Study Notes Summarizer & Quiz Generator Agent

By Nida Khurram

🚀 Project Idea: Study Notes Summarizer & Quiz Generator Agent

A powerful AI agent that summarizes PDF study materials and generates interactive quizzes - perfect for students and learners!

📋 7-Day Learning Journey (AI Agent Edition)

Day 1: Setup & Foundation

Step 1: Install Required Tools

```bash
# Install Node.js (if not already installed)
# Download from nodejs.org or use:
# Windows: chocolatey install nodejs
# Mac: brew install node

# Install Gemini CLI globally
npm install -g @google/gemini-cli

# Test Gemini CLI commands
gemini /help
gemini /tools
```

```
gemini /memory list
gemini /memory add <tag>
gemini /memory refresh
```

Step 2: Create Global Context File

Create `gemini.md` in home directory:

```
markdown

## 🎯 General Rules
- Follow clear, simple, and maintainable coding practices.
- Provide short explanations unless asked for detailed ones.
- Always ask before making changes that can affect multiple files.
- When unsure about user intent, ask a clarifying question.

---

## 🛠️ Coding Style (Python)
- Prefer readability over complexity.
- Use descriptive variable, function, and file names.
- Include comments when logic isn't obvious.
- Keep functions small and modular.
- Follow Pythonic conventions (PEP 8).

---

## 🎯 Project Specific Rules
- Use Streamlit for web UI (simple and effective)
- Implement proper error handling for file operations
- Create modular code structure
- Use type hints where possible
- Follow clean architecture principles
```

Step 3: Initialize Project

```
bash
```

```
# Create project directory
mkdir study-notes-agent
cd study-notes-agent

# Initialize Python environment with UV
pip install uv
# Or install via PowerShell:
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"

# Initialize project
uv init
uv venv
# Activate virtual environment
# Windows:
.venv\Scripts\activate
# Mac/Linux:
source .venv/bin/activate

# Install required packages
uv add streamlit pypdf2 google-generativeai python-dotenv
uv add openagents@latest  # If available, otherwise install separately
```

Step 4: Create Project Structure

```
text

study-notes-agent/
├── main.py                 # Streamlit main application
├── agent/
│   ├── __init__.py
│   ├── summarizer.py      # PDF summarization logic
│   ├── quiz_generator.py  # Quiz generation logic
│   └── context7_client.py # Context7 MCP integration
├── utils/
│   ├── __init__.py
│   ├── pdf_processor.py   # PDF text extraction
│   └── file_handler.py    # File operations
├── data/                  # Uploaded PDFs storage
├── requirements.txt
└── README.md
```

Step 5: Create Project GEMINI.md

Create `GEMINI.md` in project root:

```markdown
# Study Notes Summarizer & Quiz Generator Agent

## Project Overview
AI-powered agent that summarizes PDF study materials and generates interactive quizzes using Context7 MCP and Gemini.

## Core Features
- PDF text extraction using PyPDF
- Intelligent summarization using Gemini
- Multiple quiz types (MCQs, True/False, Mixed)
- Context7 MCP integration for enhanced capabilities
- Streamlit web interface
- Downloadable summaries and quizzes

## Tech Stack
- **Framework**: OpenAgents SDK
- **UI**: Streamlit
- **PDF Processing**: PyPDF2
- **AI**: Gemini CLI + Context7 MCP
- **Language**: Python 3.11+

## Agent Architecture
```

Study Notes Agent

├── PDF Upload & Processing

├── Text Extraction (PyPDF)

├── Summary Generation (Gemini)

├── Quiz Generation (Context7 + Gemini)

└── Streamlit UI

```text
```

```
## Key Components
1. **PDF Processor**: Extracts clean text from uploaded PDFs
2. **Summarizer Agent**: Creates concise, meaningful summaries
3. **Quiz Generator**: Creates various quiz formats from content
4. **Context7 Integration**: Enhances AI capabilities
5. **Streamlit UI**: Clean, user-friendly interface

## UI Design Guidelines
- Card-based layout for summaries
- Clean, educational color scheme
- Responsive design
- Progress indicators for long operations
- Download buttons for generated content
```

Prompt for Gemini

```
text

gemini

"Read all GEMINI.md files and initialize my Study Notes Agent project. Create the basic project structure with Streamlit setup, PDF processing utilities, and the main application file."
```

Save conversation:

```
bash

gemini /save features/init
```

Day 2: PDF Processing & Text Extraction

Create `utils/GEMINI.md`**:**

```
markdown
```

Day 2: PDF Processing Engine

Today's Goal
Build robust PDF text extraction system using PyPDF.

