Step-by-Step Project Execution Checklist: Hierarchical Organizational Influence Propagation

Hilal Hussain, Mahmoud Abdelmoneum, Theo Chen

October 29, 2025

0. Initial Preparation and Planning

Objective: Establish shared understanding, infrastructure, and organizational structure to enable efficient parallel work throughout the project lifecycle. This foundation ensures all team members can contribute effectively without blockers.

Tasks:

Read through all project materials, technical plan, and literature review to ensure all team members share a common understanding of objectives, techniques, and deliverables.

Decide and assign concrete roles (e.g., data wrangling, modeling, system integration, evaluation, report writing) among team members for parallel progress.

Ensure all team members have access to necessary computing resources, accounts (e.g., Google Colab Pro, institutional clusters, AWS/Azure credits), and software licenses if applicable.

Establish a version-controlled GitHub repository for code, data pipeline, and documentation. Set up project management tools (e.g., Notion, Trello) for task tracking.

Completion Metrics:

- All team members can access GitHub repository and have write permissions
- Computing environment verified (all team members can run a simple PyTorch GPU test)
- Project management board created with all major tasks mapped to team members
- Initial team meeting held with documented role assignments and communication schedule

1. Dataset Curation and Generation

Objective: Construct a comprehensive, diverse, and high-quality dataset that captures organizational structures, recommendations, and multi-step agent responses. This dataset is the foundation

for training and validating our hierarchical multi-agent model. Without sufficient data covering various organizational types, sizes, and response patterns, the model cannot learn generalizable representations of influence propagation.

1.1. Search for Open Datasets Related to Organizational Decision-Making

Objective: Identify and collect real-world datasets that provide ground-truth organizational responses, team decision-making data, or similar multi-agent interaction patterns. Real data anchors our synthetic generation and provides validation targets that test whether our model captures actual organizational behavior rather than artifacts of synthetic generation.

Tasks:

Explore the following sources for actual or proxy datasets:

- A dataset of human decision-making in teamwork management (PMC)
- Public Recommender Systems Datasets (GitHub)
- UCSD Recommender Systems Datasets
- Awesome Multi-Agent Papers (with dataset links)
- Top Datasets for AI Agents (Coresignal 2025)
- Amazon Bedrock multi-agent business datasets
- JASSS multi-agent org models (case studies)
- TalkToData multi-agent system data collection tools

For case study and text-based datasets, check archives of the Harvard Business Review, Stanford GSB, business publications, and government agency filings (SEC EDGAR, public board reports, etc).

Download and examine at least 5 candidate datasets, documenting their structure, size, and relevance to organizational response modeling.

Completion Metrics:

- Document created listing at least 5 datasets with descriptions, sizes, formats, and relevance scores (1-5)
- At least 500 real-world organizational case studies collected and stored in standardized format (JSON/CSV)
- Successfully extracted organizational context, recommendation text, and documented responses for at least 300 cases
- Initial analysis completed showing distribution of organization sizes, industries, and response types

1.2. Synthetic Data Generation

Objective: Generate a large-scale synthetic dataset (target: 10,000 organizations, 750,000 agent responses) that covers diverse organizational structures, recommendation types, and response dynamics. Synthetic data allows us to control ground-truth influence relationships and scale beyond limited real-world data, enabling robust model training and controlled hypothesis testing.

Tasks:

Design prompts for GPT-4, Claude-3, or other LLMs (LLM synthetic data guide) to generate:

- Organizational charts with member attributes (role, authority level ρ_i, department, seniority)
- Company context descriptions (industry, size, culture, recent history, values)
- Detailed external recommendations spanning strategic, operational, policy, and crisis domains
- Multi-step agent responses ($t = 0, 1, 2, 3$) with explicit reasoning and influence acknowledgment

Script randomized generation ensuring diversity across: organization size (10-500 agents), hierarchy depth (2-6 levels), span of control (2-10 reports per manager), industries (tech, healthcare, finance, manufacturing, government), and organizational cultures (risk-averse, innovative, bureaucratic, agile).

Implement validation checks: coherence tests (does agent reasoning match context?), diversity metrics (coverage of parameter space), and quality filters (remove degenerate or contradictory responses).

