Day 1: Foundation & Setup

Building the Foundation of Your RAG System

1. Understanding RAG Architecture

Retrieval-Augmented Generation (RAG) is a powerful AI architecture that enhances Large Language Models (LLMs) by combining them with external knowledge retrieval. This combination allows for more accurate, up-to-date, and contextually relevant responses.

Core Components:

- Document Processing System
- Vector Database (Elasticsearch)
- LLM Integration (OpenAI)

How RAG Works:

1. Document Processing Phase: Document → Text Extraction → Chunking → Embedding Generation
2. Query Processing Phase: User Query → Query Embedding → Similarity Search → Context Retrieval
3. Response Generation Phase: Retrieved Context + User Query → LLM Processing → Final Response

2. Development Environment Setup

To get started, you need to set up your development environment with the necessary tools and dependencies.

Prerequisites:

- PHP 8.2 or higher
- Composer
- Node.js & npm
- Docker & Docker Compose (for Elasticsearch)
- OpenAI API Key
- Git

Installation Steps:

PHP and Composer Setup

```
# Install PHP 8.2 (commands may vary based on OS)
sudo apt update
sudo apt install php8.2 php8.2-cli php8.2-common php8.2-curl php8.2-mbstring
php8.2-xml php8.2-zip

# Install Composer
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php composer-setup.php
php -r "unlink('composer-setup.php');"
sudo mv composer.phar /usr/local/bin/composer
```

Node.js and npm Setup

```
# Install Node.js and npm (commands may vary based on OS)
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Docker Setup

```
# Install Docker (commands may vary based on OS)
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

3. Project Structure and Configuration

Our RAG system is built on the Laravel framework. Here's how to initialize the project and configure the environment.

Project Initialization:

```
# Create new Laravel project
composer create-project laravel/laravel rag-system
cd rag-system

# Install dependencies
composer require openai-php/client
composer require elasticsearch/elasticsearch
npm install
```

Directory Structure (Key Directories):

```
rag-system/
├── app/            # Application logic (Models, Services, etc.)
├── config/         # Configuration files
├── database/       # Database migrations and seeders
├── public/         # Publicly accessible assets
├── resources/      # Views and frontend assets
├── routes/         # Web and API routes
└── storage/        # File storage and logs
```

Environment Configuration: Create or update your `.env` file with the following essential configurations:

```
APP_NAME=RAGSystem
APP_ENV=local
APP_DEBUG=true

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=rag_system
DB_USERNAME=root
DB_PASSWORD=

OPENAI_API_KEY=your_api_key_here
ELASTICSEARCH_HOST=localhost:9200
```

4. Hands-on Implementation

Let's set up the basic components needed for our RAG system.

Setting Up Elasticsearch: Create a `docker-compose.yml` file at the project root:

```yaml
version: '3.8'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.7.0
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
    ports:
      - "9200:9200"
    volumes:
      - elasticsearch-data:/usr/share/elasticsearch/data

volumes:
  elasticsearch-data:
```

Run the container:

```
docker-compose up -d
```

Creating Basic Models: Create a `Document` model (`app/Models/Document.php`):

```php
<?php
```

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Document extends Model
{
    protected $fillable = [
        'title',
        'content',
        'embedding',
        'metadata'
    ];

    protected $casts = [
        'metadata' => 'array'
    ];
}
```

Setting Up Services: Create a basic `RagService` class (`app/Services/RagService.php`):

```php
<?php

namespace App\Services;

use OpenAI\Client;
use Elasticsearch\Client as ElasticsearchClient;

class RagService
{
    protected $openai;
    protected $elasticsearch;

    public function __construct(Client $openai, ElasticsearchClient
$elasticsearch)
    {
        $this->openai = $openai;
        $this->elasticsearch = $elasticsearch;
    }

    public function generateEmbedding(string $text)
    {
        // Implementation will be covered in Day 3
    }
}
```

5. Testing Your Setup

Verify that your environment and basic components are working correctly.

Verify Environment: Run these commands in your terminal:

```
php -v      # Check PHP version
composer -V # Check Composer version
node -v     # Check Node.js version
docker --version
docker-compose --version
```

Test Elasticsearch Connection:

```
curl http://localhost:9200
```

You should see a JSON response with Elasticsearch cluster information.

Run Laravel Development Server:

```
php artisan serve
npm run dev
```

Visit `http://localhost:8000` (or the address shown by `php artisan serve`) in your browser. You should see the Laravel welcome page.