### **1. Database Schema Updates**

The Setarcos app uses Supabase (PostgreSQL) as its database, so all schema changes must be compatible with this system and properly indexed for performance.

* **New Table: user\_rating\_interactions** Create a table to store rating data and interaction metrics, including response\_id, user\_id, rating\_value (-1, 1, 2), philosopher\_tone, and various engagement metrics (e.g., hover\_duration\_ms, time\_to\_rate\_ms). Add a unique constraint on (user\_id, response\_id) to allow only one rating per user per response, with updates permitted if the rating changes (tracked via rating\_changed). Indexes will be added on response\_id, user\_id, philosopher\_tone, model\_used, and created\_at to optimize query performance.
* **Existing Table Updates**
  + **ai\_responses**: Add columns user\_rating (default 0), rating\_timestamp, and user\_engagement\_duration\_ms to store rating metadata.
  + **response\_quality\_metrics**: Add user\_rating\_correlation and rating\_prediction\_accuracy to track quality correlations.
* **Implementation Steps**
  + Define the schema changes using SQL migrations (via Alembic, per the blueprint).
  + Update SQLAlchemy ORM models in the backend to reflect these changes.
  + Test migrations to ensure backward compatibility with existing data.

### **2. Backend Services**

The blueprint specifies FastAPI for the backend, so new services will be implemented as part of the FastAPI application. Asynchronous tasks will leverage Celery to maintain responsiveness.

* **UserRatingService**
  + **Purpose**: Handle rating submissions, validate inputs, calculate engagement metrics, and trigger downstream processes.
  + **Key Endpoints**:
    - POST /ratings/submit: Submit a rating with interaction metrics.
    - GET /ratings/analytics: Retrieve aggregated rating insights (e.g., by philosopher tone or model).
  + **Behavior**: Store ratings in the database, queue Celery tasks for quality analysis, and send analytics events to PostHog.
* **QualityCorrelationService**
  + **Purpose**: Analyze the correlation between predicted quality scores and user ratings, updating prediction algorithms.
  + **Key Functionality**: Calculate prediction accuracy and feed insights into the AI Router’s quality assessment pipeline.
  + **Execution**: Run as a Celery task triggered by rating submissions.
* **RatingAnalyticsService**
  + **Purpose**: Process rating data for PostHog and generate insights for AI Router optimization.
  + **Key Functionality**: Format and send events (e.g., rating\_submitted) asynchronously via Celery.
* **Integration**:
  + Services will interact via FastAPI endpoints and Celery tasks. For example, a rating submission will call UserRatingService, which queues tasks for QualityCorrelationService and RatingAnalyticsService.

### **3. Frontend Integration**

The Setarcos app uses React Native with Zustand for state management. The Orb Rating System requires a new component integrated into the existing UI.

* **ContemplativeOrb Component**
  + **Purpose**: Provide a minimalist, expandable orb for users to rate AI responses (-1, 1, 2).
  + **Features**:
    - Capture mobile-friendly interaction metrics (e.g., touch duration, time to rate) instead of hover-based metrics.
    - Submit ratings to the backend via API calls.
    - Display the current rating state (if already rated).
  + **Implementation**:
    - Build in React Native as a reusable component.
    - Integrate into screens displaying AI responses (e.g., Ask feature screens).
    - Use Zustand to manage rating state locally (e.g., currentRating, isSubmitted).
* **Event Tracking**:
  + Send PostHog events (e.g., rating\_interaction\_started, rating\_submitted) with properties like response\_id, rating\_value, and interaction metrics.

### **4. AI Router Integration**

The Orb Rating System enhances the AI Router by incorporating user feedback into quality assessment and model selection.

* **Quality Assessment Service**
  + Update to include user rating data in quality scoring, adjusting base scores with a correlation factor derived from historical ratings.
* **Decision Matrix Service**
  + Add user satisfaction weighting to model selection, prioritizing models with higher ratings (especially transformative ratings) for specific philosopher tones.
* **Cost Optimization Service**
  + Balance cost with user satisfaction, ensuring even cost-constrained routing avoids poorly rated models.
* **Implementation**:
  + Modify existing services to query user\_rating\_interactions and adjust algorithms dynamically based on rating trends.

### **5. Testing and Validation**

Ensure the feature meets technical, business, and user experience goals through rigorous testing.

* **Unit and Integration Tests**: Validate new services, database interactions, and frontend components.
* **A/B Testing**: Compare user engagement and satisfaction with and without the rating system.
* **Performance Monitoring**: Track metrics like rating submission latency (< 200ms) and cache hit ratio (> 85%).
* **User Experience Testing**: Confirm the orb is unobtrusive and aligns with the app’s contemplative aesthetic.

### **6. Documentation and Compliance**

Integrate the feature into the Setarcos Technical Blueprint and ensure ethical data handling.

* **Blueprint Updates**:
  + Add user\_rating\_interactions and table alterations to the database schema section.
  + Document new API endpoints in the API design section.
  + Include ContemplativeOrb in the frontend file structure.
  + Update the system architecture diagram with new services and interactions.
* **Compliance**:
  + Inform users how ratings influence AI behavior (e.g., via privacy policy).
  + Implement rate limiting for submissions to prevent abuse.

### **Implementation Timeline**

The rollout follows a phased approach, integrated into the app’s sprint planning:

* **Phase 1: Foundation (Weeks 1-2)**
  + Database schema and migrations.
  + Core UserRatingService and basic ContemplativeOrb.
  + PostHog event setup.
* **Phase 2: Integration (Weeks 3-4)**
  + QualityCorrelationService and AI Router updates.
  + Frontend interaction tracking and analytics dashboard.
* **Phase 3: Optimization (Weeks 5-6)**
  + Enhance quality prediction and decision matrix weighting.
  + Optimize performance (e.g., caching, indexing).
* **Phase 4: Validation (Weeks 7-8)**
  + Conduct A/B testing, validate correlations, and refine UX.

### **Key Considerations**

* **Mobile Adaptation**: Replace hover metrics with touch-based equivalents (e.g., tap duration).
* **User Flow**: Ensure the orb is optional and non-disruptive, expanding only on interaction.
* **Performance**: Use Celery for async tasks and proper indexing to handle scale.
* **Ethics**: Transparently use rating data to improve AI, respecting user privacy.

This plan ensures the Orb Rating System enhances the Setarcos app’s functionality, aligns with its minimalist philosophy, and integrates seamlessly into the existing technical framework. By following these steps, we’ll deliver a robust feature that improves user engagement and AI performance while maintaining the app’s core identity.