International Journal of AI, BigData, Computational and Management Studies
Noble Scholar Research Group | Volume 7, Issue 1, 70-73, 2026
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V7I1P111

Original Article

Agentic Leave and Dispatch Automation for Trucking Fleets Using MCP and LLMs

Vamshi Krishna Malthummeda
Independent Researcher, USA.

Received On: 11/12/2025 ***Revised On: 13/01/2026*** ***Accepted On: 20/01/2026*** ***Published On: 04/02/2026***

Abstract: *Large trucking companies employing thousands of drivers operate in a complex work pattern with weekend and night shift operations. These kinds of large fleets face operational disruptions due to unplanned absences by the drivers. Looking up the leave balances in the HR system manually and rescheduling dispatches are slow, error-prone, and often result in unexpected payroll deductions due to insufficient leave balances. This paper presents Conversational Dispatcher Bot installed as an app on mobile phone which leverages Large Language Models (LLM) to understand driver queries and use Model Context Protocol(MCP) for secure enterprise tool calling. Using MCP server bot connects to HR system to check leave balances, leave eligibility, estimates the payroll deductions, process the leave and connects to Scheduling system to reassign affected pickups/drop-offs to the alternate qualified drivers. The proposed architecture transforms driver's phone into operational control interface reducing the dispatcher workload immensely, improving scheduling continuity, increasing policy transparency and lowering driver attrition through fair and explainable decision-making.*

Keywords: *Large Language Models (Llms), Python, Flutter, Model Context Protocol, Postgres, REST API, Conversational Bot, Lang Graph, Tool Calling.*

1. Introduction

Trucking companies operating large fleets need to ensure driver availability to take care of committed pickup and drop-offs. Fleet operators will have some kind of contingency plans for the planned absences by the drivers. But unplanned absences are challenging and get even more complicated when the drivers operate on various types of schedules as described below:

- 4-day work followed by 3 days off pattern and work can start on any day of the week.
- 2-days work followed by 1 day off pattern
- Similarly, 5 day work followed by 2 day off pattern and 6-day work followed by 1 day off pattern
- On top of the above patterns, some drivers work on night shifts

In such kinds of scenarios, dispatchers manually verify leave balance via HR portals, validate policy eligibility, and then reschedule loads using dispatch systems. This entire process is time-consuming, error-prone and not scalable, which introduces delays and increases the likelihood of missed delivery deadlines, and customer dissatisfaction. At the same time, unexpected payroll deductions for exhausted leaves frequently cause disputes. Lack of transparency damages trust and contributes to higher driver turnover.

To solve the above problems, we propose the self-service LLM-based mobile chatbot app that automates the leave management and dispatch continuity workflow. The bot uses MCP tool calling to interact with enterprise systems and execute actions such as leave posting, unpaid leave confirmation, deducting leave balance, payroll deduction initiation and load reassignment.

2. Problem Statement

Large trucking companies struggle with endemic pain points like severe driver shortages, high turnover, and retention issues[1]. Unplanned absence management is very important to address high turnover and retention issues. Below is the breakdown of the challenges faced in Unplanned absence management:

- Leave Balance Verification at Scale: There will be lot many factors considered before processing a driver's leave application which includes leave balance, type of the driver (Part-Time/Full-Time), jurisdiction of the driver (each jurisdiction will have its own leave policy), work schedule of the driver, weekend and night shift entitlements, carry-over, accrual, leave replenishment and black out and various other rules. Validating these rules manually or via HR portal by dispatcher or via self-servicing HR portal by a driver is a time-consuming process (which is rare as most drivers never login to a company HR portal).
- Payroll risk and low trust: Most of the time drivers contact logistics division dispatcher to let them know about their unplanned absence which can span multiple days and dispatcher prioritizes dispatch continuity, works on identifying the alternate qualified driver to take the assignment,

applies the time-off on behalf of the driver. HR system checks the leave balance for the driver, if enough balance is available then leave balance is decreased by those many days in the leave period. If the driver does not have enough leave balance then payroll deductions will be triggered. This results in drivers feeling treated unfairly and increases the risk of driver attrition.

