# 🎓 BITS Pilani - API Based Products Assignment (EC-1)

**Course:** SE*ZG504 - API-based Products
**Semester:** First Semester 2025-2026
**Weightage:** 30%
**Duration:** 2 Weeks

---

## 📋 Assignment Overview

This assignment consists of three questions covering different aspects of API development and management:

1. **Question 1 (10 marks):** Plagiarism Checker using Cosine Similarity + ML
2. **Question 2 (10 marks):** Rate Limiting and Request Size Limiting with KONG API Gateway
3. **Question 3 (10 marks):** Implementation of Token Bucket and Leaky Bucket Rate Limiting Algorithms

**Total:** 30 Marks

---

## 🎯 Learning Objectives

By completing this assignment, you will learn:

- ✅ Text similarity using TF-IDF and Cosine Similarity
- ✅ Machine Learning classification (Logistic Regression)
- ✅ Building web applications with Streamlit and Flask
- ✅ API Gateway configuration and management
- ✅ Rate limiting techniques and implementations
- ✅ Request validation and size limiting
- ✅ Algorithm implementation and analysis
- ✅ Docker containerization

---

## 📦 Complete Deliverables

**Question 1 Deliverables:**

✅ `utils.py` - Helper functions (cosine similarity, highlighting)
✅ `data_prep.py` - Training data preparation
✅ `model.py` - ML model training
✅ `app.py` - Streamlit web application

- ✅ `plagiarism_model.pkl` - Trained ML model
- ✅ `plagiarism_dataset.csv` - Training dataset
- ✅ Sample test files (3 text files)
- ✅ Screenshots (8 total)

**Bonus:** Flask API version with separate frontend

**Question 2 Deliverables:**

- ✅ `docker-compose.yml` - KONG configuration
- ✅ `test_rate_limit.py` - Rate limiting test script
- ✅ `test_size_limit.py` - Size limiting test script
- ✅ Configuration commands documentation
- ✅ Screenshots (8 total)

**Question 3 Deliverables:**

- ✅ `token_bucket.py` - Token Bucket algorithm implementation
- ✅ `leaky_bucket.py` - Leaky Bucket algorithm implementation
- ✅ Code snippets with explanations
- ✅ Screenshots (8 total)

---

## 🏗 Technologies Used

**Programming Languages:**

- Python 3.8+

**Libraries & Frameworks:**

- **scikit-learn** - Machine Learning
- **pandas** - Data manipulation
- **numpy** - Numerical computing
- **Streamlit** - Web application framework
- **Flask** - RESTful API framework
- **TfidfVectorizer** - Text vectorization
- **difflib** - Text comparison

**Tools & Platforms:**

- **KONG API Gateway** - API management
- **Docker & Docker Compose** - Containerization
- **PostgreSQL** - KONG database

---

# 🚀 Quick Start Guide

### Prerequisites Check:

bash

```bash
# Python version
python --version  # Should be 3.8 or higher

# Docker (for Question 2)
docker --version
docker-compose --version

# pip
pip --version
```

### Installation Steps:

### 1. Clone/Download the Project

bash

```bash
# Create main directory
mkdir BITS_Assignment
cd BITS_Assignment
```

### 2. Set up Question 1 (Streamlit Version)

bash

```
# Create and navigate to directory
mkdir Question1_Streamlit
cd Question1_Streamlit

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate

# Install dependencies
pip install numpy pandas scikit-learn streamlit joblib

# Copy all Question 1 files here
# (utils.py, data_prep.py, model.py, app.py, sample files)

# Prepare data and train model
python data_prep.py
python model.py

# Run application
streamlit run app.py
```

### 3. Set up Question 2 (KONG)

```
bash

# Navigate to Question 2 directory
cd ../Question2_KONG

# Create docker-compose.yml
# (Copy from provided artifact)

# Start KONG
docker-compose up -d

# Follow configuration steps in guide
```

### 4. Set up Question 3 (Algorithms)

```
bash
```

```
# Navigate to Question 3 directory
cd ../Question3_Algorithms

# Copy algorithm files
# (token_bucket.py, leaky_bucket.py)

# Run demonstrations
python token_bucket.py
python leaky_bucket.py
```