Generate data in phases: Phase 1 (1,000 orgs for initial development), Phase 2 (5,000 orgs for hypothesis testing), Phase 3 (10,000 orgs for final training).

Completion Metrics:

- Phase 1: 1,000 organizations with 50+ recommendations each (50k agent responses) generated and validated
- Diversity check passed: organizations span all intended sizes (10-500), depths (2-6), and industries (5+)
- Quality check: manual review of 100 random samples shows >90% coherent and realistic scenarios
- Data format standardized: all data stored in consistent JSON schema with documented fields
- Phase 2: 5,000 organizations (375k responses) with documented quality metrics
- Phase 3: 10,000 organizations (750k responses) with final validation report

1.3. Agent-Based Simulation

Objective: Create a rule-based agent-based model (ABM) that serves dual purposes: (1) generate additional training data with known causal influence structure for validation, and (2) provide a transparent baseline for comparing learned model behavior against interpretable heuristics. This allows us to verify whether our deep learning model captures fundamental organizational dynamics encoded in established organizational behavior theory.

Tasks:

Implement ABM using Python (Mesa documentation, NetLogo, or custom PyTorch simulation) with agent decision rules based on:

- Personal risk tolerance (influences support for risky recommendations)
- Departmental goal alignment (agents support recommendations benefiting their unit)
- Hierarchical influence (agents weight opinions of higher-status colleagues by power differential)
- Coalition dynamics (agents coordinate with peers before committing)
- Organizational norms (culture-specific response patterns)

Design authority graph generation with configurable topology (star, hierarchical tree, matrix, hybrid) and influence weights derived from formal authority and informal ties.

Run parameter sweeps varying: noise levels, influence weight distributions, response determinism, and organizational structures.

Generate 100,000 simulation runs with full trace logs (agent states, observations, influence weights, final responses) for training data augmentation and ground-truth validation.

Completion Metrics:

- Working ABM implemented that can simulate organizational response for arbitrary org charts and recommendations
- Parameter space documented with at least 5 key parameters and their ranges
- 100,000 simulation runs completed with full traces logged
- Validation: ABM reproduces qualitative patterns from organizational behavior literature (e.g., hierarchical cascades, consensus time vs. centralization)
- Comparative analysis document: ABM behavior vs. real case studies shows alignment on key metrics (consensus time, support distribution)

2. Data Pipeline and Preprocessing

Objective: Build robust, efficient, and reusable data infrastructure that transforms raw organizational data (text descriptions, graphs, responses) into model-ready formats. A well-designed pipeline enables rapid experimentation, ensures reproducible splits, and handles the complexity of multi-modal data (text + graphs + tabular features).

Tasks:

Develop data loaders supporting all formats: JSON (organizational scenarios), CSV (agent responses), adjacency lists/GraphML (authority graphs), and text files (recommendations and context).

Implement text preprocessing pipeline using standard NLP tools (spaCy, NLTK, or HuggingFace tokenizers): lowercasing, tokenization, stopword handling (optional), sentence segmentation, and encoding compatible with chosen transformer models (BERT/RoBERTa).

Build graph encoding utilities using NetworkX and PyTorch Geometric to convert organizational charts into graph objects with node features (agent attributes) and edge features (influence weights).

Implement stratified dataset splitting ensuring balanced representation across: organization size bins (small <50, medium 50-150, large >150), industries, hierarchy depths, and response distribution types. Use 70/15/15 split for train/validation/test.

Create PyTorch Dataset and DataLoader classes supporting batching of variable-size graphs, padding of text sequences, and efficient GPU transfer.

Design utilities for generating contrastive pairs (positive: recommendation-org with support; negative: random pairings) for dual encoder training.

