Detailed Project Plan

Phase 1: Planning & Infrastructure Setup (Week 1-2)

1. **Requirements Gathering**
    - Define types of PII to detect (e.g., government IDs, emails, phone numbers, addresses).
    - Identify all data sources (e.g., S3 buckets, local storage).
    - Study Presidio's requirements, architecture, and setup.
2. **Tool & Framework Setup**
    - Install dependencies like `tesseract-ocr`, `spacy`, `nltk`, `opencv`.
    - Set up local and cloud infrastructure for S3, Azure, and Google Cloud storage.
    - Install and configure OCR tools (`Keras-OCR`, `EasyOCR`, `docTR`).
    - Explore and test OCR options:
        - **Keras-OCR**: Setup, pre-trained model tests, and fine-tuning for Devanagari and Latin scripts.
        - **PyTesseract**: Implement OCR with image enhancement using CV techniques.
        - **EasyOCR**: Test for PII extraction and map results.
        - **docTR**: Test its KIE model for advanced document parsing.
3. **Architecture Design**
    - Design data pipeline for file retrieval, OCR processing, and PII classification using Presidio.

Phase 2: Data Collection & Preprocessing (Week 3)

4. **Data Input Handling**
    - Develop scripts to fetch documents (images, PDFs, text files) from local directories and cloud storage.
5. **Preprocessing Images**
    - Use OpenCV for image enhancement: grayscale conversion, auto-rotation, deskewing, thresholding.

Phase 3: PII Detection (Week 4-5)

6. **OCR & Text Extraction**
    - Apply Tesseract OCR and other tools to extract text into structured formats.
7. **Regex for PII Identification**

- Write regular expressions to detect emails, phone numbers, and ID formats.
- Implement custom regex for regional identifiers like PAN cards or SSNs.

8. **NLP for Advanced PII Detection**
   - Leverage Spacy/NLTK for contextual data extraction (addresses, names, etc.).
   - Use Presidio's NER for enhanced PII classification.
9. **Confidence Scoring & Classification**
   - Assign confidence scores to PII using predefined keyword matches and patterns.
   - Map files to specific PII categories (e.g., passport, driver's license).

Phase 4: Automation & Integration (Week 6)

10. **Automated Scanning**
    - Automate periodic scans via AWS Lambda triggers for S3 or cron jobs for local storage.
11. **Error Handling & Resilience**
    - Implement robust logging and handling for low-quality/corrupted files.

Phase 5: Output Generation & Reporting (Week 7)

12. **File Processing & Result Storage**
    - Store outputs in JSON/CSV with details like file path, detected PII, and classification.
13. **Reporting**
    - Create stakeholder reports summarizing findings, with features like email notifications or a monitoring dashboard.

Phase 6: Testing, Documentation & Deployment (Week 8)

14. **Testing**
    - Validate accuracy with dummy data across formats (JPG, PNG, PDF, DOC, TXT).
    - Minimize false positives/negatives by refining logic.
15. **Documentation**
    - Document installation, usage, and troubleshooting.
    - Create user guides and technical references.
16. **Deployment**
    - Deploy the system in production with necessary security protocols.
    - Set up monitoring for new PII leaks or errors.

Phase 9: Frontend Development (Week 9-10)

17. **Initial Frontend Development Using Streamlit**
    - Create an interactive interface for:
        - Uploading files for scanning.
        - Displaying detected PII and associated confidence scores.
        - Offering options to download results in JSON/CSV formats.
    - Make the interface intuitive with tabs or sections for different functionalities (OCR preview, PII summary, etc.).

Additionally,

- **UI Enhancement with ReactJS**
    - Transition the basic Streamlit interface to a professional ReactJS-based UI:
        - Create responsive designs using Material-UI or Bootstrap.
        - Add dynamic visualizations for PII reports (e.g., graphs, charts).
        - Integrate filtering and search functionalities for PII details.
        - Enable real-time file processing progress indicators.
    - Set up backend integration with the processing pipeline via REST APIs.

- **Frontend Testing & Deployment**
    - Test the ReactJS-based frontend for responsiveness, cross-browser compatibility, and smooth integration with the backend.
    - Deploy the frontend on a hosting platform (e.g., AWS Amplify, Netlify).

Timeline Summary

- **Weeks 1-8**: Core pipeline, OCR, PII detection, and reporting.
- **Weeks 9-10**: Frontend development and deployment.