d0531779-2ed9-4771-a181-a795708a2527

Job Analyzer Pro - Complete Setup Guide

🚀 Quick Start Instructions

1. Create Project Structure

```bash
# Navigate to your XAMPP htdocs directory
cd C:\xampp\htdocs  # Windows
# or
cd /Applications/XAMPP/htdocs  # Mac

# Create project directory
mkdir job-analyzer
cd job-analyzer
```

2. Install Dependencies

```bash
# Initialize Composer
composer init

# Install all required packages
composer require slim/slim:"4.*"
composer require slim/psr7
composer require illuminate/database
composer require vlucas/phpdotenv
composer require smalot/pdfparser
composer require spatie/pdf-to-text
composer require tecnickcom/tcpdf
composer require guzzlehttp/guzzle
composer require respect/validation
composer require firebase/php-jwt
composer require league/flysystem
composer require twig/twig
composer require symfony/process
composer require ramsey/uuid
composer require nesbot/carbon
composer require monolog/monolog
```

3. Database Setup

1. Start XAMPP (Apache + MySQL)

2. Go to http://localhost/phpmyadmin
3. Create database: `job_analyzer`
4. Import the provided SQL schema
5. Update `.env` with your database credentials

4. Create Folder Structure

Create all folders as shown in the folder structure artifact, then create the missing essential files:

app/Models/User.php

```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;

class User extends Model
{
    protected $fillable = ['username', 'email', 'password_hash'];
    protected $hidden = ['password_hash'];

    public function jobs()
    {
        return $this->hasMany(Job::class);
    }
}
```

app/Models/AnalysisLog.php

```php
```

```php
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;

class AnalysisLog extends Model
{
    protected $table = 'analysis_logs';
    protected $fillable = ['job_id', 'analysis_id', 'log_level', 'message', 'additional_data'];
    protected $casts = ['additional_data' => 'array'];

    public function job()
    {
        return $this->belongsTo(Job::class);
    }
}
```

app/Services/PDFParserService.php

```php
<?php
namespace App\Services;
use Smalot\PdfParser\Parser;

class PDFParserService
{
    public function extractText(string $filePath): string
    {
        try {
            $parser = new Parser();
            $pdf = $parser->parseFile($filePath);
            return $pdf->getText();
        } catch (\Exception $e) {
            throw new \Exception("Failed to extract PDF text: " . $e->getMessage());
        }
    }
}
```

app/Services/AIAnalysisService.php

```php
```

```php
<?php
namespace App\Services;
use GuzzleHttp\Client;

class AIAnalysisService
{
    private $client;
    private $serviceUrl;

    public function __construct()
    {
        $this->client = new Client(['timeout' => 30]);
        $this->serviceUrl = $_ENV['AI_SERVICE_URL'] ?? 'http://localhost:5000';
    }

    public function analyzeWithAI(string $jobDescription, string $resumeText, string $coverLetterText): array
    {
        try {
            $response = $this->client->post($this->serviceUrl . '/analyze', [
                'json' => [
                    'job_description' => $jobDescription,
                    'resume_text' => $resumeText,
                    'cover_letter_text' => $coverLetterText
                ]
            ]);

            return json_decode($response->getBody()->getContents(), true);
        } catch (\Exception $e) {
            throw new \Exception("AI analysis failed: " . $e->getMessage());
        }
    }

    public function checkStatus(): array
    {
        try {
            $response = $this->client->get($this->serviceUrl . '/health');
            return ['status' => 'running', 'message' => 'AI service is operational'];
        } catch (\Exception $e) {
            return ['status' => 'stopped', 'message' => 'AI service is not running'];
        }
    }

    public function startService(): array
    {
        // Implementation to start Python service
        return ['success' => true, 'message' => 'Service started'];
```

```
    }
}
```

app/Validators/JobValidator.php

```php
<?php
namespace App\Validators;

class JobValidator
{
    public function validate(array $data, ?int $jobId = null): array
    {
        $errors = [];

        if (empty($data['job_title'])) {
            $errors['job_title'] = 'Job title is required';
        }

        if (empty($data['job_description'])) {
            $errors['job_description'] = 'Job description is required';
        } elseif (strlen($data['job_description']) < 50) {
            $errors['job_description'] = 'Job description must be at least 50 characters';
        }

        return [
            'valid' => empty($errors),
            'errors' => $errors
        ];
    }
}
```

app/Validators/FileValidator.php

```php
```

```php
<?php
namespace App\Validators;

class FileValidator
{
    public function validateResume($uploadedFile): array
    {
        return $this->validateFile($uploadedFile, ['pdf', 'doc', 'docx']);
    }

    public function validateCoverLetter($uploadedFile): array
    {
        return $this->validateFile($uploadedFile, ['pdf', 'doc', 'docx']);
    }

    private function validateFile($uploadedFile, array $allowedTypes): array
    {
        $errors = [];

        if ($uploadedFile->getSize() >
```