Mechanism for Reviewing and Integrating LLM-Collected Facts

Objective: Provide a user-facing interface and backend pipeline to surface the *learned* or *suggested* facts generated by the LLM ensemble (currently hidden), allow manual review, and integrate selected facts via the `learn` command into the permanent knowledge base.

---

1. Backend: Capturing LLM Suggestions

1.1 **Extend Portfolio Manager**
- After each `ask` or `explain` call, capture any *temporary facts* emitted by the ensemble (e.g., from RAG context enrichment).
- Store these in a new in-memory list `learning_suggestions` alongside timestamps and source metadata.

1.2 **API Endpoint**
- Expose a new Flask route `GET /api/learning_suggestions`
- Returns JSON: `[{ "fact": "Predicate(args)", "source": "RAG|Ensemble", "time": "..." }, ...]`

1.3 **Integration Command**
- Extend `learn` command handler to accept an `id` or array of IDs from `learning_suggestions`.
- For each selected suggestion, invoke existing `learn_fact(fact)` routine; move from `learning_suggestions` to `permanentKnowledge`.

---

2. Frontend: User Interface

2.1 **New Panel: "Learning Suggestions"**
- Tab alongside Knowledge Base and Data Sources.
- Table listing:
- Fact string
- Source (RAG, Ensemble)
- Confidence or rank
- Timestamp
- Checkbox for selection

2.2 **Actions**
- **Integrate Selected** button: sends `POST /api/command` with `{ "cmd": "learn", "args": [<fact1>, <fact2>, ...] }`.
- **Dismiss**: remove from suggestions list without integration.

2.3 **Real-time Updates**
- Use WebSocket or periodic polling (`/api/learning_suggestions`) to refresh list.

---

3. Data Flow

1. User issues query → ensemble produces temporary facts → stored in `learning_suggestions`.
2. User navigates to "Learning Suggestions" panel, reviews facts.
3. User ticks checkboxes and clicks "Integrate" → frontend calls `learn` command.
4. Backend moves facts to permanent store, clears from suggestions.
5. PermanentKnowledge increases; UI updates accordingly.

---

4. Implementation Checklist

- [ ] Extend backend models: add `learning_suggestions` store
- [ ] Implement `/api/learning_suggestions` GET endpoint
- [ ] Enhance `learn` command handler to accept specific suggestions
- [ ] Frontend: create "Learning Suggestions" React component
- [ ] Add selection UI and integrate/dismiss actions
- [ ] Wire up polling or WebSocket for live updates
- [ ] Write unit tests for API and command integration

Outcome: This mechanism ensures full transparency over LLM-derived facts and empowers the user to curate the permanent knowledge base in a controlled, scientific workflow.