ML/AI/NLP Task: Build a Multi-Task Q&A API Using Hugging Face's Pre-Trained Models

Timeline: 2 Days

Objective: Your task is to create a Q&A API using Hugging Face's pre-trained models. The API, given a user's question, should perform multiple tasks to generate an answer. This task aims to evaluate your ability to implement pre-trained models, design complex backends, and apply logical thinking to problem-solving.

Scenario: Suppose we are building an internal tool for a large multinational corporation. The tool should be able to answer questions about the company's operations, such as details about various departments, employee roles, recent company news, and so on. The data can be fictional but should be diverse enough to cover a wide range of topics.

Requirements:

1. **Question Classification:** When a question is submitted to the API, the first task is to analyze and categorize the question using a pre-trained model. For example, you could use Hugging Face's 'distilbert-base-uncased' model, fine-tuned on a question classification task. The categories could include "Company Department", "Employee Role", "Company News", etc.
2. **Information Retrieval:** Depending on the type of question, the API should retrieve relevant information. This could involve querying an SQL database for employee details, making a request to a news API for company news, or fetching information from a CSV file for departmental details.
3. **Answer Generation:** Use Hugging Face's 'bert-large-uncased-whole-word-masking-finetuned-squad' model to generate a potential answer based on the retrieved information. The input to the model would be the user's question and the retrieved information as context.
4. **Answer Validation:** Implement a mechanism to validate or rank the generated answer(s). This could involve checking if the answer is present in the context, or if it contains any keywords related to the question.

5. **API Structure:** The API should accept a question as input and return a validated answer as output. It should be designed to handle multiple requests concurrently and should follow RESTful principles.

Deliverables:

1. A link to the source code in a public repository on GitHub.
2. Documentation for the API, explaining how to make requests and what responses to expect.
3. A brief report detailing your approach, the models used, the logic behind the architecture, and any challenges you faced and how you overcame them.

Evaluation Criteria:

1. **Functionality:** Does the API work as expected without any errors or bugs?
2. **Complexity:** How efficiently and effectively have you built the backend for the API?
3. **Interpretation Skills:** How well does the API classify questions, retrieve relevant information, generate potential answers, and validate the final answer?
4. **Code Quality:** Is the code well-organized, easy to understand, and properly commented?

Additional Notes:

- You may use any additional libraries or tools as necessary.
- You should focus on writing clean, efficient, and well-documented code.