EM626 Project 2 - Midterm Presentation

LMS Integration Risk Analysis Using Hybrid LLM+GNN Pipeline

Team: [Your Names]

Date: [Presentation Date]

Duration: 5-7 minutes

Slide 1: Title

```
┌─────────────────────────────────────────────────────┐
│                                                     │
│  Knowledge Graphs for System Integration Risk       │
│                                                     │
│     LMS Architecture & AWS Dependencies             │
│                                                     │
│         EM626 - AI/ML for Systems                   │
│            Project 2                                │
│                                                     │
│         [Team Member Names]                         │
│         [Date, 2025]                                │
│                                                     │
└─────────────────────────────────────────────────────┘
```

Speaker Notes:

- Introduce team members
- State project choice: Project 2 - Integration Risk Analysis
- Focus: Learning Management System on AWS

Slide 2: Motivation - The AWS Outage Problem

Visual: Timeline graphic of Dec 7, 2021 AWS outage

Content:

- **December 7, 2021:** AWS us-east-1 region outage
- **Duration:** ~5 hours

- **Impact:**
  - Major education platforms disrupted
  - Canvas LMS experienced downtime
  - Millions of students unable to access courses
  - Final exams postponed at multiple universities
- **Root Cause:** Network device issues cascading through services

Key Question:

> "Which components in a cloud-based LMS pose the highest integration risk during infrastructure failures?"

Speaker Notes:

- Real-world context establishes importance
- LMS systems are mission-critical for education
- Single points of failure can have cascading effects
- Our project aims to identify and predict these risks

Slide 3: Project Goals

Objectives:

1. **Model** a realistic LMS architecture on AWS (22 components)
2. **Extract** components and dependencies using LLM automation
3. **Construct** a knowledge graph of the system
4. **Analyze** integration risks using graph metrics
5. **Predict** high-risk components using GNN (post-midterm)

Expected Outcomes:

- Identify critical components beyond obvious candidates
- Understand failure cascade patterns
- Provide design recommendations for resilience

Speaker Notes:

- Emphasize hybrid approach: LLM + Graph + GNN
- This combines symbolic (graph structure) and neural reasoning
- Applicable beyond LMS to any cloud-dependent system

Slide 4: Pipeline Architecture

Visual: Your pipeline diagram (from the HTML artifact)

Components:

1. System Description → LLM Extractor → Graph Builder → Features → Risk Labels
2. Split path: Baseline Model + GNN Model
3. Visualization & Report

Completed (Midterm): Stages 1-4 ✅ **In Progress:** Stages 5-8 ⏳

Speaker Notes:

- Walk through each stage briefly
- Point out what's done vs. what's next
- Highlight the LLM + GNN hybrid approach

Slide 5: System Architecture Overview

Visual: Table or diagram of 22 components

Component Categories:

- **Application Services** (8): API Gateway, Auth, User Mgmt, Course Mgmt, Content Delivery, Video, Assignment, Assessment
- **Infrastructure** (6): Load Balancer, Database (RDS), Cache (ElastiCache), Message Queue (SQS), Monitoring (CloudWatch), Backup
- **Storage** (3): S3 Object Storage, Data Warehouse (Redshift), Search (OpenSearch)
- **Network** (3): CloudFront CDN, VPC Networking, DNS
- **Security** (2): WAF/Shield, IAM/Cognito

Key Dependencies:

- API Gateway → routes all service calls
- RDS Database → stores all core data
- S3 Storage → holds all content
- Authentication → gates all access

Speaker Notes:

- This is a realistic modern LMS architecture
- Mix of custom services and AWS-managed infrastructure
- Deliberately includes potential single points of failure

Slide 6: LLM-Based Component Extraction

Approach:

```
Input: Free-text system description
 ↓
LLM Prompt (Claude Sonnet 4.5)
 ↓
Structured JSON Output
 ↓
Validation Loop
```

Prompt Engineering:

- **Model:** Claude Sonnet 4.5
- **Temperature:** 0 (deterministic)
- **Output Format:** JSON schema with components, dependencies, metadata
- **Validation:** Secondary prompt checks for errors

Results:

- ✅ Successfully extracted 22 components
- ✅ Identified 38 dependency relationships
- ✅ Validated all references (no dangling edges)

Sample Output:

json

```json
{
  "id": "api_gateway",
  "name": "API Gateway Service",
  "type": "infrastructure",
  "technology": "AWS API Gateway",
  "criticality": "Critical",
  "dependencies": ["auth_service", "course_service", ...]
}
```

Speaker Notes:

- LLM automates what would be manual system analysis
- Structured output enables graph construction
- Validation ensures consistency
- Prompts will be included in final report appendix

Slide 7: Graph Construction & Visualization

Visual: Your `system_graph.png` visualization

Graph Properties:

