Phase 3 v2 – Baseline Predictor
(Merged and Expanded from Phase 3 v1)

---

1. Purpose of Phase 3
Phase 3 is the first true “prediction” phase of the system.
It does not involve machine learning yet.
Instead, it constructs a deterministic, mathematically defined baseline predictor that uses all the features engineered in Phase 2.
This baseline predictor serves three critical purposes:
1) It produces stable, interpretable predictions.
These predictions use understandable logic, like:

- recent form vs long-term baseline
- opponent team-defense deltas
- defender matchup difficulty
- archetype interactions

This lets us explain *why* a certain projection happened using human-friendly rules.
2) It acts as a fallback model when ML outputs are unstable.
Some players have extremely limited historical data: rookies, injuries, backups.
ML models struggle with sparse data.
The baseline will always generate sane, conservative predictions.
3) It acts as a base learner input to the ensemble meta-model.
Phase 4 models generate predictions.
Phase 6 ensemble combines them.
The baseline predictor is one of the models being weighted in that ensemble.
The ensemble will learn if the baseline tends to underpredict/overpredict certain stats and adjust accordingly.

---

2. Conceptual Philosophy of the Baseline Predictor
The Phase 3 v1 idea was:
“Use rolling averages weighted by recency and adjust for matchup difficulty.”
Phase 3 v2 significantly expands this:
The baseline predictor must incorporate:

- Long-term player averages
- Short-term form (last 3 games exponentially weighted)
- Opponent team-defense strength
- Opponent team-defense RECENT form
- Effective defender metrics
- Archetype vs archetype matchup adjustments
- Usage share projections (snap share, target share, carry share)
- Game context (if later added: pace, script, etc.)

Then combine these components into:
A weighted composite “expected statline,”
producing predicted counts for the stats needed by ML and fantasy.
The baseline must do all of this in a rule-based, transparent, non-ML way.

---

3. Inputs to the Baseline Predictor
The baseline predictor consumes:
1. The player_game_features table generated in Phase 2.
This includes:
- player usage features
- player efficiency features
- recent and long-term deltas
- defensive team metrics + deltas
- defender-level features + deltas
- coverage probabilities
- effective defender metrics
- archetype metadata

2. Player positional context
QB, RB, WR, TE determine:
- which stats we predict
- which feature families matter most
- how strongly defender matchups influence the prediction

3. Archetype interaction matrices (from the Archetype Modeling document)
A matrix such as:

Offensive Archetype	Defensive Archetype	Effect on Expected Yards
Deep WR	Press-Man Boundary CB	-12%
Slot WR	Slot Zone CB	+8%
Pass-Catching RB	Soft LB	+15%

The baseline uses this matrix to adjust projections before ML sees the results.
4. Effective defender metrics
These incorporate:
- coverage probabilities
- defender performance levels
- defender form deltas

These create a final “defender difficulty score” per player.

---

4. Stats the Baseline Must Predict
For each (player_id, game_id) pair, the baseline must output:
WR / TE
- targets
- receptions
- receiving yards
- receiving TDs

RB
- carries
- rush yards
- rush TDs
- targets
- receptions
- rec yards
- rec TDs

QB
- pass attempts

- completions
- pass yards
- pass TDs
- interceptions
- rush yards
- rush TDs

These baseline predictions will serve:

- as a standalone prediction
- as an input to ML models (optional)
- as an input to the ensemble meta-model (mandatory)

---

5. High-Level Structure of the Baseline Formula

Each predicted stat follows the same template:

```
Predicted Stat =
    Base Player Expectation
    + Player Form Adjustment
    + Team Context Adjustment
    + Opposing Defense Adjustment
    + Defender Matchup Adjustment
    + Archetype Interaction Adjustment
    + Residual Stability Adjustment
```

Let's break each piece down.

---

6. Base Player Expectation

This is the long-term, leakage-safe average stat for the player.

Examples:

- player_targets_season_avg
- player_yards_per_target_season
- player_carries_season_avg
- player_yards_per_carry_season

For rookies or players with limited history:

- use league-average baseline weighted toward player archetype
- e.g., a non-slot deep WR rookie should start nearer the *deep WR* archetype baseline
- this was mentioned conceptually in v1; v2 formalizes it

---

7. Player Form Adjustment (Recency)

Compute exponential-weighted recent averages:

```
form_last3_exp = (stat_week(n-1)*0.55
                  + stat_week(n-2)*0.30
                  + stat_week(n-3)*0.15)
```

Then compute deltas:

```
delta_last3_vs_season = form_last3_exp - season_avg
```

Then scale by a position-specific coefficient:

Example:

```
targets_form_adj = delta_targets_last3_vs_season * 0.65
```

For RB yards/carry:

```
ypr_form_adj = delta_ypr_last3_vs_season * 0.45
```

This recency logic was in v1 in a loose conceptual form; v2 formalizes:

- EWMA

- delta formulation
- scaling coefficients

---

8. Team Context Adjustment
Examples:
For pass-heavy teams:
+ (team_pass_rate_last3_exp − league_avg_pass_rate) * 0.8
For run-heavy teams:
+ (team_rush_rate_last3_exp − league_avg_rush_rate) * 0.6
For high-paced teams:
+ (team_plays_per_game_last3_exp − league_avg) * 0.4
This part existed in v1, but v2 requires explicitly coding it per stat:
- Targets depend heavily on pass rate
- Carries depend on run rate
- Yards depend on play volume
- TDs depend on red-zone tendencies

This must be parameterized per stat.

