This Assignment is only for interview purposes. It has 2 parts. Part A is for 70 marks and Part B is for 30 marks.

PART A: Task: Extract and Categorize Tasks from Unannotated Text (70 marks)

Objective:
Create an NLP pipeline to identify and categorize tasks from unstructured text. Eg. *Rahul wakes up early every day. He goes to college in the morning and comes back at 3 pm. At present, Rahul is outside. He has to buy the snacks for all of us.* In this, task is that "Rahul has to buy snacks for all of us" and this is the only sentence that should be extracted. For a task, the bare minimum is what action has to be performed. Who has to perform this (entity extraction) and when should the task get over are good to have. A better example of a task would be: *Rahul should clean the room by 5 pm today*.

So, the main objective of this assignment is to create a program which takes a paragraph (or even larger text) as input, and the program has to figure out whether there is any task(s) in that paragraph or text, and if there is one (or more), extract it. Also, extract who has to do that task and by when, if this information is present. [Note: Task and action item are the same.]

Details:
You are tasked with building a system to analyze unstructured text and extract potential tasks/action items. Since annotated datasets are unavailable for this task, use an heuristic-based approach. [Note: You can't use any LLM or any model. In this, you first have to come up with an algorithm/way for what and how to extract and then use NLP to extract that information.]

Steps:

1. **Preprocessing:**
    - Clean the text (remove stop words, punctuation, and irrelevant metadata).
    - Tokenize sentences and perform POS tagging to identify actionable verbs (e.g., "schedule," "discuss," "review").
2. **Task Identification:**
    - Develop heuristics to identify sentences likely representing tasks.
    - Example heuristics:
        - Sentences starting with imperative verbs.
        - Sentences mentioning deadlines or names (e.g., "John to review").
3. **Categorization:**
    - Use word embeddings (e.g., BERT, Word2Vec) to cluster tasks into categories. Please come up with "useful categories" on your own.
    - Define categories dynamically using topic modelling (e.g., Latent Dirichlet Allocation).

4. **Output:**
    - Generate a structured list of tasks with categories. Also, for the tasks that have information, list who is to do that task and when is the deadline.

Deliverables:
1. A short video of a code walkthrough and testing. You can upload the video on Google Drive (and share access) or any of the many sites that are available these days.

2. A link to your code or your code itself (Jupyter notebook or Python script with all steps, well-documented). Include modular functions for preprocessing, task extraction, and categorization. Validate results with a small, manually curated sample. Include insights or challenges faced during the task.

PART B: Build a machine learning model to classify customer reviews as positive or negative. This is a basic text classification problem. The candidate will demonstrate skills in text preprocessing, feature extraction, and training a simple ML model. (30 marks)

Steps:

1. **Data Collection**:
    - Use a publicly available dataset like **IMDb Reviews** or **Amazon product reviews** (or any other public dataset but share the link).
2. **Data Preprocessing**:
    - Clean the text data by removing unnecessary characters, digits, and stop words.
    - Convert text to lowercase, and tokenize it (split text into words).
3. **Feature Extraction**:
    - Use **TF-IDF** (Term Frequency-Inverse Document Frequency) to convert the text into numerical features.
    - Briefly explain why TF-IDF is chosen over other methods like Bag of Words.
4. **Model Selection and Training**:
    - Train a classification model using any one of the following:
        - Logistic Regression
        - Naive Bayes
        - Support Vector Machine
    - Train the model on the dataset and test it on a separate portion (e.g., 80/20 train-test split).
    - Tune hyperparameters to optimize performance.

5. **Evaluation**:
    - Evaluate the model using **accuracy**, **precision**, and **recall**.
    - Discuss how you would improve the model (e.g., trying different algorithms, parameter tuning, adding more data).

Deliverables:
1. A short video of a code walkthrough and testing. You can upload the video on Google Drive (and share access) or any of the many sites that are available these days.

2. A link to your code or your code itself (Jupyter notebook or Python script with all steps, well-documented). Include modular functions for preprocessing, task extraction, and categorization. Validate results with a small, manually curated sample. Include insights or challenges faced during the task.