- Dispatch Instability Due to Absences: When the driver becomes unavailable, the dispatcher scrambles to identify the impacted pickup/drop-offs during the leave period, locate alternate qualified drivers, ensure shift and Hours of service feasibility (sometimes the violations will be tagged to the driver for driving more than stipulated hours) and finally dispatcher need to avoid SLA violations. This adds significant operational stress and may result in missed deliveries.

3. Proposed System Overview

The above problems listed in the problem statement can be addressed using a Dispatcher Chatbot mobile app that executes the following workflow:

1. Driver launches the Dispatcher Chatbot on his phone and applies for a leave of absence
2. LLM parses the input provided by the driver and extracts the leave period start and end datetimes
3. Bot calls the HR system via MCP tools to verify the leave balance
4. Bot applies policy rules based on schedule pattern
5. If leave balance is sufficient → approve leave
6. If leave is insufficient → then driver will be notified about insufficient leave balance and driver will be given a chance to revisit his leave plans or give the consent for unpaid leave which results in payroll deductions.
7. After confirmation from the driver either leave balance will be decreased or unpaid leave request will be submitted, which results in the payroll deduction.
8. Bot calls scheduling system via MCP tools to retrieve impacted loads.
9. Bot selects replacement drivers and reassigns tasks automatically.
10. System sends confirmations to driver + dispatch + replacement drivers

This provides closed-loop operational automation, not just a chatbot response engine.

Contributions of this paper

The key novelty of this work lies in converting a driver-facing chatbot into a real operational engine:

1. Mobile-first conversational dispatch automation: drivers initiate leave actions directly from their phones.
2. LLM-driven MCP tool calling: dynamic orchestration of HR and scheduling tools rather than hardcoded workflows.
3. Closed-loop leave-to-dispatch workflow: leave decisions automatically trigger load reassignment.
4. Consent-based payroll deduction mechanism: unpaid leave is applied only after explicit driver confirmation.
5. Scalable for complex rosters: explicitly supports multi-pattern schedules including weekend and night shifts.
6. Auditability and enterprise compliance: all actions are logged through MCP tool traces.

Core Workflow: Paid Leave Verification & Approval

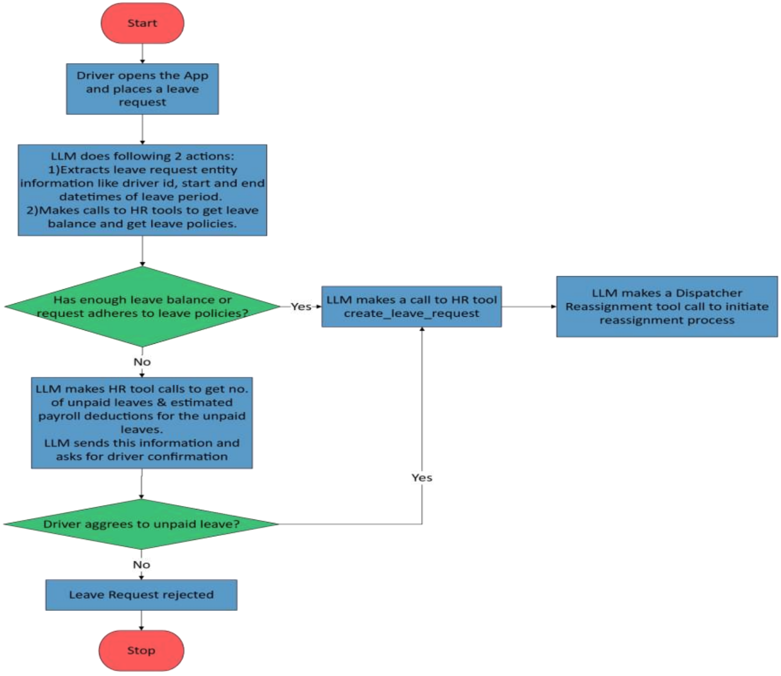
Start
Driver opens the App and places a leave request
LLM does following 2 actions:
1)Extracts leave request entity information like driver id, start and end datetimes of leave period.
2)Makes calls to HR tools to get leave balance and get leave policies.
Has enough leave balance or request adheres to leave policies?
Yes
LLM makes a call to HR tool create_leave_request
LLM makes a Dispatcher Reassignment tool call to initiate reassignment process
No
LLM makes HR tool calls to get no. of unpaid leaves & estimated payroll deductions for the unpaid leaves.
LLM sends this information and asks for driver confirmation
Yes
Driver aggrees to unpaid leave?
No
Leave Request rejected
Stop

