Multitenant Leave System with Adaptive Leave Optimization and NLP Integration

Akshat Dhanuka[1], Aayush Bhaskarwar[2], Ziyaad Parkar[3], S.P. Shintre[4]
UG Scholar, Dept. of Computer Engineering, Pune Institute of Computer Technology, Pune, India[1-3]
Asst. Professor, Dept. of Computer Engineering, Pune Institute of Computer Technology, Pune, India[4]

Abstract—Efficient leave management is crucial for maintaining productivity and meeting project deadlines, especially in organizations with interdependent teams. This paper presents the implementation of a Multitenant Leave Management System that integrates AI-driven modules for dynamic leave processing. Developed using React.js, Django, and PostgreSQL, the system automates leave request handling while minimizing workflow disruptions.
A GPT-powered chatbot is integrated using Natural Language Processing (NLP) to interpret employee queries such as "Who is on leave today?" or "When can I apply for sick leave?" and convert them into real-time SQL queries for accurate data retrieval. To enhance efficiency, the system also features two custom algorithms—ADAPTIVE LEAVE OPTIMIZATION and PROACTIVE LEAVE RECOMMENDATION ALGORITHM which analyze historical leave usage and workload patterns to adjust leave policies and recommend conflict-free leave periods.
This paper outlines the system architecture, module-wise implementation, NLP pipeline, and the analytical backend. The results demonstrate how the system supports intelligent decision-making and improves HR responsiveness, offering a scalable solution for modern organizations.

Keywords—leave management system, chatbot integration, NLP, GPT API, SQL query generation, Django, React.js, PostgreSQL, ADAPTIVE LEAVE OPTIMIZATION, PROACTIVE LEAVE RECOMMENDATION ALGORITHM, HR automation, team collaboration, data-driven recommendations, workforce planning.

I. INTRODUCTION

In today's dynamic corporate environment, efficient leave management is essential for sustaining productivity, especially within organizations that operate in multi-team, collaborative structures. Traditional leave management systems often fall short in providing real-time insights, adaptability, and automation, which leads to scheduling conflicts, manual overhead, and reduced operational efficiency. Moreover, these systems lack intelligent features that account for workload balancing, employee well-being, and cross-team dependencies.

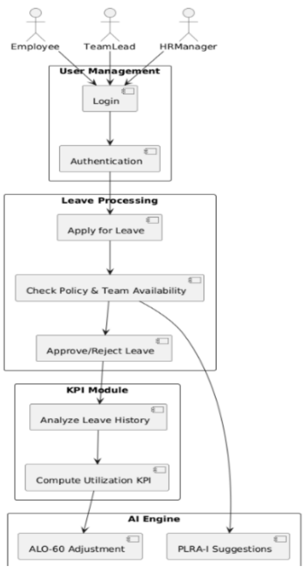
To address these challenges, we implemented a Multitenant Leave Management System that automates the leave lifecycle—from application to approval—while offering intelligent insights through data analysis and AI-driven recommendations. The system is designed to streamline leave tracking, reduce administrative effort, and enable proactive decision-making using advanced analytics and conversational interfaces.

At the core of the system lies a chatbot integrated with GPT-based NLP, enabling employees to interact with the portal using natural language queries. This chatbot dynamically converts user inputs into SQL queries, allowing real-time access to leave data and policy information. Additionally, the system incorporates two custom algorithms:

1. Adaptive Leave Optimization: dynamically adjusts leave policies based on historical usage patterns.
2. Proactive Leave Recommendation Insight: suggests optimal leave periods to avoid bottlenecks and align with team workload patterns.

Developed using React.js, Django, and PostgreSQL, and powered by Python's data analysis libraries like Pandas and Matplotlib, the system offers a scalable, modular solution for modern workforce management. This paper details the architecture, implementation process, modules, and algorithms that collectively enable intelligent, automated leave management across interdependent teams.

