Arabic PII Redaction: An Accelerated Implementation Plan with Replit & Claude

This document provides a detailed, week-by-week implementation plan to execute the strategic blueprint for the Arabic PII Redaction Challenge. It is designed to be executed by a small, agile team, leveraging Replit as the core development and collaboration platform and Claude as an AI coding partner to maximize speed and efficiency.

Core Tenets of this Plan

- **Speed through Acceleration:** We will use Claude to generate boilerplate code, complex logic (like regex and model architecture), and synthetic data templates, allowing the team to focus on high-level strategy and tuning.
- **Collaboration-First:** Replit will serve as the "single source of truth," providing a zero-setup, collaborative environment where the entire team can code, test, and deploy simultaneously.
- **Iterative MVP Development:** The plan is structured to build and test components incrementally, starting with a simple rule-based MVP and progressively layering on more sophisticated models.

Tech Stack & Environment

- **Development Environment: Replit**
  - **Workspace:** A shared Replit workspace for all team members.
  - **Repl Type:** A Python Repl, upgraded to a "Core" plan for access to more powerful machines ("Boosted Repls") and background execution for model training.
  - **Package Management:** The replit.nix file will be used to define and manage all Python dependencies (PyTorch, Hugging Face, etc.), ensuring a consistent environment for everyone.
- **AI Coding Assistant: Claude (Pro or API)**
  - **Role:** Code generation, regex creation, synthetic data templating, debugging, and documentation.
- **Core Libraries (to be listed in replit.nix):**
  - torch, torchvision, torchaudio
  - transformers, datasets, seqeval
  - pandas
  - scikit-learn
  - pytorch-crf
  - fastapi or flask (for creating a simple testing API)

Week 1: Setup, Data Acquisition, and Rule-Based MVP

Goal: Establish the development environment and build a functioning, high-precision rule-based system. This provides an immediate, testable baseline.

Day	Task	Key Activities & Claude Prompts
Day 1	**Environment Setup**	1. Create a shared Replit Workspace and a new Python Repl. 2. Upgrade to the Replit Core plan to enable more powerful hardware. 3. Configure the replit.nix file to install all necessary Python libraries. Claude Prompt: "Create a replit.nix file for a Python project. It needs to include torch, transformers, datasets, seqeval, pandas, and pytorch-crf."
Day 2-3	**Data Acquisition & Exploration**	1. Download the Wojood NER dataset. 2. Write Python scripts in Replit to load, parse, and analyze the dataset's structure, entity distribution, and text formats. Claude Prompt: "Write a Python script using pandas to load a TSV file named 'wojood.tsv', inspect the first 50 rows, and print a count of unique values in the 'entity_type' column."
Day 4-5	**Rule-Based Engine (MVP)**	1. Develop the rule-based engine in a rules.py file within Replit. 2. Focus on high-precision regex for structured PII: phone numbers, emails, National IDs, and IBANs. Claude Prompt: "Write a Python function that takes a

		string of Arabic text and uses the re module to find and return all occurrences of Jordanian mobile numbers. The numbers can start with +962, 00962, or 07 and may contain spaces." (Repeat for other PII types).
Day 6-7	**Test & Deploy MVP**	1. Create a simple web interface in Replit using FastAPI to test the rule-based engine interactively. 2. Thoroughly test the regex patterns with edge cases. Claude Prompt: "Create a simple FastAPI app in a main.py file. It should have one POST endpoint /mask that accepts a JSON body {'text': '...'}. The endpoint should import the functions from rules.py, apply them to the text, and return the masked text."

Week 2: Data Augmentation & Baseline Model Training

Goal: Create the comprehensive training dataset and train the first transformer-based NER model to establish a performance baseline.

Day	Task	Key Activities & Claude Prompts
Day 8-9	**Data Augmentation**	1. Implement the schema mapping from Wojood to the 7 PII types. 2. Build the synthetic data generator. Claude Prompt: "I need to generate synthetic Arabic sentences for PII detection. Give me 20 diverse Python f-string templates for sentences that would contain a [NAME], for example: f'تم تعيين

		{name} في منصب جديد.'. Then, give me templates for [ADDRESS] and [ID_NUMBER]." 3. Use the generated regex and gazetteers to populate these templates and create a large train_augmented.csv file.
Day 10-11	**Preprocessing & Label Alignment**	1. Write the full preprocessing pipeline as described in the strategic plan (normalization, subword tokenization). 2. Implement the critical label alignment logic (mapping word-level labels to subword tokens). This is a complex step and an excellent use case for Claude. Claude Prompt: "Write a Python function that takes a list of words and a corresponding list of NER labels. It should use a Hugging Face tokenizer to tokenize the words. It must then align the labels with the subword tokens according to the IOB2 scheme. The first token of a word gets the original label, and subsequent sub-tokens get -100. Handle special tokens like [CLS] and [SEP]."
Day 12-14	**Baseline Model Training**	1. Write the fine-tuning script using the Hugging Face Trainer API. 2. Configure TrainingArguments and the compute_metrics function using seqeval. Claude Prompt: "Generate a complete Python script to fine-tune aubmindlab/bert-base-arabertv2 for token classification

		using the Hugging Face Trainer. The script should load a pre-tokenized dataset, define TrainingArguments, include a compute_metrics function with seqeval for precision, recall, and F1, and start the training." 3. Start the training run on a Boosted Repl. Monitor progress via logs.

