رواد مصر الرقمية

وزارة الاتصالات
وتكنولوجيا المعلومات

# Intelligent IT Support Ticket Classification and Response System using RAG

*An End-to-End Production-Ready NLP System*

## Prepared by

| Student ID | Student Name |
|---|---|
| 21073603 | محمد إبراهيم سعد (Leader) |
| 21039291 | ايمان موسي محمود |
| 21070610 | هنا محمد وصفى |
| 21006901 | محمد أشرف محمد |
| 21129786 | احمد عبداللاه ابراهيم |
| 21056479 | أشرف معوض رمضان |

## Supervised By

Eng. Abdelrahman Elmashtoly

## Date

1/2026 To 6/2026

# Project Proposal

**1. Executive Summary**

This project aims to design and deploy an intelligent IT support automation system that classifies incoming support tickets and generates accurate, context-aware responses using Retrieval-Augmented Generation (RAG).

The system integrates natural language processing (NLP), vector search, and transformer-based large language models (LLMs) to retrieve relevant historical IT incidents and produce grounded responses.

The final solution will be deployed as a scalable cloud API on Microsoft Azure and monitored using MLOps practices to ensure reliability, accuracy, and continuous improvement.

---

**2. Project Objectives**

The primary objectives of this project are:

- Automatically classify IT support tickets into predefined categories
- Retrieve semantically similar historical IT issues from a knowledge base
- Generate context-aware technical responses grounded in retrieved data
- Deploy the system as a production-ready Azure API
- Implement monitoring, logging, and retraining pipelines

**3. Project Scope**

**In Scope**

- IT support ticket dataset ingestion and preprocessing
- Text embeddings generation and vector database indexing
- Retrieval-Augmented Generation (RAG) pipeline implementation
- Transformer-based classification and response generation
- Azure cloud deployment (API + vector search integration)
- Monitoring dashboard and retraining triggers

**Out of Scope**

- Real company proprietary data integration
- Multilingual support
- Voice or chatbot interface
- On-premise deployment

**4. System Overview**

The system processes IT support tickets through the following pipeline:

1. Ticket ingestion and preprocessing
2. Embedding generation
3. Vector similarity retrieval
4. Context injection into LLM
5. Response generation
6. API delivery to user

This architecture enables responses grounded in real IT incidents rather than generic model knowledge.

---

**5. Target Users**

- IT helpdesk teams
- Technical support agents
- Enterprise IT service desks
- Managed service providers (MSPs)

**6. Expected Deliverables**

By July 2026, the project will deliver:

- Cleaned and structured IT support ticket dataset
- Vector search index and embeddings repository
- Trained classification and RAG models
- Deployed Azure API endpoint
- Monitoring dashboard with KPIs
- Final technical documentation and presentation

---

**7. Technology Stack**

**NLP & ML**

- Python
- HuggingFace Transformers
- Sentence-Transformers
- FAISS / Azure Cognitive Search

**Backend & API**

- FastAPI
- REST APIs

**Cloud & MLOps**

- Microsoft Azure ML
- Azure Cognitive Search
- MLflow
- Monitoring Dashboard

## 8. Project Timeline

Project duration: **January 2026 → July 2026 (6 months)**
Milestones:

1. Data Collection & Preprocessing
2. Model Development (RAG)
3. Azure Deployment
4. MLOps & Monitoring
5. Final Documentation & Demo

---

## 9. Project Team

Team size: **6 members**

- محمد ابراهيم سعد (Leader)
- ايمان موسي محمود
- هنا محمد وصفى
- محمد أشرف محمد
- احمد عبداللاه ابراهيم
- أشرف معوض رمضان

---

## 10. Success Criteria

The project will be considered successful if:

- Ticket classification accuracy meets target threshold
- Retrieved contexts are relevant to user queries
- Generated responses are technically correct and grounded
- API latency meets real-time requirements
- System runs reliably in Azure cloud environment

