**Lebanese university**

الجامعة اللبنانية

كلّيّة الهندسة

الفرع الثالث

**Faculty of engineering**

**Branch 3**

**Mini Project Report**

**“Leave Management System”**

**2024-2025**

Presented for:

Doctor Mohammad Aoude

Prepared by:

* Fatima Moselmani (6468)
* Fatima Jawad (6496)

Abstract

This report presents the design, implementation, and evaluation of a smart Leave Management System (LMS) that integrates both a rule-based engine and an AI-powered decision engine using a Large Language Model (LLM). The dual AI approach allows the system to process employee leave requests intelligently, ensuring consistency, transparency, and adaptability. The system aims to improve decision-making efficiency while allowing comparative analysis between traditional and AI-driven methodologies.

1. **General Introduction:**

Managing employee leave requests is a critical process in any organization, and crucial aspect of human resource operations. Traditional leave approval systems often rely on manual reviews or static rule-based software, which may result in delays, inconsistencies, or lack of scalability as organizations grow. To address these challenges, this project proposes a smart **Leave Management System** that integrates both **rule-based decision engine** and **an AI-powered decision engine** based on a Large Language Model (LLM). The system allows employees to submit leave requests through a web interface, it process these leave requests and then tells whether to accept or reject the request with the reason if rejected.

The combination of deterministic rules and AI reasoning provides a platform for evaluating decisions from both procedural and intelligent perspectives. This dual approach improves automation, fairness, and explainability in leave management while reducing the administrative burden on human resources.

*Objective:*

The primary objective of this project is to:

* To build a dual-engine Leave Management System that processes leave requests using:
* A traditional rule-based engine (based on organizational policies)
* An AI agent powered by an LLM that evaluates requests based on context, previous patterns, and natural language explanations.
* To compare both engines in terms of performance, decision accuracy, and adaptability.
* To build a user-friendly web-based interface that shows decisions and justifications.

*Report Plan*

This report is organized as follows:

* **Section 1: General Introduction** – Explains the motivation, objectives, and scope of the project.
* **Section 2: System Architecture** – Describes the overall design and structure of the system, including major components and their interactions.
* **Section 3: Implementation Details** – Covers the tools and technologies used, backend logic, interface design, and integration of the rule-based and AI engines.
* **Section 4: Results and Discussion** – Presents case studies and examples, compares the decisions made by the rule-based engine and the LLM-based AI engine, and discusses the observed outcomes.
* **Section 5: Conclusion and Future Work** – Summary of outcomes and discussion of potential improvements, such as fine-tuning the LLM, introducing user feedback.

1. **System Architecture:**
   1. Overview

The system is composed of the following core modules:

* + Web frontend
  + Flask backend API
  + Relational database
  + Rule-based engine
  + AI engine (LLM integration)
  1. Component Breakdown
  + Frontend: A web-based user interface (UI) built with HTML/CSS/JS to allow employees to submit leave requests and view decision outcomes.
  + Backend: Developed using Python Flask, it handles request processing, API routing, and system logic.
  + Database: MySQL used to store user data, leave requests, and decision logs.
  + Rule Engine: Encodes policies such as leave balances, holidays, and maximum duration.
  + AI Engine: Uses a prompt-based interface to an LLM to analyze request descriptions and provide decisions with justifications. (we use the model qwen3:1.7b)
  1. Data Flow
  + User submits a leave request.
  + Backend stores data and passes it to the decision engine.
  + Rule-based or AI engine processes the request.
  + Decision and justification are stored and returned to the user.
  + Dashboard presents decision history and statistics.

1. **Implementation Details**

* We started by identifying common rules followed by HR departments. Based on these rules, the leave requests might be accepted or rejected.

Rules are:

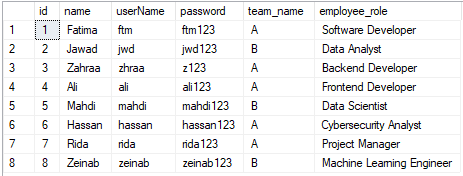
* + past leave history (max 15 days per year)
  + project deadline (no leave allowed during critical periods)
  + A maximum of 3 people from the same team can be on leave at the same time
  + Duration of leave must not exceed 7 consecutive days
  + Reason for leave: accepted directly if it's a sick leave or an emergency
  + Notice period must be at least 2 days before the requested leave start date
* Then, we designed the database and created the following tables:
  + employees: contains employee information (name, username, password, team\_name)

