# 5.8 Lumen AI Agentic Companion

## Feature Goal

To provide users with an intelligent, non-intrusive AI companion that enhances philosophical exploration by offering contextual guidance, personalized recommendations, and cross-feature integration throughout the app experience.

## 5.8.1 Definitions and Glossary

* **Lumen**: An AI agentic companion represented as a small, glowing orb with a soft golden light and subtle starry particles, designed to guide users through their philosophical journey.
* **Intervention**: A moment when Lumen appears to provide guidance, insights, or navigation assistance.
* **Philosophical Profile**: A data model representing the user's philosophical interests, knowledge areas, and exploration patterns.
* **Contextual Awareness**: Lumen's ability to understand the user's current location, activity, and philosophical context within the app.
* **Visual Evolution**: Changes in Lumen's appearance that reflect the user's progression and philosophical growth.

## 5.8.2 System Architecture

### Frontend Components

* **LumenOrbComponent.tsx** - Primary visual representation of Lumen
* **LumenDialogueComponent.tsx** - Text and interaction interface for Lumen communications
* **LumenAnimationController.tsx** - Manages Lumen's animations and transitions
* **LumenInterventionManager.tsx** - Controls when and how Lumen appears
* **LumenAccessibilityProvider.tsx** - Ensures Lumen is accessible to all users
* **LumenPreferencesScreen.tsx** - Settings screen for Lumen customization
* **LumenEvolutionDisplay.tsx** - Visualizes Lumen's evolution stages
* **LumenContextProvider.tsx** - Provides Lumen context throughout the app

### Backend Components

* **lumen\_core\_service.py** - Central orchestration service for Lumen functionality
* **user\_context\_service.py** - Builds and maintains user philosophical context
* **intervention\_service.py** - Determines appropriate interventions
* **content\_generation\_service.py** - Creates philosophical content for Lumen
* **visual\_presentation\_service.py** - Manages Lumen's visual appearance
* **feature\_integration\_service.py** - Enables cross-feature functionality
* **notification\_integration\_service.py** - Connects Lumen to notification system
* **analytics\_service.py** - Tracks and analyzes Lumen's effectiveness

### Database Models

-- Stores user's philosophical profile and preferences

CREATE TABLE lumen\_user\_profiles (

user\_id UUID PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,

philosophical\_interests JSONB DEFAULT '[]'::jsonb,

knowledge\_areas JSONB DEFAULT '[]'::jsonb,

preferred\_learning\_style VARCHAR(50),

sophistication\_level INTEGER DEFAULT 1,

motivation\_factors JSONB DEFAULT '[]'::jsonb,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

CONSTRAINT valid\_sophistication CHECK (sophistication\_level BETWEEN 1 AND 5)

);

-- Stores user preferences for Lumen behavior

CREATE TABLE lumen\_preferences (

user\_id UUID PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,

lumen\_enabled BOOLEAN DEFAULT TRUE,

intervention\_frequency VARCHAR(20) DEFAULT 'medium',

notifications\_enabled BOOLEAN DEFAULT TRUE,

preferred\_intervention\_types JSONB DEFAULT '["navigation", "insight", "reflection"]'::jsonb,

disabled\_features JSONB DEFAULT '[]'::jsonb,

accessibility\_preferences JSONB DEFAULT '{}'::jsonb,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

CONSTRAINT valid\_frequency CHECK (intervention\_frequency IN ('low', 'medium', 'high'))

);

-- Records Lumen interventions and their effectiveness

CREATE TABLE lumen\_interventions (

intervention\_id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,

intervention\_type VARCHAR(50) NOT NULL,

trigger\_context JSONB NOT NULL,

content JSONB NOT NULL,

timestamp TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

user\_response VARCHAR(20),

effectiveness\_score INTEGER,

followup\_actions JSONB DEFAULT '[]'::jsonb,

CONSTRAINT valid\_response CHECK (user\_response IN ('accepted', 'dismissed', 'ignored')),

CONSTRAINT valid\_effectiveness CHECK (effectiveness\_score IS NULL OR effectiveness\_score BETWEEN 1 AND 10)

);

