Phase 1: Foundation & Scoping (Weeks 1-2)

Goal: Establish the logical core of the project and set up the technical environment.

- **Week 1: Rule Scoping & Prolog Backend**
    - **Task 1: Research and Finalize Legal Aid Rules.**
        - Source: The Legal Services Authorities Act, 1987 (India).
        - Action: Identify a clear, manageable subset of 5-7 eligibility rules.
        - **Example Rule Set:**
            - **Categorical:** Is the applicant a woman or child? Are they a member of a Scheduled Caste (SC) or Scheduled Tribe (ST)? Are they an industrial workman?
            - **Financial:** Is their annual income below a specific threshold (e.g., ₹1,00,000)?
            - **Case Type Exclusions:** Is the case related to defamation, economic offenses, or malicious prosecution?
    - **Task 2: Develop the Prolog Knowledge Base (legal_aid_eligibility.pl).**
        - Action: Write the Prolog script based on the rules defined above. Use your REFERENCE_prolog_system.pl as a structural guide. Define dynamic predicates for facts the NLP model will provide (e.g., income_annual/2, is_woman_or_child/2).
- **Week 2: Environment Setup & Python-Prolog Bridge**
    - **Task 1: Set Up Your Coding Environment.**
        - Action: Install Python, SWI-Prolog, and the necessary Python libraries (pyswip, pandas, scikit-learn, spacy, torch, transformers).
    - **Task 2: Build the Python-Prolog Interface.**
        - Action: Create a Python module (prolog_bridge.py) that uses pyswip.
        - **Key Functions:**
            - load_rules(filepath)
            - clear_facts()
            - assert_fact(fact_string)
            - query_eligibility(person_id) (returns True/False)

Phase 2: Data Creation & Annotation (Weeks 3-4)

Goal: Create a high-quality, novel dataset—the foundation of your research.

- **Week 3: Dataset Generation**
    - **Task 1: Design the Annotation Schema.**

        - Action: Create a spreadsheet (legal_queries.csv) with the columns: query_id, query_text, entities_present (a list of strings), and one column for each ground truth Prolog fact (e.g., fact_income, fact_case_type).
    - **Task 2: Write Realistic Queries.**
        - Action: Write 100-150 queries. Ensure variety in phrasing, legal issues, and socio-economic details.
        - *Example Query:* "I am a widow and my son is 12. My former employer fired me without paying my last two months' salary. My yearly income is around 80,000 rupees. Can I get help?"
- **Week 4: Data Annotation**
    - **Task 1: Annotate the Entire Dataset.**
        - Action: Go through each query and meticulously fill in the annotation columns.
        - *Annotation for Example Query:*
            - entities_present: ["categorical_status", "income", "case_type"]
            - fact_categorical_status: is_woman_or_child(q1, true).
            - fact_income: income_annual(q1, 80000).
            - fact_case_type: case_type(q1, labor_dispute).
    - **Task 2: Split the Dataset.**
        - Action: Programmatically split your .csv file into train.csv (70%), validation.csv (15%), and test.csv (15%).

Phase 3: NLP Pipeline Implementation (Weeks 5-8)

Goal: Build the core NLP system that translates natural language into logical facts.

- **Week 5: High-Precision Extractors (Stage 2)**
    - **Task 1: Build Keyword and Regex-Based Extractors.**
        - Action: Create a Python module (extractors.py).
        - extract_income(text): Uses regex to find numbers near currency symbols or words like "income," "salary," "earn."
        - extract_categorical_status(text): Uses keyword matching (e.g., "woman," "child," "adivasi," "dalit") to return facts like is_woman_or_child(true).
- **Week 6: Novel Component - Case Type Classifier**
    - **Task 1: Train the Case Type Classifier.**
        - Action: Using your train.csv, train a simple text classification model (e.g., TF-IDF + Logistic Regression from scikit-learn).

        - Input: query_text.
        - Output: A single case type label (e.g., "labor_dispute", "property_dispute"). This is a key part of your Stage 2 extractors.

- **Week 7: Coarse-Grained Classifier (Stage 1)**
    - **Task 1: Train the Entity Presence Classifier.**
        - Action: Using your train.csv, train a multi-label classification model.
        - Input: query_text.
        - Output: A list of entities present (e.g., ["income", "case_type"]). This model will orchestrate your pipeline.
- **Week 8: Pipeline Integration**
    - **Task 1: Write the Main Processing Logic (main.py).**
        - Action: Create a function process_legal_query(text) that performs the full end-to-end process:
            1. Calls the Stage 1 classifier to get the list of entities.
            2. Loops through the list, calling the appropriate Stage 2 extractor for each entity.
            3. Collects all generated Prolog facts.
            4. Uses your prolog_bridge to clear old facts, assert the new ones, and query for the final eligibility decision.

Phase 4: Evaluation and Paper Writing (Weeks 9-12)

Goal: Prove your system's effectiveness and document your research contribution.

- **Week 9: Baseline Implementation & Evaluation**
    - **Task 1: Implement Baseline Models.**
        - **LLM-Only Baseline:** Use a model API (e.g., from Hugging Face or another provider) with a detailed prompt asking it to extract all facts in the correct Prolog format.
        - **Regex-Only Baseline:** A script that uses only regular expressions to try and extract all facts.
    - **Task 2: Run Evaluation Suite.**
        - Action: Write a script (evaluate.py) that iterates through your test.csv and runs your HybEx-Law pipeline, plus both baselines, on each query. Store all results.
- **Week 10: Results Analysis & Paper Writing (Methodology)**
    - **Task 1: Calculate and Tabulate Results.**

        - Action: Calculate the Fact-Level F1-Score and the End-to-End Task Success Rate for all three models. Create a clear results table for your paper.
    - **Task 2: Write the "Methodology" and "Experiments" Sections.**
        - Action: Detail your pipeline architecture, dataset creation process, and the evaluation protocol.
- **Week 11: Paper Writing (Introduction & Conclusion)**
    - **Task 1: Write the Introduction and Related Work.**
        - Action: Introduce the "access to justice" problem, state your contribution, and position your work within the context of neuro-symbolic AI research.
    - **Task 2: Write the Conclusion and Future Work.**
        - Action: Summarize your findings, re-emphasize the superiority of the hybrid approach, and suggest future improvements (e.g., expanding the rule set, improving the case type classifier).
- **Week 12: Final Review and Submission**
    - **Task 1: Assemble and Polish the Full Paper.**
        - Action: Combine all sections, add diagrams (like the pipeline architecture), proofread, and check formatting.
    - **Task 2: Prepare Code for Submission.**
        - Action: Clean up your code, add comments, and create a README.md file explaining how to run your project.