---

## 📖 Detailed Question Breakdown

### Question 1: Plagiarism Checker

**Objective:** Build an application that detects plagiarism using text similarity and ML classification.

**Key Components:**

1. **TF-IDF Vectorization** - Converts text to numerical vectors
2. **Cosine Similarity** - Measures document similarity (0 to 1)
3. **Logistic Regression** - Binary classification (plagiarized/original)
4. **Text Highlighting** - Uses difflib to show matching segments

**How it Works:**

```
Input: Two text documents
  ↓
TF-IDF Vectorization
  ↓
Calculate Cosine Similarity
  ↓
ML Model Prediction
  ↓
Output: Plagiarism score + Highlighted matches
```

**Threshold:** Similarity ≥ 0.8 indicates plagiarism

**Features:**

- ✅ File upload interface
- ✅ Real-time similarity calculation
- ✅ ML-based classification

- ✅ Visual highlighting of matching text
- ✅ Document statistics
- ✅ Responsive UI

**Test Cases Provided:**

1. **Plagiarized:** `submission_plagiarized.txt` (High similarity)
2. **Original:** `submission_original.txt` (Low similarity)

---

### Question 2: KONG API Gateway

**Objective:** Configure rate limiting and request size limiting using KONG to protect APIs.

**Setup Architecture:**

```
Client Request
  ↓
KONG Gateway (Port 8000)
  ├─ Rate Limiting Plugin
  ├─ Request Size Limiting Plugin
  └─ Routes
    ↓
Flask API (Port 5000)
  ↓
Response
```

**Key Concepts:**

**Rate Limiting:**

- Limits number of requests per time period
- Prevents API abuse and DoS attacks
- Configuration: 5 requests per minute (configurable)
- Returns 429 (Too Many Requests) when exceeded

**Request Size Limiting:**

- Limits maximum payload size
- Prevents memory exhaustion
- Configuration: 5MB limit (configurable)
- Returns 413 (Payload Too Large) when exceeded

**KONG Components:**

1. **Service** - Upstream API endpoint
2. **Route** - URL path mapping
3. **Plugins** - Rate limiting, size limiting

**Testing:**

- Rate limit test: Send 10 requests rapidly
- Size limit test: Upload 6MB file
- Both should show appropriate error responses

---

### Question 3: Rate Limiting Algorithms

**Objective:** Implement and compare Token Bucket and Leaky Bucket algorithms.

### Token Bucket Algorithm

**Concept:**

```
Bucket: [🪙 🪙 🪙 🪙 🪙] (Tokens)
         ↓
Refill Rate: +2 tokens/second
Request: Consumes 1 token
If tokens available: ✅ Allow
If no tokens: ❌ Reject
```

**Characteristics:**

- ✅ Allows burst traffic (up to bucket capacity)
- ✅ Tokens refill continuously
- ✅ Variable output rate
- ✅ Good for API rate limiting

**Use Cases:**

- API request throttling
- User quotas
- Resource access control

### Leaky Bucket Algorithm

**Concept:**

```
Bucket: [📦📦📦📦📦] (Request Queue)
        ↓
Leak Rate: Process 2 requests/second
Request: Added to queue
If space available: ✅ Add to queue
If bucket full: ❌ Reject
```

**Characteristics:**

- ✅ Smooths burst traffic
- ✅ Constant output rate
- ✅ Has request queue
- ✅ Good for traffic shaping

**Use Cases:**

- Network packet scheduling
- Traffic smoothing
- Bandwidth limiting

### Comparison Table

| Feature | Token Bucket | Leaky Bucket |
|---|---|---|
| Burst Handling | Allows bursts | Queues bursts |
| Output Rate | Variable | Constant |
| After Idle | Immediate burst allowed | No burst allowance |
| Latency | Immediate | May add queueing delay |
| Memory | Low (just tokens) | Higher (queue) |

---

## 📸 Screenshot Requirements

**Question 1 (8 Screenshots):**

1. ✅ Streamlit app homepage
2. ✅ File upload interface
3. ✅ High similarity result (plagiarized)
4. ✅ Highlighted matching segments
5. ✅ Low similarity result (original)

