5.2 Quest Feature

Feature Goal: Guide users through structured philosophical explorations with clear learning objectives, interactive elements, and rewards using an engaging skill tree design.

5.2.1 Definitions and Glossary

5.2.1.1 Quest

A structured philosophical exploration with defined learning objectives, interactive elements, and rewards. Quests guide users through philosophical concepts in a systematic way, combining educational content with interactive challenges.

5.2.1.2 Skill Tree

A visual representation of quest progression showing relationships between quests, prerequisites, and unlockable content. The skill tree provides a spatial metaphor for learning progression and allows users to visualize their philosophical journey.

5.2.1.3 Quest Step

An individual component within a quest that contains specific content and interaction requirements. Steps may include readings, reflections, multiple-choice questions, or other interactive elements.

5.2.1.4 Challenge Node

A specialized quest step that offers additional difficulty and rewards bonus XP. Challenge nodes test deeper understanding and encourage users to engage more thoroughly with philosophical concepts.

5.2.1.5 Quest Badge

A visual achievement awarded for reaching specific milestones in quest completion or mastery. Badges serve as extrinsic motivation and visible markers of progress.

5.2.2 System Architecture

5.2.2.1 Frontend Components

QuestScreen.tsx: Quest browsing and selection

- Implements filtering by difficulty, concept, and completion status

- Renders quest cards with visual indicators for status and rewards

- Integrates with recommendation engine for personalized suggestions

QuestDetailScreen.tsx: Individual quest view with skill tree visualization

- Renders quest metadata, description, and prerequisites

- Displays estimated duration and XP rewards

- Provides start/continue buttons based on user progress

- Shows concept relationships and learning objectives

SkillTreeView.tsx: Zoomable visualization of the quest progression system

- Implements WebGL-based rendering for smooth performance

- Supports touch and mouse-based zoom and pan interactions

- Renders nodes, connections, and visual state indicators

- Handles node selection and navigation events

QuestStep.tsx: Individual step in a quest

- Dynamically renders different step types (reading, multiple-choice, reflection)

- Handles user input validation and submission

- Displays concept links and related resources

- Shows progress indicators and navigation controls

QuestProgress.tsx: Progress indicator

- Renders progress bar with percentage completion

- Shows step indicators with completion status

- Displays XP earned and remaining

- Animates progress updates for positive reinforcement

QuestNodeComponent.tsx: Individual node in skill tree

- Renders node based on completion status (locked, available, completed)