Figure 1 employees table

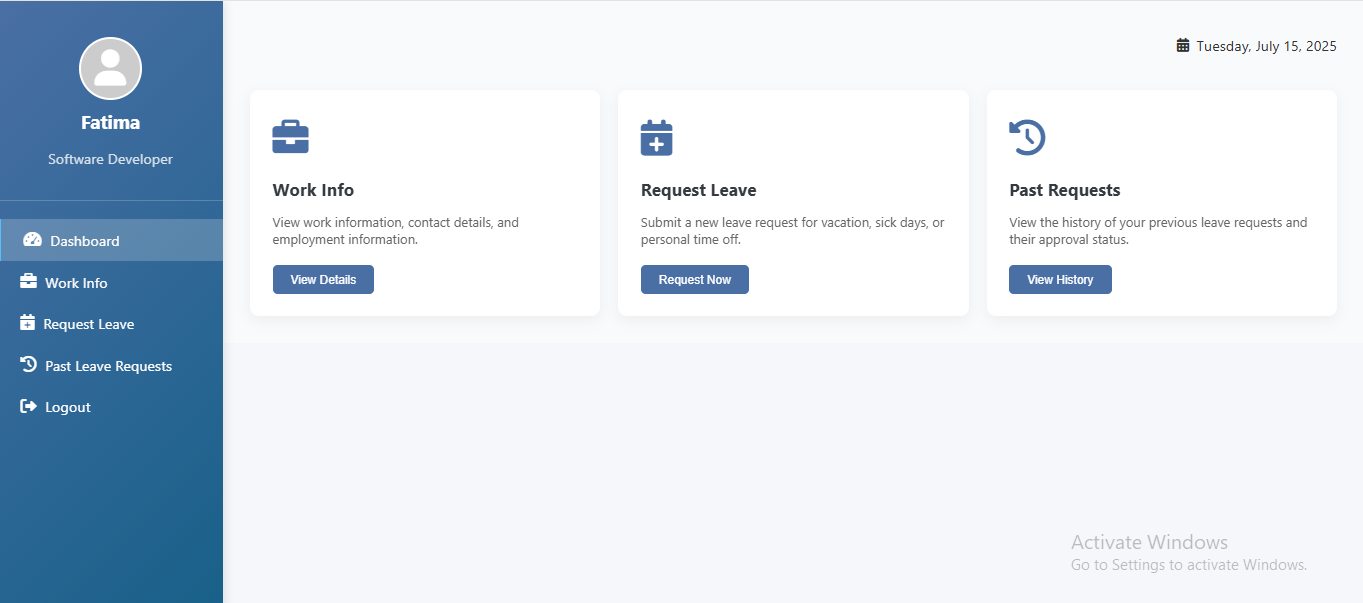
* + leave\_requests: stores each leave request (id, employee\_id, start\_date, end\_date, reason, request\_date, rule\_decision, rule\_reason, ml\_decision, ml\_reason)
  + project\_deadlines: links teams to their ongoing projects and deadline dates (id, team, project\_name, deadline\_date)
* After setting up the database, we designed the dashboard that allows users and admins to interact with the system.

Figure 2 Employee Dashboard

The dashboard includes:

* A login interface for employees
* A leave request submission form (a form to enter leave requests)

It takes these as inputs:

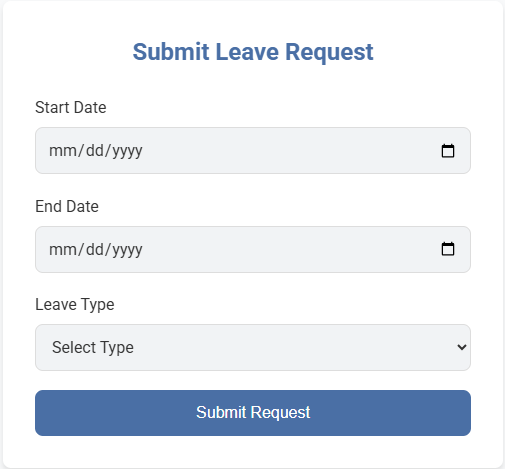
Start date

Figure 3 Submit Leave Request Form

End date

Reason

* A page to show the result whether the request is accepted or rejected
* A history table showing past leave requests and their status

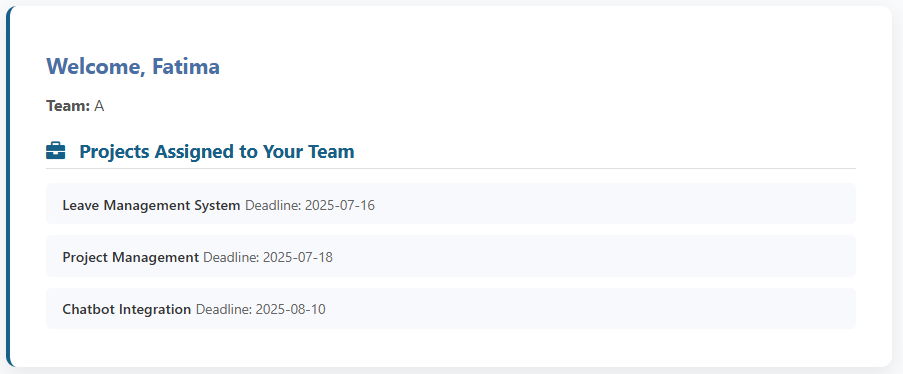
The dashboard is implemented using a simple and responsive web interface with HTML, CSS, and JavaScript, connected to the backend through Flask APIs. It serves as the central point of interaction, where decisions, logs, and employee data are visualized clearly and efficiently.

