**1. Architectural Philosophy**

**Key Constraints You Mentioned**

* UWAMP (Windows + Apache + MySQL + PHP)
* Python support (limited on shared hosting)
* Deployment on **GoDaddy shared hosting**
* Integration into **Astraal LXP** (likely PHP-based)

**Design Decision**

1. We **separate concerns**:

| **Layer** | **Technology** | **Responsibility** |
| --- | --- | --- |
| LXP UI & Auth | PHP | Learner interaction |
| Data Store | MySQL | Progress, scores, events |
| ML Engine | Python (offline + lightweight API) | Recommendation logic |
| Integration | REST / CLI | PHP ↔ Python bridge |

1. This avoids persistent Python servers (not allowed on GoDaddy shared hosting).

**Correct Reframing of the Problem (Journey, not Assessments)**

**What your UI clearly communicates**

* Your learner journey includes:
* Learning Path
* Skills & Competencies
* Critical Thinking
* Problem Solving Skills
* Project Studio
* Collaborative Learning
* Work Life Experience
* Career Path & Compliance
* Mentorship & Social Learning
* Coding Ground
* Edu 5.0 Lifelong Learning
* This means:
* **Assessment is just one signal, not the driver.**
* The **driver is progression maturity** across multiple dimensions.

**2. Redefined Problem Statement (Journey-Aligned)**

**❌ Traditional (incorrect for Astraal)**

* “Adapt learning paths based on test scores and completion.”

**✅ Astraal-Aligned (correct)**

* Learner journeys are currently linear and static, while real learner growth is multi-dimensional and non-linear across skills, thinking, collaboration, and career readiness.

**3. Revised Solution Concept**

**“Learner Journey Orchestration Engine”**

* This is **not a recommendation engine alone**.  
  It is a **Journey State Engine + Recommendation Layer**.

**4. Journey Dimensions (Mapped 1:1 to Your Sidebar)**

* Each learner has **parallel progression states**.

| **Journey Area** | **What You Track (Examples)** |
| --- | --- |
| Learning Path | Module flow, pacing, depth |
| Skills & Competencies | Skill evidence, rubrics |
| Critical Thinking | Case handling, reflections |
| Problem Solving | Scenario attempts, solution quality |
| Project Studio | Project milestones, reviews |
| Collaborative Learning | Peer interactions, roles |
| Work Life Experience | Simulations, internships |
| Career & Compliance | Readiness, certifications |
| Mentorship | Sessions, guidance uptake |
| Coding Ground | Practice depth, consistency |
| Edu 5.0 | Lifelong, cross-domain learning |

* **Each dimension evolves independently but influences others.**

**5. Core Concept: Learner Journey State (LJS)**

* Instead of “marks”, define a **Learner Journey State Vector**.

LJS = {

learning\_depth,

skill\_maturity,

thinking\_complexity,

problem\_solving\_confidence,

project\_readiness,

collaboration\_index,

work\_exposure\_level,

career\_alignment,

mentorship\_dependency,

practice\_consistency,

lifelong\_learning\_index

}

* This is what your ML works on — **not raw assessments**.

**6. How Adaptivity Actually Works in Astraal**

**Example (Realistic)**

* A learner:
* Performs well in Coding Ground
* Struggles in Collaborative Learning
* Is active in Project Studio
* Has low Mentorship engagement

**Adaptive Response**

* Do **not** push harder content
* Instead:
  + Recommend **team-based project**
  + Surface **mentor interaction**
  + Add **reflection task** in Critical Thinking
  + Keep coding difficulty stable
* ➡️ This is *journey balance*, not acceleration.

**7. ML Model You Actually Need (Simple & Powerful)**

**Model Role**

* Rank **next best journey actions**, not just modules.

**Output Examples**

* “Join collaborative sprint”
* “Attempt industry simulation”
* “Request mentor review”
* “Deepen skill X”
* “Pause progression → reflection”

**ML Approach (Shared Hosting Friendly)**

* Feature-weighted ranking
* Rule-governed ML (ML suggests, rules decide)

Final Recommendation Score =

0.35 \* Skill Readiness

+ 0.25 \* Journey Balance

+ 0.20 \* Engagement Stability

+ 0.20 \* Career Alignment

**8. UWAMP + Python Architecture (Journey Version)**

**What Python Does**

* Computes **Journey State**
* Generates **Next Journey Actions**
* Writes back ranked actions

**What PHP Does**

* Captures events
* Displays adaptive UI
* Maintains learner trust

Learner Action

↓

PHP Event Logger

↓

MySQL (Journey Tables)

↓

Python Journey Engine (batch / cron)

↓

Journey Recommendations

↓

Adaptive UI Sections

**9. Database Design (Journey-Oriented)**

learner\_journey\_state (

learner\_id INT,

learning\_depth FLOAT,

skill\_maturity FLOAT,

thinking\_complexity FLOAT,

collaboration\_index FLOAT,

project\_readiness FLOAT,

career\_alignment FLOAT,

updated\_on DATETIME

);

journey\_recommendations (

learner\_id INT,

journey\_area VARCHAR(50),

recommended\_action TEXT,

confidence\_score FLOAT

);

* This maps **directly to your sidebar sections**.