- Handles click/tap interactions for node selection

- Displays tooltips with quest information on hover/focus

- Animates state changes for visual feedback

ConceptMiniTree.tsx: Mini visualization of related concepts

- Renders hierarchical concept relationships

- Provides interactive navigation to concept details

- Shows mastery level indicators for each concept

- Supports collapsible/expandable nodes for complex hierarchies

5.2.2.2 State Management

QuestStore: Central state management for quest-related data

- Maintains quest list, details, and user progress

- Handles filtering and sorting operations

- Manages skill tree state and navigation

- Coordinates with API service for data operations

ProgressStore: Tracks and manages user progression data

- Maintains XP totals and level information

- Tracks badge awards and unlocks

- Manages quest completion statistics

- Coordinates with notification system for achievements

5.2.2.3 Frontend Services

QuestApiService: Handles all quest-related API communications

- Fetches quest lists and details

- Submits quest start and completion requests

- Retrieves user progress and recommendations

- Manages error handling and retry logic

SkillTreeRenderer: Manages skill tree visualization

- Handles layout calculations and positioning

- Manages zoom and pan interactions

- Optimizes rendering performance

- Supports different device capabilities

5.2.2.4 Backend Components

quest\_service.py: Quest management and progress tracking

- Handles quest retrieval, filtering, and recommendation

- Manages user quest progress and completion

- Processes step submissions and validation

- Coordinates with XP service for rewards

- Implements skill tree data structure and navigation

- Manages quest prerequisites and unlocking logic

- Handles quest content delivery and caching

- Provides analytics data for quest engagement

concept\_service.py: Concept integration

- Maps concepts to quest content and steps

- Tracks concept mastery through quest completion

- Provides concept relationship data for mini-tree

- Updates concept engagement metrics

xp\_service.py: XP tracking and badge management

- Awards XP for quest-related activities

- Tracks user XP totals and level progression

- Manages badge eligibility and awards

- Provides XP thresholds for quest unlocking

5.2.2.5 Data Access

QuestRepository: Data access layer for quest-related operations

- Implements CRUD operations for quest entities

- Provides optimized queries for quest filtering and search

- Manages caching for frequently accessed quest data

- Handles transaction management for quest operations

UserQuestRepository: Data access for user-quest relationships

- Tracks user progress across quests and steps

- Manages user response data and submissions

- Provides analytics queries for user engagement

- Implements efficient progress tracking algorithms

5.2.3 Database Models

5.2.3.1 quests: Quest definitions and metadata

CREATE TABLE quests (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

title VARCHAR(100) NOT NULL,

description TEXT NOT NULL,

difficulty VARCHAR(20) NOT NULL,

estimated\_duration INTEGER NOT NULL,

xp\_reward INTEGER NOT NULL,

concepts JSONB NOT NULL,

prerequisites JSONB,

is\_premium BOOLEAN DEFAULT FALSE,

skill\_tree\_structure JSONB DEFAULT '{}',

xp\_threshold INTEGER DEFAULT 0,

accessibility\_options JSONB DEFAULT '{}',

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

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

CREATE INDEX idx\_quests\_difficulty ON quests(difficulty);

CREATE INDEX idx\_quests\_is\_premium ON quests(is\_premium);

CREATE INDEX idx\_quests\_xp\_threshold ON quests(xp\_threshold);

CREATE INDEX idx\_quests\_concepts ON quests USING GIN(concepts);

5.2.3.2 quest\_steps: Individual steps within quests

CREATE TABLE quest\_steps (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

quest\_id UUID NOT NULL REFERENCES quests(id),

step\_number INTEGER NOT NULL,

title VARCHAR(100) NOT NULL,

content TEXT NOT NULL,

step\_type VARCHAR(50) NOT NULL,

interaction\_data JSONB,

xp\_reward INTEGER NOT NULL,

node\_position JSONB DEFAULT '{}',

connected\_nodes JSONB DEFAULT '[]',

concept\_links JSONB DEFAULT '[]',

is\_challenge\_node BOOLEAN DEFAULT FALSE,

bonus\_xp INTEGER DEFAULT 0,

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

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

UNIQUE(quest\_id, step\_number)

);

CREATE INDEX idx\_quest\_steps\_quest\_id ON quest\_steps(quest\_id);

CREATE INDEX idx\_quest\_steps\_step\_type ON quest\_steps(step\_type);

CREATE INDEX idx\_quest\_steps\_is\_challenge\_node ON quest\_steps(is\_challenge\_node);

CREATE INDEX idx\_quest\_steps\_concept\_links ON quest\_steps USING GIN(concept\_links);

5.2.3.3 user\_quests: User progress on quests

CREATE TABLE user\_quests (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

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

quest\_id UUID NOT NULL REFERENCES quests(id),

status VARCHAR(20) NOT NULL DEFAULT 'not\_started',

current\_step INTEGER,

started\_at TIMESTAMP WITH TIME ZONE,

completed\_at TIMESTAMP WITH TIME ZONE,

xp\_earned INTEGER DEFAULT 0,

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

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

UNIQUE(user\_id, quest\_id)

);

CREATE INDEX idx\_user\_quests\_user\_id ON user\_quests(user\_id);

CREATE INDEX idx\_user\_quests\_quest\_id ON user\_quests(quest\_id);

CREATE INDEX idx\_user\_quests\_status ON user\_quests(status);

CREATE INDEX idx\_user\_quests\_completed\_at ON user\_quests(completed\_at);

5.2.3.4 user\_quest\_steps: User progress on individual steps

CREATE TABLE user\_quest\_steps (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

user\_quest\_id UUID NOT NULL REFERENCES user\_quests(id),

step\_id UUID NOT NULL REFERENCES quest\_steps(id),

status VARCHAR(20) NOT NULL DEFAULT 'not\_started',

user\_response JSONB,

started\_at TIMESTAMP WITH TIME ZONE,

completed\_at TIMESTAMP WITH TIME ZONE,

xp\_earned INTEGER DEFAULT 0,

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

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

UNIQUE(user\_quest\_id, step\_id)

);