-- Stores philosophical prompts for Lumen to use

CREATE TABLE lumen\_philosophical\_prompts (

prompt\_id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

prompt\_type VARCHAR(20) NOT NULL,

content TEXT NOT NULL,

complexity INTEGER NOT NULL DEFAULT 1,

related\_concepts JSONB DEFAULT '[]'::jsonb,

appropriate\_contexts JSONB DEFAULT '[]'::jsonb,

followup\_prompts JSONB DEFAULT '[]'::jsonb,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

is\_active BOOLEAN DEFAULT TRUE,

CONSTRAINT valid\_type CHECK (prompt\_type IN ('question', 'insight', 'reflection', 'challenge')),

CONSTRAINT valid\_complexity CHECK (complexity BETWEEN 1 AND 5)

);

-- Tracks Lumen's visual evolution for each user

CREATE TABLE lumen\_visual\_states (

user\_id UUID PRIMARY KEY REFERENCES users(id) ON DELETE CASCADE,

evolution\_stage INTEGER DEFAULT 1,

base\_appearance VARCHAR(50) DEFAULT 'standard',

glow\_intensity INTEGER DEFAULT 50,

particle\_effects JSONB DEFAULT '["basic"]'::jsonb,

color\_palette JSONB DEFAULT '{"primary": "#1a237e", "secondary": "#ffd700", "accent": "#f5f5dc"}'::jsonb,

unlocked\_appearances JSONB DEFAULT '[]'::jsonb,

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

CONSTRAINT valid\_evolution CHECK (evolution\_stage BETWEEN 1 AND 10),

CONSTRAINT valid\_glow CHECK (glow\_intensity BETWEEN 0 AND 100)

);

-- Stores intervention rules for Lumen behavior

CREATE TABLE lumen\_intervention\_rules (

rule\_id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

trigger\_conditions JSONB NOT NULL,

intervention\_type VARCHAR(50) NOT NULL,

priority INTEGER DEFAULT 5,

cooldown\_period INTEGER DEFAULT 3600, -- seconds

personalizable BOOLEAN DEFAULT TRUE,

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

CONSTRAINT valid\_priority CHECK (priority BETWEEN 1 AND 10)

);

## 5.8.3 API Endpoints

### Lumen Core API

GET /api/v1/lumen/status

Response:

{

"user\_id": "550e8400-e29b-41d4-a716-446655440000",

"lumen\_enabled": true,

"evolution\_stage": 3,

"current\_visual\_state": {

"base\_appearance": "scholar",

"glow\_intensity": 65,

"particle\_effects": ["starry", "wisdom"],

"color\_palette": {

"primary": "#1a237e",

"secondary": "#ffd700",

"accent": "#f5f5dc"

}

},

"intervention\_frequency": "medium",

"philosophical\_interests": ["ethics", "epistemology", "existentialism"]

}

### Lumen Preferences

PUT /api/v1/lumen/preferences

Request:

{

"lumen\_enabled": true,

"intervention\_frequency": "high",

"notifications\_enabled": true,

"preferred\_intervention\_types": ["navigation", "insight", "reflection"],

"accessibility\_preferences": {

"screen\_reader": false,

"reduced\_motion": true,

"high\_contrast": false

}

}

Response:

{

"success": true,

"message": "Lumen preferences updated successfully"

}

### Lumen Intervention

POST /api/v1/lumen/intervention

Request:

{

"intervention\_type": "philosophical\_prompt",

"context": {

"current\_feature": "explore",

"current\_concept": "epistemology",

"session\_duration": 450

}

}

Response:

{

"intervention\_id": "550e8400-e29b-41d4-a716-446655440001",

"content": {

"prompt\_type": "reflection",

"text": "As you explore epistemology, consider how your own process of knowing shapes what you discover. How do you determine what constitutes knowledge?",

"related\_concepts": ["knowledge", "justification", "truth"],

"animations": ["gentle\_pulse", "thought\_sparkle"]

},

"visual\_state": {

"position": "bottom\_right",

"animation": "appear\_from\_bottom",

"emphasis\_level": "medium"

}

}

### Lumen Feedback

POST /api/v1/lumen/feedback

Request:

