributions

REFERENCES

1. M. Cataldo and J. D. Herbsleb, "Coordination breakdowns and their impact on development productivity," IEEE Trans. Software Eng., vol. 39, no. 3, pp. 343–360, 2013.
2. A. Ziegler et al., "Productivity assessment of neural code completion," ACM SIGPLAN Symp. Machine Programming, 2022.
3. M. Honnibal et al., "spaCy: Industrial-strength NLP in Python," Zenodo, 2020.
4. X. Wang et al., "Can LLMs plan? Benchmarking LLMs in calendar scheduling," arXiv:2404.13678, 2024.
5. P. Dominic and R. Bhatt, "Skill-graph-based developer-task matching in distributed agile teams," J. Syst. Software, vol. 189, 2022.
6. Y. Zhang, F. Wu, and S. Ji, "Reinforcement learning for dynamic task assignment in software projects," IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 5, 2023

HOW TO CITE: Dnyaneshwari Rode, Snehal Kamble, Ashwini Kauthale, Sakshi Darandale, S. S. Shinde*, S. R. Patil, Developing A Virtual Leader Using NLP And AI, Int. J. Sci. R. Tech., 2026, 3 (5), 298-301. https://doi.org/10.5281/zenodo.15179749