CREATE INDEX idx\_user\_quest\_steps\_user\_quest\_id ON user\_quest\_steps(user\_quest\_id);

CREATE INDEX idx\_user\_quest\_steps\_step\_id ON user\_quest\_steps(step\_id);

CREATE INDEX idx\_user\_quest\_steps\_status ON user\_quest\_steps(status);

CREATE INDEX idx\_user\_quest\_steps\_completed\_at ON user\_quest\_steps(completed\_at);

5.2.3.5 quest\_badges: Badges for quest achievements

CREATE TABLE quest\_badges (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

name VARCHAR(100) NOT NULL,

description TEXT NOT NULL,

xp\_threshold INTEGER NOT NULL,

quest\_category VARCHAR(50) NOT NULL,

icon\_url VARCHAR(255) NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE INDEX idx\_quest\_badges\_quest\_category ON quest\_badges(quest\_category);

CREATE INDEX idx\_quest\_badges\_xp\_threshold ON quest\_badges(xp\_threshold);

5.2.3.6 user\_quest\_badges: User earned badges

CREATE TABLE user\_quest\_badges (

id UUID PRIMARY KEY DEFAULT uuid\_generate\_v4(),

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

badge\_id UUID NOT NULL REFERENCES quest\_badges(id),

earned\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

UNIQUE(user\_id, badge\_id)

);

CREATE INDEX idx\_user\_quest\_badges\_user\_id ON user\_quest\_badges(user\_id);

CREATE INDEX idx\_user\_quest\_badges\_badge\_id ON user\_quest\_badges(badge\_id);

CREATE INDEX idx\_user\_quest\_badges\_earned\_at ON user\_quest\_badges(earned\_at);

5.2.4 API Endpoints

5.2.4.1 GET /api/v1/quests

Description: Retrieve a list of available quests with filtering options

Authentication: Required

Parameters:

- page: Pagination page number (default: 1)

- limit: Items per page (default: 20, max: 50)

- difficulty: Filter by difficulty level (beginner, intermediate, advanced)

- concept\_id: Filter by related concept

- status: Filter by user status (not\_started, in\_progress, completed)

- skill\_tree: Boolean to return skill tree view (default: false)

Response:

```json

{

"quests": [

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"title": "Introduction to Stoicism",

"description": "Explore the foundational principles of Stoic philosophy",

"difficulty": "beginner",

"estimated\_duration": 30,

"xp\_reward": 100,

"concepts": ["stoicism", "virtue", "nature"],

"is\_premium": false,

"user\_status": "not\_started"

},

{

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Epicurean Philosophy",

"description": "Discover the philosophy of pleasure and tranquility",

"difficulty": "intermediate",

"estimated\_duration": 45,

"xp\_reward": 150,

"concepts": ["epicureanism", "pleasure", "tranquility"],

"is\_premium": false,

"user\_status": "in\_progress"

}

],

"total": 15,

"page": 1,

"pages": 1,

"limit": 20

}

```

5.2.4.2 GET /api/v1/quests/{id}

Description: Retrieve detailed information about a specific quest

Authentication: Required

Parameters: None

Response:

```json

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"title": "Introduction to Stoicism",

"description": "Explore the foundational principles of Stoic philosophy",

"difficulty": "beginner",

"estimated\_duration": 30,

"xp\_reward": 100,

"concepts": [

{

"id": "123",

"name": "stoicism",

"description": "An ancient Greek philosophy that teaches virtue and resilience"

},

{

"id": "124",

"name": "virtue",

"description": "Moral excellence and righteousness"

},

{

"id": "125",

"name": "nature",

"description": "The physical world and its phenomena"

}

],

"prerequisites": [],

"is\_premium": false,

"steps": [

{

"id": "650e8400-e29b-41d4-a716-446655440001",

"step\_number": 1,

"title": "The Stoic Worldview",

"content": "Stoicism teaches that virtue is the only true good...",

"step\_type": "reading",

"xp\_reward": 20

},

{

"id": "650e8400-e29b-41d4-a716-446655440002",

"step\_number": 2,

"title": "The Four Cardinal Virtues",

"content": "Stoics identified four primary virtues...",

"step\_type": "multiple\_choice",

"interaction\_data": {

"question": "Which of the following is NOT one of the four cardinal virtues in Stoicism?",

"options": ["Wisdom", "Justice", "Courage", "Happiness", "Temperance"],

"correct\_answer": 3

},

"xp\_reward": 25,

"is\_challenge\_node": true,

"bonus\_xp": 10

}

],

"user\_progress": {

"status": "not\_started",

"current\_step": null,

"started\_at": null,

"completed\_at": null,

"xp\_earned": 0

},

"skill\_tree\_structure": {

"position": {"x": 100, "y": 200},

"connected\_quests": ["550e8400-e29b-41d4-a716-446655440001"]

},

"xp\_threshold": 0,

"accessibility\_options": {

"high\_contrast\_available": true,

"audio\_narration\_available": true

}

}