Completion Metrics:

- Data loader successfully loads and batches data from all formats (JSON, CSV, graphs) without errors
- Pipeline processes full dataset (10k orgs) in under 10 minutes on single GPU
- Stratified splits created with documented statistics: train (7k orgs, 525k responses), val (1.5k orgs, 112k responses), test (1.5k orgs, 112k responses)
- Split validation: no org leakage across splits, distributions of size/industry/depth similar across splits (KS test $p > 0.05$)
- Sample batch visualization notebook created showing text, graphs, and labels correctly formatted
- Unit tests passing for all data loading, preprocessing, and batching functions

3. Model Development

Objective: Implement the core hierarchical multi-agent model architecture consisting of dual encoders (text), graph neural networks (influence propagation), and hierarchical policy networks (agent response prediction). This is the central technical contribution that learns to map organizational contexts and recommendations to agent-level response predictions while capturing strategic interaction and influence dynamics.

3.1. Baseline Models

Objective: Establish performance lower bounds and sanity checks through simple baseline models. Baselines provide reference points for evaluating whether the complexity of our proposed architecture yields meaningful improvements, and help identify whether the task is learnable at all from the available data.

Tasks:

Implement non-learning baselines:

- Random prediction: uniform random over 5 response categories
- Majority class: always predict most common response in training set
- Organizational culture heuristic: predict support/oppose based on culture-recommendation alignment score

Implement simple learning baselines:

- Logistic regression on bag-of-words features (TF-IDF of recommendation + org description)
- Feedforward MLP (3 layers, 512 hidden units) on concatenated text embeddings (no graph structure, no temporal dynamics)
- Independent Q-learning or single-agent RL baseline (agents learn independently without coordination)

Train all baselines on training set and evaluate on validation set with consistent metrics (accuracy, F1).

Completion Metrics:

- All baselines implemented and tested on sample data
- Baseline results documented: random (20% accuracy), majority (25-30%), BoW logistic regression (40-50%), MLP (50-60%)
- Validation results table created comparing all baselines on accuracy, macro F1, and per-level accuracy

- Baseline training completes in under 2 hours for full training set
- Baseline performance establishes that task is learnable (non-trivial improvement over random/majority)

3.2. Dual Encoder Setup for Text

Objective: Learn semantic embeddings of recommendations and organizational contexts that capture compatibility and response likelihood. The dual encoder maps variable-length text to fixed-dimensional vectors in a shared space where semantically similar or compatible pairs are close, enabling the model to generalize to new recommendation-organization combinations.

Tasks:

Select pre-trained language models from HuggingFace Model Hub (link): BERT-large, RoBERTa-large, or Sentence-BERT variants.

Implement Siamese/dual encoder architecture following best practices from:

- EmergentMind: Dual Encoder Architecture
- DeepLearning.AI Embedding Models course

Design training objectives: contrastive loss (maximize similarity for positive pairs, minimize for negatives), triplet loss, or InfoNCE loss. Experiment with temperature parameters and negative sampling strategies (random, hard negatives).

Fine-tune encoders on organizational data with batch size 64, learning rate 2e-5, warmup steps 1000, and early stopping on validation loss.

Evaluate embedding quality: measure retrieval accuracy (given org embedding, retrieve correct recommendation among distractors) and embedding space structure (t-SNE visualization shows clustering by domain/industry).

Completion Metrics:

- Dual encoder implemented and training converges (validation loss decreases for at least 10 epochs)
- Embedding dimension set to 768 (BERT/RoBERTa standard)
- Retrieval evaluation: top-1 accuracy $> 60\%$, top-5 accuracy $> 85\%$ on validation set
- t-SNE visualization created showing clear clustering of similar organizations and recommendations
- Ablation study completed: contrastive loss vs. triplet loss vs. InfoNCE (document best-performing variant)
- Fine-tuned encoders saved with documented hyperparameters

3.3. Graph Neural Networks for Influence Propagation

Objective: Model how information and influence propagate through organizational authority graphs over time. The GNN learns to aggregate information from neighboring agents weighted by learned attention, capturing both structural properties (who influences whom) and dynamic properties (how responses evolve through organizational levels).

Tasks:

Implement Graph Attention Network (GAT) or GraphSAGE using PyTorch Geometric (documentation). Reference implementations:

- Graph Attention Networks (Veličković et al., 2018)
- Enhance Information Propagation for GNNs (Yang et al., 2021)

Design GNN architecture: 4 layers, 8 attention heads per layer, 256 hidden dimensions, ELU activations, residual connections between layers.