Fig 1: Leave Approval Workflow

4. Architecture Design and Core Components

The proposed mobile Dispatcher Bot **will be** developed entirely using Flutter, a cross-platform framework that ensures the app functions smoothly across Android and iOS devices. Flutter was chosen for its flexibility, real-time rendering capabilities, and ability to create a highly responsive user interface[2]. The core intelligence layer is implemented using an LLM-driven agent framework in Python, where LangGraph is used to model the end-to-end workflow as a stateful, multi-step execution graph with policy checkpoints and conditional transitions (e.g., paid leave approval vs. unpaid leave confirmation)[3]. The bot leverages LangChain agent and tool abstractions to enable structured tool invocation, dynamic tool selection, and safe multi-step execution with error handling[4][5]. Enterprise connectivity is achieved through a dedicated MCP (Model Context Protocol) Server, which exposes authenticated HR and scheduling capabilities as standardized tools[6][7].

- Mobile Chatbot Application (Driver Interface): The application will be deployed on the driver's phone. Driver will be launching the Chatbot app and will be conversing and making requests in chat window in the natural language such as "I need leave from today till next Monday"
- LLM Orchestration Layer: In this layer LLM initially tries to understand the intent of the driver (leave request or schedule change), next it will extract the entity information like driver identification, start and end datetimes of the leave period etc. then decides to call which tool (HR system related tool or Scheduling system tool)
- MCP Server (Enterprise Tool Gateway): The MCP server exposes authenticated enterprise capabilities as tools. Provides consistent interface for multiple enterprise vendors, provides centralized tool permissioning, safer abstraction compared to raw API exposure and audit friendly tool traces.

HR Tools

- get_leave_balance(driver_id)
- get_leave_policy(driver_id, date_range, shift_type)
- create_leave_request(driver_id, leave_type, date_range)
- estimate_pay_deduction(driver_id, hours, pay_period)
- apply_unpaid_leave(driver_id, date_range)
- register_payroll_deduction(driver_id, amount, pay_period)

Dispatch/Scheduling Tools

- get_driver_schedule(driver_id, date_range)
- get_assigned_pickups_dropoffs(driver_id, date_range)
- find_available_drivers(criteria)
- reassign_load(load_id, new_driver_id)
- notify_dispatch_team(event_payload)

MCP enables clean separation between the agent and external systems, improving modularity and security.

External Systems: The external systems host all the authoritative data.HR system maintains the leave rules and accruals, employee information (driver identification, hire date, work anniversary date, region, shift type, employment type etc.), payroll integration, unpaid leave deductions. Scheduling / Dispatch System maintains routes, loads, pickups/drop-offs, driver availability and assignments

Audit and Logging Layer: Stores tool calls + responses, driver confirmation history, leave approvals and payroll actions, load reassignment transactions.