Learning Focus
- PDF file handling
- Text extraction techniques
- Error handling for corrupted PDFs
- Text cleaning and preprocessing

Technical Requirements

PDF Processor Features:
1. **File Upload Support**
   - Accept PDF files via Streamlit
   - Validate file type and size
   - Handle multiple pages

2. **Text Extraction**
   - Extract text from all pages
   - Preserve logical structure
   - Handle encoding issues
   - Remove unnecessary whitespace

3. **Text Cleaning**
   - Remove header/footer artifacts
   - Fix line breaks and spacing
   - Handle special characters
   - Normalize text format

4. **Error Handling**
   - Corrupted PDF files
   - Password-protected PDFs
   - Image-based PDFs (warn user)
   - Large file handling

Success Criteria
☑ Can upload and validate PDF files

```
☑ Extracts clean text from multi-page PDFs
☑ Handles common PDF issues gracefully
☑ Provides meaningful error messages
☑ Text is properly formatted for AI processing

### Example Output Structure:
```python
{
    "filename": "biology_notes.pdf",
    "page_count": 15,
    "total_chars": 24500,
    "extracted_text": "Cleaned text content...",
    "metadata": {
        "author": "",
        "title": "",
        "creation_date": ""
    }
}
```
text

Prompt for Gemini
```

gemini

"Build the PDF processing system. Read @utils/GEMINI.md and create the PDF text extraction utilities with proper error handling and text cleaning."

```
text

**Save conversation:**
```bash
gemini /save features/pdf_processing
```

Day 3: Summary Generation Agent

Create `agent/summarizer/GEMINI.md`:
```

````
markdown

# Day 3: Intelligent Summary Generator

## Today's Goal
Create AI-powered summarization agent using Gemini CLI.

## Learning Focus
- Gemini API integration
- Prompt engineering for summaries
- Content structuring
- Quality optimization

## Summary Agent Features

### 1. Summary Types
- **Concise Summary**: 1-2 paragraphs, key points only
- **Detailed Summary**: Section-wise breakdown
- **Bullet Points**: Quick revision format
- **Concept Map**: Hierarchical structure

### 2. Prompt Engineering
```python
SUMMARY_PROMPTS = {
    "concise": "Create a concise 2-paragraph summary highlighting main concepts...",
    "detailed": "Provide a detailed section-by-section summary covering...",
    "bullet": "Extract key points as bullet points for quick revision...",
    "academic": "Academic-style summary with thesis, evidence, conclusion..."
}
````

3. Quality Controls

- Maintain original meaning
- Preserve technical terminology
- Appropriate length (10-20% of original)
- Logical flow and structure
- Highlight key concepts and definitions
````

4. Output Formats

- **Card Layout**: Beautiful Streamlit cards
- **Section Tabs**: Different summary types in tabs
- **Download Options**: TXT, PDF, Markdown
- **Copy to Clipboard**: One-click copying

UI Display Example:

```
text

📋 Summary Card
┌─────────────────────────────────────┐
│ Chapter 3: Cellular Biology         │
│                                     │
│ • Key Concepts: Cell structure,     │
│   organelles, membrane transport    │
│ • Main Topics: Mitochondria         │
│   function, DNA replication         │
│ • Important Definitions: 15 terms   │
│ • Difficulty: Intermediate          │
└─────────────────────────────────────┘
```

Success Criteria

☑ Generates meaningful, accurate summaries

☑ Multiple summary format options

☑ Maintains academic integrity

☑ Beautiful UI presentation

☑ Download and share functionality

```
text
```

```
## **Prompt for Gemini**
```

gemini

"Build the summary generation agent. Read @agent/summarizer/GEMINI.md and implement the Gemini-powered summarization system with multiple output formats and quality controls."

```
text

**Save conversation:**
```bash
gemini /save features/summarizer
```

Day 4: Context7 MCP Integration

Create `agent/context7/GEMINI.md`**:**

```
markdown

# Day 4: Context7 MCP Integration

## Today's Goal
Integrate Context7 MCP to enhance AI capabilities for educational content.

## Learning Focus
- MCP (Model Context Protocol) setup
- Context7 tool integration
- Enhanced AI capabilities
- Educational content optimization

## Context7 Integration Features

### 1. MCP Setup
- Configure Context7 MCP server
- Establish connection protocol
```
```

```
- Tool registration and discovery
- Error handling and fallbacks

### 2. Enhanced Capabilities
- **Educational Context**: Subject-specific knowledge
- **Learning Patterns**: Adaptive to educational content
- **Quiz Optimization**: Better question generation
- **Content Understanding**: Deeper material comprehension

### 3. Tool Integration
```python
CONTEXT7_TOOLS = {
    "educational_analyze": "Analyze educational content complexity",
    "learning_objectives": "Extract learning objectives",
    "concept_mapping": "Create concept relationships",
    "difficulty_assessment": "Assess content difficulty level"
}
```

4. Fallback Strategy

- Primary: Context7 + Gemini
- Fallback: Gemini alone
- Graceful degradation
- User notification system

Implementation Flow:

1. Upload PDF → Extract text
2. Send to Context7 for educational analysis
3. Enhanced processing with subject context
4. Generate superior summaries and quizzes
5. Fallback to basic Gemini if Context7 unavailable
```