Implement temporal unrolling: process $T = 3$ time steps where each step aggregates information from previous step's hidden states, modeling cascade dynamics.

Add node features to GNN input: concatenate [agent embedding (768-d from text encoder), power level (1-d), hierarchy level (1-d), previous hidden state (256-d)].

Train GNN jointly with encoders, monitoring: attention weight distributions (check for concentration on high-influence agents), hidden state evolution (measure temporal consistency), and gradient flow (check for vanishing gradients in deep GNN).

Completion Metrics:

- GNN module implemented and integrates with dual encoders without errors
- Forward pass on sample batch (32 organizations) completes in < 1 second on GPU
- Attention weights visualized: qualitative check shows higher attention to higher-power neighbors
- Temporal dynamics verified: hidden states evolve across time steps (correlation between t and $t + 1$ is 0.6-0.8, not 1.0)
- Ablation study: 2-layer vs. 4-layer vs. 6-layer GNN (document depth vs. performance tradeoff)
- Ablation study: GAT vs. GraphSAGE vs. GCN (document best-performing architecture)
- Gradient norms logged and stable (no vanishing/exploding gradients)

3.4. Hierarchical Policy Network

Objective: Predict individual agent response distributions conditioned on organizational context, recommendation, neighbor influences, and higher-level agent decisions. The hierarchical conditioning ensures lower-level agents' predictions respect organizational authority structure and capture strategic interaction (agents anticipate and respond to leaders' positions).

Tasks:

Design policy network architecture: MLP with 3 layers ($2048 \to 1024 \to 512 \to 5$), ReLU activations, dropout 0.2, layer normalization.

Implement hierarchical conditioning: for each agent i at level ℓ, concatenate to input all predictions from agents at levels $< \ell$. Use masking to handle variable numbers of higher-level agents.

Alternative exploration: implement hierarchical attention mechanism where lower-level agents attend over higher-level agents' hidden states rather than hard conditioning.

Add agent-specific features to input: one-hot role encoding, continuous power level, department embeddings (if available).

Output 5-class probability distribution via softmax: strongly oppose, oppose, neutral, support, strongly support.

Implement prediction rollout: predict level-by-level starting from highest hierarchy level, feeding predictions into next level's inputs.

Completion Metrics:

- Hierarchical policy network implemented and tested on sample data
- Forward pass correctly handles variable organization sizes and hierarchy depths
- Hierarchical conditioning verified: ablation shows removing higher-level inputs degrades lower-level prediction accuracy by $> 10\%$
- Policy network predictions sum to 1.0 and are non-negative (valid probability distributions)
- Hierarchical attention variant implemented for comparison (if time permits)
- Qualitative analysis: predicted responses show hierarchical cascade patterns (higher-level support predicts lower-level support)

3.5. Full Model Integration and Training

Objective: Integrate all components into a unified end-to-end model and train using multi-objective loss to simultaneously optimize prediction accuracy and game-theoretic properties. Successful training yields a model that can predict organizational responses to novel recommendations while exhibiting equilibrium-consistent behavior.

Tasks:

Integrate dual encoders, GNN, and hierarchical policies into single PyTorch nn.Module with documented forward pass.

Implement multi-objective loss function:

$$\mathcal{L} = \mathcal{L}_{pred} + 0.1\mathcal{L}_{consistency} + 0.05\mathcal{L}_{equilibrium}$$

where prediction loss is cross-entropy, consistency loss penalizes temporal discontinuities, and equilibrium loss encourages best-response properties.

Set up training loop with AdamW optimizer (learning rate 1e-4, weight decay 1e-5), linear warmup (5000 steps), cosine annealing schedule, gradient clipping (max norm 1.0), and early stopping (patience 5 epochs).

Configure logging: loss components, accuracy metrics, learning rate, gradient norms, sample predictions. Use TensorBoard or Weights & Biases for visualization.

Train on full dataset (10k organizations) for up to 50 epochs, monitoring validation accuracy and loss. Checkpoint best model by validation accuracy.

Implement distributed training (DataParallel or DistributedDataParallel) if using multiple GPUs to accelerate training.