```

5.2.4.3 GET /api/v1/quests/skill-tree

Description: Retrieve the full skill tree visualization data

Authentication: Required

Parameters: None

Response:

```json

{

"quests": [

{

"id": "550e8400-e29b-41d4-a716-446655440000",

"title": "Introduction to Stoicism",

"description": "Explore the foundational principles of Stoic philosophy",

"xp\_threshold": 0,

"user\_status": "completed",

"position": {"x": 100, "y": 200},

"connected\_quests": ["550e8400-e29b-41d4-a716-446655440001"]

},

{

"id": "550e8400-e29b-41d4-a716-446655440001",

"title": "Advanced Stoic Practices",

"description": "Deepen your understanding of Stoic philosophy through practical exercises",

"xp\_threshold": 500,

"user\_status": "locked",

"position": {"x": 300, "y": 200},

"connected\_quests": ["550e8400-e29b-41d4-a716-446655440002"]

},

{

"id": "550e8400-e29b-41d4-a716-446655440002",

"title": "Stoicism and Modern Life",

"description": "Apply Stoic principles to contemporary challenges",

"xp\_threshold": 1000,

"user\_status": "locked",

"position": {"x": 500, "y": 200},

"connected\_quests": []

}

],

"user\_xp": 350,

"next\_unlock": {

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

"xp\_required": 500,

"xp\_remaining": 150

},

"categories": [

{

"name": "Stoicism",

"quests": ["550e8400-e29b-41d4-a716-446655440000", "550e8400-e29b-41d4-a716-446655440001", "550e8400-e29b-41d4-a716-446655440002"],

"color": "#4A90E2"

},

{

"name": "Epicureanism",

"quests": ["550e8400-e29b-41d4-a716-446655440003", "550e8400-e29b-41d4-a716-446655440004"],

"color": "#50E3C2"

}

]

}

```

5.2.4.4 POST /api/v1/quests/{id}/start

Description: Start a quest or resume progress

Authentication: Required

Request Body: None

Response:

```json

{

"success": true,

"user\_quest\_id": "750e8400-e29b-41d4-a716-446655440000",

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

"status": "in\_progress",

"current\_step": 1,

"xp\_earned": 5,

"total\_xp": 1255,

"xp\_to\_next\_level": 745

}

```

5.2.4.5 POST /api/v1/quests/{id}/steps/{step\_id}/complete

Description: Complete a quest step and submit user response

Authentication: Required

Request Body:

```json

{

"user\_response": {

"answer": "Virtue is living in accordance with nature.",

"reflection": "I find this concept particularly relevant because..."

}

}

```

Response:

```json

{

"success": true,

"step\_id": "650e8400-e29b-41d4-a716-446655440001",

"status": "completed",

"xp\_earned": 20,

"next\_step": {

"id": "650e8400-e29b-41d4-a716-446655440002",

"step\_number": 2,

"title": "The Four Cardinal Virtues",

"content": "Stoics identified four primary virtues...",

"step\_type": "multiple\_choice",

"interaction\_data": {

"question": "Which of the following is NOT one of the four cardinal virtues in Stoicism?",

"options": ["Wisdom", "Justice", "Courage", "Happiness", "Temperance"],

"correct\_answer": 3

},

"xp\_reward": 25

},

"quest\_progress": {

"completed\_steps": 1,

"total\_steps": 5,

"percentage": 20

},

"total\_xp": 1275,

"xp\_to\_next\_level": 725

}

```

5.2.4.6 GET /api/v1/quests/recommended

Description: Get personalized quest recommendations

Authentication: Required

Parameters:

- limit: Maximum number of recommendations (default: 5, max: 10)