Success Criteria

- ☑ Successful Context7 MCP connection
- ☑ Enhanced educational content processing
- ☑ Proper error handling and fallbacks
- ☑ Improved summary and quiz quality
- ☑ Seamless user experience

```
text

## **Prompt for Gemini**
```

gemini

"Integrate Context7 MCP into the agent. Read @agent/context7/GEMINI.md and implement the MCP connection with enhanced educational capabilities and proper fallback mechanisms."

```
text

**Save conversation:**
```bash
gemini /save features/context7_integration
```

Day 5: Quiz Generation Engine

Create `agent/quiz_generator/GEMINI.md`**:**

```
markdown

# Day 5: Intelligent Quiz Generator
```
```

```
## Today's Goal
Build sophisticated quiz generation system using original PDF content.

## Learning Focus
- Question generation algorithms
- Multiple quiz formats
- Difficulty balancing
- Answer validation

## Quiz Generator Features

### 1. Quiz Types
- **Multiple Choice (MCQ)**: 4 options, single correct
- **True/False**: Fact-based statements
- **Fill-in-the-Blank**: Key term completion
- **Mixed Quiz**: Combination of all types
- **Chapter-wise**: Organized by content sections

### 2. Question Quality
- Based on original PDF (not summary)
- Cover key concepts and definitions
- Varying difficulty levels
- Clear, unambiguous questions
- Plausible distractors for MCQs

### 3. Generation Parameters
```python
QUIZ_CONFIG = {
    "question_count": 10,
    "difficulty": "mixed",  # easy, medium, hard, mixed
    "quiz_type": "mcq",     # mcq, true_false, mixed
    "focus_areas": []       # specific topics to focus on
}
```

4. Output Formats

- **Interactive Quiz**: Streamlit form with instant feedback
- **Printable Version**: PDF format for offline use
```

- **Answer Key**: Separate answer sheet
- **Performance Analytics**: Score and weak areas

5. UI Components

- Progress bar during generation
- Question cards with clean layout
- Instant answer validation
- Score summary at the end
- Review incorrect answers

Example Question Card:

```
text

Q1: What is the primary function of mitochondria?
○ A) Protein synthesis
○ B) Energy production ✓
○ C) DNA storage
○ D) Waste elimination

📝 Explanation: Mitochondria are known as the
powerhouse of the cell, producing ATP through
cellular respiration.
```

Success Criteria

☑ Generates high-quality questions from original content

☑ Multiple quiz formats available

☑ Appropriate difficulty levels

☑ Interactive user experience

☑ Performance tracking and analytics

```
text

## **Prompt for Gemini**
```

gemini

"Build the quiz generation engine. Read @agent/quiz_generator/GEMINI.md and implement the intelligent quiz generator with multiple formats, difficulty levels, and interactive features."

```
text

**Save conversation:**
```bash
gemini /save features/quiz_generator
```

Day 6: Streamlit UI & User Experience

Create `ui/GEMINI.md`**:**

```
markdown

# Day 6: Streamlit Web Interface

## Today's Goal
Create beautiful, responsive Streamlit interface for the study agent.

## Learning Focus
- Streamlit components and layout
- User experience design
- Responsive web design
- Interactive elements
```
```

```
## UI Design Specifications

### 1. Page Layout
- **Sidebar**: File upload, settings, navigation
- **Main Area**: Content display, summaries, quizzes
- **Header**: App title, user guide
- **Footer**: Status, credits

### 2. Color Scheme
```python
COLOR_THEME = {
    "primary": "#4B8BBE",    # Blue - learning
    "secondary": "#306998",  # Dark blue
    "accent": "#FFD43B",     # Yellow - highlight
    "success": "#50C878",    # Green - correct
    "danger": "#DC143C",     # Red - incorrect
    "background": "#F0F2F6", # Light gray
    "card": "#FFFFFF"        # White cards
}
```

3. Key Components

A. File Upload Section

- Drag-and-drop PDF upload
- File validation and preview
- Upload progress indicator

B. Summary Display

- Tabbed interface for different summary types
- Card-based layout with shadows
- Copy-to-clipboard functionality
- Download buttons (TXT, PDF)
```

C. Quiz Interface

- Clean question cards
- Progress indicator
- Instant feedback with explanations
- Score summary and analytics
- Review mode for incorrect answers

D. Navigation & Flow

1. Upload PDF → Show extraction status
2. Generate Summary → Display in beautiful cards
3. Create Quiz → Interactive quiz interface
4. View Results → Analytics and recommendations

