UniAssist API Documentation

Version: 1.0.0
Base URL: `http://localhost:8000`
Project: Multi-Agent RAG-Based AI Assistant for Programming Fundamentals Course
Team: i210433-Nida Azam, i221338-Raba Alvi, i221296-Eraj Zaman, i221179-Amna Saeed

---

Table of Contents

Overview

UniAssist is a Multi-Agent RAG (Retrieval-Augmented Generation) system designed to help students with Programming Fundamentals (C++) coursework. The system provides:

- **Question Answering** from course materials and past papers
- **Quiz Generation** on various topics

- **Automatic Quiz Grading** with detailed feedback
- **Document Summarization** with multiple formats
- **PDF Upload** for custom course materials

Key Features

- ✅ Real-time Q&A from 22 past papers (36K words)
- ✅ AI-powered quiz generation (3-5 questions per topic)
- ✅ Automatic grading with instant feedback
- ✅ Multiple summarization modes (concise, detailed, bullet points)
- ✅ Custom PDF upload and processing
- ✅ Vector-based semantic search (ChromaDB)
- ✅ Fast responses (<5 seconds)

Authentication

Current Version: No authentication required (demo mode)

Production: Will require API key in header:

```
Authorization: Bearer YOUR_API_KEY
```

Base URL

Local Development:

```
http://localhost:8000
```

Production: (To be deployed)

```
https://uniassist.api.com
```

Endpoints

1. Health Check

Check if the API is running and get system status.

Endpoint: `GET /health`

Request:

```bash
curl -X GET "http://localhost:8000/health"
```

Response:

```json
{
  "status": "healthy",
  "assistant": "ready",
  "vector_store": 92
}
```

Response Fields:

- `status` (string): System health status
- `assistant` (string): RAG system status
- `vector_store` (integer): Number of documents in vector store

---

2. Ask Question

Ask a question about Programming Fundamentals course content.

Endpoint: `POST /ask`

Request Body:

```json
```

```
{
  "question": "string"
}
```

Example Request:

```bash
curl -X POST "http://localhost:8000/ask" \
  -H "Content-Type: application/json" \
  -d '{
    "question": "What are pointers in C++?"
  }'
```

Example Response:

```json
{
  "question": "What are pointers in C++?",
  "answer": "A pointer in C++ is a variable that stores the memory address of another variable. Pointers are declared using the
  "sources": [
    "PF Final Exam (Solution) (Spring-2024).pdf",
    "PF Final-Exam Spring 2019.pdf",
    "PF Sessional-II (Fall-20) (Solution).pdf"
  ],
  "context_used": 5
}
```

Response Fields:

- `question` (string): The original question
- `answer` (string): AI-generated answer based on course materials
- `sources` (array): List of source documents used
- `context_used` (integer): Number of document chunks used

Supported Topics:

- Pointers
- Arrays

- Loops (for, while, do-while)
- Recursion
- Functions
- Classes and Objects
- File Handling
- Data Structures

Response Time: 2-5 seconds

---

3. Generate Quiz

Generate a multiple-choice quiz on a specific topic.

Endpoint: `POST /quiz/generate`

Request Body:

```json
{
  "topic": "string",
  "num_questions": 5
}
```

Parameters:

- `topic` (string, required): Topic for quiz generation
- `num_questions` (integer, optional): Number of questions (default: 5, max: 5)

Example Request:

```bash
curl -X POST "http://localhost:8000/quiz/generate" \
  -H "Content-Type: application/json" \
  -d '{
    "topic": "pointers",
    "num_questions": 3
  }'
```

Example Response:

```json
```

```json
{
  "topic": "pointers",
  "num_questions": 3,
  "questions": [
    {
      "question": "What does a pointer store in C++?",
      "options": {
        "A": "The memory address of another variable",
        "B": "The value of a variable directly",
        "C": "The name of a variable",
        "D": "The data type of a variable"
      },
      "correct": "A"
    },
    {
      "question": "Which operator is used to dereference a pointer?",
      "options": {
        "A": "&",
        "B": "*",
        "C": "->",
        "D": "."
      },
      "correct": "B"
    },
    {
      "question": "What is a dangling pointer?",
      "options": {
        "A": "A pointer that points to deallocated memory",
        "B": "A pointer that is never used",
        "C": "A pointer with no data type",
        "D": "A pointer that points to NULL"
      },
      "correct": "A"
    }
  ],
  "sources": [
    "PF Past Papers Database",
    "Course Materials"
  ]
}
```

Response Fields:

- `topic` (string): Quiz topic
- `num_questions` (integer): Number of questions generated
- `questions` (array): Array of question objects
  - `question` (string): Question text
  - `options` (object): Four options (A, B, C, D)
  - `correct` (string): Correct answer (A, B, C, or D)
- `sources` (array): Source materials used

Supported Topics:

- `pointers`
- `arrays`
- `loops`
- `recursion`
- `functions`

Response Time: 1-3 seconds

---

4. Grade Quiz

Grade a student's quiz submission and return detailed results.