# Project Plan

### 1. Official Project Timeline (DEPI)

| Phase | Period | Description |
|---|---|---|
| Project Planning & Management | Jan → 20 Feb 2026 | Proposal, plan, roles, risks, KPIs |
| Literature & Requirements | 21 Feb → 20 Apr 2026 | Research + requirements |
| System Analysis & Design | 21 Apr → 1 May 2026 | Architecture & diagrams |
| Implementation | 2 May → 10 Jul 2026 | RAG system development |
| Testing & Final Delivery | 11 Jul → 17 Jul 2026 | Testing + presentation |

---

### 2. Milestones & Deliverables (Aligned with DEPI)

**Milestone 1 — Project Planning & Management**
**Deadline:** 20 Feb 2026
Deliverables:

- Project Proposal
- Project Plan
- Roles & Tasks
- Risk Plan
- KPIs

---

**Milestone 2 — Literature Review & Requirements**
**Deadline:** 20 Apr 2026
Deliverables:

- RAG & NLP literature review
- Stakeholder analysis
- User stories
- Functional requirements
- Non-functional requirements

**Milestone 3 — System Analysis & Design**
**Deadline:** 1 May 2026
Deliverables:

- System architecture diagram
- Use case diagram
- Data flow diagram
- Component diagram
- Deployment diagram

---

**Milestone 4 — Implementation**
**Deadline:** 10 Jul 2026
Deliverables:

- Data preprocessing pipeline
- Embedding & vector DB
- RAG retrieval system
- LLM generation
- Azure deployment
- Source code (GitHub)

---

**Milestone 5 — Testing & Final Delivery**
**Deadline:** 17 Jul 2026
Deliverables:

- Test cases & results
- Performance evaluation
- Final report
- Presentation slides
- System demo

### 3. Student Gantt Chart (DEPI Schedule)

```
Month:  Jan   Feb   Mar   Apr   May   Jun   Jul
---------------------------------------------
Planning ████
Liter+Req   ███████
Design       ██
Implementation    █████████
Testing & Final     ██
```

---

### 4. Work Distribution (Even Across Students)

Since DEPI requires each member contribution:

- Every phase → all 6 members contribute
- Tasks divided equally
- Deliverables merged collaboratively

Example distribution per phase:

| Phase | Work Split |
|---|---|
| Planning | 1 doc per member |
| Literature | papers divided |
| Requirements | features divided |
| Design | diagrams divided |
| Implementation | modules divided |
| Testing | cases divided |

---

### 5. Phase Dependencies

DEPI workflow is sequential:
Planning → Requirements → Design → Implementation → Testing

### 6. Key Submission Dates (DEPI)

| Deliverable | Date |
|---|---|
| Planning Docs | 20 Feb 2026 |
| Literature & Requirements | 20 Apr 2026 |
| Design | 1 May 2026 |
| Implementation | 10 Jul 2026 |
| Final Presentation | 17 Jul 2026 |

---

# Task Assignment & Roles

### 1. Team Structure (XP Pairs)

| Pair | Members |
|---|---|
| Pair 1 | ايمان موسي محمود — هنا محمد وصفى |
| Pair 2 | محمد ابراهيم سعد — احمد عبداللاه ابراهيم |
| Pair 3 | محمد أشرف محمد — أشرف معوض رمضان |

Each pair works together on all tasks through continuous collaboration, code review, and shared ownership, following XP principles.

---

### 2. XP Role Model in Project

In Extreme Programming, roles rotate rather than being fixed.
Each pair alternates between:

- Driver (coding)
- Navigator (reviewing & guiding)

This ensures equal contribution and knowledge sharing.

## 3. Responsibilities by Pair (Project Phases)

### Pair 1 — Data & NLP Pipeline

**Members:** ايمان موسي محمود — هنا محمد وصفى
Responsibilities:

- Dataset collection & cleaning
- Text preprocessing
- Tokenization & normalization
- Embedding generation
- EDA analysis

Deliverables:

- Clean dataset
- Preprocessing scripts
- EDA report

---

### Pair 2 — RAG Retrieval & Modeling

**Members:** محمد ابراهيم سعد — احمد عبداللاه ابراهيم
Responsibilities:

- Vector database construction
- FAISS / Azure Cognitive Search indexing
- Similarity search implementation
- Ticket classification model
- Retrieval evaluation

Deliverables:

