Educational AI Platform - Intern Project Documentation

Project Overview

This document outlines four key development tasks for building an intelligent educational platform that leverages AI technologies to enhance learning experiences. Each task focuses on different aspects of educational content creation and analysis.

Task 1: Lesson Plan Generation System

Objective

Develop a Retrieval-Augmented Generation (RAG) system for automated lesson plan creation with class-wise document organization.

Technical Specifications

Core Components

- **Vector Database**: Quadrant DB (Qdrant)
- **Embedding Model**: NVIDIA nv-embed
- **Document Structure**: Class-wise categorization (Class 1-12)
- **RAG Architecture**: Query → Retrieve → Generate

System Architecture

```
Document Ingestion → Text Processing → nv-embed Encoding → Qdrant Storage
                                              ↓
User Query → Query Processing → Semantic Search → Context Retrieval → LLM Generation
```

Key Features Required

1. **Document Management**
    - Class-wise document categorization
    - Metadata tagging (subject, chapter, difficulty level)
    - Version control for curriculum updates

2. **Embedding Pipeline**
   - Text chunking with overlap (512 tokens, 50 token overlap)
   - nv-embed integration for high-quality embeddings
   - Batch processing for large document sets
3. **Retrieval System**
   - Semantic search with similarity scoring
   - Filter by class, subject, and curriculum standards
   - Top-k retrieval with relevance ranking
4. **Generation Module**
   - Contextual lesson plan creation
   - Structured output (objectives, activities, assessments)
   - Curriculum alignment verification

Deliverables

- [ ] Qdrant database setup and configuration
- [ ] Document ingestion pipeline
- [ ] nv-embed integration
- [ ] RAG query system
- [ ] Lesson plan generation API
- [ ] Class-wise filtering functionality
- [ ] Performance testing and optimization

Timeline: Day 1 (8 hours)

MVP Scope for Day 1

- Basic Qdrant setup with sample documents
- Simple text chunking and embedding pipeline
- Basic retrieval demo with hardcoded queries
- Simple lesson plan template generation

Task 2: Exam Creator MVP

Objective

Build a basic exam creation prototype with simple evaluation.

Simplified Requirements (Day 2)

Input Parameters (Minimal)

- **Topic**: Text input
- **Exam Type**: MCQ only (simplified)
- **Difficulty**: Easy/Medium/Hard dropdown
- **Total Score**: Number input

Core Features (MVP)

1. **Simple Question Generator**
    - Template-based MCQ creation
    - Topic keyword matching
    - Basic difficulty adjustment
2. **Basic Exam Config**
    - Fixed question count (5-10 questions)
    - Simple scoring (equal weightage)
    - JSON output format
3. **Basic Evaluation**
    - MCQ auto-scoring only
    - Simple percentage calculation
    - Basic pass/fail logic

Quick Implementation

```
# Simplified API Structure
class SimpleExamCreator:
    def create_mcq_exam(self, topic, difficulty, score):
        questions = self.generate_simple_mcqs(topic, difficulty)
        return {"questions": questions, "total_score": score}

    def evaluate_mcqs(self, answers, correct_answers):
        score = sum(1 for a, c in zip(answers, correct_answers) if a == c)
        return {"score": score, "percentage": (score/len(answers))*100}
```

Deliverables (Day 2)

- [ ] Simple MCQ generation
- [ ] Basic web interface
- [ ] Auto-evaluation for MCQs
- [ ] Score calculation
- [ ] Demo with sample questions

Timeline: Day 2 (8 hours)

Task 3: Homework Creator MVP

Objective

Create a simple homework assignment generator with basic evaluation.

Simplified Specifications (Day 3 Morning)

Input Parameters

- **Topic**: Text input
- **Difficulty**: Easy/Medium/Hard
- **Score**: Total marks

MVP Components

1. **Simple Assignment Generator**
   - Template-based problems
   - Topic keyword matching
   - Fixed question formats
2. **Basic Submission**
   - Text input answers
   - Simple file upload
   - Timestamp recording
3. **Simple Evaluation**
   - Keyword-based checking
   - Manual score input option
   - Basic feedback templates

Deliverables (Day 3 Morning - 4 hours)

- [ ] Homework template system
- [ ] Simple problem generator
- [ ] Basic submission form
- [ ] Simple evaluation logic
- [ ] Score calculation

Timeline: Day 3 Morning (4 hours)

Task 4: Nipun Lakshya Analysis MVP

Objective

Build a basic analysis tool for assessment data with simple recommendations.