**10. UI Adaptivity (What Changes Visibly)**

* Your sidebar **does not change**, but:
* Badges appear (“Focus Needed”, “Ready to Advance”)
* Sections reorder subtly (non-disruptive)
* CTA text adapts:
  + “Explore”
  + “Strengthen”
  + “Collaborate”
  + “Reflect”
* This preserves learner dignity and autonomy.

**1. What Your UI Is Already Telling Us (Key Insight)**

* From your screen, your learning path is driven by **six orchestration signals**:
* Learning Intent (Declared / Revised)
* Skill Gap (Active)
* Learning Direction (Accepted)
* Milestones (Generated)
* Guided Pathways (Adaptive)
* Learning Status (Tracking On)
* 👉 **These are not UI labels**  
  👉 **These are system states**
* So the engine you are building is not “recommend courses”  
  It is:
* **A Learning Journey Orchestration Engine that ranks next best actions across a guided pathway**

**2. Target System Architecture (Infra-Realistic)**

**High-Level Architecture**

Learner UI (Astraal LXP - PHP)

↓

Journey Event Capture Layer (PHP)

↓

Journey State Store (MySQL)

↓

Adaptive Recommendation Engine (Python)

↓

Ranked Pathway Decisions (MySQL)

↓

Adaptive Learning Path UI

**Why this works on GoDaddy Shared Hosting**

* No long-running Python services
* Python runs **on-demand / cron**
* PHP remains the primary runtime
* ML is **lightweight, explainable, batch-based**

**3. Core Engine Components (Mapped to Your UI)**

**3.1 Learning Intent Engine**

* **Source:** Learner declaration + revisions
* Stored as:

intent\_strength

intent\_stability

intent\_alignment\_score

* Used to:
* Lock or unlock pathway changes
* Prevent random recommendations

**3.2 Skill Gap Engine**

* **Source:** Skills & Competencies, Coding Ground, Project Studio
* Computed as:

skill\_gap = required\_skill\_level − demonstrated\_skill\_level

* This directly feeds **ranking weights**.

**3.3 Learning Direction Resolver**

* **Source:** Accepted direction (guided / exploratory / accelerated)
* Acts as a **constraint layer**:
* What can be recommended
* What cannot be recommended

**3.4 Milestone Generator**

* Milestones are **derived**, not fixed.
* Examples:
* Concept mastery
* Project checkpoint
* Collaboration exposure
* Reflection completion
* Milestones gate progression.

**3.5 Guided Pathway Adapter (ML Layer)**

* This is where **ML ranking** is applied.
* It ranks:
* Next module
* Next project
* Next collaboration
* Next intervention (mentor / revise)

**3.6 Learning Status Tracker**

* Ensures:
* No learner is “lost”
* Inactivity triggers adaptive nudges
* Pace remains humane

**4. Data Model (Journey-Centric, Not Exam-Centric)**

**Core Tables**

learner\_journey\_state (

learner\_id INT PRIMARY KEY,

intent\_score FLOAT,

skill\_gap\_score FLOAT,

engagement\_score FLOAT,

milestone\_progress FLOAT,

pathway\_confidence FLOAT,

last\_updated DATETIME

);

learning\_events (

learner\_id INT,

event\_type VARCHAR(50),

event\_value FLOAT,

source\_module VARCHAR(50),

created\_on DATETIME

);

adaptive\_recommendations (

learner\_id INT,

recommendation\_type VARCHAR(50),

recommended\_item\_id INT,

rank\_score FLOAT,

rationale TEXT,

generated\_on DATETIME

);

* This allows **explainable adaptivity** — critical for trust and compliance.

