Compiler Practical Projects

1. Chatbot Command Language

Goal: Build a tiny language for chatbot commands.

- **Lexical analysis:** Tokenize commands like `/remind`, `/schedule`, `/translate` and their arguments.
- **Parsing:** Convert `/remind "Buy milk" at 5pm` into a structured object (action, message, time).
- **Application:** Use the parsed structure to trigger simple actions (print reminders, schedule messages).

---

2. Math Plotting Language

Goal: Build a tiny language where a user can type math expressions and plotting commands.

- **Lexical analysis:** Tokenize `PLOT sin(x)*x FROM 0 TO 10`.
- **Parsing:** Build an AST describing the function, domain, and options.
- **Application:** Use `matplotlib` to plot the function after parsing.

---

3. Simple Music/Sequence Generator

Goal: Design a tiny language for composing very simple musical sequences or sound effects.

- **Lexical Task:** Tokenize musical notes (`C4`, `G3`), durations (`.25s`, `half`), and simple commands (`PLAY`, `WAIT`).
- **Parsing Task:** Parse the sequence of commands, ensuring a valid structure. Example input: `PLAY C4 .5s; WAIT .1s; PLAY G3 .5s;`.
- **Application:** Use a Python sound library (like **`Pygame.mixer`** or **`Midiutil`**) to interpret the parsed AST and actually generate the sound or a MIDI file.

---

4. Neural Network Architecture DSL

Goal: Create a tiny Domain-Specific Language (DSL) to define a simple, sequential neural network model.

- **Lexical analysis:** Tokenize keywords like `MODEL`, `INPUT`, `LAYER`, `DENSE`, `CONV` and numerical parameters (e.g., neuron counts, kernel size).

- **Parsing:** Parse the sequence of layers into a structured Model AST (Abstract Syntax Tree).

Example Syntax:

```
MODEL MyNet:
    INPUT 784
    LAYER DENSE 128 ACTIVATION ReLU
    LAYER DENSE 10 ACTIVATION Softmax
```

Application: The resulting AST is traversed to call the appropriate functions in an ML library like TensorFlow/Keras or PyTorch to construct the actual model.

---

5. Feature Engineering Script Parser

Goal: Allow a user to define a sequence of data preprocessing steps using a simple script.

- **Lexical analysis:** Tokenize commands like `LOAD`, `DROP`, `SCALE`, `NORMALIZE`, column names (identifiers), and numerical arguments.
- **Parsing:** Define a grammar for feature transformation pipelines. Ensure logical flow (e.g., `DROP` before `SCALE`).

Example Syntax:

```
LOAD "data.csv"
DROP column_A, column_B
SCALE feature_C MINMAX
OUTPUT "transformed_data.csv"
```

Application: Use Pandas to load the data, then execute the methods (`df.drop()`, `MinMaxScaler().fit_transform()`) based on the parsed AST.

---

6. Basic Prompt Template Validator (Generative LLMs)

Goal: Design a parser for a structured prompt template used to communicate with a Generative LLM.

- **Lexical analysis:** Define tokens for literal text (strings), reserved symbols `{` and `}`, and parameter names (`{topic}`, `{style}`).
- **Parsing:** Ensure that the template contains all required placeholders and that nesting (if allowed) is correct.

Example Syntax:

```
"Please write a {style} poem about {topic}. The author should be {author}."
```

Application: After validating the structure, the program accepts a dictionary of values (e.g. `{"style": "haiku", "topic": "rain"}`) and performs substitution, replacing placeholders with actual values to generate the final LLM prompt string.

---

7. Simple Image Filter Pipeline Language

Goal: Create a language to define a chain of image filters.

- **Lexical analysis:** Tokenize keywords like `LOAD`, `SAVE`, `BLUR`, `CROP`, `RESIZE`, numerical arguments (e.g., size values, kernel radius), and file paths (strings).
- **Parsing:** Parse the command sequence and its parameters.

Example Syntax:

```
LOAD "photo.jpg"
RESIZE 500x300
BLUR radius=5
SAVE "output.png"
```

Application: Use OpenCV or Pillow (PIL) to load the image and apply the functions sequentially based on the parsed list of instructions.

---

8. Color Palette Generator DSL

Goal: Allow users to define a color palette using a simple, human-readable language.

- **Lexical analysis:** Tokenize color names (red, blue), adjectives/tones (light, dark, vibrant), and numerical counts (e.g., `5 colors`).
- **Parsing:** Parse the descriptive request into a structured data object.

Example Syntax:

```
GENERATE PALETTE 5 colors from (light blue and dark orange)
```

Application: The parsed parameters are used as inputs to a logic function (or a tiny pre-trained ML model, as a stretch goal) that produces a list of hexadecimal color codes.