Completion Metrics:

- Full model forward pass runs without errors on full batch size (32 organizations)
- Training converges: validation loss decreases for > 20 epochs, validation accuracy plateaus
- Target performance achieved: validation accuracy $> 75\%$, macro F1 > 0.70
- Training completes within 48 hours on 4x A100 GPUs (or equivalent compute)
- Best model checkpoint saved with validation accuracy, loss, and hyperparameters documented
- Training curves visualized showing smooth convergence (no instabilities or divergence)
- All loss components contribute: removing any component degrades performance by $> 3\%$

4. Baseline and Advanced Model Comparison

Objective: Rigorously compare our proposed hierarchical model against established MARL algorithms and architectural variants to validate that our design choices (hierarchical conditioning, graph attention, dual encoders) yield meaningful improvements. This establishes the scientific contribution and identifies which components drive performance.

Tasks:

Reproduce or leverage classic MARL baselines using open-source implementations:

- Independent Q-learning (agents learn without coordination)
- Centralized critic methods (QMIX, COMA) from MARLlib (documentation)
- Mean field MARL approximation (Yang et al., ICML 2018)
- Standard GNN without hierarchical structure or temporal dynamics
- Dual encoders + MLP (no graph component)
- Full model without equilibrium regularization

Train all models on identical data splits with consistent hyperparameters where applicable (learning rate, batch size, epochs).

Evaluate all models on test set using comprehensive metrics: individual agent accuracy, organizational outcome accuracy, per-level accuracy, consensus time prediction, attention-influence correlation (for GNN models).

Conduct statistical significance tests (paired t-test or Wilcoxon signed-rank) to verify improvements are not due to random variation.

Completion Metrics:

- At least 5 baseline/variant models implemented and trained to convergence
- Comparison table created with all models' performance on test set across > 8 metrics
- Statistical tests completed: our full model significantly outperforms ($p < 0.01$) all baselines on primary metric (agent accuracy)
- Ablation results documented: removing hierarchical conditioning reduces accuracy by $> 8\%$, removing GNN reduces accuracy by $> 10\%$
- Analysis document: identify which organizational types/sizes each model excels at
- Computational cost comparison: inference time and parameter count for each model

5. Evaluation and Analysis Tools

Objective: Develop comprehensive evaluation infrastructure to assess model performance across multiple dimensions: predictive accuracy, game-theoretic properties, influence propagation fidelity, generalization, and interpretability. These tools enable hypothesis testing and provide evidence for research claims about organizational response dynamics.

Tasks:

Implement standard classification metrics: accuracy, precision, recall, F1 (macro and per-class), confusion matrices for agent-level and organizational-level predictions, stratified by hierarchy level and organization size.

Create influence-flow analysis tools: compute Spearman correlation between learned GAT attention weights and ground-truth influence indicators (from ABM or human annotation), visualize attention heatmaps overlaid on organizational charts, identify most influential agents by incoming attention weights.

Develop equilibrium analysis scripts: measure exploitability (maximum gain from unilateral deviation), compute best-response deviation rate, calculate equilibrium gap (utility under predicted strategies vs. equilibrium strategies), test Stackelberg constraint satisfaction (lower-level responses optimal given higher-level actions).

Implement generalization evaluation: cross-domain tests (train on 4 industries, test on 5th), cross-size tests (train on medium orgs, test on small/large), cross-culture tests, measure domain shift via embedding space visualization.

Build interpretability tools:

- SHAP analysis (GitHub repo) for feature importance across recommendation features, agent features, and contextual features
- Attention weight visualization showing information flow through org chart over time steps
- Counterfactual generation: vary individual features (power level, urgency, culture) and measure prediction sensitivity

Implement adversarial robustness tests: synonym replacement in text (using NLP augmentation libraries), graph edge perturbations (add/remove random edges), agent feature noise (Gaussian noise on power levels).

