

MCA 3rd Semester Minor Project Proposal Document

This document contains **five ML-powered project proposals**, each fully detailed with:

- Abstract
 - Problem Statement
 - Objectives
 - System Architecture
 - Proposed Technology Stack
 - ML Pipeline
 - Workflows
 - Expected Outcomes
 - Future Enhancements
-

1. Interview Profiling ML Backend

Abstract

This project focuses on creating an AI-powered backend that performs **profiling of an interview candidate** using real-time **video + audio** data. The system extracts behavioral, emotional, linguistic, and confidence metrics to help interviewers evaluate candidates more effectively.

Problem Statement

Interviewers often follow inconsistent evaluation patterns. Human bias, lack of structure, and subjective judgment lead to unreliable hiring decisions. There is a need for a consistent, data-driven profiling system.

Objectives

- Process real-time video to detect facial expressions, attention, and micro-emotions
- Process audio to analyze tone, pace, sentiment
- Provide a profiling report of the candidate
- Deliver API-based output usable by any frontend

System Architecture

- **Frontend:** Web/video stream sender
- **Backend:** ML engine for video/audio processing
- **Modules:**
 - Face Detection
 - Emotion Recognition
 - Eye Contact Analysis

- Audio Sentiment
- Speech-to-Text
- Confidence Scoring
- **Database:** Store scoring logs for later audit
- **Output:** JSON profile report

Technology Stack

- Python (FastAPI/Flask)
- OpenCV + Mediapipe
- TensorFlow/PyTorch models
- Speech-to-text (Vosk/Whisper local)
- Scikit-learn for scoring

ML Pipeline

1. Frame capture
2. Face landmark detection
3. Emotion inference
4. Eye contact metrics
5. Audio MFCC extraction
6. Sentiment + tone analysis
7. Score aggregation

Expected Output

A coherent profiling dashboard containing:

- Confidence score
- Stress/Anxiety measure
- Communication clarity
- Facial expression chart
- Engagement timeline

Future Enhancements

- Bias detection
- Real-time interviewer recommendation

2. System Log Anomaly Detection (ML-Based)

Abstract

A machine learning system designed to analyze system logs and detect suspicious or abnormal activity. Helps in early detection of attacks, malware, or performance issues.

Problem Statement

System logs generate huge volumes of data. Humans cannot manually detect anomalies. Traditional rule-based detection fails on unknown patterns.

Objectives

- Parse multiple log formats
- Detect unusual login attempts
- Flag suspicious resource usage patterns
- Provide an anomaly score

System Architecture

- Log Parser → Feature Extractor → ML Model → Alert System → Dashboard

Technology Stack

- Python
- Pandas, NumPy
- Scikit-Learn (Isolation Forest, One-Class SVM)
- Elasticsearch / Simple SQLite
- FastAPI

ML Pipeline

1. Log ingestion
2. Tokenization
3. Feature engineering
4. Outlier model training
5. Real-time anomaly scoring

Expected Output

- Alerts for high-risk events
- Log clusters (normal vs abnormal)
- Heatmap of anomalies

Future Enhancements

- Integrate with firewalls
- Live Linux kernel alerts via eBPF

3. Micro AI Router (Query Analyzer + Best LLM Selector)

Abstract

This system acts as a "micro-AI" layer that receives a user query, analyzes it, and decides which major LLM (GPT, Claude, Mistral, Llama) is best suited. It also rewrites the query for optimal results.

Problem Statement

Most users input unclear or unoptimized queries. Different LLMs excel at different tasks. A routing layer is needed.

Objectives

- Classify the query type (code, math, logic, creative)
- Select the best LLM
- Rewrite the prompt to improve output
- Provide response quality prediction

System Architecture

- Input → Intent Classifier → LLM Selector → Prompt Rewriter → API Caller → Output

Technology Stack

- Python
- MiniLM / DistilBERT
- FastAPI
- Vector embeddings
- Router logic

ML Pipeline

1. Text embedding
2. Query classification
3. Prompt optimization
4. Confidence scoring

Expected Output

- Optimized prompt
- Selected LLM
- Final response delivered neatly

Future Enhancements

- Latency-based LLM selection
- Cost-optimized routing

4. AI-Generated Survey System + Answer Validity Analyzer

Abstract

A web-based survey creator that uses AI to auto-generate relevant survey questions and then analyze user responses for genuineness, consistency, and randomness.

Problem Statement

Surveys often have:

- Repetitive or poorly structured questions
- Users giving random or dishonest answers

Objectives

- Auto-generate context-aware survey questionnaires
- Analyze user responses using ML
- Detect inconsistent/fake answers
- Provide survey insights

System Architecture

- Survey Generator → Response Collector → ML Analyzer → Insights Dashboard

Technology Stack

- React/Next.js frontend
- Python FastAPI backend
- ML models:
 - LLM for question generation
 - Scikit-Learn for answer anomaly detection

ML Pipeline

1. Generate questions using LLM
2. Encode user answers
3. Detect randomness/inconsistency
4. Generate authenticity score

Expected Output

- Genuine answer percentage
- Individual inconsistency score
- Survey-level insights

Future Enhancements

- Multi-lingual survey generation
- Bias-aware question generation

5. AI-Based Notes Search for Zettelkasten System

Abstract

This project provides an AI-powered semantic search system for Zettelkasten-style note-taking. Instead of keyword search, it understands context and relationships.

Problem Statement

Zettelkasten notes are highly linked and conceptual. Traditional search fails to retrieve meaningful connections.

Objectives

- Semantic search for notes
- Topic clustering
- Auto-tagging
- Relationship detection between notes

System Architecture

- Note Parser → Embedding Model → Vector Store → Search Engine → Result Ranking

Technology Stack

- Python backend
- Sentence-BERT embeddings
- FAISS vector index
- Markdown note parser
- Next.js frontend (optional)

ML Pipeline

1. Extract text from notes
2. Generate embeddings
3. Store vectors in FAISS
4. Perform vector similarity search

Expected Output

- Ranked semantic search results
- Concept linking suggestions
- Auto-tag recommendations

Future Enhancements

- Full Zettelkasten graph visualization
- AI-based note generation and linking

