Task Description

You are tasked with building a Flask API that serves data from a SQLite database containing information about employees of a fictional company. The API should allow for basic CRUD (Create, Read, Update, Delete) operations on the employee data, and also provide a few additional endpoints for data analysis purposes.

Requirements

1. The Flask API should have the following endpoints:
    - `GET /employees`: Returns a list of all employees in the database.
    - `GET /employees/<int:id>`: Returns the employee with the specified ID.
    - `POST /employees`: Creates a new employee with the specified data (name, department, salary, hire_date). The API should return the ID of the newly created employee.
    - `PUT /employees/<int:id>`: Updates the employee with the specified ID with the specified data (name, department, salary, hire_date).
    - `DELETE /employees/<int:id>`: Deletes the employee with the specified ID.
    - `GET /departments`: Returns a list of all unique departments in the database.
    - `GET /departments/<string:name>`: Returns a list of all employees in the specified department.
    - `GET /average_salary/<string:department>`: Returns the average salary of employees in the specified department.
    - `GET /top_earners`: Returns a list of the top 10 earners in the company based on their salary.
    - `GET /most_recent_hires`: Returns a list of the 10 most recently hired employees.
2. The API should include error handling for common scenarios (e.g. invalid ID in a `GET` request, missing data in a `POST` request).
3. The database should be generated using the SQLAlchemy library and should contain a table called "employees" with the following columns: `id`, `name`, `department`, `salary`, and `hire_date`. The `id` column should be an auto-incrementing integer and serve as the primary key. The data in the table should be randomly generated using the Faker library. The generated data should include at least 1000 employees.

   - **name**: a string with a maximum length of 50 characters
   - **department**: a string with a maximum length of 50 characters
   - **salary**: a float with a minimum value of 0 and maximum value of 1000000
   - **hire_date**: a datetime object in the format of 'YYYY-MM-DD HH:MM:SS', with a range from 01-01-2020 00:00:00 to today.

4. The data science component of the task is to provide an implementation of a machine learning model that can be used to predict the salary of a new employee based on their department, hire date, and job title. You can use any machine learning library of your choice for this task (e.g. scikit-learn). The trained model should be exposed via a new endpoint:
    - `POST /predict_salary`: Takes in data for a new employee (department, hire date, and job title) and returns the predicted salary.

Constraints

- The implementation should use Python 3 and the Flask framework.
- You may use any additional libraries as long as they are freely available and can be installed via pip.
- The implementation should use SQLAlchemy to generate the database and the Faker library to generate fake data.
- The implementation should use SQLite as the database engine.
- The implementation should be provided as a Git repository that includes all necessary code.