Response:

```json

{

"recommended\_quests": [

{

"id": "550e8400-e29b-41d4-a716-446655440002",

"title": "Introduction to Epicureanism",

"description": "Discover the philosophy of pleasure and tranquility",

"difficulty": "beginner",

"estimated\_duration": 30,

"xp\_reward": 100,

"reason": "Based on your interest in Stoicism"

},

{

"id": "550e8400-e29b-41d4-a716-446655440003",

"title": "Aristotle's Ethics",

"description": "Explore Aristotle's views on virtue and the good life",

"difficulty": "intermediate",

"estimated\_duration": 45,

"xp\_reward": 150,

"reason": "Popular with users who completed Introduction to Stoicism"

}

]

}

```

5.2.4.7 POST /api/v1/quests/{id}/concepts/{concept\_id}/explore

Description: Track concept exploration from quest and retrieve concept details

Authentication: Required

Request Body: None

Response:

```json

{

"success": true,

"concept": {

"id": "123",

"name": "stoicism",

"description": "An ancient Greek philosophy that teaches virtue and resilience"

},

"related\_concepts": [

{

"id": "124",

"name": "virtue",

"description": "Moral excellence and righteousness",

"relationship": "core principle"

},

{

"id": "125",

"name": "nature",

"description": "The physical world and its phenomena",

"relationship": "guiding force"

}

],

"mastery": {

"level": 2,

"percentage": 40,

"next\_level\_at": 75

}

}

```

5.2.5 User Experience Flow

5.2.5.1 Quest Discovery and Selection

- User navigates to the Quest screen from the main navigation

- System displays available quests with filtering options

- "Recommended for You" filter is auto-applied based on user progress and interests

- User can sort by difficulty, estimated duration, or XP reward

- Each quest card shows title, brief description, difficulty, duration, and XP reward

- Premium quests are clearly marked with a visual indicator

- User selects a quest to view details

5.2.5.2 Quest Detail Exploration

- System displays comprehensive quest information

- User sees description, learning objectives, and related concepts

- Prerequisites are clearly indicated with completion status

- Estimated duration and XP rewards are prominently displayed

- Skill tree position is visualized to show relationship to other quests

- User can see their current progress if the quest was previously started

- "Start Quest" or "Continue Quest" button is displayed based on progress

5.2.5.3 Skill Tree Navigation

- User can access the full skill tree view from the Quest screen

- System renders an interactive, zoomable visualization of all quests

- Quests are organized spatially to show progression paths and relationships

- Color coding indicates different philosophical traditions or themes

- Locked quests show XP thresholds required for unlocking

- Completed quests are visually distinguished from in-progress and not-started quests

- User can tap/click on any quest to view its details

- Zooming and panning controls allow exploration of the full tree

5.2.5.4 Starting a Quest

- User taps "Start Quest" button on the Quest Detail screen

- System creates a user\_quest record and awards initial XP (+5)

- User receives visual feedback of XP gain

- System presents the first quest step with clear instructions

- Progress indicator shows 0% completion

- User can access quest overview at any time via an info button

5.2.5.5 Completing Quest Steps

- System presents each step sequentially with appropriate UI based on step type:

- Reading: Text content with concept links and optional images

- Multiple Choice: Question with selectable options and feedback

- Reflection: Prompt with text input area and optional pre-filled examples

- Challenge: Special step with bonus XP and increased difficulty

- User completes the required interaction for the step

- System validates input where applicable (e.g., multiple choice answers)

- User submits their response via the "Submit" or "Continue" button

- System records completion, awards XP, and shows visual feedback

- Progress indicator updates to reflect advancement

- System presents the next step automatically

5.2.5.6 Concept Exploration

- User can tap on concept links within quest steps to explore related concepts

- System opens the Concept Detail view as an overlay or modal

- Mini-tree visualization shows the concept's position in the hierarchy

- Related concepts are displayed with their relationships

- User's current mastery level for the concept is shown

- "Back to Quest" button allows seamless return to the quest step

- System tracks this exploration for analytics and mastery progression

5.2.5.7 Quest Completion

- After completing the final step, system shows a celebration animation

- Total XP earned is displayed with level progress

- Any badges earned are showcased with descriptions

- Concept mastery progress is summarized

- Recommendations for next quests are presented

- User can share their achievement via the Share feature

- "Explore More Quests" button returns to the Quest screen

5.2.6 XP Integration

5.2.6.1 XP Award Structure

- Starting quests: +5 XP

- Completing quest steps: +10-30 XP per step (based on difficulty)