{

"intervention\_id": "550e8400-e29b-41d4-a716-446655440001",

"response": "accepted",

"effectiveness\_rating": 8,

"feedback\_text": "This prompt helped me think more deeply about epistemology"

}

Response:

{

"success": true,

"message": "Feedback recorded successfully",

"xp\_earned": 5

}

### Lumen Navigation Guidance

GET /api/v1/lumen/guidance

Request Parameters:

current\_feature=explore

current\_concept=ethics

guidance\_type=navigation

Response:

{

"guidance\_id": "550e8400-e29b-41d4-a716-446655440002",

"guidance\_type": "navigation",

"content": {

"text": "Based on your interest in ethics, you might enjoy exploring these related concepts:",

"recommendations": [

{

"concept\_id": "550e8400-e29b-41d4-a716-446655440003",

"name": "Virtue Ethics",

"relevance\_score": 0.92,

"path\_description": "A branch of ethics focusing on character development"

},

{

"concept\_id": "550e8400-e29b-41d4-a716-446655440004",

"name": "Moral Responsibility",

"relevance\_score": 0.87,

"path\_description": "Examining accountability for ethical decisions"

}

],

"animations": ["path\_highlight", "gentle\_point"]

},

"visual\_state": {

"position": "near\_concept\_map",

"animation": "float\_and\_glow",

"emphasis\_level": "medium"

}

}

## 5.8.4 User Experience Flow

1. **Initial Introduction**
   * Lumen appears during onboarding with a warm welcome
   * Introduces its purpose as a philosophical guide
   * Explains how to summon or dismiss it
   * Sets initial preferences based on user choices
2. **Ambient Presence**
   * Lumen remains subtly present in the app interface
   * Glows gently to indicate availability
   * Positions itself contextually based on current activity
   * Respects user's attention and focus
3. **Navigation Guidance**
   * Appears when user seems uncertain about navigation
   * Highlights paths on the explore map
   * Suggests relevant features based on current context
   * Guides users through complex philosophical territories
4. **Philosophical Engagement**
   * Offers contextual philosophical prompts
   * Provides thought-provoking insights related to current content
   * Suggests connections between concepts being explored
   * Adapts philosophical depth to user's sophistication level
5. **Cross-Feature Integration**
   * Suggests relevant actions across app features
   * Creates bridges between isolated philosophical explorations
   * Maintains context when transitioning between features
   * Builds coherent philosophical narratives across the app
6. **Personalized Evolution**
   * Evolves visually as user progresses
   * Adapts behavior based on user preferences and feedback
   * Develops specialized knowledge of user's philosophical interests
   * Creates increasingly personalized philosophical guidance
7. **User Control**
   * Can be summoned via help icon or "ASK" input
   * Easily dismissed with a gesture
   * Preferences adjustable in settings
   * Respects user's desired intervention frequency

## 5.8.5 Cross-Feature Integration

### 5.8.5.1 Ask Feature Integration

* Suggests philosophical questions based on user interests
* Recommends appropriate philosophical tones for questions
* Highlights concepts mentioned in AI responses
* Suggests follow-up questions to deepen philosophical exploration
* Connects responses to relevant concepts in Explore feature

### 5.8.5.2 Explore Feature Integration

* Highlights recommended paths on concept map
* Suggests connections between concepts being explored
* Provides contextual information about philosophical concepts
* Guides exploration based on user's philosophical interests
* Creates personalized concept exploration journeys

### 5.8.5.3 Quest Feature Integration

* Provides hints and guidance for quest progression
* Highlights philosophical significance of quest challenges
* Suggests quests based on user's philosophical interests
* Celebrates quest milestones with philosophical context
* Creates bridges between quests and other app features

### 5.8.5.4 Journal Feature Integration

* Suggests reflection prompts based on recent exploration
* Highlights philosophical themes in journal entries
* Connects journal entries to relevant concepts
* Recommends journaling topics based on philosophical interests
* Creates continuity between journaling and other activities

### 5.8.5.5 Forum Feature Integration

* Suggests relevant discussions based on user's interests
* Prompts sharing of insights at meaningful moments
* Highlights community connections around philosophical topics
* Guides users to discussions matching their questions
* Facilitates philosophical dialogue between users