Simplified Scope (Day 3 Afternoon)

Data Sources (Simplified)

- **Sample JSON Files**: 2-3 sample assessments
- **Simple Excel**: Basic score mapping
- **Fixed Benchmarks**: Hardcoded thresholds

MVP Components

1. Basic Data Processing

JSON Input → Parse Scores → Compare with Benchmarks → Generate Simple Report

2. Simple Analysis

A. Score Comparison

- JSON data parsing (basic)
- Excel lookup (simple)
- Above/below benchmark classification

B. Gap Identification

- Simple if-else rules
- Basic categorization (weak/average/strong)
- Fixed recommendation templates

C. Simple Recommendations

- **For Students**: Template-based suggestions
- **For Teachers**: Basic improvement tips

3. Basic Reports

Simple Output

```
{
  "student_id": "123",
  "literacy_level": "Below Average",
  "numeracy_level": "Average",
  "recommendations": [
    "Focus on reading practice",
    "Continue current math level"
  ]
}
```

Deliverables (Day 3 Afternoon - 4 hours)

- [ ] JSON parser for sample data
- [ ] Basic Excel integration
- [ ] Simple score comparison
- [ ] Template-based recommendations
- [ ] Basic report generation

Timeline: Day 3 Afternoon (4 hours)

3-Day Sprint Implementation Plan

Day 1: RAG Lesson Plan System

Team: 2 interns

Morning (4 hours)

- Set up Qdrant locally
- Create sample document collection (5-10 docs)
- Implement basic text chunking

Afternoon (4 hours)

- Integrate nv-embed for embeddings
- Build simple retrieval function
- Create basic lesson plan template

Deliverable: Working demo with search and basic lesson generation

Day 2: Exam Creator System

Team: 2 interns

Morning (4 hours)

- Create MCQ question templates
- Build simple web interface
- Implement topic-based question selection

Afternoon (4 hours)

- Add basic evaluation logic
- Create scoring system
- Build demo interface

Deliverable: Working exam creator with 5-10 sample questions

Day 3: Homework Creator + Nipun Analysis

Team: 4 interns (2 per task)

Morning (4 hours) Homework Team: Build assignment generator and evaluation **Analysis Team**: JSON parsing and score mapping

Afternoon (4 hours) Homework Team: Demo interface and testing **Analysis Team**: Recommendation engine and reports

Deliverable: Two working prototypes

Simplified Resource Requirements

Quick Setup Stack

- **Backend**: Python + Flask (fastest setup)
- **Database**: Local Qdrant + SQLite
- **Frontend**: Simple HTML/Bootstrap
- **ML**: Pre-trained models only

Team Structure (8 interns)

- **Day 1**: 2 interns on RAG system
- **Day 2**: 2 interns on Exam creator

- **Day 3**: 4 interns (2 on Homework, 2 on Analysis)
- **Support**: Floating support for blockers

Success Metrics (MVP)

- Working demos for all 4 tasks
- Basic functionality demonstrated
- Simple UI interfaces
- Core algorithms implemented

Risk Mitigation (3-Day Sprint)

High-Priority Risks

- **Setup delays** → Pre-configured environments
- **Scope creep** → Strict MVP boundaries
- **Integration issues** → Independent demos
- **Technical blockers** → Immediate mentor intervention

Fallback Plans

- Pre-built templates ready
- Simplified mock data
- Manual processes as backup
- Hardcoded examples for demos

Daily Checkpoints

End of Day 1

- RAG system demo
- Document search working
- Basic lesson plan output

End of Day 2

- Exam creator interface
- MCQ generation working
- Basic evaluation functional

End of Day 3

- All 4 prototypes complete
- Demo presentations ready
- Basic documentation done

Final Goal: 4 working prototypes demonstrating core concepts in 3 days.