- Beginner quest steps: 10-15 XP

- Intermediate quest steps: 15-20 XP

- Advanced quest steps: 20-30 XP

- Challenge nodes: Additional bonus XP (+5-15 XP)

- Completing full quests: +25-100 XP (based on difficulty)

- Beginner quests: 25-50 XP

- Intermediate quests: 50-75 XP

- Advanced quests: 75-100 XP

5.2.6.2 XP Thresholds for Quest Unlocking

- Beginner quests: 0 XP (available immediately)

- Intermediate quests: 250-500 XP

- Advanced quests: 500-1000 XP

- Expert quests: 1000+ XP

5.2.6.3 XP Visualization

- Standard XP bar shows progress toward next user level

- Quest-specific progress indicators show completion percentage

- Skill tree displays XP thresholds for locked quests

- XP gains are animated to provide positive reinforcement

- Total XP and XP to next level are displayed on profile and quest screens

5.2.6.4 Badge Integration

- Quest Novice: Complete 1 quest (any difficulty)

- Quest Enthusiast: Complete 5 quests

- Quest Adept: Complete 15 quests

- Quest Master: Complete 30 quests

- Quest Sage: Complete 50 quests

- Beginner's Mind: Complete all beginner quests

- Philosophical Journeyer: Complete quests across 3 different philosophical traditions

- Challenge Seeker: Complete 10 challenge nodes

- Perfect Philosopher: Achieve 100% on all multiple-choice questions in a quest

5.2.7 Concepts Integration

5.2.7.1 Concept Linking

- Quest steps contain hyperlinked concepts within content

- Each concept link opens the concept detail view when tapped

- Concepts table stores relationships between concepts and quests

- Quest completion increases mastery of related concepts

- Concept exploration from quests is tracked for analytics

5.2.7.2 Concept Visualization

- Mini-tree visualization shows hierarchical relationships

- Concept detail view displays definition, description, and examples

- Related concepts are shown with relationship types

- User's mastery level is indicated with visual progress bar

- Concepts learned through quests are highlighted in the user's profile

5.2.7.3 Mastery Progression

- Completing quest steps with concept links increases mastery by 5-10%

- Challenge nodes related to concepts provide larger mastery boosts (10-15%)

- Completing full quests increases mastery of primary concepts by 15-25%

- Mastery levels unlock additional content and advanced quests

- Mastery is visualized consistently across the app

5.2.7.4 Navigation Integration

- "Back to Quest" button ensures seamless return from concept exploration

- Breadcrumb navigation shows path from quest to concept

- Recently explored concepts are saved for quick access

- Concept search is available within the quest interface

- Related quests are shown on concept detail pages

5.2.8 Accessibility Features

5.2.8.1 Visual Accessibility

- High-contrast mode enhances visibility of quest elements

- Color blindness considerations in skill tree visualization

- Adjustable text sizes for all quest content

- Visual indicators do not rely solely on color

- Animations can be reduced or disabled

- Sufficient contrast ratios (WCAG AA compliance)

5.2.8.2 Screen Reader Support

- ARIA labels for all interactive elements

- Semantic HTML structure for proper navigation

- Descriptive alt text for images and diagrams

- Announcement of XP gains and progress updates

- Keyboard focus management for modal dialogs

- Screen reader optimization mode

5.2.8.3 Input Accessibility

- Voice input for reflections and text responses

- Alternative input methods for multiple choice (keyboard, voice)

- Touch targets sized appropriately (minimum 44x44 pixels)

