Phase 6 v2 – UI + Ollama Integration
(Merged & Expanded from Phase 6 v1)

---

1. Purpose of Phase 6

Phase 6 is where the entire backend prediction system becomes a real product—a local desktop/web application with:

- a navigable UI
- player projections
- matchup analysis
- start/sit recommendations
- trade and waiver intelligence
- team/game predictions
- explanations powered by Ollama

Phase 6 v1 described a basic UI with Ollama generating text.
Phase 6 v2 transforms that into a complete architectural system with:

- standardized APIs
- a modular React/Flask UI
- advanced explanation generators
- player cards and matchup dashboards
- defender + archetype influence visualization
- game projection views
- support for offline desktop use

This phase must support:
A fast, responsive, local-first prediction assistant with explainability built in.

---

2. UI Philosophy

The UI is not a data-entry tool.
It is a decision-support interface.
Design goals:

- fast
- context-rich
- actionable
- transparent
- no clutter
- mobile-friendly
- desktop-first

Phase 6 must provide:

✓ Player-level views

- statline projection
- fantasy projection
- volatility, floor/ceiling
- defender impact
- archetype impact
- recent form
- matchup difficulty

✓ Weekly overview

- top players by position
- streaming options
- matchup tiers

✓ Side-by-side comparisons
- for start/sit
- for trade evaluation

✓ Team-level game projection views
- predicted score
- pace
- top performers

✓ Natural-language explanations
- generated using Ollama
- directly tied to structured metadata

---

3. UI Architecture
The UI and backend communicate through a simple REST API layer.
3.1 Frontend (Recommended Stack)
You can choose between:
Option A (recommended): Local Web App
- React or Svelte frontend
- Runs locally
- Beautiful and fast
- No app store nonsense
- Fully cross-platform

Option B: Desktop app wrapper
- Electron (wrap the web app)
- or Tauri (lighter)

This satisfies your requirement:
“Eventually I want this as a desktop popup.”
Option C: Simple HTML/Bootstrap app
- Quickest to build
- Good enough for internal tools

---

4. Backend API (Flask or FastAPI)
The backend exposes endpoints for the UI to load predictions.
Base path:
/api/
4.1 Player Projection Endpoint
GET /api/player/{player_id}/projection?season=&week=
Returns:
- statline predictions
- fantasy projections
- matchup difficulty
- defender impact score
- archetype adjustments
- explanation metadata

4.2 Top Players Endpoint
GET /api/players/top?position=WR&week=6
Returns:
- list of top players by fantasy points
- streaming rankings
- matchup tier badges

4.3 Start/Sit Comparison

```
POST /api/compare
Body:
{
  "playerA_id": "...",
  "playerB_id": "...",
  "season": 2025,
  "week": 8
}
```
Returns:
- side-by-side predicted stats
- start probability
- matchup difficulty
- explanation metadata

4.4 Game Projection

```
GET /api/game/{game_id}/projection
```
Returns:
- projected score
- pace
- expected leaders
- matchup notes

4.5 Explanation Endpoint (Ollama call)

```
POST /api/explain
```
Body contains:
- predicted stats
- defender + archetype metadata
- matchup difficulty
- recent form deltas
- model/ensemble contributions

Backend passes structured info to Ollama model.

---

5. UI Page Structure

5.1 Home Dashboard (Weekly Overview)

Shows:
- top positional projections
- breakout candidates
- boom/bust indicators
- matchup tiers
- start/sit recommendations

5.2 Player Detail Page

For each player:
- predicted statline
- fantasy projection
- floor/ceiling
- volatility score
- defender impact breakdown
- archetype matchup explanation
- recent trend graphs
- schedule difficulty chart
- Ollama-generated natural language explanation

5.3 Compare Page

Used for:
- start/sit
- trade evaluation

Two columns:
- Player A
- Player B

Show:
- projections
- matchup tiers
- defender comparisons
- ROS value
- natural-language comparison summary

5.4 Waivers Page

Rank players by:
- rest-of-season value
- opportunity trend
- positional scarcity
- schedule difficulty
- upside rating
- breakout probability

5.5 Trades Page
- team-by-team trade suggestions
- trade value score
- projected ROS gain
- “synergy” with team needs
- natural-language trade explanations

5.6 Game Projections Page

Show:
- predicted score
- pace
- top fantasy performers
- defender matchup notes
- exploitable weaknesses

---

6. Natural Language Explanation Layer (Ollama)

6.1 What Ollama Does

Ollama does *not* produce new numerical predictions.

