Project Timeline: AI Resume Parser and Enhancement System

Week 1: Project Setup

Goals

- Establish the foundation for the project.

Tasks

1. Finalize the tech stack for both backend and frontend.
2. Set up backend project structure (Flask/Django/FastAPI).
3. Configure the frontend environment with React.js.
4. Set up version control (Git/GitHub) and initial documentation.

Week 2: Basic Resume Parsing

Goals

- Extract essential information from uploaded resumes.

Tasks

1. Integrate libraries like spaCy, PyPDF2, and pdfminer for basic parsing.
2. Extract name, contact info, and education details.
3. Test parsing functionality with a variety of resume formats.

Week 3: Advanced Resume Parsing

Goals

- Expand parsing capabilities to include skills and work experience.

Tasks

1. Use advanced techniques (e.g., ResumeParser API) for structured information extraction.
2. Enhance the parser to recognize keywords like certifications, achievements, and languages.
3. Test and refine the parsing logic with real-world resumes.

Week 4: Database and Storage

Goals

- Configure secure data storage.

Tasks

1. Choose and configure a database (PostgreSQL, MongoDB, or SQLite).

2. Set up cloud storage (AWS S3 or Google Cloud Storage) for uploaded resumes.
3. Implement database schema for user data and resume metadata.

Week 5: File Upload Module

Goals

- Enable users to upload resumes.

Tasks

1. Develop a file upload module supporting PDF and DOCX formats.
2. Write backend logic to validate file types and sizes.
3. Create a basic frontend interface for file upload.

Week 6: Skill Matching and Recommendation

Goals

- Build a recommendation engine for skills.

Tasks

1. Implement a model using Hugging Face Transformers or BERT to analyze skills.
2. Integrate LinkedIn API to fetch job descriptions for comparison.
3. Develop a system to suggest skills that align with industry trends.

Week 7: Grammar and Language Correction

Goals

- Enhance resumes by detecting and correcting language issues.

Tasks

1. Integrate Grammarly API or LanguageTool for grammar correction.
2. Use Hugging Face models (e.g., t5-base or BART) for rephrasing suggestions.
3. Design a UI for users to view and accept corrections.

Week 8: Resume Template Generator

Goals

- Create visually appealing resume templates.

Tasks

1. Design multiple resume templates using ReportLab, FPDF, or LaTeX.
2. Implement backend logic for generating resumes in PDF format.
3. Develop a frontend feature to preview and select templates.

Week 9: Authentication and Security

Goals

- Ensure secure user access and data management.

Tasks

1. Implement user registration and login using OAuth 2.0 or JWT.
2. Secure file uploads with encryption.
3. Add role-based access control for admin and users.

Week 10: Testing and Deployment

Goals

- Finalize and deploy the application.

Tasks

1. Conduct unit and integration testing for all modules.
2. Use Docker to containerize the application.
3. Deploy the project on AWS, Heroku, or a similar platform.
4. Gather feedback from users and make final adjustments.