- Keyboard navigation for all interactive elements

- Reduced motion option for animations

- Extended timeouts for user interactions

5.2.8.4 Content Accessibility

- Audio narration for reading content

- Simplified language option for complex philosophical concepts

- Transcripts for any audio or video content

- Structured content with clear headings and sections

- Progress saving for users who need to take breaks

- Offline access to downloaded quest content

5.2.9 Implementation Considerations

5.2.9.1 Step Types

- Reading: Text-based content with concept links and optional media

- Multiple Choice: Questions with selectable options and feedback

- Reflection: Open-ended prompts with text input

- Matching: Pairing related concepts or ideas

- Ordering: Arranging concepts or steps in correct sequence

- Interactive Diagram: Visual exploration with interactive elements

- Discussion: Prompt for community engagement

- Challenge: Special step with increased difficulty and rewards

5.2.9.2 Offline Support

- Quest content is cached after initial download

- Progress is stored locally and synced when connection is restored

- Offline mode indicator shows sync status

- Essential functionality works without internet connection

- Background sync when connection is reestablished

5.2.9.3 Premium Content

- Premium quests are clearly marked with visual indicators

- Free users can view quest details but not start premium quests

- Upgrade prompts are non-intrusive but clear

- Some quest steps may be premium within otherwise free quests

- Premium content offers higher XP rewards and exclusive badges

5.2.9.4 Performance Optimization

- Progressive loading for large skill trees

- Asset preloading for smooth step transitions

- WebGL rendering for complex visualizations

- Efficient caching strategy for quest content

- Optimized database queries for quest filtering and recommendations

- Lazy loading for quest steps until needed

5.2.9.5 Mobile Optimization

- Responsive design adapts to different screen sizes

- Touch-friendly controls throughout the interface

- Gesture support for skill tree navigation

- Offline capability for on-the-go learning

- Battery-efficient animations and rendering

- Reduced data usage option for metered connections

5.2.10 Celery Tasks

5.2.10.1 XP and Badge Processing

@celery\_app.task(name="xp.award\_quest\_xp")

def award\_quest\_xp(user\_id, quest\_id, step\_id=None, is\_completion=False):

"""Award XP for quest activities using the standard XP system"""

db = get\_db\_connection()

quest\_service = QuestService(db)

xp\_service = XPService(db)

if is\_completion:

# Award XP for completing the entire quest

quest = quest\_service.get\_quest(quest\_id)

xp\_amount = quest["xp\_reward"]

action\_type = "quest\_complete"

elif step\_id:

# Award XP for completing a quest step

step = quest\_service.get\_step(step\_id)

xp\_amount = step["xp\_reward"]

if step.get("is\_challenge\_node"):

xp\_amount += step.get("bonus\_xp", 0)

action\_type = "quest\_step\_complete"

else:

# Award XP for starting a quest

xp\_amount = 5

action\_type = "quest\_start"

# Award XP using the standard XP service

xp\_earned = xp\_service.award\_xp(

user\_id=user\_id,

action\_type=action\_type,

context={"quest\_id": quest\_id, "step\_id": step\_id},

amount=xp\_amount

)

# Check for badge eligibility

check\_quest\_badge\_eligibility.delay(user\_id)

return xp\_earned

@celery\_app.task(name="quests.check\_badge\_eligibility")

def check\_quest\_badge\_eligibility(user\_id):

"""Check if user is eligible for new quest badges"""

db = get\_db\_connection()

quest\_service = QuestService(db)

notification\_service = NotificationService(db)

# Get user's quest statistics

stats = quest\_service.get\_user\_quest\_stats(user\_id)

# Get all available badges

badges = quest\_service.get\_quest\_badges()

# Check each badge for eligibility

newly\_awarded = []

for badge in badges:

if quest\_service.is\_eligible\_for\_badge(user\_id, badge["id"], stats):

# Award the badge if not already awarded

if quest\_service.award\_badge(user\_id, badge["id"]):

newly\_awarded.append(badge)

# Send notifications for newly awarded badges

for badge in newly\_awarded:

notification\_service.send\_notification(

user\_id=user\_id,

notification\_type="badge\_awarded",

title=f"New Badge: {badge['name']}",

body=f"You've earned the {badge['name']} badge: {badge['description']}",

data={"badge\_id": badge["id"]}

)

return [badge["id"] for badge in newly\_awarded]

5.2.10.2 Quest Unlocking

@celery\_app.task(name="quests.check\_unlockable\_quests")

def check\_unlockable\_quests(user\_id):

"""Check if user has enough XP to unlock new quests"""

db = get\_db\_connection()

quest\_service = QuestService(db)

xp\_service = XPService(db)

notification\_service = NotificationService(db)

# Get user's current XP

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

# Get quests with XP thresholds

locked\_quests = quest\_service.get\_locked\_quests\_for\_user(user\_id)