Week 3: Advanced Model (CRF) and Hybrid System Integration

Goal: Improve sequence tagging consistency with a CRF layer and integrate all components into the final hybrid architecture.

Day	Task	Key Activities & Claude Prompts
Day 15-17	**Advanced Model (AraBERT+CRF)**	1. Refactor the baseline model into a custom PyTorch class that incorporates a CRF layer. Claude Prompt: "I need to add a CRF layer to my Hugging Face transformer model for NER. Show me how to create a custom PyTorch nn.Module class that wraps a BERT model and a CRF layer from the pytorch-crf library. The forward pass should handle calculating the loss during training and decoding the best path during evaluation." 2. Modify the training script to use a custom training loop for the new CRF model.
Day 18-19	**Hybrid System Integration**	1. Write the final inference pipeline script (pipeline.py). 2. This script should implement the "rules-first, model-second" logic. Claude Prompt: "Write a Python class

		PIIPipeline that takes a fine-tuned model and a tokenizer. It should have a mask(text) method that first applies the functions from rules.py to the text, and then passes the remaining unmasked text to the transformer model for further masking. It should then combine the results and return the final masked sentence."
Day 20-21	**End-to-End Testing**	1. Update the FastAPI app in Replit to use the new PIIPipeline. 2. Rigorously test the full hybrid system with a variety of sentences from your validation set, checking for both accuracy and correct integration.

Week 4: Optimization, Final Tuning, and Submission

Goal: Maximize the final competition score by optimizing for speed and fine-tuning the precision/recall balance.

Day	**Task**	**Key Activities & Claude Prompts**
Day 22-23	**Speed Optimization (Quantization)**	1. Implement Post-Training Dynamic Quantization on the fine-tuned AraBERT+CRF model. Claude Prompt: "Write a Python script that loads my fine-tuned PyTorch model from a saved directory. It should then apply post-training dynamic quantization using torch.quantization.quantize_dynamic to the Linear layers

		and save the quantized model." 2. Benchmark the inference speed of the FP32 vs. INT8 models in Replit to quantify the speed-up.
Day 24-25	**Score Optimization (Threshold Tuning)**	1. Modify the inference pipeline to expose a confidence threshold for the transformer model's predictions. 2. Write a script to iterate through different threshold values (e.g., from 0.5 to 0.99), run predictions on the validation set, and calculate the final weighted competition score for each threshold. This will identify the optimal balance.
Day 26-27	**Error Analysis & Final Polish**	1. Generate a confusion matrix on the validation set predictions. 2. Analyze the most common errors. If time permits, generate a small set of new training examples that specifically target these errors and run a final, brief re-training. 3. Clean up the code, add documentation and comments. Claude Prompt: "Generate docstrings for all functions in my pipeline.py script."
Day 28	**Packaging & Submission**	1. Prepare the final submission package as per the competition's rules. 2. Ensure the inference script is clean and runs correctly in a fresh environment. 3. Submit the solution.

Cost Analysis for MVP (1-Month Sprint)

This plan is designed to be highly cost-effective while delivering a competitive result.

Item	Service	Estimated Cost (Low End)	Estimated Cost (High End)	Justification
Development Platform	Replit Core Plan	$20	$20	**Essential.** Provides the necessary compute power (Boosted Repls), background execution for training, and collaboration features.
AI Assistant	Claude Pro	$20	$20	**Essential.** For a single developer or small team, the Pro subscription offers enough usage for all code generation, debugging, and templating needs.
GPU Compute (Optional)	Google Colab Pro	$10	$50+	**Optional but Recommended.** While Replit's boosted CPUs are fast, GPU access (via Colab or a cloud provider) can drastically reduce training times from hours to minutes, allowing for

				more iterations. The high-end reflects using a dedicated cloud GPU service for a few hours.
Total Estimated MVP Cost		**~$50**	**~$90+**	The $1000 prize comfortably covers even an aggressive development budget, a very high-ROI decision.

By adhering to this accelerated plan, your team can systematically build upon the strategic research, leveraging Replit and Claude to handle the heavy lifting of implementation and allowing you to focus on what truly wins competitions: superior data, intelligent architecture, and meticulous optimization.