AI-Based Email Deadline and Task Reminder System for Students

An Intelligent Approach to Academic Email Monitoring, Deadline Extraction, And Task Prioritization

Mr. Sagar Dnyanadeo Shingote Ms. Chikhale Shivani Subodh
Department of Master of Computer Applications
JCEI's Jaihind Institute of Management and Research, Vadgaon Sahani Gmail -sagarshingote20@gmail.com
-chikhaleshivani25@gmail.com

ABSTRACT
Students receive important academic information through email, including assignment instructions, submission deadlines, examination notices, project milestones, and meeting schedules. These messages can become difficult to manage when students receive many emails from different teachers and departments. Important dates may be overlooked, causing missed submissions and avoidable academic stress. This paper proposes an AI-Based Email Deadline and Task Reminder System for Students that organizes academic email information and converts actionable messages into a structured task list. The proposed system uses email access with user permission, natural-language processing to identify task-related messages, date and deadline extraction, priority classification, and a reminder scheduler. Students can review extracted tasks, correct inaccurate dates, set reminder preferences, and track task status through a dashboard. A rule-based baseline can be implemented first, with machine-learning or language-model components added for classification and extraction. The paper presents the system architecture, methodology, evaluation plan, expected benefits, and limitations. Since a deployed system and measured user study are not provided here, performance values are not claimed; the evaluation metrics are proposed for future testing. The system aims to reduce manual email checking and support timely academic task completion while protecting student privacy.

Keywords: Artificial Intelligence, Email Classification, Deadline Extraction, Task Management, Natural Language Processing, Student Productivity, Reminder System.

1. INTRODUCTION

Email is a common channel through which students receive assignment details, examination announcements, project updates, timetable changes, and instructions from faculty. Unlike a dedicated task manager, an email inbox primarily stores messages rather than presenting all academic responsibilities in one organized timeline. Students must read messages, identify actions, remember dates, and create reminders manually. When this process is inconsistent, a deadline can be missed even if the original email is available.

Artificial intelligence (AI) and natural-language processing (NLP) can help transform unstructured email text into useful task information. A system that identifies action-oriented messages, extracts dates and task descriptions, and schedules reminders could provide students with a single view of pending work. The proposed approach keeps students in control: extracted details are shown for confirmation before a task is treated as final, and reminders can be edited or dismissed.

The objective of this research is to design a practical and privacy-aware workflow for academic email monitoring. The proposed design can be developed as a web application or desktop application and connected to an email provider through its authorized API. AI-generated outputs are treated as suggestions, not as guaranteed facts.

1.1 Problem Statement

Students often need to manually search emails for academic tasks and deadlines. Important information may be buried in long message threads, written in different formats, or expressed with relative dates such as "next Friday." Basic calendar reminders require students to enter the information themselves. Therefore, a system is needed to detect relevant academic emails, extract task and deadline details, organize tasks by urgency, and notify students at suitable times with user review and privacy safe guards.

1.2 Objectives

- To design an AI-assisted system for identifying academic emails that contain actionable tasks.
- To extract task descriptions, dates, times, courses, and sender details were available.
- To allow students to verify, edit, and confirm extracted task information.
- To prioritize tasks using deadline proximity and user-defined importance.
- To send configurable reminders before deadlines and show overdue tasks.
- To provide a dashboard for pending, completed, and overdue academic tasks.
- To consider privacy, access permissions, data minimization, and secure storage.

1.3 Contributions

- A modular architecture linking email ingestion, NLP analysis, task management, and notifications.
- A human-in-the-loop process that asks students to confirm uncertain dates and extracted tasks.
- A priority model based on urgency, task importance, and completion status.
- An evaluation framework using extraction precision/recall, date accuracy, reminder delivery, and usability measures.

2. LITERATURE REVIEW

Email filtering and text classification are established applications of machine learning and NLP. Classification methods can distinguish message categories by learning from text features, while rule-based approaches can identify common phrases such as "submit by," "assignment due," or "exam scheduled." Rules are easy to explain and useful for an initial prototype, but they may fail when wording is indirect or context-dependent. (Manning et al., 2008; Oard & Webber, 2008).

Information extraction methods identify entities such as dates, times, course names, and action phrases within unstructured text. Date extraction is challenging because messages may contain multiple dates, ambiguous formats, or relative expressions. A system must distinguish the actual submission deadline from dates mentioned as examples or in earlier replies. Context-aware models may improve interpretation, but their outputs still require validation. (Jurafsky & Martin, n.d.).

Task-management and calendar applications support manual entry, notifications, and status tracking. However, they commonly rely on the user to convert each email into a task. The proposed system combines email classification and deadline extraction with task management. Its distinguishing design principle is confirmation by the student before a low-confidence extraction is committed. (Google, n.d.; Microsoft, n.d.).