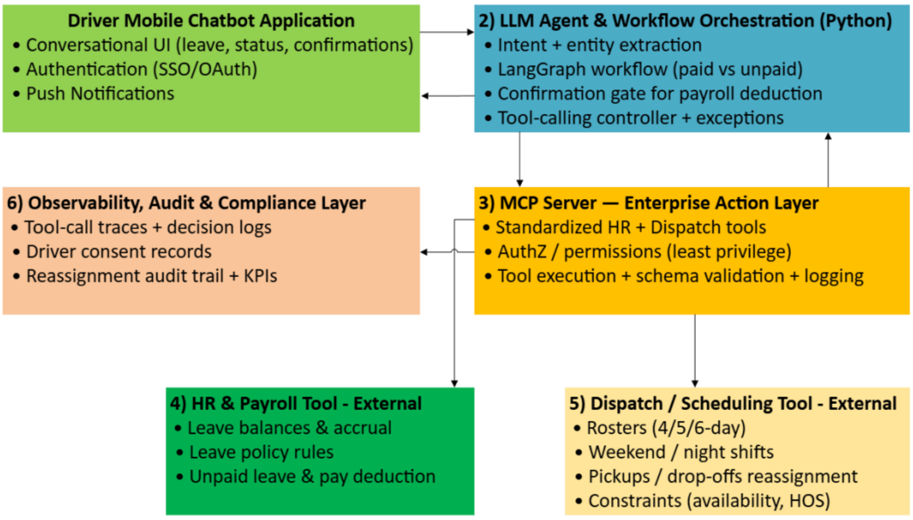
Driver Mobile Chatbot Application
• Conversational UI (leave, status, confirmations)
• Authentication (SSO/OAuth)
• Push Notifications
2) LLM Agent & Workflow Orchestration (Python)
• Intent + entity extraction
• LangGraph workflow (paid vs unpaid)
• Confirmation gate for payroll deduction
• Tool-calling controller + exceptions
6) Observability, Audit & Compliance Layer
• Tool-call traces + decision logs
• Driver consent records
• Reassignment audit trail + KPIs
3) MCP Server — Enterprise Action Layer
• Standardized HR + Dispatch tools
• AuthZ / permissions (least privilege)
• Tool execution + schema validation + logging
4) HR & Payroll Tool - External
• Leave balances & accrual
• Leave policy rules
• Unpaid leave & pay deduction
5) Dispatch / Scheduling Tool - External
• Rosters (4/5/6-day)
• Weekend / night shifts
• Pickups / drop-offs reassignment
• Constraints (availability, HOS)

Fig 2: Core Components of the Chatbot

5. Conclusion

This paper introduced an MCP-enabled mobile LLM Dispatcher Bot designed for large trucking fleets with complex schedules. By combining natural language interaction, secure enterprise tool calling via MCP, consent-driven payroll deduction workflows, and automated dispatch reassignment, the proposed system improves operational continuity while enhancing driver trust. The architecture reduces dispatcher workload, prevents scheduling disruption, decreases payroll disputes, and offers a scalable approach to reduce driver attrition in high-frequency logistics environments.

References

1. Procter, D., & Sousa Jr, P. (2021). Goldilocks and the Three Dispatchers: Quantifying the Impact of Dispatcher Management on Truck Driver Performance.
2. Agarwal, K., Ananthanarayanan, S., & Srinivasan, S. (2024). Enhancing iot based plant health monitoring through advanced human plant interaction using large language models and mobile applications. *arXiv preprint arXiv:2409.15910*.
3. Sapkota, R., Shrestha, R., Rijal, M., & Karkee, M. LangChain vs. LangGraph vs. LangSmith: Taxonomies of Agentic AI Toolchains for End-to-End Orchestration. *Authorea Preprints*.
4. Jeong, C. (2025). A Study on the MCP x A2A Framework for Enhancing Interoperability of LLM-based Autonomous Agents. *arXiv preprint arXiv:2506.01804*.
5. Agents - Docs by LangChain
6. Ding, P. (2025). Toolregistry: A protocol-agnostic tool management library for function-calling llms. *arXiv preprint arXiv:2507.10593*.
7. Model Context Protocol (MCP) - Docs by LangChain.