### 5.8.5.6 Profile Feature Integration

* Visualizes philosophical journey and growth
* Highlights achievements and milestones
* Suggests next steps for philosophical development
* Provides insights about philosophical interests and patterns
* Creates personalized philosophical identity visualization

### 5.8.5.7 Notification Integration

* Delivers personalized philosophical prompts via notifications
* Schedules notifications based on user engagement patterns
* Creates "philosophical question of the day" notifications
* Sends re-engagement prompts for inactive users
* Celebrates philosophical milestones with special notifications

## 5.8.6 Gamification & XP Integration

* **XP Awards**
  + Engaging with Lumen prompts: 5 XP
  + Following Lumen navigation guidance: 10 XP
  + Providing feedback on Lumen interventions: 5 XP
  + Completing Lumen-suggested cross-feature actions: 15 XP
  + Reaching Lumen evolution milestones: 25 XP
* **Achievements**
  + "Philosophical Companion": First Lumen interaction
  + "Guided Explorer": Follow 10 Lumen navigation suggestions
  + "Deep Thinker": Engage with 20 philosophical prompts
  + "Evolving Wisdom": Reach evolution stage 5 with Lumen
  + "Philosophical Dialogue": Complete 50 Lumen interactions
* **Visual Evolution Stages**
  + Stage 1: Basic orb with gentle glow
  + Stage 2: Enhanced glow with subtle particle effects
  + Stage 3: Distinct constellation patterns within orb
  + Stage 4: Dynamic color shifts reflecting philosophical areas
  + Stage 5: Complex particle effects and unique animations

## 5.8.7 Accessibility Features

* **Screen Reader Support**
  + Descriptive labels for Lumen's appearance and actions
  + Text alternatives for all visual guidance
  + Keyboard navigation for all Lumen interactions
  + Announcement of Lumen's appearance and purpose
* **Visual Accessibility**
  + High contrast mode for Lumen's appearance
  + Adjustable size options for better visibility
  + Color blind friendly visual indicators
  + Text-based alternatives to visual guidance
* **Motor Accessibility**
  + Voice command options for Lumen interaction
  + Adjustable timing for interaction responses
  + Simplified gesture alternatives
  + Persistent controls for users with motor limitations
* **Cognitive Accessibility**
  + Adjustable complexity of philosophical prompts
  + Clear, concise language options
  + Reduced animation mode
  + Consistent and predictable interaction patterns

## 5.8.8 Implementation Considerations

* **AI Integration**
  + Leverages existing AI Router system (Section 9.1)
  + Extends Context-Specific Tone Prompt Workflow (Section 9.3)
  + Implements specialized prompt templates for guidance
  + Maintains compatibility with all AI models
* **Performance Optimization**
  + Lightweight visual implementation to minimize battery impact
  + Efficient animation system using React Native Reanimated
  + Background processing of complex personalization operations
  + Caching of common philosophical prompts and guidance
* **Offline Support**
  + Core Lumen functionality available offline
  + Cached philosophical content for offline guidance
  + Synchronization of interactions when connection restored
  + Graceful degradation of advanced features when offline
* **Mobile Optimization**
  + Touch-optimized interaction design
  + Appropriate sizing for all screen dimensions
  + Battery-efficient animations and effects
  + Responsive positioning across device orientations

## 5.8.9 Celery Tasks

# src/tasks/lumen\_tasks.py

@celery\_app.task

def update\_philosophical\_profile(user\_id):

"""

Analyzes user activity to update their philosophical profile.

Parameters:

- user\_id: UUID of the user

Returns:

- Updated philosophical profile

"""

# Get user's recent activity

user\_service = UserService()

activity = user\_service.get\_recent\_activity(user\_id, days=30)

# Extract philosophical interests

concept\_service = ConceptService()

interests = concept\_service.extract\_interests\_from\_activity(activity)

# Update user's philosophical profile

lumen\_service = LumenService()

profile = lumen\_service.update\_user\_profile(user\_id, interests)

return profile

@celery\_app.task

def generate\_personalized\_prompts(user\_id):

"""

Creates personalized philosophical prompts based on user's profile.