It takes:
- structured context
- metadata
- statlines
- model contributions

and generates:
- clean, readable insights
- matchup analysis
- player-specific reasoning
- defender/archetype explanations

6.2 Structured Prompt Template (Backend Assembles Context)

Backend creates JSON-ish context like:

{

```
  "player": {
    "name": "CeeDee Lamb",
    "position": "WR",
    "team": "DAL"
  },
  "projections": {
    "targets": 10.2,
    "receptions": 6.8,
    "yards": 89.4,
    "tds": 0.6
  },
  "meta": {
    "recent_form": "+12% vs season baseline",
    "defender_matchup": {
        "effective_defender_ypt": 1.22,
        "alignment": "Slot-heavy",
        "note": "Opposing slot CB trending down recently"
    },
    "archetype_interaction": "Favorable (+6%)",
    "matchup_difficulty": 41,
    "volatility": "moderate"
  },
  "model_contributions": {
    "baseline": 0.32,
    "xgb": 0.41,
    "ridge": 0.11,
    "defender_model": 0.09,
    "arch_model": 0.07
  }
}
```

And sends:

```
prompt = EXPLANATION_TEMPLATE + json.dumps(context)
```

Ollama then produces something like:

“CeeDee Lamb projects for 10 targets and about 90 yards this week.
His recent form (+12%) is trending upward, and the matchup against a struggling slot CB is favorable.
Archetype interaction also boosts his projection by about 6%.
Overall, expect WR1 production with moderate volatility.”

---

7. Explanation Types

Phase 6 v2 must support multiple explanation types:

7.1 Projection Explanation

Why this projection is the way it is.

7.2 Start/Sit Explanation

Short paragraphs comparing usefulness and safety.

7.3 Matchup Difficulty Explanation

Describing defender and defensive-team trends.

7.4 Trade Evaluation Explanation

Natural language summary of value gained/lost.

7.5 Waiver Recommendation Explanation

Opportunity trends, injuries, role changes, upside signals.

---

8. UI Visualization Requirements
8.1 Radar Charts
- defender strength vs archetype vulnerabilities
- player usage vs league average

8.2 Line Graphs
- recent form (last N weeks)
- opponent defense trending over time

8.3 Bar Graphs
- stat breakdown by model contribution
- start/sit probability weights

8.4 Icons/Badges
- matchup difficulty colors (green/yellow/red)
- volatility index icons
- defender alignment icons (slot, boundary, deep)
- archetype labels

Phase 6 must output graphics-ready metadata.

---

9. Desktop App / Local Popup Mode
You mentioned:
“The goal is eventually to have a desktop app I can open as a popup.”
Phase 6 v2 supports this via:
Option A — Electron/Tauri wrapper
Wrap the local web app (React + Flask) into a desktop application.
Option B — PySide6/PyQt6 native
Embed predictions + web view + icons + menu.
Option C — Tkinter lightweight (simple)
Not recommended for aesthetics, but fastest.
Offline-first design
All predictions run locally via:
- local SQLite DB
- local Python models
- local Ollama instance

No internet required.

---

10. Folder Structure for UI/Backend/Ollama

```
app/
   backend/
      api/
         projections.py
         compare.py
         games.py
         explain.py
      models/
      database/
      static/
      templates/
   frontend/
      src/
```

```
   public/
ollama/
   prompt_templates/
   system_prompts/
dist/ (for desktop builds)
```

---

11. Versioning & Testing

Each UI version is tied to:

- data_version
- model_version
- ensemble_version

UI should display:

“Predictions: Week 6 (Model 2025W06_Ensemble_v2.1)”

Testing includes:

- API schema tests
- Page rendering tests
- Mock Ollama tests
- Regression tests for explanation templates
- Benchmarking UI load times

---

12. Summary of Phase 6 v2

Phase 6 v2 expands Phase 6 v1 into a complete, production-grade user interface and explanation system.

It includes:

✓ A modular backend (Flask/FastAPI)

✓ A modern frontend (React/Svelte)

✓ Player detail pages

✓ Weekly dashboards

✓ Comparisons, waivers, trades, ROS views

✓ Game predictions

✓ Defender/archetype visualizations

✓ Full integration with ensemble predictions

✓ A robust natural-language explanation engine

✓ Support for local popup/desktop mode

✓ A structured prompt generator for Ollama

Phase 6 v2 is the “user experience layer” that brings all prior phases to life.