Endpoint: `POST /quiz/grade`

Request Body:

```
json
```

```json
{
  "quiz": {
    "questions": [
      {
        "question": "string",
        "correct": "string"
      }
    ]
  },
  "answers": {
    "1": "string",
    "2": "string"
  }
}
```

Parameters:

- `quiz` (object, required): The quiz object from `/quiz/generate`
- `answers` (object, required): Student's answers (question number → answer)

Example Request:

```
bash
```

```
curl -X POST "http://localhost:8000/quiz/grade" \
  -H "Content-Type: application/json" \
  -d '{
    "quiz": {
      "questions": [
        {
          "question": "What does a pointer store?",
          "correct": "A"
        },
        {
          "question": "What is dereferencing?",
          "correct": "B"
        }
      ]
    },
    "answers": {
      "1": "A",
      "2": "B"
    }
  }'
```

Example Response:

```
json
```

```
{
 "total_questions": 2,
 "correct": 2,
 "incorrect": 0,
 "score": 100,
 "results": [
  {
   "question_num": 1,
   "question": "What does a pointer store?",
   "user_answer": "A",
   "correct_answer": "A",
   "is_correct": true
  },
  {
   "question_num": 2,
   "question": "What is dereferencing?",
   "user_answer": "B",
   "correct_answer": "B",
   "is_correct": true
  }
 ]
}
```

Response Fields:

- `total_questions` (integer): Total number of questions
- `correct` (integer): Number of correct answers
- `incorrect` (integer): Number of incorrect answers
- `score` (float): Percentage score (0-100)
- `results` (array): Detailed results for each question
  - `question_num` (integer): Question number
  - `question` (string): Question text
  - `user_answer` (string): Student's answer
  - `correct_answer` (string): Correct answer
  - `is_correct` (boolean): Whether answer was correct

Response Time: <1 second

---

5. Summarize Document

Generate a summary of provided text content.

Endpoint: `POST /summarize`

Request Body:

```json
{
  "text": "string",
  "summary_type": "concise"
}
```

Parameters:

- `text` (string, required): Text to summarize (max 5000 characters)
- `summary_type` (string, optional): Type of summary
  - `concise` (default): 2-3 sentences
  - `detailed`: 4-6 sentences
  - `bullet_points`: Key points as bullets

Example Request (Concise):

```bash
curl -X POST "http://localhost:8000/summarize" \
  -H "Content-Type: application/json" \
  -d '{
    "text": "Pointers are variables that store memory addresses. They are declared using asterisk symbol. Dereferencing accesse
    "summary_type": "concise"
  }'
```

Example Response (Concise):

```json
```

```
{
  "summary": "Pointers are variables that store memory addresses of other variables in C++. They enable direct memory mani
  "summary_type": "concise"
}
```

Example Request (Detailed):

```bash
curl -X POST "http://localhost:8000/summarize" \
  -H "Content-Type: application/json" \
  -d '{
    "text": "Pointers store memory addresses. They use asterisk for declaration and dereferencing. Essential for dynamic memo
    "summary_type": "detailed"
  }'
```

Example Response (Detailed):

```json
{
  "summary": "Pointers in C++ are powerful variables that store memory addresses rather than direct values. They are declare
  "summary_type": "detailed"
}
```

Response Fields:

- `summary` (string): Generated summary
- `summary_type` (string): Type of summary generated

Response Time: 1-3 seconds

---

6. Upload PDF

Upload a PDF document to add to the knowledge base.

Endpoint: `POST /upload-pdf`

Request: Multipart form data

Parameters:

- `file` (file, required): PDF file to upload
- `doc_type` (string, optional): Document type
  - `notes` (default)
  - `past_paper`
  - `slides`
  - `handbook`
  - `course_outline`

Example Request:

```bash
curl -X POST "http://localhost:8000/upload-pdf?doc_type=notes" \
  -F "file=@/path/to/document.pdf"
```

Example Response:

```json
{
  "status": "success",
  "filename": "document.pdf",
  "pages": 15,
  "chunks_added": 12,
  "message": "PDF processed and added to knowledge base"
}
```

Response Fields:

- `status` (string): Upload status
- `filename` (string): Name of uploaded file
- `pages` (integer): Number of pages in PDF
- `chunks_added` (integer): Number of text chunks added to vector store
- `message` (string): Success message

Limitations:

- Max file size: 10MB
- Supported formats: PDF only
- Processing time: ~10-30 seconds depending on PDF size

---

7. Get Statistics

Get system statistics and usage information.