Parameters:

- user\_id: UUID of the user

Returns:

- List of generated prompts

"""

# Get user's philosophical profile

lumen\_service = LumenService()

profile = lumen\_service.get\_user\_profile(user\_id)

# Generate prompts using AI service

ai\_service = AIService()

prompts = ai\_service.generate\_philosophical\_prompts(

interests=profile.get('interests'),

sophistication\_level=profile.get('sophistication\_level'),

learning\_style=profile.get('preferred\_learning\_style'),

count=10

)

# Store prompts for later use

lumen\_service.store\_personalized\_prompts(user\_id, prompts)

return prompts

@celery\_app.task

def schedule\_philosophical\_notifications(user\_id):

"""

Schedules philosophical notifications based on user's engagement patterns.

Parameters:

- user\_id: UUID of the user

Returns:

- Scheduled notification details

"""

# Get user preferences

user\_service = UserService()

preferences = user\_service.get\_notification\_preferences(user\_id)

if not preferences.get('notifications\_enabled'):

return {"status": "notifications\_disabled"}

# Get optimal notification times

analytics\_service = AnalyticsService()

optimal\_times = analytics\_service.get\_optimal\_engagement\_times(user\_id)

# Get personalized prompts

lumen\_service = LumenService()

prompts = lumen\_service.get\_personalized\_prompts(user\_id)

# Schedule notifications

notification\_service = NotificationService()

scheduled = notification\_service.schedule\_philosophical\_notifications(

user\_id=user\_id,

prompts=prompts,

times=optimal\_times,

frequency=preferences.get('notification\_frequency', 'medium')

)

return scheduled

@celery\_app.task

def analyze\_intervention\_effectiveness(days=7):

"""

Analyzes the effectiveness of Lumen interventions across users.

Parameters:

- days: Number of days to analyze

Returns:

- Analysis results

"""

# Get intervention data

lumen\_service = LumenService()

interventions = lumen\_service.get\_recent\_interventions(days=days)

# Analyze effectiveness by type

analytics\_service = AnalyticsService()

effectiveness = analytics\_service.analyze\_intervention\_effectiveness(interventions)

# Update intervention rules based on effectiveness

lumen\_service.optimize\_intervention\_rules(effectiveness)

return effectiveness

@celery\_app.task

def update\_lumen\_visual\_evolution(user\_id):

"""

Updates Lumen's visual appearance based on user's philosophical journey.

Parameters:

- user\_id: UUID of the user

Returns:

- Updated visual state

"""

# Get user's progression data

user\_service = UserService()

progression = user\_service.get\_user\_progression(user\_id)

# Calculate appropriate evolution stage

lumen\_service = LumenService()

evolution\_stage = lumen\_service.calculate\_evolution\_stage(

xp\_level=progression.get('xp\_level'),

concept\_mastery=progression.get('concept\_mastery'),

days\_active=progression.get('days\_active')

)

# Update visual state

visual\_state = lumen\_service.update\_visual\_state(

user\_id=user\_id,

evolution\_stage=evolution\_stage,

philosophical\_interests=progression.get('top\_interests')

)

return visual\_state

@celery\_app.task

def generate\_cross\_feature\_recommendations(user\_id):

"""

Creates personalized cross-feature recommendations based on user activity.

Parameters:

- user\_id: UUID of the user

Returns:

- List of cross-feature recommendations

"""

# Get user's recent activity

user\_service = UserService()

activity = user\_service.get\_recent\_activity(user\_id, days=14)

# Analyze activity for patterns

analytics\_service = AnalyticsService()

patterns = analytics\_service.identify\_activity\_patterns(activity)

# Generate cross-feature recommendations

lumen\_service = LumenService()

recommendations = lumen\_service.generate\_cross\_feature\_recommendations(

user\_id=user\_id,

patterns=patterns,

max\_recommendations=5

)

return recommendations

@celery\_app.task

def sync\_lumen\_with\_wisdom\_xp(user\_id):

"""

Synchronizes Lumen's behavior with the Wisdom XP system.