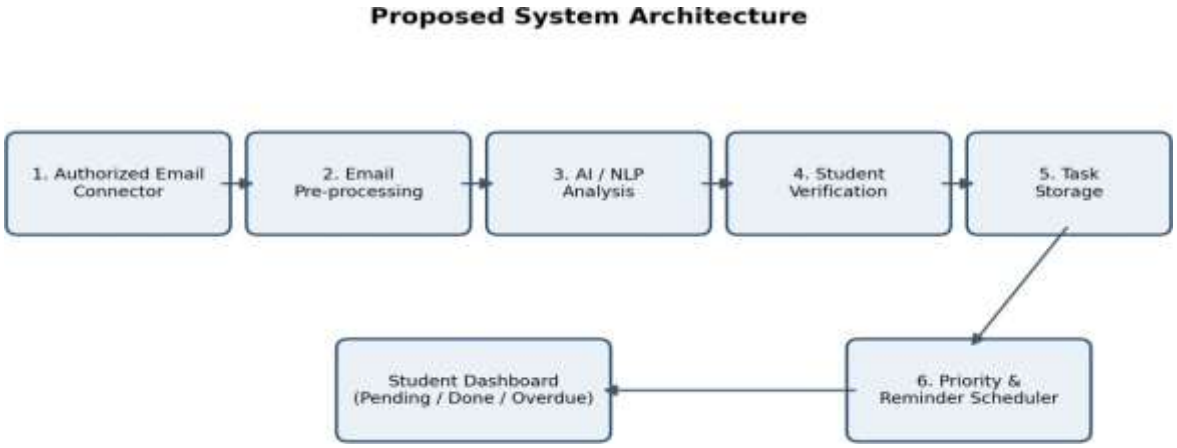
Approach	Strengths	Limitations
Manual email checking	No extra setup; user reads original context	Time-consuming; relies on memory
Calendar / task manager	Clear schedule and reminders	Tasks usually need manual entry
Keyword/rule-based extraction	Simple, explainable, low computational cost	Limited handling of varied language and ambiguity
AI-assisted proposed system	Automates candidate detection and task creation	May misread dates; needs permission, validation, and testing

3. METHODOLOGY

3.1 System Architecture

The proposed system is organized into six modules: (1) Authorized Email Connector, (2) Email Pre-processing, (3) AI/NLP Analysis, (4) Task Verification and Storage, (5) Priority and Reminder Scheduler, and (6) Student Dashboard. The email connector retrieves only the message fields authorized by the student. Pre-processing removes unnecessary formatting and identifies message content. The AI/NLP module classifies academic relevance and extracts possible task details. The verification interface presents the source email and confidence indicators.

Confirmed tasks are stored and passed to the scheduler, which creates notifications according to the student's preferences.



Module	Main responsibility
Email connector	Authenticate securely and retrieve permitted messages
Pre-processing	Clean text, handle quoted replies, identify language and metadata
AI/NLP analysis	Classify academic task emails and extract action, date, time, course
Verification and storage	Allow edits; save confirmed tasks and source-message reference
Priority and scheduler	Rank tasks and schedule reminders based on user settings
Dashboard	Show upcoming, completed, and overdue tasks

Figure 2. Email-to-reminder processing workflow



3.2 AI and Analysis Techniques

The initial prototype can use a hybrid approach. Keyword rules identify likely academic messages, while a text classifier assigns categories such as assignment, exam, project, meeting, or general notice. Date extraction identifies explicit dates and times; contextual checks compare the extracted date with surrounding phrases such as "deadline," "submit," or "exam." Relative dates should be interpreted using the message timestamp and the student's configured time zone. If multiple candidate dates are found or confidence is low, the system should ask the student to choose rather than silently schedule a reminder.

A simple priority score may combine urgency and importance: Priority = 0.60 × Urgency Score + 0.40 × Importance Score. Urgency can be mapped from remaining time (for example, higher scores for deadlines within 24 hours), while importance is set by the student or inferred as a suggestion from message context. These weights are design choices and should be tuned through evaluation, not treated as scientifically validated values.

3.3 Data and Task Fields

A prototype dataset can contain anonymized or synthetic email examples with fields such as message ID, subject, sanitized body, received date, category label, task description, expected deadline, extracted deadline, confidence, and verification status. Real email content is sensitive; testing should prefer synthetic or consented, anonymized examples. No actual student email dataset or measured sample is supplied with this paper.