Endpoint: `GET /stats`

Request:

```bash
curl -X GET "http://localhost:8000/stats"
```

Response:

```json
{
  "total_documents": 92,
  "system_status": "operational",
  "model": "qwen2:0.5b"
}
```

Response Fields:

- `total_documents` (integer): Total documents in knowledge base
- `system_status` (string): System operational status
- `model` (string): LLM model being used

Response Codes

Code	Description
200	Success

Code	Description
400	Bad Request - Invalid input
404	Not Found - Resource doesn't exist
422	Unprocessable Entity - Validation error
500	Internal Server Error
503	Service Unavailable - System not initialized

Error Handling

All errors return a JSON object with details:

```json
{
  "detail": "Error message describing what went wrong"
}
```

Common Errors:

422 Validation Error

```json
{
  "detail": [
    {
      "loc": ["body", "question"],
      "msg": "field required",
      "type": "value_error.missing"
    }
  ]
}
```

503 Service Unavailable

```json
```

```
{
  "detail": "Assistant not initialized"
}
```

Solution: Ensure vector store is created by running:

```bash
python src/embeddings/embedding_generator.py
```

404 No Results

```json
{
  "detail": "No relevant content found for this topic"
}
```

Rate Limits

Current Version: No rate limits (demo mode)

Production Limits:

- 100 requests per minute per IP
- 1000 requests per day per API key
- PDF uploads: 10 per hour

Examples

Complete Workflow Example

```bash
```

```
# 1. Check system health
curl http://localhost:8000/health

# 2. Ask a question
curl -X POST http://localhost:8000/ask \
  -H "Content-Type: application/json" \
  -d '{"question": "What are pointers?"}'

# 3. Generate a quiz
curl -X POST http://localhost:8000/quiz/generate \
  -H "Content-Type: application/json" \
  -d '{"topic": "arrays", "num_questions": 3}'

# 4. Grade the quiz
curl -X POST http://localhost:8000/quiz/grade \
  -H "Content-Type: application/json" \
  -d '{
    "quiz": {...},
    "answers": {"1": "A", "2": "B", "3": "C"}
  }'

# 5. Summarize text
curl -X POST http://localhost:8000/summarize \
  -H "Content-Type: application/json" \
  -d '{
    "text": "Your text here...",
    "summary_type": "concise"
  }'
```

Interactive Documentation

Visit the interactive API documentation at:

Swagger UI: `http://localhost:8000/docs`
ReDoc: `http://localhost:8000/redoc`

SDK & Libraries

Python Example

python

```python
import requests

BASE_URL = "http://localhost:8000"

# Ask a question
def ask_question(question: str):
    response = requests.post(
        f"{BASE_URL}/ask",
        json={"question": question}
    )
    return response.json()

# Generate quiz
def generate_quiz(topic: str, num_questions: int = 3):
    response = requests.post(
        f"{BASE_URL}/quiz/generate",
        json={"topic": topic, "num_questions": num_questions}
    )
    return response.json()

# Usage
answer = ask_question("What are pointers?")
print(answer['answer'])

quiz = generate_quiz("arrays", 3)
print(f"Generated {quiz['num_questions']} questions")
```

JavaScript Example

javascript

```javascript
const BASE_URL = "http://localhost:8000";

// Ask a question
async function askQuestion(question) {
  const response = await fetch(`${BASE_URL}/ask`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ question })
  });
  return await response.json();
}

// Generate quiz
async function generateQuiz(topic, numQuestions = 3) {
  const response = await fetch(`${BASE_URL}/quiz/generate`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ topic, num_questions: numQuestions })
  });
  return await response.json();
}

// Usage
const answer = await askQuestion("What are pointers?");
console.log(answer.answer);

const quiz = await generateQuiz("arrays", 3);
console.log(`Generated ${quiz.num_questions} questions`);
```

Technical Details

Architecture

- **Backend:** FastAPI (Python 3.10+)
- **Vector Store:** ChromaDB (Persistent)
- **Embeddings:** sentence-transformers/all-MiniLM-L6-v2 (384 dimensions)
- **LLM:** qwen2:0.5b (via Ollama)
- **PDF Processing:** PyMuPDF

- **Search:** Semantic similarity search with cosine distance

Data Sources

- 22 Programming Fundamentals past papers
- 280 pages of course materials
- 36,367 words of content
- 193 images extracted
- 92 intelligent text chunks

Performance

- Question answering: 2-5 seconds
- Quiz generation: 1-3 seconds
- Quiz grading: <1 second
- Summarization: 1-3 seconds
- PDF upload processing: 10-30 seconds

Troubleshooting

Issue: "Collection does not exist"

Solution:

```bash
python src/embeddings/embedding_generator.py
```

Issue: "Could not connect to Ollama"

Solution:

```bash
ollama serve
```

Issue: Slow responses

Solutions:

1. Use smaller model: `ollama pull qwen2:0.5b`
2. Reduce `num_questions` in quiz generation
3. Shorten text in summarization

Support & Contact

Project Team:

- Nida Azam (i210433)
- Raba Alvi (i221338)
- Eraj Zaman (i221296)
- Amna Saeed (i221179)

Institution: FAST NUCES Islamabad
Course: Generative AI Final Project
Version: 1.0.0
Last Updated: December 2024

Changelog

Version 1.0.0 (December 2024)

- Initial release
- Core Q&A functionality
- Quiz generation and grading
- Document summarization
- PDF upload support
- 92 documents in knowledge base
