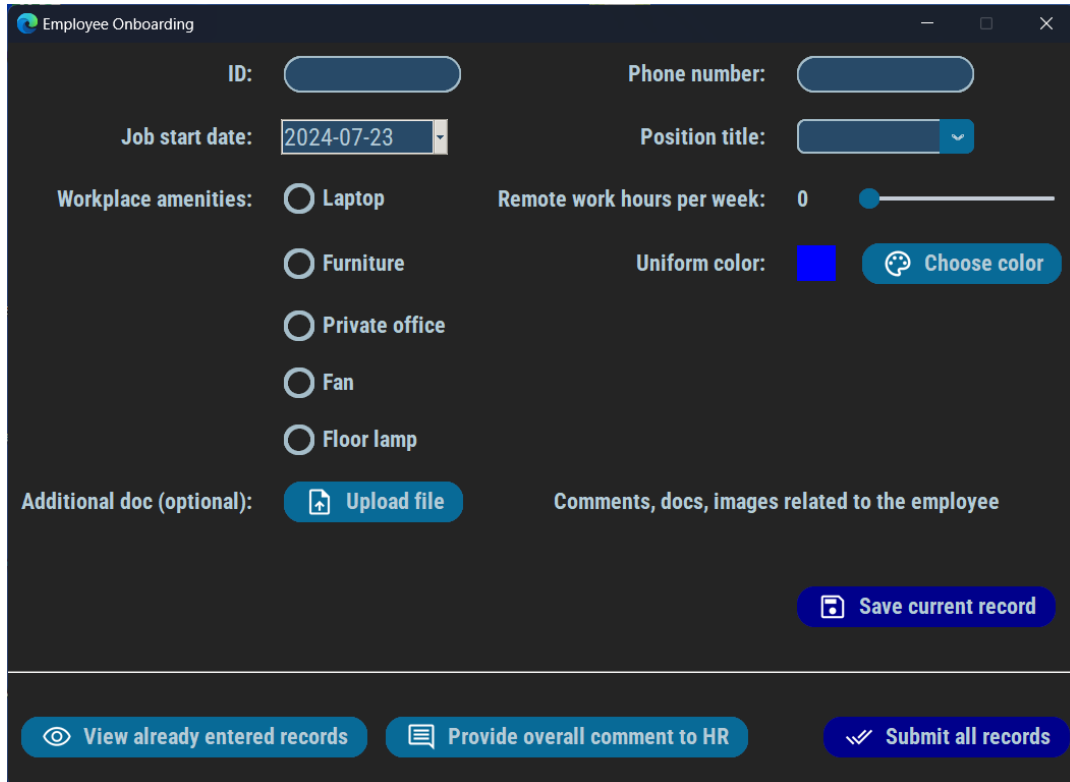


# Employee

## Short description of the application

This is an application written in Python, designed to support HR Assistants in entering and managing employee onboarding information through a structured form. It also allows assistants to provide overall feedback to the HR manager, which applies to all employee data as a final message rather than feedback on individual records.



The screenshot shows a web application titled "Employee Onboarding" with a dark theme. The form includes the following fields and controls:

- ID:** A text input field.
- Phone number:** A text input field.
- Job start date:** A date picker showing "2024-07-23".
- Position title:** A dropdown menu.
- Workplace amenities:** A group of radio buttons for "Laptop", "Furniture", "Private office", "Fan", and "Floor lamp".
- Remote work hours per week:** A slider control set to "0".
- Uniform color:** A color selection area with a blue swatch and a "Choose color" button.
- Additional doc (optional):** A section with an "Upload file" button.
- Comments, docs, images related to the employee:** A large text area for notes.
- Save current record:** A button with a floppy disk icon.
- View already entered records:** A button with an eye icon.
- Provide overall comment to HR:** A button with a speech bubble icon.
- Submit all records:** A button with a checkmark icon.

## Instructions

Users of the employee application sometimes encounter situations where the application does not align with their goals or needs. For example, they may need to enter a type of data that the form does not directly accommodate. In such cases, they often rely on *workarounds* or improvised solutions. These workarounds are deliberate deviations from the intended use of the software that arise when application forms and workflows fail to meet user needs.

We use large language models (LLMs) to identify user workarounds in the employee application and then generate feature requests that address these workarounds. In this study, we experiment with *eight* different LLMs, and you take on the role of a developer of the application that is looking to improve it. Your task is to evaluate their generated feature requests based on three criteria: *completeness*, *clarity*, and *difficulty*. All workarounds identified by the LLM are presented together with the corresponding feature

requests. Each feature request has been generated by the LLM to address these workarounds. In the following questions, indicate the degree to which you agree with the statement, from 1 "completely disagree" to 5 "completely agree".

The responses generated by the LLMs are provided in the accompanying folders, where the file **results/LLMi/employee.txt** contains the output of the **LLMi**.

- 1. Completeness:** The feature proposed by the LLM addresses all the identified workarounds. (1: completely disagree, 5: completely agree)

LLM	LLM1	LLM2	LLM3	LLM4	LLM5	LLM6	LLM7	LLM8
Rating								

- 2. Clarity:** The feature description proposed by the LLM clearly explains how I, as a developer, should implement the feature. (1: completely disagree, 5: completely agree)

LLM	LLM1	LLM2	LLM3	LLM4	LLM5	LLM6	LLM7	LLM8
Rating								

- 3. Difficulty:** The feature proposed by the LLM is likely to be difficult to implement. (1: completely disagree, 5: completely agree)

LLM	LLM1	LLM2	LLM3	LLM4	LLM5	LLM6	LLM7	LLM8
Rating								

If you have any feedback, please enter it in the box below.

--