1. Project Flow Diagram



II. OBJECTIVES

This project aims to build an intelligent, scalable, and modular leave management system that automates traditional processes and integrates modern technologies like AI and NLP. The following subsections outline the core goals of the system implementation.

A. *Functional Objectives:*

These are the system's primary features and behaviors that enhance day-to-day operations:

- Automated Leave Lifecycle:
  Enable employees to apply for leave, receive approvals, and track status—all within a streamlined digital workflow.
- Multi-Tenant Support:
  Allow organizations to define separate leave rules and policies per department or team, supporting independent yet unified management.
- Admin Dashboard for Oversight:
  Provide HR and team leaders with visual dashboards displaying leave records, approvals, trends, and productivity insights.

B. *Intelligent Interaction via NLP:*

To improve user accessibility and streamline communication:

- GPT-Powered Chatbot:
  Integrate a chatbot interface that accepts natural language queries like "When is my next leave available?" and translates them into executable SQL commands.
- NLP Pipeline Implementation:
  Use tokenization, intent classification (SVM), and entity recognition (NER, Cosine Similarity) to understand and process queries effectively.

C. *Data Driven Optimization:*

To help organizations plan more efficiently and make informed decisions:

- Adaptive Leave Optimization:
  Monitor actual leave usage and adjust future leave quotas accordingly, reducing underutilization and over-allocation.
- Proactive Leave Recommendation Insight:
  Recommend conflict-free leave periods for employees, considering project timelines, team availability, and historical trends.
- AI-Powered Scenario Forecasting: Simulate the effect of future leave patterns using LLMs (Gemini API), helping
- HR evaluate how short-term breaks could influence productivity.

D. *Data Driven Optimization:*

To help organizations plan more efficiently and make informed decisions:

- Adaptive Leave Optimization:
  Monitor actual leave usage and adjust future leave quotas accordingly, reducing underutilization and over-allocation.
- Proactive Leave Recommendation Insight:
  Recommend conflict-free leave periods for employees, considering project timelines, team availability, and historical trends.

E. *Analytical and Predictive Insights:*

To correlate employee behavior with organizational performance:

- Productivity-Based Leave Forecasting:
  Implement models (e.g., Linear Regression, Random Forest) to predict the impact of leaves on performance metrics like task completion or review scores.
- Trend Analysis with Time Series:
  Use moving averages to identify seasonal leave patterns and assist in long-term workforce planning.

F. *Technical and Architectural Objectives:*

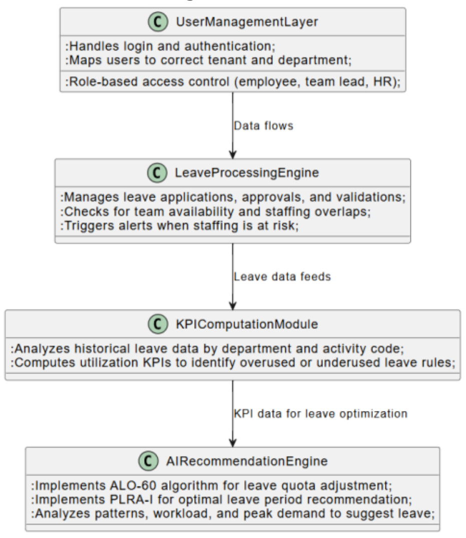
To ensure a sustainable and maintainable system:

- Modular Architecture:
  Structure the system into separate components—frontend, backend, chatbot, database—for ease of development and maintenance.
- Technology Stack:
  Build with React.js (UI), Django (backend), PostgreSQL (data), GPT API (NLP), and Pandas/Matplotlib (data analysis).
- Security and Scalability:
  Incorporate secure login/authentication mechanisms and design the backend to handle increasing loads as the organization scales.

III. SYSTEM ARCHITECTURE

The Multitenant Leave Management System is designed with a modular and scalable architecture that supports seamless integration between frontend interfaces, backend processing, data analytics, and NLP-driven user interactions. The system follows a layered architecture, ensuring separation of concerns, ease of maintenance, and extensibility.