- Vector index
- Retrieval module
- Classification model

### Pair 3 — Generation, API & Deployment

**Members:** محمد أشرف محمد — أشرف معوض رمضان

Responsibilities:

- LLM prompt design
- RAG response generation
- FastAPI backend
- Azure deployment
- System integration

Deliverables:

- RAG generation module
- REST API
- Azure deployment

---

## 4. Shared Responsibilities (All Pairs)

All pairs collaborate on:

- System architecture design
- Integration testing
- Evaluation & metrics
- Documentation
- Final presentation

---

## 5. XP Practices Applied in Project

The team follows Extreme Programming practices:

- Pair programming
- Continuous integration
- Small iterative releases
- Refactoring
- Collective code ownership
- Frequent testing

**6. Workload Balance Assurance**

Balance is ensured by:

- Equal pair size (2 each)
- Rotating driver/navigator roles
- Cross-phase collaboration
- Shared final deliverables

Thus, all six members contribute equally to planning, development, and delivery.

---

# Risk Assessment & Mitigation Plan

**1. Risk Management Approach**

The team follows proactive risk management:

- Early identification
- Probability & impact evaluation
- Preventive mitigation
- Monitoring during milestones

Risks are reviewed at the end of each project phase.

---

**2. Risk Matrix**

| ID | Risk | Probability | Impact | Level |
|---|---|---|---|---|
| R1 | Poor data quality | Medium | High | High |
| R2 | Weak embedding/retrieval accuracy | Medium | High | High |
| R3 | Azure deployment complexity | Medium | Medium | Medium |
| R4 | LLM hallucination | Medium | High | High |
| R5 | Integration failures | Low | High | Medium |
| R6 | Uneven contribution in pairs | Low | Medium | Low |
| R7 | Time constraints before deadlines | Medium | High | High |
| R8 | System latency (slow response) | Medium | Medium | Medium |

### 3. Risk Details & Mitigation

**R1 — Poor Data Quality**
**Description:** IT tickets may contain noise, missing labels, or inconsistent text.
**Impact:** Weak training and retrieval performance.
**Mitigation:**

- Text cleaning pipeline
- Manual inspection samples
- Remove duplicates
- Normalize categories

**Owner:** Pair 1 (Data)

---

**R2 — Weak Retrieval Accuracy**
**Description:** Retrieved tickets may not match user query semantics.
**Impact:** Incorrect generated responses.
**Mitigation:**

- Use sentence-transformer embeddings
- Tune similarity metric & top-k
- Evaluate retrieval accuracy
- Compare models

**Owner:** Pair 2 (Retrieval)

---

**R3 — Azure Deployment Complexity**

**Description:** Cloud services integration may fail or be misconfigured.
**Impact:** System not deployable.
**Mitigation:**

- Local prototype first
- Use Azure tutorials & templates
- Incremental deployment
- Versioned configuration

**Owner:** Pair 3 (Deployment)

**R4 — LLM Hallucination**
**Description:** Model generates unsupported or incorrect IT advice.
**Impact:** Loss of reliability.
**Mitigation:**

- Strict RAG context grounding
- Limit generation temperature
- Include retrieved evidence
- Human validation samples

**Owner:** Pair 3 (Generation)

---

**R5 — Integration Failures**
**Description:** Retrieval, LLM, and API modules may not connect correctly.
**Impact:** End-to-end pipeline breaks.
**Mitigation:**

- Modular interfaces
- Integration testing early
- API contracts defined
- Continuous integration

**Owner:** All pairs

---

**R6 — Uneven Contribution in XP Pairs**
**Description:** One student may dominate coding tasks.
**Impact:** Unfair workload & grading issues.
**Mitigation:**

- Driver/navigator rotation
- Weekly role switching
- GitHub commit tracking
- Pair reviews

**Owner:** All pairs

**R7 — Time Constraints**
**Description:** Implementation may exceed DEPI deadlines.
**Impact:** Late submission.
**Mitigation:**

- Phase-based milestones
- MVP first approach
- Weekly progress checks
- Scope control

**Owner:** All pairs

---

**R8 — High Response Latency**
**Description:** RAG pipeline may respond slowly due to vector search or LLM.
**Impact:** Poor user experience.
**Mitigation:**

- Limit context size
- Optimize top-k
- Cache embeddings
- Measure response time KPI

**Owner:** Pair 2 & 3

---

### 4. Risk Monitoring Plan

Risks reviewed at:

- Phase completion meetings
- Before each DEPI submission
- Integration milestones

If risk level increases → mitigation updated.