**5. ML Design (Simple, Explainable, Ranking-Based)**

**Why Ranking Models (and not “AI magic”)**

* You are **ordering options**, not predicting truth
* You need **transparency**
* You need **low compute**

**Feature Vector (per learner × pathway option)**

features = [

intent\_score,

skill\_gap\_score,

engagement\_score,

milestone\_progress,

pathway\_confidence,

recency\_of\_activity,

pathway\_difficulty\_delta

]

**Ranking Formula (Hybrid: ML + Rules)**

final\_score =

0.30 \* skill\_readiness

+ 0.25 \* milestone\_alignment

+ 0.20 \* engagement\_stability

+ 0.15 \* intent\_consistency

+ 0.10 \* career\_alignment

* Rules always override ML if:
* Milestone incomplete
* Compliance requirement unmet
* Learner explicitly revises intent

**6. Python Engine (GoDaddy-Safe)**

**Folder Structure**

/python/

├── feature\_builder.py

├── ranker.py

├── model.pkl

├── run\_engine.py

**run\_engine.py (Entry Point)**

from feature\_builder import build\_features

from ranker import rank\_pathways

features = build\_features()

recommendations = rank\_pathways(features)

recommendations.to\_mysql()

**Execution Modes**

* PHP exec() on learner action
* Nightly cron for recalibration

**7. PHP ↔ Python Integration**

**Trigger Recommendation Update**

exec("python3 /home/user/python/run\_engine.py " . $learner\_id);

**Fetch Recommendations**

SELECT \* FROM adaptive\_recommendations

WHERE learner\_id = ?

ORDER BY rank\_score DESC;

**8. UI Adaptation Logic (What Learners See)**

* Your UI already supports this beautifully.
* Examples:
* **Guided Pathways → ADAPTIVE**
* Buttons change context:
  + *Revise* (intent unstable)
  + *Explore* (safe branching)
  + *View Progress* (tracking mode)
* No sudden jumps. No confusion.

**9. Compliance, Ethics & Institutional Readiness**

* This architecture is:
* **Explainable** (rationale stored)
* **Non-manipulative**
* **Learner-controlled**
* **Faculty-override friendly**
* Which makes it suitable for:
* Universities
* Corporate L&D
* BFSI-aligned training ecosystems

**1. Project Folder Structure (UWAMP / GoDaddy Shared Hosting)**

/public\_html/

│

├── astraal-lxp/

│ ├── config/

│ │ └── db.php

│ │

│ ├── learning-path/

│ │ ├── learning\_intent.php

│ │ ├── skill\_gap.php

│ │ ├── milestones.php

│ │ ├── guided\_pathways.php

│ │ └── learning\_status.php

│ │

│ ├── api/

│ │ ├── log\_event.php

│ │ ├── get\_recommendations.php

│ │ └── trigger\_engine.php

│ │

│ └── dashboard.php

│

├── python/

│ ├── feature\_builder.py

│ ├── ranker.py

│ ├── run\_engine.py

│ └── model.pkl

│

└── cron/

└── nightly\_engine.php

This structure:

* Works on **UWAMP**
* Works on **GoDaddy shared hosting**
* Keeps Python isolated and secure

**2. MySQL Database Tables (Journey-First)**

**2.1 Learner Master**

CREATE TABLE learners (

learner\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

email VARCHAR(100),

created\_on DATETIME DEFAULT CURRENT\_TIMESTAMP

);

**2.2 Learning Intent**

CREATE TABLE learning\_intent (

learner\_id INT,

intent\_text TEXT,

intent\_score FLOAT,

revised\_count INT DEFAULT 0,

updated\_on DATETIME,

PRIMARY KEY (learner\_id)

);

**2.3 Skill Gap State**

CREATE TABLE skill\_gap\_state (

learner\_id INT,

skill\_code VARCHAR(50),

required\_level INT,

current\_level INT,

gap\_score FLOAT,

updated\_on DATETIME

);

**2.4 Milestones**

CREATE TABLE milestones (

learner\_id INT,

milestone\_code VARCHAR(50),

status ENUM('pending','completed'),

completion\_percent FLOAT,

updated\_on DATETIME

);

**2.5 Journey Events (Very Important)**

CREATE TABLE learning\_events (

event\_id INT AUTO\_INCREMENT PRIMARY KEY,

learner\_id INT,

event\_type VARCHAR(50),

event\_value FLOAT,

source VARCHAR(50),

created\_on DATETIME DEFAULT CURRENT\_TIMESTAMP

);

**2.6 Adaptive Recommendations (Python writes here)**

CREATE TABLE adaptive\_recommendations (

learner\_id INT,

recommendation\_type VARCHAR(50),

recommendation\_text TEXT,

rank\_score FLOAT,

rationale TEXT,

generated\_on DATETIME

);

**3. PHP Configuration (DB Connection)**

**/config/db.php**

<?php

$conn = new mysqli("localhost", "db\_user", "db\_pass", "astraal\_lxp");

if ($conn->connect\_error) {

die("DB Connection failed");