Completion Metrics:

- Evaluation script runs on full test set in < 5 minutes and outputs comprehensive report with all metrics
- Influence-flow analysis shows significant correlation ($\rho > 0.6$) between learned attention and ground-truth influence
- Equilibrium analysis: exploitability $< 15\%$, best-response deviation rate $< 25\%$
- Generalization results: cross-domain accuracy drop $< 10\%$ relative to in-domain performance
- SHAP analysis completed with feature importance rankings documented (top 10 features identified)
- Attention visualization notebook created with interactive plots for > 20 sample organizations
- Counterfactual analysis: measure sensitivity to each feature type (documented in report)
- Adversarial robustness: accuracy drop $< 15\%$ under moderate perturbations (20% text changes, 10% edge changes)

6. Hypothesis Testing and Experimental Validation

Objective: Test specific research hypotheses about organizational influence dynamics using controlled experiments on our trained model and datasets. This moves beyond generic performance metrics to answer targeted questions about how organizational structure, culture, and power dynamics affect response propagation.

Tasks:

Test Hypothesis 1 (Hierarchical Cascades): Measure correlation between higher-level and lower-level agent predictions across hierarchy levels. Plot cascade strength vs. hierarchy depth. Statistical test: is correlation significantly higher for adjacent levels vs. distant levels?

Test Hypothesis 2 (Centralization and Consensus): Compute network centralization metrics for each organization's authority graph. Correlate with predicted consensus time (time steps to stable collective decision). Expected: higher centralization $\rightarrow$ faster consensus.

Test Hypothesis 3 (Culture Alignment): Generate synthetic data varying culture-recommendation alignment systematically. Measure predicted support uniformity (variance in response distribution) vs. alignment score. Expected: high alignment $\rightarrow$ low variance (uniform support/oppose).

Test Hypothesis 4 (Power Moderation): Analyze interaction effects between agent power level and neighbor influence strength. Regression analysis: does high power reduce sensitivity to peer influence? Expected: high-power agents weight peers less.

For each hypothesis, generate visualizations (scatter plots, regression lines, distribution plots), conduct statistical tests (correlation tests, t-tests, ANOVA), and document effect sizes.

Completion Metrics:

- H1 tested: correlation between adjacent levels > 0.7, significantly higher ($p < 0.01$) than distant levels
- H2 tested: centralization-consensus correlation $\rho > 0.5$, $p < 0.01$
- H3 tested: alignment-uniformity correlation $\rho > 0.6$, $p < 0.01$
- H4 tested: power-influence interaction significant ($p < 0.05$), effect size > 0.3
- All hypotheses documented in report with visualizations, statistical tests, and interpretation
- At least 3 out of 4 hypotheses confirmed (results align with predictions)
- Negative results (if any) analyzed and explained (potential model limitations or incorrect assumptions)

7. Real-World Validation

Objective: Validate model predictions against real organizational case studies to assess ecological validity and practical applicability. This tests whether patterns learned from synthetic data

generalize to actual organizational behavior and whether the model provides useful predictions for real-world scenarios.

Tasks:

Select 50-100 high-quality real cases with detailed documentation of organizational context, recommendation, and actual responses (from Harvard Business Review, Stanford cases, or collected data).

For each case, encode organizational description and recommendation using our dual encoders, construct authority graph from org chart, and run model inference to predict agent responses.

Compare predictions to documented actual responses: compute accuracy metrics, identify systematic biases (over-predicting support? under-predicting opposition?), analyze failure modes (what types of cases does model fail on?).

Conduct expert evaluation: recruit 2-3 organizational behavior researchers or experienced managers to review 20 sample predictions and rate plausibility on 1-5 scale.

Perform fine-tuning experiment: fine-tune model on real cases (if sufficient data), measure performance improvement, assess whether synthetic pre-training provides useful initialization.

Qualitative case study analysis: select 5 cases where model succeeded and 5 where it failed, analyze in depth to understand model strengths and limitations.

Completion Metrics:

- Model predictions generated for all 50-100 real cases
- Real-case accuracy $> 60\%$ (lower than synthetic test set but well above baselines)
- Expert evaluation: average plausibility rating $> 3.5/5.0$, inter-rater agreement (Krippendorff's alpha) > 0.6
- Fine-tuning on real cases improves accuracy by $> 5\%$ (validates transfer learning)
- Failure mode analysis completed: document top 3 failure patterns (e.g., misses informal influence, fails on crisis scenarios)
- Case study write-ups completed for 10 organizations (5 successes, 5 failures) with detailed analysis
- Validation report documents model limitations and applicability boundaries

8. Documentation and Code Management

Objective: Ensure project reproducibility, maintainability, and accessibility for future work or external review. High-quality documentation enables others to understand, use, and extend our work, which is critical for scientific contribution and potential real-world deployment.