**5. Contingency Strategy**

If major technical risk occurs:

- Switch to smaller embedding model
- Replace Azure with FAISS local
- Use simpler classifier baseline
- Reduce dataset size

Ensures project completion within deadline.

---

# Key Performance Indicators (KPIs)

**1. KPI Categories**

Project success is evaluated across five dimensions:

1. Retrieval Quality
2. Classification Accuracy
3. Response Generation Quality
4. System Performance
5. Project Delivery & Usability

---

**2. Retrieval KPIs (RAG Core)**

**KPI 1 — Top-K Retrieval Accuracy**
**Definition:** Percentage of queries where relevant ticket appears in top-K results
**Formula:**
Relevant_in_topK / Total_queries
**Target:** ≥ 80%
**Why:** Measures vector search effectiveness.

---

**KPI 2 — Similarity Score Quality**

**Definition:** Average cosine similarity between query and retrieved tickets
**Target:** ≥ 0.70
**Why:** Ensures semantic closeness of retrieved context.

### KPI 3 — Retrieval Latency

**Definition:** Time to retrieve top-K documents
**Target:** ≤ 300 ms
**Why:** Important for real-time API.

---

## 3. Classification KPIs

### KPI 4 — Ticket Classification Accuracy

**Definition:** Correct category predictions / total tickets
**Target:** ≥ 85%

---

### KPI 5 — Recall (Critical Classes)

**Definition:** Ability to detect key IT issue categories
**Target:** ≥ 80%
**Why:** Missing important IT issues is costly.

---

## 4. Generation KPIs (LLM + RAG)

### KPI 6 — Response Groundedness

**Definition:** % of responses supported by retrieved tickets
**Measurement:** Manual evaluation sample
**Target:** ≥ 85%
**Why:** Prevent hallucination.

---

### KPI 7 — Response Relevance Score

**Definition:** Human rating (1–5) of answer usefulness
**Target:** ≥ 4.0

---

### KPI 8 — BLEU / ROUGE (Optional)

**Definition:** Overlap with known resolutions
**Target:** Moderate (reference only)
**Note:** Human evaluation preferred for RAG text.

### 5. System Performance KPIs

**KPI 9 — End-to-End Response Time**
**Definition:** Time from ticket input → final response
**Target:** ≤ 2 seconds

**KPI 10 — API Availability**
**Definition:** % uptime of Azure endpoint
**Target:** ≥ 95%

**KPI 11 — System Stability**
**Definition:** Failure rate during requests
**Target:** ≤ 5% errors

### 6. Project Execution KPIs (DEPI)

**KPI 12 — Milestone Completion Rate**
**Definition:** Deliverables completed on deadline
**Target:** 100%

**KPI 13 — Code Contribution Balance**
**Definition:** Even GitHub commits across members
**Target:** Balanced across 6 students

**KPI 14 — Documentation Completeness**
**Definition:** All DEPI required documents submitted
**Target:** 100%

### 7. KPI Summary Table

| Category | KPI | Target |
|---|---|---|
| Retrieval | Top-K accuracy | ≥ 80% |
| Retrieval | Similarity | ≥ 0.70 |
| Classification | Accuracy | ≥ 85% |
| Classification | Recall | ≥ 80% |
| Generation | Groundedness | ≥ 85% |
| Generation | Relevance | ≥ 4/5 |
| Performance | Response time | ≤ 2 s |
| Performance | API uptime | ≥ 95% |
| Project | Milestones | 100% |
| Project | Contribution | Balanced |

---

### 8. KPI Measurement Plan

KPIs measured at:

- After model training
- After deployment
- Before final submission

Tools:

- sklearn metrics
- retrieval evaluation scripts
- latency logging
- human evaluation forms
- GitHub analytics