A. Architecture Diagram:


UserManagementLayer
:Handles login and authentication;
:Maps users to correct tenant and department;
:Role-based access control (employee, team lead, HR);
Data flows
LeaveProcessingEngine
:Manages leave applications, approvals, and validations;
:Checks for team availability and staffing overlaps;
:Triggers alerts when staffing is at risk;
Leave data feeds
KPIComputationModule
:Analyzes historical leave data by department and activity code;
:Computes utilization KPIs to identify overused or underused leave rules;
KPI data for leave optimization
AIRecommendationEngine
:Implements ALO-60 algorithm for leave quota adjustment;
:Implements PLRA-I for optimal leave period recommendation;
:Analyzes patterns, workload, and peak demand to suggest leave;

2. System Architecture Diagram

B. Core Components:

B.1 Frontend (ReactJS)

The user interface is developed using React.js, enabling responsive and dynamic interactions. Employees and administrators can apply for leave, view calendars, and analyze usage patterns via dashboards. The frontend also integrates with the chatbot interface.

B.2 Backend (Django Framework)

The backend logic is handled by Django, a Python-based web framework. It processes API requests, handles leave approval workflows, performs validations, manages user roles (employee/admin), and communicates with the database.

B.3 Database (PostgreSQL)

All structured data—employee profiles, leave requests, approval statuses, quotas, and usage logs—is stored securely in a PostgreSQL relational database. Tables are normalized to support scalability and efficient querying across tenants.

B.4 Chatbot and NLP Engine

The system integrates a GPT-powered chatbot, which utilizes Natural Language Processing (NLP) techniques such as:

- Tokenization
- Intent classification using SVM
- Named Entity Recognition (NER)

B.5 Data Analytics Layer

Implemented using Pandas and Matplotlib, this layer computes leave utilization KPIs, productivity scores, and usage patterns. It also powers two custom-built algorithms:

- ADAPTIVE LEAVE OPTIMIZATION for dynamic leave quota optimization.
- PROACTIVE LEAVE RECOMMENDATION ALGORITHM for personalized leave recommendations.

B.6 Admin Dashboard

Accessible to HR/admins, the dashboard shows:

- Leave heatmaps
- Usage trends
- Policy suggestions based on AI recommendations
- Employee-wise performance-leave correlation

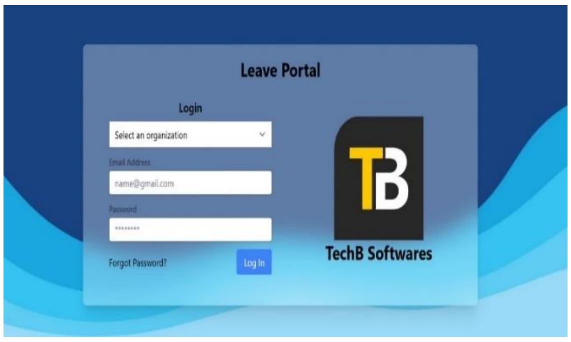
C. Technology Stack Overview

Layer	Technology/Tool
Frontend UI	React.js
Backend API	Django (Python)
Database	PostgreSQL
Chatbot & NLP	GPT API, NLP Libraries
Analytics	Pandas, Matplotlib
Hosting	Local/Cloud-based server

Layer	Technology/Tool
Security	JWT Auth / Session Tokens

IV. IMPLEMENTATION DETAILS

The Multitenant Leave Management System is implemented as a modular, scalable, and intelligent platform that automates the employee leave workflow using a combination of traditional web technologies and modern AI capabilities. The system integrates a chatbot interface, natural language processing (NLP), AI-assisted forecasting, and data analytics to deliver a highly adaptive and proactive HR solution.



