



IJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: <https://doi.org/10.22214/ijraset.2026.79532>

www.ijraset.com

Call:  08813907089

E-mail ID: ijraset@gmail.com

AI-Assisted Intelligent Task Manager with ML-Based Priority Classification and Conversational AI

Manikandan, Vishal SZ, AL Raafhath RK, Roselin Lourd

Department of Artificial Intelligence and Data Science, Sri Manakula Vinayagar Engineering College

Abstract: *In today's fast-paced professional and academic environments, effective task management is a critical challenge. This paper presents the design and implementation of an AI-Assisted Intelligent Task Manager, a web-based application that leverages machine learning to automatically classify and prioritize user-submitted tasks. The system employs a multi-feature ML model trained on task metadata including deadlines, dependencies, estimated effort, and contextual keywords to generate a dynamic priority score for each task. A natural language chatbot interface, powered by a large language model (LLM), enables users to interact conversationally with the system querying task status, asking for priority explanations, and receiving intelligent scheduling suggestions. The system additionally supports smart reminders, workload forecasting, and integration with calendar APIs. Evaluation results demonstrate high classification accuracy and strong user satisfaction, making the system a robust solution for intelligent personal and team productivity management.*

Keywords: *Task Management, Machine Learning, Priority Classification, Natural Language Processing, Conversational AI, Intelligent Scheduling, LLM, Chatbot*

I. INTRODUCTION

The proliferation of digital workspaces and the increasing complexity of modern workflows have made manual task prioritization both inefficient and error-prone. Individuals and teams routinely face scenarios where hundreds of tasks compete for attention simultaneously, making it nearly impossible to determine optimal execution order without external assistance.

Traditional task management tools such as Trello, Asana, and Todoist allow users to manually assign priority levels, but they do not offer intelligent, data-driven prioritization. Users are left to apply their own judgment, which can be influenced by cognitive biases, emotional states, and incomplete information.

This paper introduces a novel AI-Assisted Intelligent Task Manager that automates priority scoring through a machine learning pipeline. The system accepts natural-language task descriptions from users and uses a trained classification model to assign a quantitative priority score, subsequently ranking tasks in an actionable queue. Furthermore, an integrated conversational AI assistant enables users to interact with the system through chat, ask questions about task prioritization logic, request rescheduling, and receive proactive recommendations.

The primary contributions of this work are: (1) a multi-feature ML model for task priority classification, (2) a chatbot interface powered by an LLM for explainable AI interactions, (3) a full-stack web application integrating these capabilities, and (4) a smart notification and calendar synchronization layer for comprehensive task lifecycle management.

II. LITERATURE REVIEW

A. Existing Task Management Systems

Conventional task management platforms rely on manual priority assignment. While tools like Jira offer workflow automation rules, they lack the ability to infer priority from task content semantics. Studies by Mäntylä et al. [1] have shown that human-assigned priorities in issue trackers are frequently inconsistent, motivating the need for automated classification approaches.

B. ML-Based Priority Classification

Several works have explored machine learning for issue and task prioritization. Researchers have applied Support Vector Machines (SVMs), Random Forests, and gradient boosting algorithms to classify software bug severity with reasonable accuracy [2].

More recently, transformer-based models such as BERT have been employed for semantic understanding of task descriptions, significantly improving classification performance [3].

C. Conversational Interfaces for Productivity

The integration of conversational AI into productivity tools has been explored in enterprise settings. Virtual assistants such as Microsoft Cortana and Google Assistant have demonstrated the feasibility of natural-language task scheduling. LLM-based agents, as discussed in [4], offer more flexible and context-aware interactions, making them well-suited for explainable productivity systems.

III. SYSTEM ARCHITECTURE

A. Overview

The AI-Assisted Task Manager follows a three-tier architecture comprising a frontend client layer, a RESTful backend API layer, and a machine learning inference layer. All components communicate over HTTPS, ensuring secured data transmission. The system architecture is illustrated conceptually below.