4. Responsive Design

- Mobile-friendly layout
- Adaptive columns and spacing
- Touch-friendly buttons
- Readable typography on all devices

5. User Experience

- Loading spinners for long operations
- Success/error messages
- Tooltips and help text
- Session state management

- Previous uploads memory

Success Criteria

☑ Beautiful, professional interface

☑ Intuitive user journey

☑ Responsive mobile design

☑ Smooth interactions and feedback

☑ Accessible and user-friendly

```
text

## **Prompt for Gemini**
```

gemini

"Build the Streamlit UI interface. Read @ui/GEMINI.md and create a beautiful, responsive web interface with proper layout, color scheme, and interactive components."

```
text

**Save conversation:**
```bash
gemini /save features/streamlit_ui
```

Day 7: Advanced Features & Polish

Create `features/advanced/GEMINI.md`**:**

```
markdown
```
```

Day 7: Advanced Features & Final Polish

Today's Goal
Add advanced features and polish the application for production readiness.

Learning Focus
- Advanced AI features
- Performance optimization
- Error handling
- User customization

Advanced Features

1. Smart Content Analysis
- **Difficulty Assessment**: Auto-detect content complexity
- **Subject Classification**: Identify subject area (Science, History, Math, etc.)
- **Key Concept Extraction**: Automatically identify important concepts
- **Learning Objective Generation**: Suggest learning goals

2. Personalization
- **Learning Style Adaptation**: Visual, auditory, reading/writing preferences
- **Difficulty Adjustment**: Adapt quiz difficulty to user performance
- **Progress Tracking**: Track user improvement over time
- **Custom Study Plans**: Generate personalized study schedules

3. Enhanced Quiz Features
- **Timed Quizzes**: Add time limits for practice
- **Question Banks**: Store and reuse questions
- **Adaptive Testing**: Dynamic difficulty based on performance
- **Peer Comparison**: Anonymous performance benchmarking

4. Export & Integration
- **Anki Deck Export**: Create importable flashcards
- **Google Classroom Integration**: Share directly to classrooms
- **API Endpoints**: REST API for other applications
- **Batch Processing**: Handle multiple PDFs simultaneously

5. Performance Optimization
- **Caching**: Cache processed PDFs and summaries
- **Async Processing**: Background processing for large files

```
- **Memory Management**: Efficient handling of large documents
- **Progress Tracking**: Real-time progress updates

### 6. Production Ready Features
- **Error Logging**: Comprehensive error tracking
- **User Analytics**: Usage patterns and feature popularity
- **A/B Testing**: Interface and algorithm improvements
- **Documentation**: User guide and API documentation

### Final Polish Checklist
- [ ] All features tested and working
- [ ] Error handling for edge cases
- [ ] Performance optimized
- [ ] UI/UX polished and responsive
- [ ] Documentation complete
- [ ] Deployment ready

### Success Criteria
☑ Advanced AI features implemented
☑ Performance optimized for production
☑ Comprehensive error handling
☑ Professional documentation
☑ Ready for deployment and user testing
```

Prompt for Gemini

```
text

gemini

"Add advanced features and final polish. Read @features/advanced/GEMINI.md and implement smart content analysis, personalization, performance optimization, and production-ready features."
```

Save conversation:

```
bash

gemini /save features/final_polish
```

🚀 Complete Implementation Commands

Final Setup & Run Commands:

```bash
# Navigate to project directory
cd study-notes-agent

# Activate virtual environment
.venv\Scripts\activate  # Windows
source .venv/bin/activate  # Mac/Linux

# Install all dependencies
uv sync

# Run the Streamlit application
streamlit run main.py

# For production deployment
streamlit run main.py --server.port 8501 --server.address 0.0.0.0
```

Testing the Application:

1. Open browser to http://localhost:8501
2. Upload a PDF study material
3. Generate summary (try different formats)
4. Create quiz (test all question types)
5. Experience the full interactive flow

🎯 Student Success Path

By the end of 7 days, you'll have:

- ☑ A working AI Study Notes Agent
- ☑ PDF processing and text extraction skills
- ☑ Gemini CLI and Context7 MCP experience
- ☑ Streamlit web development skills
- ☑ AI prompt engineering knowledge
- ☑ A impressive portfolio project
- ☑ Confidence to build more AI agents!

This is real AI agent development experience! 🚀

📝 Gemini CLI Commands Summary

```bash
# Daily development workflow
gemini /memory list
gemini /memory add "Study Notes Agent Project"
gemini "Read GEMINI.md and implement [feature]"
gemini /save features/[day_feature]

# Project management
gemini /tools  # Check available tools
```

```
gemini /help   # Get help on commands
gemini /clear  # Clear conversation history
```

Start building your AI Study Notes Agent today! 🎓✨