6. ✅ Document statistics
7. ✅ (Optional) Flask API running
8. ✅ (Optional) API response

**Question 2 (8 Screenshots):**

1. ✅ KONG admin API response
2. ✅ Service registration
3. ✅ Route configuration
4. ✅ Rate limiting plugin
5. ✅ Size limiting plugin
6. ✅ Rate limit exceeded (429)
7. ✅ Size limit exceeded (413)
8. ✅ Docker containers running

**Question 3 (8 Screenshots):**

1. ✅ Token Bucket - Burst requests
2. ✅ Token Bucket - Refill mechanism
3. ✅ Token Bucket - Multiple clients
4. ✅ Token Bucket - API simulation
5. ✅ Leaky Bucket - Filling bucket
6. ✅ Leaky Bucket - Leak mechanism
7. ✅ Leaky Bucket - Comparison table
8. ✅ Leaky Bucket - Network simulation

**Total Screenshots:** 24 minimum

---

## 🎨 Sample Output Examples

**Question 1 Output:**

```
Cosine Similarity Score: 0.87
Plagiarism Probability: 0.92
Status: 🔴 PLAGIARIZED

Highlighted Matching Text:
[Text with <mark> highlighted sections]
```

```
Document Statistics:
Original: 2,543 characters, 421 words
Submission: 2,398 characters, 405 words
```

**Question 2 Output:**

```
bash

# Rate Limit Test
Request 1: Status 200
Request 2: Status 200
Request 3: Status 200
Request 4: Status 200
Request 5: Status 200
Request 6: Status 429 ⚠️ Rate limit exceeded!
Headers: {'X-RateLimit-Limit-Minute': '5', 'X-RateLimit-Remaining-Minute': '0'}

# Size Limit Test
Status: 413
Response: Request size exceeded 5MB limit
```

**Question 3 Output:**

```
Token Bucket - Burst Test:
Request 1: ✅ ALLOWED | Tokens remaining: 9.00
Request 2: ✅ ALLOWED | Tokens remaining: 8.00
...
Request 11: ❌ REJECTED | Retry after: 0.50s

Leaky Bucket - Queue Test:
Request 1: ✅ ADDED | Queue size: 1/10
Request 2: ✅ ADDED | Queue size: 2/10
...
Request 11: ❌ REJECTED | Bucket is full!
```

---

## 📝 PDF Submission Format

**Document Structure (25-35 pages):**

```
1. Cover Page (1 page)
  - Name, ID, Course, Date

2. Table of Contents (1 page)
```

```
3. Question 1 (8-12 pages)
   - Introduction & Objectives
   - Architecture & Design
   - Code Implementation (key snippets)
   - Screenshots with explanations
   - Results & Analysis

4. Question 2 (6-10 pages)
   - Introduction to KONG
   - Configuration Steps
   - Commands & Setup
   - Screenshots with explanations
   - Testing Results

5. Question 3 (8-12 pages)
   - Algorithm Explanations
   - Implementation Details
   - Code Snippets
   - Screenshots with outputs
   - Comparison & Analysis

6. Conclusion (1-2 pages)

7. References (1 page)
```

**Formatting Tips:**

- **Font:** Arial or Times New Roman, 11-12pt
- **Margins:** 1 inch all sides
- **Line Spacing:** 1.5
- **Code:** Monospace font, syntax highlighting
- **Screenshots:** Clear, annotated, properly sized
- **Headers:** Use consistent formatting
- **Page Numbers:** Bottom center

---

## 🔍 Testing & Verification

### Question 1 Tests:

```
bash
```

```
# Test 1: High similarity
Upload: original.txt + submission_plagiarized.txt
Expected: Similarity > 0.8, Plagiarized

# Test 2: Low similarity
Upload: original.txt + submission_original.txt
Expected: Similarity < 0.5, Original

# Test 3: Identical files
Upload: original.txt + original.txt
Expected: Similarity = 1.0, Plagiarized
```

**Question 2 Tests:**

```bash
# Test 1: Rate limiting
Run test_rate_limit.py
Expected: 429 after configured limit

# Test 2: Size limiting
Run test_size_limit.py
Expected: 413 for oversized payload

# Test 3: Normal request
Single request within limits
Expected: 200 OK
```

