1. Problem Definition (6 points)

Hypothetical AI Problem: *Detecting biased language in online job postings.*

Objectives:

1. Identify and flag gender-coded or discriminatory phrases.
2. Suggest neutral language alternatives in real time.
3. Improve diversity in applicant pools over time.

Stakeholders:

- Recruitment teams
- Job seekers from underrepresented groups

KPI: *Reduction in flagged biased phrases per 100 job postings over time.*

2. Data Collection & Preprocessing (8 points)

Data Sources:

1. Public job listing datasets (e.g., Kaggle's job descriptions dataset)
2. Web-scraped job ads from company websites or job boards (e.g., Indeed, LinkedIn)

Potential Bias: Training data may already reflect historical biases (e.g., overuse of "aggressive," "dominant," "rockstar"), which could reinforce the status quo if not corrected.

Preprocessing Steps:

- Text cleaning (punctuation removal, lowercasing)
- Tokenization and lemmatization
- Handling class imbalance (e.g., SMOTE or stratified sampling)

3. Model Development (8 points)

Chosen Model: *Fine-tuned BERT (Bidirectional Encoder Representations from Transformers)*
Justification: BERT is highly effective for NLP tasks and captures contextual meaning in language—essential for identifying subtle bias.

Data Split Strategy:

- 70% training
- 15% validation
- 15% testing Stratified sampling ensures representation of different bias types.

Hyperparameters to Tune:

1. Learning rate – to optimize training stability and convergence.
2. Maximum sequence length – to control context depth for longer job descriptions.

4. Evaluation & Deployment (8 points)

Evaluation Metrics:

- *F1-Score:* Balances precision and recall—critical in detecting rare biased phrases.
- *False Positive Rate:* Ensures model doesn't over-flag neutral language.

Concept Drift: Occurs when the statistical properties of the data change over time (e.g., new slang, shifting definitions of inclusivity). **Monitoring Solution:** Use active learning or continuous evaluation pipelines with periodic re-training using recent data.

Technical Deployment Challenge: Scalability – processing large volumes of job ads in real time may require distributed computing or efficient model serving via APIs like FastAPI + TorchServe.