---

9. Opposing Team-Defense Adjustment
For example, projecting WR receiving yards:
def_wr_yards_adj =
    − (def_wr_yards_allowed_per_game_season − league_avg_wr_yards) * 0.35
    − (delta_def_wr_yards_last3_vs_season) * 0.45
Meaning:
- If a defense is better than average → subtract
- If they are trending upward (worse recently) → add
- If they are trending downward (improving recently) → subtract more

This appeared vaguely in v1 as “adjust for matchup difficulty.”
v2 requires:
- separate season-long and recent-form adjustments
- each with explicit weights

---

10. Defender Matchup Adjustment (Huge v2 addition)
Based on effective defender metrics, e.g.:
defender_ypt = effective_def_yards_allowed_per_target
delta_defender_ypt_form = effective_def_delta_yards_allowed_last3_vs_season
Then:
matchup_adj =
    − (defender_ypt − league_avg_def_ypt) * 0.55
    − (delta_defender_ypt_form) * 0.40
Meaning:
- If the likely matchup defender is strong → predict fewer yards
- If the defender is performing poorly recently → adjust upwards

This did not exist at all in Phase 3 v1; it is fully mandated by v2.

---

11. Archetype Interaction Adjustment
From archetype matrix:
arch_adj =

```
    archetype_matrix[off_archetype][def_archetype]
```
Examples:

- Deep WR vs press-man CB → −12%
- Slot WR vs zone → +8%

Apply multiplicative effect:

```
predicted_yards *= (1 + arch_adj)
```
This supports:

- explainability
- baseline interpretability
- downstream ensemble context

---

12. Residual Stability Adjustment

Because baseline predictions must be conservative and stable, introduce:

- shrinkage toward long-term mean
- shrinkage toward archetype mean when data is sparse
- positional caps (e.g., WR targets cannot exceed team pass attempts × logical max share)

Examples:

```
predicted_targets =
    0.85 * predicted_targets + 0.15 * season_avg_targets
```
or for rookies:

```
predicted_rec_yards =
    0.7 * predicted_rec_yards + 0.3 * archetype_avg_rec_yards
```
This was lightly implied in v1; v2 formalizes it as a required stabilizer.

---

13. Positional Stat Outputs

For WR/TE:

- targets = baseline formula
- receptions = targets × catch_rate_prediction
- rec_yards = receptions × ypr_prediction
- rec_tds = touchdown formula

For RB:

- carries = baseline formula
- rush_yards = carries × yards_per_carry_prediction
- rush_tds = TD formula
- receiving stats same as WR/TE logic

For QB:

- pass_attempts = baseline formula
- completions = attempts × completion% prediction
- pass_yards = attempts × yards_per_attempt_prediction
- pass_tds = TD formula
- interceptions = int_rate_prediction × attempts
- rush yards / TDs computed like RB but weighted lightly

---

14. Rule-Based Consistency Constraints

The baseline must enforce logical constraints:

- receptions ≤ targets
- completions ≤ pass_attempts
- rec_yards ≥ receptions (unless negative EPA weirdness allowed)

- rush_yards ≥ carries × (−2) *(soft floor)*
- TDs are integers in final fantasy output logic (though baseline can output expectations)
- no negative predictions

---

15. Output Format
The baseline predictor outputs a table:
baseline_predictions
Columns:
- player_id
- game_id
- pred_targets
- pred_receptions
- pred_rec_yards
- pred_rec_tds
- pred_carries
- pred_rush_yards
- pred_rush_tds
- pred_pass_attempts
- pred_completions
- pred_pass_yards
- pred_pass_tds
- pred_interceptions
- metadata fields used for explanation
- baseline_version
- data_version

This table is stored as:
baseline_predictions_<data_version>
in the database.

---

16. Why This Baseline Is Critical for the Ensemble
The ensemble meta-model (Phase 6) takes as input:
- baseline predictions
- ML model predictions
- defender-aware submodel predictions
- archetype-response predictions
- other model outputs

The ensemble learns:
- when baseline is trustworthy
- when ML is better
- when defender-aware adjustments matter most

The baseline predictor therefore provides:
- interpretability
- sanity/stability
- base signal for ensemble weighting

This creates a hybrid system far superior to traditional fantasy projections.

---

17. Versioning and Reproducibility

Every time Phase 2/Phase 3 recompute features and predictions for a new NFL week:

- baseline predictor version must update
- code version must be logged
- baseline outputs must store data_version

A typical version might be:
baseline_version = 2025W06_B1.0
This helps:

- debugging
- reproducibility
- ensemble consistency

---

18. Summary of Phase 3 v2
Phase 3 v2 expands Phase 3 v1 by adding:

- defender-aware logic (effective defender metrics)
- archetype interaction adjustments
- explicit coefficients and scaling for each adjustment
- delta-weighted recent form
- explicit constraints
- fully defined baseline formula structure
- positional logic for QB/RB/WR/TE
- integration with ensemble training
- versioning and reproducibility rules

The end result is a fully deterministic, explainable, stable prediction system that produces the first meaningful stat projections in the application pipeline.