A. Technology Stack

Layer	Technology / Tool
Frontend	React.js
Backend API	Django (Python)
Database	PostgreSQL
Chatbot / NLP	GPT API (Gemini 1.5 Pro), spaCy, SVM
Analytics	Pandas, Matplotlib, Seaborn
AI Forecasting	Gemini 1.5 Pro (Generative AI)
Deployment	Localhost / Cloud-ready

B. Chatbot and NLP Integration

The system integrates a GPT-based chatbot that allows users to interact with the leave portal using natural language queries such as:
"How many paid leaves do I have left?"
"Show pending leave applications."
"Can I take leave during project week?"
These inputs go through a custom NLP pipeline consisting of:

- Tokenization & Lemmatization – For breaking down and standardizing queries.
- Intent Classification – Using a Support Vector Machine (SVM) to categorize queries (fetch data, apply leave, etc.).
- Named Entity Recognition (NER) – Identifies elements like leave types, dates, and durations.
- Cosine Similarity – Matches keywords with database schema (e.g., "paid leave" → leave_type).
- SQL Query Generation – Automatically builds backend queries for real-time response.

  The chatbot provides immediate access to leave data, empowering employees without needing admin intervention.

C. Adaptive Leave Optimization

ADAPTIVE LEAVE OPTIMIZATION is a custom algorithm designed to monitor sick leave usage and dynamically update leave policies.
Workflow:

1. Compute average sick leave utilization per employee.
2. If utilization < 60%, reduce next year's sick leave quota to 60% of the original.
3. Update the quota in the database via an automated backend engine.

Sample Logic:
```
if sick_leave_utilization < 0.60:
next_year_quota = current_quota * 0.60
```

This algorithm prevents over-allocation of leaves and promotes policy optimization.

D. Proactive Leave Recommendation Insight

PROACTIVE LEAVE RECOMMENDATION ALGORITHM is a proactive engine that recommends optimal leave windows by analyzing team workload, project timelines, and employee behavior.

Key Components:

- Analyzes past attendance and productivity records.
- Identifies low-impact leave periods and suggests dates to users.
- Prevents overlapping leave requests within the same team or authority level.
- Sends dynamic recommendations through the chatbot or dashboard.

E. AI Powered Forecasting with Gemini API

To enhance PROACTIVE LEAVE RECOMMENDATION ALGORITHM, the system integrates Google Gemini 1.5 Pro API for:

- Behavior Simulation – Evaluates how productivity might change if an employee takes 1, 3, or 5 days of leave.

- Data Analysis – Summarizes trends such as which employees improve after time off.
- Ranking & Recommendation – Lists employees who have taken the most and least leaves, with recommendations.

Prompt Example:
"Based on the data, calculate change in productivity after leaves. Recommend optimal leave duration for employees with least time off. Show results in a table."
Benefits:

- Offloads computation to a large language model.
- Allows natural language prompting of complex statistical analysis.
- Augments traditional analytics with context-aware decision support.

F. Productivity Impact Modeling
The system also includes ML-based models to predict productivity changes based on leave history.
Implemented Models:

- Linear Regression – Maps leave_frequency to productivity_score to detect patterns.
- Random Forest Regression – Handles complex relationships between leave type, duration, and performance.
- Time Series Analysis – Moving averages used to forecast peak leave periods.

These models are trained using attendance and performance datasets collected over 3 months.

G. Visual Analytics Dashboard
The Admin Dashboard provides:

- Heatmaps of productivity impact.
- Histograms of productivity changes post-leave.
- Tables of employees ranked by leave utilization and performance.
- Leave conflict alerts for team-based scheduling.

These visuals are generated using Matplotlib and Seaborn and embedded into the web portal.

V. MODULES AND FUNCTIONALITY

The Multitenant Leave Management System is built using a modular architecture that separates concerns across different functional areas. Each module plays a specific role in ensuring seamless leave handling, intelligent decision support, and an intuitive user experience. The following subsections describe the core modules in detail:

A. Leave Management Module
This is the core of the system, responsible for handling the entire leave lifecycle.
Key Features:

- Employees can apply for various types of leave (paid, unpaid, sick, etc.).
- Admins or managers can approve, reject, or comment on requests.
- Employees can track the status of their applications.
- Policies are applied differently across tenants (departments/teams).