TABLE I SYSTEM ARCHITECTURE COMPONENTS

Layer	Technology Stack	Responsibility
Frontend	React.js, Tailwind CSS	User interface, task input, chat widget
Backend API	Python FastAPI	Business logic, auth, REST endpoints
ML Engine	scikit-learn, Transformers	Priority scoring, text classification
LLM Chatbot	OpenAI GPT/LLaMA	Conversational AI, explainability
Database	PostgreSQL+Redis	Persistent storage, session caching
Notifications	Firebase, SMTP	Push alerts, email reminders

B. Machine Learning Pipeline

The ML pipeline ingests structured task features and produces a real-valued priority score between 0 and 1. The pipeline consists of three sequential stages: feature extraction, model inference, and score normalization.

Feature extraction parses raw task input to derive the following feature vector:

- Deadline proximity (days remaining, normalized)
- Estimated effort (hours, log-scaled)
- Dependency count (number of blocking tasks)
- Keyword urgency score (TF-IDF weighted term importance)
- Task category embedding (BERT sentence embedding, 384-dim)
- Historical completion rate of similar tasks

These features are concatenated into a feature vector and passed to a trained Gradient Boosted Trees (XGBoost) model for regression-based priority scoring. The model was trained on a dataset of 50,000 annotated tasks from open-source project trackers.

C. Chatbot Interface

The conversational chatbot is implemented using a retrieval-augmented generation (RAG) approach. The LLM receives the user's query along with relevant task context (current task list, priority scores, model explanations generated via SHAP values) and produces a coherent, contextually accurate response. This enables the chatbot to explain priority decisions in plain language, answer scheduling questions, and suggest task reorganization.

D. Data Flow

A user submits a task through the web interface by entering a title, description, deadline, and optional metadata tags. The backend API validates the input, extracts features, and forwards the feature vector to the ML inference service.

The model returns a priority score, which is stored in the database and displayed to the user. The chatbot service maintains a conversation context window and can access the user's full task list via the backend API to answer follow-up questions.

IV. KEY FEATURES AND FUNCTIONALITY

A. Intelligent Priority Scoring

The core feature of the system is its ability to automatically compute a priority score for each task. Rather than using a simple rule-based approach, the ML model captures non-linear interactions between features. For example, a task with a moderate deadline but high dependency count and a large estimated effort may rank higher than a task with a tight deadline but no dependencies and minimal effort.

B. Explainable AI via Chatbot

Users can ask the chatbot questions such as "Why is Task A ranked higher than Task B?" or "What can I do to deprioritize this task?" The system uses SHAP (SHapley Additive exPlanations) values from the ML model to generate feature-level explanations, which the LLM then translates into natural-language responses. This transparency builds user trust and encourages informed engagement with the system.

C. Smart Reminders and Notifications

The system tracks task deadlines and completion status in real-time. When a high-priority task is approaching its deadline or when a dependency is resolved, the system sends push notifications via Firebase Cloud Messaging and optional email alerts. Users can configure notification preferences, including quiet hours and threshold-based triggers.

D. Workload Forecasting

Using historical task completion data, the system generates a weekly workload forecast that estimates the total effort required for pending tasks. This allows users to proactively identify overloaded days and redistribute tasks accordingly. The forecast is visualized as an interactive chart on the dashboard.

E. Calendar Integration

The system offers bidirectional synchronization with Google Calendar and Microsoft Outlook via their respective REST APIs. High-priority tasks are automatically scheduled as calendar events, and changes made in the calendar are reflected in the task manager. This eliminates the need for manual data entry across multiple tools.

F. Collaborative Task Management

For team use cases, the system supports workspace-level access control. Team leaders can assign tasks to members, set shared dependencies, and monitor team workload distributions. The chatbot operates at the team level, capable of answering questions about collective task states and bottlenecks.

V. IMPLEMENTATION DETAILS

A. Model Training

The XGBoost priority regression model was trained using a dataset constructed from GitHub Issues, Jira exports, and a proprietary annotation set. The dataset was split into 80% training, 10% validation, and 10% test sets. Hyperparameter tuning was performed using Bayesian optimization over 100 iterations.

TABLE II MODEL PERFORMANCE METRICS

Metric	Value
Mean Absolute Error (MAE)	0.041
Root Mean Squared Error (RMSE)	0.063
RZ Score	0.921
Classification Accuracy (3-class)	94.7%
Inference Latency (p95)	<12 ms

B. Backend API

The backend is implemented in Python using the FastAPI framework, selected for its asynchronous request handling and automatic OpenAPI documentation generation. Task CRUD operations, user authentication (JWT-based), and ML inference requests are exposed as RESTful endpoints. The ML model is served via a dedicated microservice using ONNX Runtime for optimized inference.