}

?>

**4. PHP Forms (Mapped to Your UI Cards)**

**4.1 Learning Intent (Declared / Revise)**

**learning\_intent.php**

<form method="post" action="../api/log\_event.php">

<textarea name="intent\_text" required></textarea>

<input type="hidden" name="event\_type" value="learning\_intent">

<button type="submit">Declare / Revise</button>

</form>

**4.2 Skill Gap Trigger (Auto / Manual Review)**

**skill\_gap.php**

<form method="post" action="../api/log\_event.php">

<input type="hidden" name="event\_type" value="skill\_gap\_update">

<button type="submit">Recalculate Skill Gap</button>

</form>

**4.3 Milestone View (Read-Only with Tracking)**

**milestones.php**

<?php

include("../config/db.php");

$result = $conn->query("SELECT \* FROM milestones WHERE learner\_id=1");

while($row=$result->fetch\_assoc()){

echo $row['milestone\_code']." - ".$row['status'];

}

?>

**4.4 Guided Pathways (Adaptive CTA)**

**guided\_pathways.php**

<a href="../api/trigger\_engine.php">

<button>Explore Pathways</button>

</a>

**5. PHP API Layer (Critical Glue)**

**5.1 Log Any Learner Event**

**/api/log\_event.php**

<?php

include("../config/db.php");

$learner\_id = 1; // from session

$event\_type = $\_POST['event\_type'];

$event\_value = 1;

$conn->query("

INSERT INTO learning\_events

(learner\_id,event\_type,event\_value,source)

VALUES ($learner\_id,'$event\_type',$event\_value,'UI')

");

echo "Event Logged";

?>

**5.2 Trigger Python Engine (On Demand)**

**/api/trigger\_engine.php**

<?php

$learner\_id = 1;

exec("python3 /home/user/python/run\_engine.py $learner\_id");

header("Location: ../learning-path/guided\_pathways.php");

?>

**5.3 Fetch Recommendations**

**/api/get\_recommendations.php**

<?php

include("../config/db.php");

$result = $conn->query("

SELECT \* FROM adaptive\_recommendations

WHERE learner\_id=1

ORDER BY rank\_score DESC

");

while($row=$result->fetch\_assoc()){

echo "<p>".$row['recommendation\_text']."</p>";

}

?>

**6. Python Engine (Minimal & Explainable)**

**6.1 Feature Builder**

**feature\_builder.py**

import mysql.connector

import pandas as pd

def build\_features(learner\_id):

db = mysql.connector.connect(

host="localhost",

user="db\_user",

password="db\_pass",

database="astraal\_lxp"

)

query = f"""

SELECT event\_type, COUNT(\*) as cnt

FROM learning\_events

WHERE learner\_id={learner\_id}

GROUP BY event\_type

"""

return pd.read\_sql(query, db)

**6.2 Ranker (Rule + Score)**

**ranker.py**

def rank(features):

recommendations = []

if "learning\_intent" in features['event\_type'].values:

recommendations.append(

("Guided Learning Path", 0.85, "Strong intent declared")

)

recommendations.append(

("Skill Booster Module", 0.70, "Skill gap detected")

)

return recommendations

**6.3 Engine Runner**

**run\_engine.py**

import sys

import mysql.connector

from feature\_builder import build\_features

from ranker import rank

learner\_id = sys.argv[1]

features = build\_features(learner\_id)

recs = rank(features)

db = mysql.connector.connect(

host="localhost",

user="db\_user",

password="db\_pass",

database="astraal\_lxp"

)

cursor = db.cursor()

cursor.execute("DELETE FROM adaptive\_recommendations WHERE learner\_id=%s", (learner\_id,))

for r in recs:

cursor.execute("""

INSERT INTO adaptive\_recommendations

VALUES (%s,'pathway',%s,%s,%s,NOW())

""",(learner\_id,r[0],r[1],r[2]))

db.commit()

**7. Optional: Nightly Recalibration (Cron Safe)**

**/cron/nightly\_engine.php**

<?php

exec("python3 /home/user/python/run\_engine.py all");

?>

***daptive Learning Path Recommendation Engine (Journey-Oriented)***

**Capstone Title**

**Design & Build an Adaptive Learning Journey Orchestration Engine for Astraal LXP**

**Problem Context**

Most LMS/LXP platforms use **static learning paths** that do not respond to:

* Learner intent changes
* Skill gaps across domains
* Engagement drop-offs
* Milestone readiness

This results in **low completion, misaligned progression, and learner fatigue**.