Figure 4 work info page

* We design the file app.py that controls everything. For example, this Python module takes a leave request (start date, end date, reason) and calculates these parameters: team\_on\_leave, has\_deadline\_conflict, used\_annual\_leave, notice\_days, leave\_requested\_days, reason. These parameters are sent to both **rule\_based\_engine.py** and **ai\_agent.py.**
* After these, we designed the file **rule\_based\_engine.py**, which contains the logic to process leave requests based on the predefined company policies. This Python module takes the above parameters and evaluates them against each of the rules, returning a decision (Accept or Reject) and a justification.
* Also we design the **ai\_agent.py** file, that uses a large language model via ollama (we use the model qwen3:1.7b) to analyze the leave request in context, understand reasons, and provide a human-like decision with an explanation.

It follow this logic:

* Prompt Template: Each leave request is converted to a formatted prompt, including:
* Request details (dates, reason)
* Employee profile (leave history, role)
* Company leave policy
* The LLM responds with:
* A decision (Accept / Reject)
* A natural language justification
* Both the decision and the reason from the **rule\_based\_engine** and **ai\_agent**, are stored in database and displayed to employee for comparison.

1. **Results and Discussion**

*Case Study 1:*

The employee submit this form:

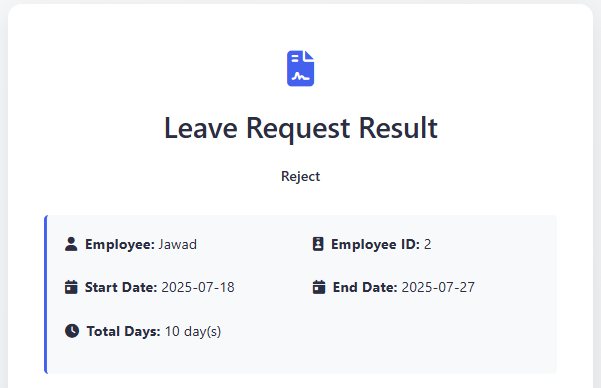
* Start Date: 18/07/2025
* End Date: 27/07/2025
* Leave type: other

Figure 5 Leave Request Result

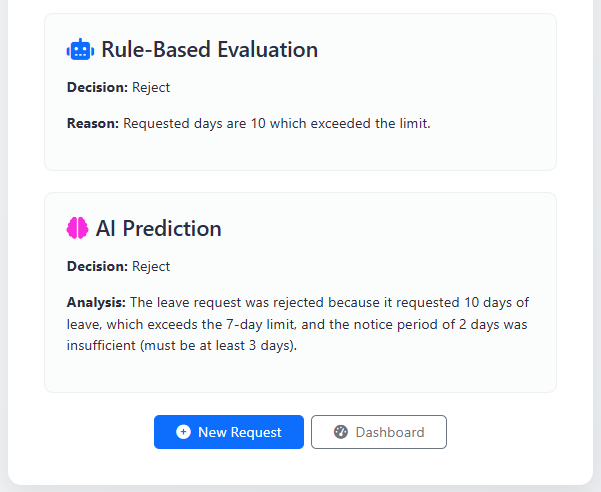
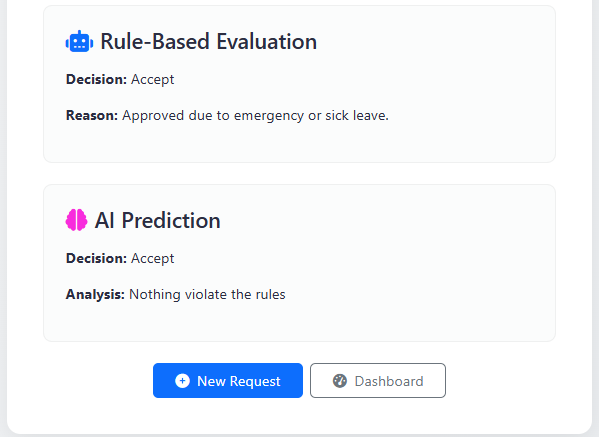


Figure 6 Rule based and AI outputs

*Case Study 2:*

The employee submit this form:

* Start Date: 17/07/2025
* End Date: 25/07/2025
* ****Leave type: other

*Comparison:*

* Rule-Based Engine: Fast, deterministic, predictable; limited flexibility.
* AI Agent: More flexible and context-aware; can handle nuanced situations and provide human-like reasoning.

The results show that using both engines side by side enhances decision-making by combining reliability with adaptability.

1. **Conclusion:**

In this report, we presented a Leave Management System that integrates both a rule-based engine and an AI-powered agent using LLMs. Through structured design, modular implementation, and real-world testing, we demonstrated how the hybrid approach improves decision-making efficiency and fairness.

*Future Work*

To enhance our project further, several features could be added:

* Add feedback loops to improve AI decision accuracy through supervised learning.
* Extend AI prompts to consider organizational culture or role-criticality.
* Implement admin review dashboards to override or approve final decisions.
* Use NLP to process unstructured requests (e.g., free-text emails).