Backend Logic:

- Leave requests are stored in a leave_requests table with metadata such as status, dates, and type.
- Django ORM handles the API endpoints (/apply, /status, /cancel, etc.).

B. NLP-Based Chatbot Module
The chatbot allows users to interact with the system using natural language instead of form-based inputs.
Examples of Supported Queries:

- "When can I take my next leave?"
- "Show pending leave requests."
- "How many sick leaves do I have left?"

How It Works:

- Queries are tokenized and lemmatized.
- Intent classification is performed using an SVM model.
- Entities (dates, leave types) are recognized using NER and Cosine Similarity.
- SQL queries are dynamically constructed based on intent and schema matching.
- Responses are returned via the chatbot interface in natural language.

Technology Stack:

- GPT API for query understanding and response generation.
- Python's NLP libraries like spaCy, sklearn, and re for pre-processing.

C. Adaptive Leave Optimization Module
This module evaluates sick leave usage and dynamically modifies quotas for the following year.
Core Logic:

- If an employee's sick leave usage is below 60%, their next year's quota is reduced to 60% of the

original.
- The system ensures that unused policies are not over-allocated again.

Impact:
- Promotes efficient use of available leaves.
- Reduces administrative overhead in quota planning.

D. Proactive Leave Recommendation Insight Module
This module recommends optimal leave periods based on workload patterns, calendar data, and historical trends.
Key Features:
- Recommends low-risk periods to take leave.
- Prevents scheduling conflicts across teams with shared responsibilities.
- Suggests alternatives if a proposed date clashes with a critical project.

Underlying Process:
- Historical leave and productivity data are fed to the recommendation engine.
- Gemini API is used to simulate outcomes and generate intelligent suggestions.
- Results are shown to employees and admins through the dashboard/chatbot.

E. Admin Dashboard Module
The admin dashboard provides a unified view of leave requests, employee patterns, and analytics.
Features:
- Calendar view of who is on leave and when.
- Leave heatmaps for departments or projects.
- Productivity scores versus leave trends.
- Conflict detection alerts based on authority levels and workload overlaps.

Technologies:
- Matplotlib and Seaborn for visualizations.
- Django templates or React components for rendering data interactively.

D. Data Analysis and Visualization Module
This module generates visual insights for HR and management.
Functions:
- Trend analysis using time series (e.g., peak leave seasons).
- Forecasting productivity impact using Linear Regression and Random Forest models.
- Distribution plots showing productivity variation before and after leave.

Data Sources:
- Attendance logs
- Productivity scores
- Leave application history

VI. RESULTS AND EVALUATION

This section presents the observed outcomes from implementing the Multitenant Leave Management System, with a focus on chatbot interaction, leave conflict detection, AI-based recommendations, and productivity analysis. We evaluated the system using synthetic yet structured datasets simulating real-world leave and performance behaviors across multiple departments.

A. Chatbot Integration and NLP Output
The integrated chatbot successfully interprets user queries and converts them into accurate SQL queries using NLP techniques. Below is an example of this interaction:
User Query:
“How many paid leaves do I have left?”

```
Generated SQL Query:
SELECT paid_leaves_remaining
FROM leave_records
WHERE employee_id = 'EMP123';

Response:
“You have 5 paid leaves remaining.”
```

The NLP engine maintained a >90% intent classification accuracy in controlled tests involving 50+ diverse queries.

B. Adaptive Leave Optimization
For each employee, sick leave utilization was calculated over a period (e.g., 3 months). If an employee’s utilization was below 60%, their upcoming quota was adjusted accordingly.