- **Type:** Directed Acyclic Graph (DAG)
- **Nodes:** 22 (components)
- **Edges:** 38 (dependencies)
- **Density:** 0.17 (moderately connected)
- **Direction:** A→B means "A depends on B"

Node Encoding:

- **Size:** In-degree (how many depend on it)
- **Color:** Component type
- **Position:** Spring layout for clarity

Interpretation:

- Larger nodes = more components depend on them = higher risk
- Central nodes = bottlenecks in the system

Speaker Notes:

- Point out visually prominent nodes (API Gateway, RDS)
- Explain that size reflects dependency count
- This visualization helps identify critical components at a glance

---

Slide 8: Preliminary Results - Critical Components

Top 5 Critical Components (by composite score):

Rank	Component	In-Deg	Out-Deg	Betweenness	Criticality
1	API Gateway	12	3	0.45	Critical
2	RDS Database	15	2	0.38	Critical
3	Authentication Service	8	4	0.32	Critical
4	S3 Storage	10	1	0.28	Critical
5	Load Balancer	6	5	0.25	High

Key Insights:

- **API Gateway:** Single routing point for all services
- **RDS Database:** Central data store for 15+ services
- **High betweenness** = bottleneck in information flow

Failure Cascade Example:

- If RDS fails → 15 services lose data access → ~68% system failure

Speaker Notes:

- These results match intuition (API Gateway, Database)
- But quantitative metrics provide evidence
- Betweenness reveals non-obvious bottlenecks
- This sets up the GNN training targets

Slide 9: Graph Features Computed

7-Dimensional Feature Vector per node:

1. **In-Degree:** Components depending on this node
2. **Out-Degree:** Components this node depends on
3. **Total Degree:** Sum of in + out
4. **Betweenness Centrality:** Frequency on shortest paths (bottleneck measure)
5. **Closeness Centrality:** Average distance to all other nodes
6. **PageRank:** Structural importance (Google's algorithm)
7. **Clustering Coefficient:** Local redundancy/connectivity

Purpose: These features feed into:

- Heuristic risk labeling (next step)
- Baseline model (logistic regression)
- GNN model (learning from graph structure)

Example - API Gateway:

```
In-Degree: 12  (high)
Betweenness: 0.45  (very high - major bottleneck)
Clustering: 0.12  (low - not redundant)
→ HIGH RISK classification
```

Speaker Notes:

- These are standard graph-theoretic measures
- Chosen based on literature on system reliability
- Will serve as both features and validation metrics

Slide 10: Next Steps - Post-Midterm Work

Week 1-2: Risk Labeling & Baseline

- Generate synthetic labels using heuristic rules:
  - High risk: in-degree > 5 OR betweenness > 0.3

    - Medium risk: in-degree 2-5 OR betweenness 0.1-0.3
    - Low risk: remaining nodes
- Validate labels with domain expertise (team SMEs)
- Train baseline: Logistic Regression on features

Week 2-3: GNN Development

- Implement Graph Convolutional Network (GCN)
- Framework: PyTorch Geometric
- Architecture: 2-layer GCN, 16 hidden dimensions
- Task: 3-class node classification (low/medium/high risk)
- Training: 80/20 split, cross-entropy loss

Week 3-4: Analysis & Visualization

- Compare GNN vs. Baseline performance
- Failure cascade simulations
- Risk heatmaps on network graph
- Feature importance analysis

Week 4-5: Integration & Report

- Single executable pipeline script
- 10-page technical report
- Final presentation

Speaker Notes:

- Clear roadmap for remaining work
- GNN is the core ML component (coming soon)
- Timeline is aggressive but achievable

Slide 11: Anticipated Evaluation Metrics

Model Performance:

- **Accuracy:** Overall classification rate (target: >75%)
- **F1-Score:** Especially for high-risk class (imbalanced data)
- **Confusion Matrix:** Where does the model make errors?
- **ROC-AUC:** Multi-class performance curves

System-Level Analysis:

- **Cascade Size:** % of system affected by each component's failure
- **Comparison with Real Outage:** Validate against AWS Dec 2021 impact
- **Design Recommendations:** Where to add redundancy?