Parameters:

- user\_id: UUID of the user

Returns:

- Synchronization results

"""

# Get user's XP data

xp\_service = XPService()

xp\_data = xp\_service.get\_user\_xp\_data(user\_id)

# Update Lumen based on XP progress

lumen\_service = LumenService()

sync\_results = lumen\_service.sync\_with\_wisdom\_xp(

user\_id=user\_id,

xp\_level=xp\_data.get('level'),

achievements=xp\_data.get('achievements'),

streaks=xp\_data.get('streaks')

)

return sync\_results

## 5.8.10 Implementation Phases

### Phase 1: Foundation (Weeks 1-4)

* Implement Lumen Core Engine
* Develop basic User Context Service
* Create initial Visual Presentation Manager
* Integrate with Explore feature for basic navigation
* Implement fundamental intervention rules

### Phase 2: Core Features (Weeks 5-8)

* Expand feature integrations to Ask and Journal
* Enhance Content Generation Engine
* Implement basic Notification integration
* Develop initial philosophical prompts
* Create first visual evolution stages

### Phase 3: Advanced Capabilities (Weeks 9-12)

* Integrate with Quest and Forum features
* Implement advanced AI capabilities
* Develop comprehensive intervention rules
* Create sophisticated visual evolutions
* Enhance personalization capabilities

### Phase 4: Refinement (Weeks 13-16)

* Implement A/B testing framework
* Optimize performance across all components
* Enhance accessibility features
* Refine intervention timing and frequency
* Develop advanced analytics dashboards

## 9.7 Lumen AI Integration

*Note: This section extends the AI Implementation section (9) with Lumen-specific AI integration details.*

### 9.7.1 Lumen AI Architecture

#### 9.7.1.1 AI Integration Points

* Extends the AI Router (Section 9.1) with specialized Lumen capabilities
* Implements dedicated prompt templates for philosophical guidance
* Utilizes Context-Specific Tone Prompt Workflow (Section 9.3) for consistent voice
* Leverages all available AI models with specialized routing for different guidance types

#### 9.7.1.2 Philosophical Intelligence Layer

* Implements concept relationship mapping for navigation guidance
* Develops philosophical question generation capabilities
* Creates insight synthesis for connecting disparate philosophical ideas
* Builds user philosophical profile through continuous learning

#### 9.7.1.3 Contextual Awareness System

* Maintains awareness of user's current app location and activity
* Tracks philosophical exploration history and patterns
* Monitors engagement with concepts and features
* Detects opportune moments for meaningful interventions

### 9.7.2 Prompt Engineering for Lumen

#### 9.7.2.1 Guidance Prompt Templates

You are Lumen, a philosophical guide in the Setarcos app. Your purpose is to {guidance\_purpose} while maintaining a {tone\_description} tone. The user is currently exploring {current\_context} and has previously shown interest in {user\_interests}.

Based on this context, provide {guidance\_type} that:

1. Connects to their current activity

2. Respects their philosophical sophistication level ({sophistication\_level}/5)

3. Aligns with their preferred learning style ({learning\_style})

4. Is concise and mobile-friendly (max 2-3 sentences)

Your guidance should feel illuminating but not intrusive, like a gentle light revealing a path rather than a spotlight demanding attention.

#### 9.7.2.2 Philosophical Prompt Categories

* **Reflective**: Prompts that encourage personal philosophical reflection
* **Connective**: Prompts that highlight relationships between concepts
* **Exploratory**: Prompts that encourage discovery of new philosophical areas
* **Practical**: Prompts that connect philosophy to everyday life
* **Historical**: Prompts that provide context about philosophical traditions

#### 9.7.2.3 Model-Specific Optimizations

* **Grok**: Optimized for creative philosophical connections and unexpected insights
* **Claude**: Optimized for nuanced ethical guidance and reflective prompts
* **Gemini**: Optimized for visual concept mapping and spatial relationships
* **ChatGPT**: Optimized for practical applications and accessible explanations