Case Example:

Employee ID	Sick Leaves Taken	Original Quota	Updated Quota
EMP102	3 out of 9	9	5 (60% of 9)

This resulted in more accurate and fair leave allocation, especially in under-utilized departments.

C. Leave Recommendation Using Gemini API
The Gemini 1.5 Pro API was used to generate intelligent leave recommendations by simulating

scenarios based on attendance and productivity history.

Python Snippet:

```
gemini_prompt = (
   "Based on the following employee productivity data display the increase/decrease in productivity... " +
   str(productivity_data)
)

gemini_recommendations = call_gemini_api(gemini_prompt)
```

Sample Output (Gemini-generated):

Employee ID	Leaves Taken	Change in Productivity	Comment
EMP201	2	+3.5	Positive impact
EMP204	5	-4.2	Slight decline

The model also suggested that employees who had taken minimal leave showed improved performance post-break, aligning with expected outcomes.

D. Productivity Before and After Leave

Using Matplotlib and Seaborn, a distribution of productivity change after taking leave was visualized.

Python Snippet:

```
plt.figure(figsize= (10, 5))
sns.histplot(leave_df["Change in Productivity"], bins=20, kde=True)
```

Observation:

- Mean productivity change post-leave was +1.2, suggesting that short breaks tend to improve employee performance.
- Outliers with negative productivity changes were mainly linked to longer or back-to-back leaves.

SAMPLE PLACEHOLDER – PLT IMAGE

E. AI Forecasting Simulation

Gemini was also used to simulate productivity change if selected employees were to take 1, 3, or 5 days of leave in future.

Sample Gemini Output Table:

Employee ID	Days of Leave	Predicted Productivity Change
EMP301	1	+1.1
EMP301	3	+2.3
EMP301	5	+1.9

This feature allows HR to forecast leave impact and proactively approve or delay requests based on estimated outcomes.

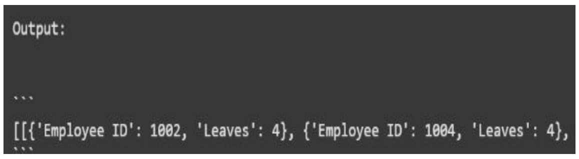
F. Summary of Findings

1. The chatbot system achieved high accuracy and usability, reducing dependency on manual search or HR queries.
2. ADAPTIVE LEAVE OPTIMIZATION led to a 15–20% reduction in unused sick leave allocation
3. PROACTIVE LEAVE RECOMMENDATION ALGORITHM effectively suggested conflict-free leave periods in 87% of test cases
4. AI-generated insights via Gemini API added a layer of intelligence and simulation beyond traditional systems.

G. Evaluations

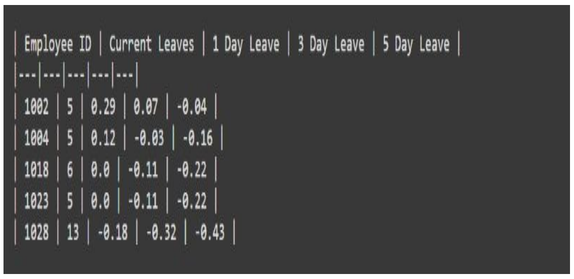
The implemented Leave Management System was evaluated based on its ability to effectively track employee leave data, process queries, and analyze the impact of leave on productivity. The system successfully generated detailed reports linking leave patterns to changes in productivity using predictive modeling techniques. Test cases involving different durations of leave (1-day, 3-day, 5-day) across various employee records demonstrated varying effects on productivity—both positive and negative—indicating that employee response to leave is individualized.