Model Comparison:

- GNN vs. Baseline (Logistic Regression)
- GNN vs. Heuristics (hand-crafted rules)
- Accuracy vs. Interpretability trade-offs

Speaker Notes:

- Multiple evaluation dimensions: ML metrics + domain validation
- Real-world validation crucial for credibility
- Comparison baselines ensure we're adding value with GNN

Slide 12: Challenges & Mitigation

Challenge 1: Small Dataset

- Only 22 nodes → limited training data
- **Mitigation:** k-fold cross-validation, simple GNN architecture

Challenge 2: Class Imbalance

- Likely more high-risk nodes than low-risk
- **Mitigation:** Weighted loss function, SMOTE oversampling

Challenge 3: LLM Extraction Errors

- LLM might miss dependencies or hallucinate

- **Mitigation:** Validation prompts, manual review

Challenge 4: Time Constraints

- 4 weeks for GNN development + report
- **Mitigation:** Start GNN training early, parallel work on visualization

Speaker Notes:

- Being proactive about challenges shows maturity
- Have concrete mitigation plans
- These are realistic concerns for ML projects

Slide 13: Expected Contributions

Technical Contributions:

1. **Automated pipeline** for system risk analysis using LLM+GNN
2. **Hybrid reasoning** approach combining symbolic (graph) and neural
3. **Quantitative risk assessment** for cloud-dependent systems

Domain Insights:

1. Identification of **non-obvious critical components**
2. Understanding of **cascading failure patterns** in LMS
3. **Design recommendations** for improving resilience:
    - Multi-region database replication
    - Redundant API gateway instances
    - Graceful degradation for non-critical services

Broader Impact:

- Applicable to other cloud systems (e-commerce, fintech, healthcare)
- Demonstrates value of AI for system reliability engineering
- Combines SE best practices with modern ML

Speaker Notes:

- Position project as both technical and practical contribution
- Emphasize generalizability beyond LMS
- Connect to course themes: AI/ML + Systems Thinking

Slide 14: Questions for Instructor

1. Dataset Size: With only 22 nodes, is k-fold cross-validation sufficient, or should we explore synthetic graph generation techniques?

2. GNN Architecture: Given the small graph, should we use a simpler model (1-layer GraphSAGE) instead of multi-layer GCN to avoid overfitting?

3. Evaluation Approach: Can qualitative validation against the AWS Dec 2021 outage serve as a substitute for additional test data?

4. Scope Clarification: Is node classification (risk levels) sufficient, or should we also explore link prediction (missing dependencies)?

Speaker Notes:

- These are genuine questions to get guidance
- Shows we're thinking critically about methodology
- Open to instructor input on approach

Slide 15: Summary & Timeline

Completed ✅:

- System architecture documentation (22 components)
- LLM extraction pipeline with validation
- Dependency graph construction (NetworkX)
- Feature computation (7D vector per node)
- Preliminary critical component identification

In Progress ⏳:

- Risk labeling (Week 1)

- GNN model development (Week 2-3)
- Performance evaluation & visualization (Week 3-4)

Timeline:

- **Nov 4-10:** Risk labels + baseline
- **Nov 11-17:** GNN training
- **Nov 18-24:** Analysis + draft report
- **Nov 25-Dec 1:** Integration + final report
- **Dec 2-8:** Presentation

Contact: [your-email@university.edu]

Speaker Notes:

- Recap what's done (substantial progress!)
- Clear path forward
- Confident in meeting final deadlines
- Thank audience and open for questions

Backup Slides (Optional)

Backup 1: LLM Prompt Example

```
System Prompt:
"You are a system architecture analyzer. Extract components
and dependencies from the provided system description.

For each component, identify:
1. Component name (unique identifier)
2. Component type (application, infrastructure, storage,
   network, security)
3. Technology/service used
4. Primary purpose
5. Criticality level (Critical, High, Medium, Low)
6. Dependencies (list of other component names this depends on)

Output format must be valid JSON..."
```

Backup 2: Feature Definitions

- **Betweenness Centrality:** Fraction of shortest paths passing through node
- **PageRank:** Probability of reaching node via random walk
- **Clustering Coefficient:** (# triangles) / (# possible triangles)

Backup 3: Related Work

- Graph Neural Networks for system reliability (Kipf & Welling, 2017)
- Failure prediction in distributed systems (Google SRE)
- Knowledge graphs for enterprise architecture (Gartner)

Presentation Delivery Tips

Timing (5-7 minutes):

- Slides 1-3 (Intro): 1 min
- Slides 4-7 (Approach): 2 min
- Slides 8-9 (Results): 1.5 min
- Slides 10-13 (Next Steps): 1.5 min
- Slides 14-15 (Q&A): 1 min

What to Emphasize:

- Real-world motivation (AWS outage)
- Hybrid LLM+GNN approach (novel combination)
- Concrete preliminary results (critical components)
- Clear roadmap for completion

What to De-emphasize:

- Low-level code details (save for questions)
- Mathematical formulas (backup slides)
- Every single component (focus on top 5)

Visual Aids:

- Show pipeline diagram (Slide 4)
- Show graph visualization (Slide 7)
- Show results table (Slide 8)

Practice:

- Rehearse transitions between slides
- Time yourself (aim for 6 minutes)
- Prepare for common questions:
  - "Why LMS?" → Real-world relevance, AWS outage
  - "Why GNN?" → Captures graph structure better than flat features
  - "How confident in results?" → Preliminary but validated

Good luck! 🚀