Field	Purpose
Email ID / received date	Reference and temporal context
Subject and sanitized text	Input for classification and extraction
Task category	Assignment, exam, project, meeting, or other
Extracted task and deadline	Candidate action and due date
Confidence / verification	Indicates uncertainty and whether user approved it
Reminder settings / status	Reminder times, completed state, overdue state

3.4 Evaluation Parameters

The system should be evaluated using a labeled test set of academic emails. Classification precision, recall, and F1-score can measure identification of task-related emails. Deadline exact-match accuracy and date-extraction precision/recall can measure temporal extraction. Reminder delivery success, duplicate-reminder rate, task correction rate, and usability feedback can assess practical performance. Evaluation should compare the AI-assisted workflow with manual task entry and a keyword-only baseline. No numerical results are reported because testing has not been conducted for this draft.

3.5 Experimental Setup

A feasible implementation may use Python for the application logic, an NLP library or model for classification and entity extraction, a relational database such as SQLite or PostgreSQL for confirmed tasks, and a web interface for task review. Email access should use the provider's official authorization flow and minimum required scopes; passwords should not be stored directly. A scheduler can run periodic checks and send in-app, email, or calendar notifications according to user preference. The exact stack may be adjusted to the institution's requirements and available resources.

4. EXPECTED RESULTS AND EVALUATION

The expected outcome is a working prototype that identifies candidate academic task emails, proposes task details and deadlines, and displays confirmed tasks in a dashboard. Students should be able to see deadlines in chronological order, filter tasks by course or status, and receive reminders at configurable intervals. The system is expected to reduce repeated inbox searching and manual transcription, but the magnitude of any improvement must be established through user testing.

Evaluation measure	How it will be measured
Task-email classification	Precision, recall, and F1-score on labeled emails
Deadline extraction	Exact date/time match against verified labels
Correction rate	Percentage of AI-suggested tasks edited by students
Reminder reliability	Successful deliveries and duplicate/late notifications
Task completion support	Comparison of missed deadlines before and during a pilot, if ethical and feasible
Usability	Short student questionnaire and task-completion observations

5. DISCUSSION

The proposed design addresses a practical student problem by connecting information already present in academic emails to a structured task workflow. Its main benefit is reducing the number of manual steps between receiving an instruction and creating a reminder. A verification step is important because an incorrect deadline can be more harmful than no reminder. Showing the original email, extracted date, and confidence level allows the student to check the result before relying on it.

The system also has important limitations. Email messages may contain several dates, forwarded content, informal language, or changed deadlines. AI models can produce incorrect or inconsistent outputs, and notification services may be delayed or unavailable. A system should not automatically send messages to faculty or alter calendar events without explicit approval. Privacy risks can be reduced by requesting limited access, encrypting stored data, offering deletion controls, and retaining only the fields necessary for task management. Students should be able to disable email scanning and use manual task entry.

Since no implementation results are available, the proposed benefits remain hypotheses. A pilot study with informed consent, anonymized data, and a predefined evaluation protocol is required before claiming improved deadline adherence or time savings.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This paper proposes an AI-Based Email Deadline and Task Reminder System for Students to convert academic email information into a verified task list with deadline-aware reminders. The design combines authorized email access, text pre-processing, AI/NLP classification and extraction, student confirmation, priority scoring, scheduling, and a dashboard. The proposed approach may help students organize academic responsibilities and reduce reliance on memory. Its reliability, usability, and impact on missed deadlines must be confirmed through implementation and empirical evaluation.

6.2 Future Work

Future work can include multilingual email understanding, improved date disambiguation, calendar integration, mobile notifications, personalized reminder timing, and learning from user corrections. Further research can compare rule-based, classical machine-learning, and language-model methods on a consented, anonymized dataset. A controlled pilot could measure extraction accuracy, task completion, perceived workload, and user trust. Security testing and a transparent data-retention policy should be completed before deployment.

REFERENCES

1) Google. (N.D.). Gmail API. Google For Developers. Https://Developers.Google.Com/Gmail/Api
2) Jurafsky, D., & Martin, J. H. (N.D.). Speech and Language Processing (3rd Ed. Draft). Https://Web.Stanford.Edu/~Jurafsky/Slp3/

3) Manning, C. D., Raghavan, P., & Schutze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

4) Microsoft. (N.D.). Mail API Overview. Microsoft Graph Documentation. Https://Learn.Microsoft.Com/Graph/Api/Resources/Mail-Api-Overview

5) Oard, D. W., & Webber, W. (2008). Information Retrieval for E-Mail. Foundations and Trends in Information Retrieval, 1(2), 141–168.

6) Pedregosa, F., Varoquaux, G., Gram fort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-Learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

7) Python Software Foundation. (N.D.). Python Documentation. Https://Docs.Python.Org/3/