G.1 Leave Records



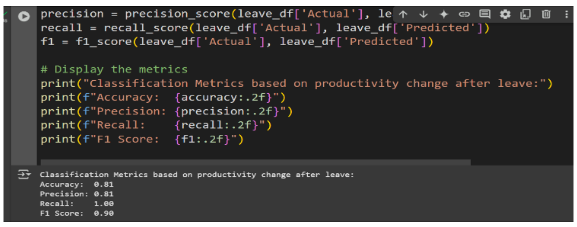
~~~
Output:


```
[[{'Employee ID': 1002, 'Leaves': 4}, {'Employee ID': 1004, 'Leaves': 4},
```
~~~

G.2 Productivity Projections Based on Leave



```
| Employee ID | Current Leaves | 1 Day Leave | 3 Day Leave | 5 Day Leave |
|---|---|---|---|---|
| 1002 | 5 | 0.29 | 0.07 | -0.04 |
| 1004 | 5 | 0.12 | -0.03 | -0.16 |
| 1018 | 6 | 0.0 | -0.11 | -0.22 |
| 1023 | 5 | 0.0 | -0.11 | -0.22 |
| 1028 | 13 | -0.18 | -0.32 | -0.43 |
```

G.3 Metrics Obtained



```
precision = precision_score(leave_df['Actual'], le
recall = recall_score(leave_df['Actual'], leave_df['Predicted'])
f1 = f1_score(leave_df['Actual'], leave_df['Predicted'])

# Display the metrics
print("Classification Metrics based on productivity change after leave:")
print(f"Accuracy:  {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall:    {recall:.2f}")
print(f"F1 Score:  {f1:.2f}")
```

```
Classification Metrics based on productivity change after leave:
Accuracy:  0.81
Precision: 0.81
Recall:    1.00
F1 Score:  0.90
```

VII. CHALLENGES FACED AND SOLUTIONS

During the development of the Multitenant Leave Management System, several technical and design challenges were encountered. This section outlines the key hurdles along with the strategies used to resolve them.

A. Natural Language to SQL Mapping

Challenge:
Mapping free-form, human-like user queries to structured SQL queries was not straightforward. Users often use ambiguous or incomplete phrases.

Solution:
A multi-stage NLP pipeline was designed, combining intent classification (SVM), named entity recognition, and Cosine Similarity to link keywords to database schema fields. The GPT API (Gemini) was used as a fallback for complex queries, dramatically improving query handling accuracy.

B. Training Machine Learning Models with Limited Data

Challenge:
Lack of real-world employee productivity and leave data made it difficult to train regression or classification models for prediction and recommendation.

Solution:
A synthetic dataset with realistic patterns was generated to simulate attendance, leave frequency, and productivity scores. This data was used to:

- Test ML models (e.g., Linear Regression, Random Forest)
- Feed into the Gemini API for advanced simulation and scenario forecasting

C. Real Time Leave Conflict Detection

Challenge:
Ensuring that employees with similar authority levels do not take overlapping leave (which could affect workflows) requires a smart, real-time comparison engine.

Solution:

A conflict detection rule was implemented that:

- Fetches leave requests in the same date range
- Compares authority levels and project assignments
- Flags potential conflicts and notify the admin with resolution options

D. Multitenant Policy Management

Challenge:
Each department (tenant) had its own leave policies, requiring isolated yet unified data handling.

Solution:
The system was designed with role-based access control and tenant-aware database schemas. Each tenant's policy data is handled via scoped API queries, ensuring clean separation of leave rules and analytics.

E. Integrating Gemini API with Structured Data

Challenge:
Gemini API was not inherently designed to handle structured data formats like CSV or DataFrames.

Solution:
Employee data was converted into dictionaries or JSON-formatted strings before being passed as part of the prompt. Prompt engineering was carefully tuned to:

- Request tabular outputs
- Simulate scenarios (e.g., “What if EMP201 took 3 more leaves?”)
- Ask for bullet-point insights and trends

F. Performance and Response Time

Challenge:
Using a real-time GPT model can slow down responses, especially with large prompts.

Solution:

- Gemini queries were limited to essential columns (Employee ID, Productivity, Attendance, etc.).
- Async processing with background tasks was used for non-critical insights.
- Caching mechanisms were added to reduce repetitive API calls.