### 9.7.3 Personalization Algorithm

#### 9.7.3.1 Philosophical Interest Mapping

* Tracks engagement with philosophical concepts
* Identifies patterns in question topics and journal entries
* Maps interests to philosophical branches and traditions
* Creates weighted interest graph for personalization

#### 9.7.3.2 Intervention Selection Logic

def select\_intervention(user\_id, current\_context):

# Get user profile and preferences

user\_profile = get\_user\_philosophical\_profile(user\_id)

preferences = get\_lumen\_preferences(user\_id)

# Generate potential interventions

potential\_interventions = []

# Navigation interventions

if current\_context.get('feature') == 'explore':

navigation\_opportunities = generate\_navigation\_opportunities(

user\_id, current\_context, user\_profile

)

potential\_interventions.extend(navigation\_opportunities)

# Philosophical prompt interventions

prompt\_opportunities = generate\_prompt\_opportunities(

user\_id, current\_context, user\_profile

)

potential\_interventions.extend(prompt\_opportunities)

# Cross-feature interventions

if has\_relevant\_cross\_feature\_opportunities(user\_id, current\_context):

cross\_feature\_opportunities = generate\_cross\_feature\_opportunities(

user\_id, current\_context, user\_profile

)

potential\_interventions.extend(cross\_feature\_opportunities)

# Filter based on user preferences

filtered\_interventions = filter\_by\_preferences(

potential\_interventions, preferences

)

# Apply cooldown rules

valid\_interventions = apply\_cooldown\_rules(

filtered\_interventions, get\_recent\_interventions(user\_id)

)

# If no valid interventions, return None

if not valid\_interventions:

return None

# Score and select best intervention

scored\_interventions = score\_interventions(

valid\_interventions, user\_profile, current\_context

)

return select\_best\_intervention(scored\_interventions)

#### 9.7.3.3 Learning from Feedback

* Incorporates explicit feedback from user ratings
* Analyzes implicit feedback from interaction patterns
* Adjusts intervention frequency based on response rates
* Refines philosophical content based on engagement metrics

### 9.7.4 Quality Assurance

#### 9.7.4.1 Philosophical Accuracy

* Review process for philosophical prompt templates
* Validation of concept relationships and connections
* Accuracy checks for philosophical references
* Balanced representation of philosophical traditions

#### 9.7.4.2 Intervention Quality Metrics

* Relevance to current context
* Alignment with user interests
* Philosophical depth appropriateness
* User response and engagement
* Cross-feature transition effectiveness

#### 9.7.4.3 A/B Testing Framework

* Tests alternative intervention strategies
* Compares effectiveness of different prompt styles
* Evaluates optimal intervention frequency
* Measures impact on overall app engagement

## 10.5 Lumen Asynchronous Tasks

*Note: This section extends the Asynchronous Processing section (10) with Lumen-specific task definitions.*

### 10.5.1 Lumen Task Configuration

* Integration with existing Celery configuration (Section 10.1)
* Task prioritization for time-sensitive interventions
* Scheduled tasks for profile updates and content generation
* Error handling and retry strategies

### 10.5.2 Lumen Task Definitions

* Philosophical profile updates
* Personalized prompt generation
* Intervention effectiveness analysis
* Visual evolution updates
* Cross-feature recommendation generation
* Notification scheduling
* Wisdom XP synchronization

### 10.5.3 Lumen Task Workflows

* Daily user profile update workflow
* Intervention optimization workflow
* Re-engagement workflow for inactive users
* Philosophical notification workflow
* Evolution milestone workflow

## 11.4 Lumen Analytics

*Note: This section extends the Analytics Implementation section (11) with Lumen-specific analytics.*

### 11.4.1 Lumen-Specific Events

* Lumen appearance events
* Intervention response events
* Guidance acceptance events
* Cross-feature transition events
* Evolution milestone events

### 11.4.2 Lumen Effectiveness Metrics

* Intervention acceptance rate
* Guidance follow-through rate
* Feature discovery attribution
* Session extension impact
* Philosophical engagement depth
* Cross-feature transition success

### 11.4.3 Lumen Analytics Dashboard

* Intervention effectiveness visualization
* User engagement patterns
* Philosophical interest mapping
* Evolution stage distribution
* A/B test results visualization