Tasks:

Write comprehensive README with: project overview, installation instructions, dataset download/generation instructions, training commands, evaluation commands, and expected results.

Document dataset schemas: JSON structure for organizational scenarios, CSV format for agent responses, graph representation formats, text encoding specifications.

Create API documentation for all modules: data loaders, model architectures, training utilities, evaluation functions. Use docstrings (NumPy or Google style) and generate HTML docs with Sphinx.

Develop Jupyter notebooks for: data exploration and visualization, model training walkthrough, inference and prediction examples, attention visualization and interpretability analysis, reproducing key experimental results.

Establish code quality standards: PEP 8 formatting (use Black or autopep8), type hints for function signatures, unit tests for critical functions (data loading, model forward pass), integration tests for training pipeline.

Create requirements.txt or environment.yml with exact package versions for reproducibility.

Set up continuous integration (GitHub Actions) for automated testing on commits.

Completion Metrics:

- README written with all sections complete (> 1000 words)
- Dataset schema documentation complete with examples
- API documentation generated and hosted (ReadTheDocs or GitHub Pages)
- At least 5 Jupyter notebooks created covering different aspects of the project
- Code coverage $> 60\%$ for unit tests (measured by pytest-cov)
- All notebooks run end-to-end without errors on fresh environment
- External team member (not on project) can install, run training, and reproduce key result following README
- Code quality: no PEP 8 violations (checked by flake8), all functions have docstrings

9. Final Reporting and Presentation

Objective: Synthesize all project work into a clear, compelling, and scientifically rigorous final report and presentation that communicates contributions, findings, and limitations to academic and practitioner audiences.

Tasks:

Write final report (15-25 pages) with structure: abstract, introduction (motivation and research questions), related work (literature review), method (model architecture and training), experiments (hypotheses, datasets, results), analysis (ablations, interpretability, case studies), discussion (implications, limitations, future work), conclusion, references.

Create comprehensive results section with: performance comparison tables, learning curves, hypothesis testing results with statistics, attention visualizations, case study narratives, failure mode analysis.

Develop slide deck (20-30 slides) for presentation covering: problem motivation, key technical ideas (dual encoders, GNN, hierarchical conditioning), main results and visualizations, case study examples, lessons learned and future directions.

Prepare supplementary materials: appendix with full hyperparameters, additional experimental results, extended case studies, dataset statistics.

Create demo (if time permits): interactive web interface where users can input organizational description and recommendation, see predicted responses visualized on org chart.

Package trained model weights and upload to hosting (Google Drive, Hugging Face Hub) with download instructions.

Record 10-minute presentation video summarizing project (for asynchronous review if needed).

Completion Metrics:

- Final report completed: 15-25 pages, all sections present, $>$ 20 references cited
- Report follows academic writing standards: clear structure, no grammatical errors, figures/tables properly captioned
- Slide deck completed: 20-30 slides with clear narrative arc and compelling visualizations
- Supplementary materials compiled: appendix, extended results, hyperparameter tables
- Model weights uploaded and accessible via public link (with download instructions in README)
- Presentation video recorded: 10 minutes, covers key contributions and results
- At least 2 rounds of internal peer review completed with revisions incorporated
- Final submission package includes: report PDF, code repository link, slide deck, weights link, video link

Appendix: Key Resources

Datasets and Data Sources:

- Teamwork decision-making dataset

- UCSD recommendation datasets
- CaseRec datasets

Pre-trained Models:

- HuggingFace Model Hub (BERT, RoBERTa)
- Sentence-BERT for embeddings

Software Libraries:

- PyTorch
- PyTorch Geometric
- HuggingFace Transformers
- Mesa (agent-based modeling)
- NetworkX (graph library)
- SHAP (interpretability)

Learning Resources:

- LLM synthetic data generation guide
- DeepLearning.AI embedding models course
- MARLlib documentation