**Question 3 Tests:**

```bash
# Test 1: Token Bucket
python token_bucket.py
Verify: Burst handling, refill mechanism

# Test 2: Leaky Bucket
python leaky_bucket.py
Verify: Queue management, constant rate

# Test 3: Comparison
Compare outputs side by side
Verify: Different behaviors explained
```

---

## 🐛 Common Issues & Solutions

### Issue 1: Module Not Found

```bash
Error: ModuleNotFoundError: No module named 'streamlit'
Solution: pip install streamlit
```

### Issue 2: Port Already in Use

```bash
Error: Address already in use
Solution:
# Find process
netstat -an | findstr :8501  # Windows
lsof -i :8501  # Mac/Linux
# Kill process or use different port
streamlit run app.py --server.port 8502
```

### Issue 3: Docker Issues

```bash
Error: Cannot connect to Docker daemon
Solution:
# Start Docker Desktop
# Verify: docker ps
```

### Issue 4: KONG Cannot Reach Flask

```bash
Error: 502 Bad Gateway
Solution: Use host.docker.internal instead of localhost
curl -X POST http://localhost:8001/services/ \
  --data "url=http://host.docker.internal:5000"
```

---

## 📚 Resources & References

**Documentation:**

- Scikit-learn: https://scikit-learn.org/
- Streamlit: https://docs.streamlit.io/
- Flask: https://flask.palletsprojects.com/

- KONG: https://docs.konghq.com/

**Tutorials:**

- TF-IDF: https://en.wikipedia.org/wiki/Tf%E2%80%93idf
- Cosine Similarity: https://en.wikipedia.org/wiki/Cosine_similarity
- Rate Limiting: https://www.cloudflare.com/learning/bots/what-is-rate-limiting/

**Additional Reading:**

- Token Bucket Algorithm: https://en.wikipedia.org/wiki/Token_bucket
- Leaky Bucket Algorithm: https://en.wikipedia.org/wiki/Leaky_bucket

---

## ✅ Pre-Submission Checklist

**Code Completeness:**

□ All Python files created and tested
□ No syntax errors
□ All dependencies listed
□ Virtual environments created
□ Models trained successfully

**Functionality:**

□ Question 1 app runs without errors
□ Question 2 KONG configured properly
□ Question 3 algorithms demonstrate correctly
□ All test cases pass

**Documentation:**

□ All screenshots captured (24 minimum)
□ Screenshots clearly labeled
□ Code snippets included
□ Explanations provided
□ Commands documented

**PDF Preparation:**

□ Cover page complete
□ Table of contents
□ All questions included
□ Proper formatting
□ File size < 25MB

□ Named: "YourName_StudentID_Assignment.pdf"

**Final Check:**

□ PDF reviewed for completeness
□ All screenshots visible and clear
□ Code formatted properly
□ No sensitive information included
□ Spell-checked
□ Ready for submission!

---

## 📧 Support & Contact

For technical issues:

1. Review this README thoroughly
2. Check the troubleshooting section
3. Verify all prerequisites
4. Test with sample data first
5. Check error messages carefully

---

## 🎓 Grading Criteria (30 Marks Total)

**Question 1 (10 marks):**

- Code Implementation: 4 marks
- Functionality: 3 marks
- Screenshots & Documentation: 3 marks

**Question 2 (10 marks):**

- KONG Configuration: 4 marks
- Testing & Verification: 3 marks
- Screenshots & Documentation: 3 marks

**Question 3 (10 marks):**

- Algorithm Implementation: 4 marks
- Correctness & Efficiency: 3 marks
- Screenshots & Analysis: 3 marks

---

## 🎉 Conclusion

This comprehensive assignment covers essential concepts in API development:

- **Machine Learning** for intelligent applications
- **API Gateway** for security and management
- **Algorithm Implementation** for efficient resource control

Complete all three questions thoroughly, document your work properly, and you'll gain valuable hands-on experience in API-based product development.

**Best of luck with your assignment!** 🚀

---

**Version:** 1.0
**Last Updated:** November 2025
**Course:** SE*ZG504 - API-based Products
**Institution:** BITS Pilani - Work Integrated Learning Programmes