C. Frontend Application

The React.js frontend presents users with an interactive task dashboard displaying tasks sorted by priority score. Color-coded priority bands (critical, high, medium, low) provide quick visual reference. The integrated chat widget is built using WebSocket connections to ensure real-time message delivery. The interface is fully responsive, supporting desktop and mobile browsers.

D. Security and Privacy

All user data is encrypted at rest using AES-256 and in transit via TLS 1.3. OAuth 2.0 is used for third-party calendar integrations. The system complies with GDPR guidelines by providing data export and deletion capabilities. No task content is used for model retraining without explicit user consent.

VI. RESULTS AND EVALUATION

A. Quantitative Results

The system was evaluated on a held-out test set of 5,000 tasks. The ML model achieved a classification accuracy of 94.7% across three priority tiers (high, medium, low), significantly outperforming baseline methods including rule-based scoring (78.3%) and simple TF-IDF + logistic regression (85.1%). The full comparison is presented in Table III.

TABLE III COMPARISON WITH BASELINE METHODS

Method	Accuracy (%)	MAE	Inference (ms)
Rule-Based Scoring	78.3	0.142	<1
TF-IDF+Logistic Reg.	85.1	0.098	3
Random Forest	90.4	0.071	8
BERT Fine-tuned	93.2	0.054	45
Proposed (XGBoost+NLP)	94.7	0.041	12

B. User Study

A user study was conducted with 40 participants across student, freelancer, and corporate professional demographics. Participants used the system for two weeks and completed a standardized usability questionnaire (SUS) and a custom productivity assessment. Results are summarized in Table IV.

TABLE IV USER STUDY SUMMARY

Metric	Score/Response
System Usability Scale (SUS)	82.5 / 100
Task Completion Rate	96%
Self-Reported Productivity Gain	31% average improvement
Chatbot Explanation Satisfaction	4.4 / 5.0
Willingness to Recommend	87.5%

VII. CONCLUSIONS

This paper has presented a comprehensive AI-assisted Intelligent Task Manager that combines machine learning-based priority classification with a conversational AI interface to create an intelligent, explainable, and user-friendly productivity tool.

The proposed system addresses the critical limitations of existing task management solutions by automating priority scoring, providing natural-language explanations, and offering proactive workload management features.

The experimental evaluation demonstrates that the system achieves state-of-the-art priority classification accuracy of 94.7% while maintaining low inference latency suitable for real-time use. The user study confirms practical utility, with participants reporting an average productivity improvement of 31% and high satisfaction with the chatbot's explanatory capabilities.

Future work will explore federated learning to enable model personalization without compromising user privacy, multi-modal task input supporting voice and image attachments, and advanced multi-agent collaboration features for enterprise deployments.

VIII. ACKNOWLEDGMENT

The authors would like to thank the Department of Artificial Intelligence and Data Science at Sri Manakula Vinayagar Engineering College for providing computational resources, and the study participants for their valuable time and feedback during the user evaluation phase.

REFERENCES

- [1] M. V. Mäntylä, B. Adams, F. Khomh, E. Engström, and K. Petersen, "On rapid releases and software testing: a case study and a semi-systematic literature review," *Empirical Software Engineering*, vol. 20, no. 5, pp. 1384–1425, 2015.
- [2] K. Tian, M. Reville, and D. Poshyanyk, "Using latent dirichlet allocation for automatic categorization of software," in *Proc. 6th IEEE Int. Working Conf. on Mining Software Repositories*, 2009, pp. 163–166.
- [3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pretraining of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT, 2019*, pp. 4171–4186.
- [4] T. B. Brown et al., "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.
- [5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2016*, pp. 785–794.
- [6] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [7] P. Lewis et al., "Retrieval augmented generation for knowledge intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [8] Google, *Firestore Cloud Messaging Documentation*. [Online]. Available: <https://firebase.google.com/docs/cloud-messaging>
- [9] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.
- [10] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)