# Check for newly unlockable quests

newly\_unlocked = []

for quest in locked\_quests:

if user\_xp >= quest["xp\_threshold"]:

quest\_service.unlock\_quest\_for\_user(user\_id, quest["id"])

newly\_unlocked.append(quest)

# Send notifications for newly unlocked quests

for quest in newly\_unlocked:

notification\_service.send\_notification(

user\_id=user\_id,

notification\_type="quest\_unlocked",

title=f"New Quest Unlocked: {quest['title']}",

body=f"You've unlocked a new quest: {quest['title']}",

data={"quest\_id": quest["id"]}

)

return [quest["id"] for quest in newly\_unlocked]

5.2.10.3 Concept Integration

@celery\_app.task(name="quests.update\_concept\_mastery\_from\_quest")

def update\_concept\_mastery\_from\_quest(user\_id, quest\_id, step\_id):

"""Update concept mastery based on quest step completion"""

db = get\_db\_connection()

quest\_service = QuestService(db)

concept\_service = ConceptService(db)

# Get step details

step = quest\_service.get\_step(step\_id)

# Get linked concepts

concept\_links = step.get("concept\_links", [])

# Update mastery for each linked concept

for concept\_id in concept\_links:

# Determine mastery increase based on step type

if step.get("is\_challenge\_node"):

percentage = 15 # Higher mastery increase for challenge nodes

else:

percentage = 10 # Standard mastery increase

# Increase concept mastery

concept\_service.increase\_concept\_mastery(

user\_id=user\_id,

concept\_id=concept\_id,

percentage=percentage

)

return len(concept\_links)

@celery\_app.task(name="quests.generate\_concept\_recommendations")

def generate\_concept\_recommendations(user\_id):

"""Generate personalized concept recommendations based on quest activity"""

db = get\_db\_connection()

quest\_service = QuestService(db)

concept\_service = ConceptService(db)

# Get user's completed quests

completed\_quests = quest\_service.get\_completed\_quests(user\_id)

# Extract concepts from completed quests

explored\_concepts = set()

for quest in completed\_quests:

quest\_concepts = quest\_service.get\_quest\_concepts(quest["id"])

explored\_concepts.update([c["id"] for c in quest\_concepts])

# Find related concepts not yet explored

recommendations = concept\_service.get\_related\_unexplored\_concepts(

user\_id=user\_id,

explored\_concept\_ids=list(explored\_concepts),

limit=5

)

return [rec["id"] for rec in recommendations]

5.2.11 Implementation Plan

5.2.11.1 Phase 1: Core Infrastructure (Weeks 1-2)

- Database schema updates

- Extend quests and quest\_steps tables

- Create quest\_badges and user\_quest\_badges tables

- Add indexes for performance optimization

- Basic API endpoint updates

- Update existing quest endpoints

- Add skill tree visualization endpoint

- Implement concept exploration tracking

- XP integration

- Connect to existing XP service

- Implement XP thresholds for quest unlocking

- Create Celery tasks for XP processing

5.2.11.2 Phase 2: Frontend Components (Weeks 3-4)

- Skill tree visualization

- Develop SkillTreeView component

- Implement zoom and pan interactions

- Create node rendering and selection

- Quest step refinements

- Update QuestStep component for all step types

- Implement challenge node UI

- Create progress visualization

- XP visualization

- Integrate with standard XP components

- Implement progress indicators

- Create unlock threshold visualization

5.2.11.3 Phase 3: Concepts Integration (Weeks 5-6)

- Concept linking in quest nodes

- Implement hyperlinked concepts

- Create concept detail overlay

- Develop navigation between quests and concepts

- Mini-tree visualization

- Develop ConceptMiniTree component

- Implement hierarchical visualization

- Create interactive navigation

- Mastery progression tracking

- Connect quest completion to concept mastery

- Implement mastery visualization

- Create mastery-based recommendations

5.2.11.4 Phase 4: Accessibility and Optimization (Weeks 7-8)

- Accessibility features

- Implement high-contrast mode

- Add screen reader support

- Create voice input and navigation

- Develop audio narration

- Performance optimization

- Implement progressive loading

- Optimize rendering performance

- Create efficient caching strategy

- Testing and refinement

- Conduct accessibility audits

- Perform performance testing

- Gather user feedback

- Make final adjustments