# **Project** This project will guide you through designing a rule-driven learning app, modeling real-world scenarios with graphs, using rigorous proofs, and conducting human-centric experiments .

# **Part A: Rule-Based Learning App Design (Shapes & Colors)**

### **Objective:**

Design a basic rule-driven app to teach primary shapes and colors.

### **Tasks:**

* Define Primary Shapes and Primary Colors.
* Define Color Mixing Rules (e.g., Red + Blue = Purple).
* Define Shape Combination Rules (e.g., Triangle + Square = House).
* Create at least **7 Basic Rules** (for primary color/shape recognition).
* Create at least **7 Advanced Rules** (for color mixing and shape merging).
* Create evaluation queries in Prolog to validate the defined rules.

### **Deliverables:**

* Written rulebook.
* Prolog facts and rules for shapes, colors, mixing, and merging.
* Prolog evaluation queries for self-testing.

### **Prolog Starter Knowledge Base:**

**Basic Facts**

primary\_color(red).

primary\_color(blue).

primary\_color(yellow).

primary\_shape(circle).

primary\_shape(square).

primary\_shape(triangle).

mix\_colors(Color1, Color2, ResultColor).

merge\_shapes(Shape1, Shape2, ResultShape).

### **Queries for Evaluation:**

|  |  |  |
| --- | --- | --- |
| **Query No** | **Command** | **Purpose** |
| 1 | ?- primary\_color(red). | Check if 'red' is defined as a primary color. |
| 2 | ?- primary\_shape(circle). | Check if 'circle' is defined as a primary shape. |
| 3 | ?- mix\_colors(red, blue, Result). | Verify mixing red and blue gives the correct secondary color. |
| 4 | ?- merge\_shapes(triangle, square, Result). | Verify merging triangle and square produces correct shape. |
| 5 | ?- mix\_colors(blue, yellow, green). | Confirm blue and yellow mix into green. |
| 6 | ?- mix\_colors(red, red, \_). | Confirm same colors should not create a new color. |
| 7 | ?- composite\_shape(triangle, triangle, Result). | Validate advanced merging of two triangles. |
| 8 | ?- valid\_color\_mix(red, yellow, Result). | Check validity of another mixed color. |
| 9 | ?- valid\_shape\_merge(circle, rectangle, Result). | Check validity of merging circle and rectangle. |
| 10 | ?- secondary\_color(Color). | Confirm a color is recognized as secondary. |

## **Part B: Graph Theory & Matrix Representation**

### **Tasks:**

* Represent shapes as nodes and their adjacency as edges.
* Create Adjacency Matrix
* Implement constraints (e.g., no two adjacent shapes have the same color).
* Write a basic program (Python/Java/C++) to validate graph coloring and adjacency rules.
* (Bonus) Find Euler and Hamiltonian Circuits in the graph.

### **Deliverables:**

* Adjacency matrix
* Graph representation diagram.
* Working validation code.
* Euler/Hamiltonian circuit analysis (optional).

## **Part C: Mathematical Proofs and Logic**

### **Tasks:**

* Prove that combining two specific shapes logically results in another shape.
* Prove that mixing two primary colors logically results in a secondary color.
* Formalize proofs using proper discrete mathematics structure.

### **Deliverables:**

* Two formal direct proofs.

## **Part D: Human-Curated Emergent Rules Discovery (Advanced Section)**

### **Tasks:**

* Conduct 20 random experiments combining different shapes and colors.
* Record observations of unexpected outcomes (e.g., new shapes, new perceived colors, visual illusions).
* Formulate at least 5-7 new rules based on observed emergent behaviors.
* Validate findings by conducting mini user-tests (optional but recommended).
* Analyze why these emergent rules could not be predicted purely through logic or AI.

### **Deliverables:**

* Experimental Log Book.
* Emergent Rules List.
* Analysis Report (2-3 pages).

# **Overall Deliverables**

Each group must submit:

1. **Rules Document** (Part A)
2. **Graph Files** (Part B)
3. **Proofs Document** (Part C)
4. **Analysis Report** (Part D)

# 