**Capstone Objective**

Build a **Journey-Aware Adaptive Recommendation Engine** that:

* Continuously evaluates learner progress
* Dynamically adjusts learning paths
* Preserves learner autonomy
* Remains explainable and faculty-governed

**Scope (What Learners Must Build)**

**Mandatory**

* PHP-based learner interaction layer
* MySQL journey state model
* Python-based ranking engine
* Adaptive learning path output
* Faculty override mechanism (read/write)

**Explicitly Out of Scope**

* Real-time streaming
* Deep neural networks
* External cloud ML services

**Deliverables**

| **Deliverable** | **Description** |
| --- | --- |
| System Architecture | PHP–MySQL–Python orchestration |
| Database Schema | Journey-centric tables |
| PHP Forms & APIs | Intent, milestones, events |
| Python Ranking Engine | Rule + ML |
| Faculty Dashboard | Override + visibility |
| Demo Walkthrough | Learner journey simulation |

**Evaluation Rubric (Industry-Grade)**

| **Criterion** | **Weight** |
| --- | --- |
| Architecture correctness | 25% |
| Adaptivity logic | 25% |
| Explainability | 15% |
| Code quality | 15% |
| UX alignment | 10% |
| Compliance awareness | 10% |

**2️⃣ ML Ranking Engine (Replacing Rules with Real ML)**

We now **upgrade** the Python engine from rule-only to **ML-assisted ranking**, while staying **shared-hosting safe**.

**2.1 ML Problem Framing (Correct Framing)**

❌ Predict score  
❌ Predict completion

✅ **Rank “next best learning action”**

This is a **Learning-to-Rank** problem.

**2.2 Training Dataset Structure**

learner\_id,

intent\_score,

skill\_gap\_score,

engagement\_score,

milestone\_progress,

pathway\_confidence,

difficulty\_delta,

label

Where:

* label = 1 if learner *successfully progressed* after recommendation

**2.3 Model Choice (Why This Works)**

| **Model** | **Reason** |
| --- | --- |
| Logistic Regression | Lightweight, explainable |
| Gradient Boosted Trees | Better ranking accuracy |
| XGBoost Ranker | Optional upgrade |

We’ll use **Logistic Regression first** (institution-friendly).

**2.4 Training Script (Offline / Local)**

**train\_model.py**

import pandas as pd

from sklearn.linear\_model import LogisticRegression

import pickle

data = pd.read\_csv("training\_data.csv")

X = data.drop("label", axis=1)

y = data["label"]

model = LogisticRegression()

model.fit(X, y)

with open("model.pkl", "wb") as f:

pickle.dump(model, f)

This runs **locally** or on UWAMP.

**2.5 Ranking at Runtime (Shared Hosting Safe)**

**ranker.py**

import pickle

import numpy as np

model = pickle.load(open("model.pkl", "rb"))

def rank(features\_df):

features\_df["rank\_score"] = model.predict\_proba(

features\_df.values

)[:,1]

return features\_df.sort\_values("rank\_score", ascending=False)

➡️ Python runs **only when triggered**  
➡️ No background services  
➡️ Fully GoDaddy compliant

**2.6 Explainability (Mandatory)**

Store **why** something was recommended:

"High skill readiness but low collaboration exposure"

This is **non-negotiable** for institutional adoption.

**3️⃣ Faculty Override & Governance Dashboard**

This is the **difference between a demo system and a deployable system**.

**3.1 Why Faculty Override Is Critical**

* Prevents blind automation
* Builds institutional trust
* Enables mentoring intervention
* Meets compliance & ethics norms

**3.2 Faculty Dashboard Tables**

CREATE TABLE faculty\_overrides (

override\_id INT AUTO\_INCREMENT PRIMARY KEY,

learner\_id INT,

original\_recommendation TEXT,

overridden\_recommendation TEXT,

reason TEXT,

overridden\_by INT,

overridden\_on DATETIME

);

**3.3 Faculty View (PHP)**

**/faculty/learner\_journey.php**

<?php

include("../config/db.php");

$result = $conn->query("

SELECT \* FROM adaptive\_recommendations

WHERE learner\_id=1

");

while($row=$result->fetch\_assoc()){

echo "<b>".$row['recommendation\_text']."</b>";

echo "<p>".$row['rationale']."</p>";

}

?>

**3.4 Faculty Override Form**

**/faculty/override.php**

<form method="post" action="save\_override.php">

<input type="hidden" name="learner\_id" value="1">

<textarea name="override\_reason"></textarea>

<input name="new\_recommendation">

<button>Override Recommendation</button>

</form>

**3.5 Override Enforcement Logic**

Priority order:

1. Faculty override
2. Compliance rules
3. Milestone gates
4. ML ranking

ML **never** has final authority.