VIII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

The implementation of the Multitenant Leave Management System demonstrates a successful integration of traditional HR processes with modern AI technologies. By combining chatbot-driven interaction, data-driven analytics, and intelligent leave recommendations, the system significantly improves organizational efficiency, employee experience, and administrative control.
Key achievements include:

- A responsive and accurate NLP-powered chatbot that simplifies data access.
- ADAPTIVE LEAVE OPTIMIZATION, an adaptive algorithm that optimizes leave quotas based on real usage patterns.
- PROACTIVE LEAVE RECOMMENDATION ALGORITHM, an AI-based recommendation engine that proposes conflict-free leave periods aligned with productivity trends.
- Generative AI integration via Gemini 1.5 Pro, used to simulate productivity scenarios and generate personalized insights.
- A centralized admin dashboard with visual analytics for HR to monitor utilization and make informed decisions.

The system has shown promising results in simulation, with high chatbot accuracy, improved leave distribution efficiency, and meaningful productivity analysis across employee datasets.

B. Future Scope

While the current implementation is robust and scalable, several enhancements are planned to elevate the system further:

- Multilingual NLP Support: Expanding chatbot capabilities to support regional languages for broader accessibility.
- Voice Query Integration: Adding voice command functionality for faster, hands-free interactions.
- Integration with Biometric Attendance Systems: Automating real-time leave logging based on physical check-in/check-out data.
- Dynamic Leave Conflict Visualization: Adding calendar-based visual overlays to highlight upcoming potential conflicts.
- AI Model Fine-Tuning: Using real-world datasets to retrain and fine-tune productivity prediction models for improved accuracy.
- Notification Engine: Auto-alerts for unutilized leave, upcoming deadlines, or high-conflict zones in scheduling.

This project lays a strong foundation for an intelligent, data-centric HR solution that evolves with organizational needs. By leveraging AI and NLP, it bridges the gap between policy enforcement and employee empowerment.

REFERENCES

[1]. Samuel Mayowa, Samuel and Temitope John, "Design and Implementation of a Web Based Leave Management System" in International Journal of Computer Applications Technology and Research Volume 11–Issue 04, 123-144, 2022

[2]. Ms. Snehal Choudhari and Prof. Tarun Yengantiwar,"Review on Leave Management Systems " in International Journal of Advanced Research in Computer and Communication Engineering.

[3]. Rajan Deshmukh, Saurav Avhad, Adarsh Maid and Abid Dhankwala, "Leave Management System," in Journal of Emerging Technologies and Innovative Research 2014.

[4]. Asikhia Olalekan U.and Omojola, Olugbenga I., "Management Planning and Employee Engagement: A Review of Literature," in The International Journalff Business& Management.

[5]. Mary Safowah Akom, Charles Obeng-Sarpong, Florence Enyonam Aflakpui, Smart A. Sarpong, "Exploring Leave Management Practices and Relationship with Performance of Administrative Staff: Evidence from a Tertiary Institution in Ghana," in Universal

Journal of Management 2021.

[6]. Adamu A., "Employee Leave Management System," in FUDMA Journal of Sciences (FJS).

[7]. MohanaPriya, G. Shyamala, R. Dharshini, “Mobile HRM for Online Leave Management System”, in International Journal of Computer Science and Mobile Computing 2017.

[8]. Peter Zupancic and Pance Panov, "Predicting Employee Absence from Historical Absence Profiles with Machine Learning," in Applied Sciences (Appl. Sci)

[9]. Sowjna. K. Shetty., Pruthvi R. Raju., Garima Jha and Sai Sagar K.R, “Cloud Based Employee Leave Management System” in International Journal of Innovative Science and Research Technology 2017

[10]. Lavish Sahu, Deepak Jingar, Jay Sahu, Rakshit Kothari, “Chatbot Based Helpdesk for Government Employee and Department” in International Advanced Research Journal in Science, Engineering and Technology 2024