## 5.2 Quest Feature Refinement

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

### System Architecture

#### Frontend Components:

- QuestScreen.tsx: Quest browsing and selection

- QuestDetailScreen.tsx: Individual quest view with skill tree visualization

- QuestStep.tsx: Individual step in a quest

- QuestProgress.tsx: Progress indicator

- SkillTreeView.tsx: Zoomable skill tree visualization

- QuestNodeComponent.tsx: Individual node in skill tree

- ConceptMiniTree.tsx: Mini visualization of related concepts

- AccessibilityControls.tsx: Controls for accessibility options

#### Backend Components:

- quest\_service.py: Quest management and progress tracking

- concept\_service.py: Concept integration

- xp\_service.py: XP tracking and badge management

#### Database Models:

- 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()

);

- 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)

);

- 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)

);

- 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)

);

- 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

);

- 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)

);

#### API Endpoints:

- GET /api/v1/quests: Get available quests

- Parameters:

- page: Pagination page number

- limit: Items per page

- difficulty: Filter by difficulty level

- concept\_id: Filter by related concept

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

- skill\_tree: Boolean to return skill tree view

- Response:

{

"quests": [

{

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

"title": "Introduction to Stoicism",

"description": "...",

"difficulty": "medium",

"estimated\_duration": 30,

"xp\_reward": 100,

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

"is\_premium": false,

"user\_status": "not\_started"

},

...

],

"total": 15

}

- GET /api/v1/quests/{id}: Get quest details

- Response:

{

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

"title": "Introduction to Stoicism",

"description": "...",

"difficulty": "medium",

"estimated\_duration": 30,

"xp\_reward": 100,

"concepts": [

{

"id": "123",

"name": "stoicism",

"description": "..."

},

...

],

"prerequisites": [],

"is\_premium": false,

"steps": [

{

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

"step\_number": 1,

"title": "The Stoic Worldview",

"content": "...",

"step\_type": "reading",

"xp\_reward": 20

},

...

],

"user\_progress": {

"status": "not\_started",

"current\_step": null,

"started\_at": null,

"completed\_at": null,

"xp\_earned": 0

}

}

- GET /api/v1/quests/skill-tree: Get full skill tree visualization

- Response:

{

"quests": [

{

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

"title": "Introduction to Stoicism",

"description": "...",

"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": "...",

"xp\_threshold": 500,

"user\_status": "locked",

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

"connected\_quests": []

}

],

"user\_xp": 350,

"next\_unlock": {

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

"xp\_required": 500,

"xp\_remaining": 150

}

}

- POST /api/v1/quests/{id}/start: Start a quest

- Response:

{

"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

}

- POST /api/v1/quests/{id}/steps/{step\_id}/complete: Complete a quest step

- Request:

{

"user\_response": {

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

"reflection": "..."

}

}

- Response:

{

"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": "...",

"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": 1250,

"xp\_to\_next\_level": 750

}

- GET /api/v1/quests/recommended: Get personalized quest recommendations

- Response:

{

"recommended\_quests": [

{

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

"title": "Introduction to Epicureanism",

"description": "...",

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

},

...

]

}

- POST /api/v1/quests/{id}/concepts/{concept\_id}/explore: Track concept exploration from quest

- Response:

{

"success": true,

"concept": {

"id": "123",

"name": "stoicism",

"description": "..."

},

"related\_concepts": [

{

"id": "124",

"name": "virtue",

"description": "..."

},

...

]

}

### User Experience Flow

1. User browses available quests on the QuestScreen

- Quests are presented in a visually engaging list

- Filters allow sorting by difficulty, concept, or status

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

2. User selects and views the skill tree on QuestDetailScreen

- Zoomable interface with thematic visuals (cosmic background)

- Nodes represent individual quests or quest steps

- Locked nodes show XP requirements for unlocking

- Current progress and available paths are clearly indicated

3. User starts the quest via POST /api/v1/quests/{id}/start

- System awards initial XP (+5 XP)

- First step is presented with clear instructions

4. System presents quest steps sequentially using the QuestStep component

- Each step has a specific type (reading, multiple\_choice, reflection, etc.)

- Challenge nodes offer bonus XP for additional effort

- Concept links allow exploration of related philosophical concepts

5. User completes interactive elements

- Readings with comprehension checks

- Multiple choice questions with immediate feedback

- Reflections with optional pre-filled examples

- Voice input available for accessibility

6. System tracks progress via the QuestProgress component and awards XP

- Standard XP system awards points for each completed step

- XP contributes to overall user level and unlocks advanced quests

- Progress is auto-saved to prevent data loss

7. User explores related concepts through hyperlinks

- Concept Detail page shows definition and relationships

- Mini-tree visualization shows concept hierarchy

- "Back to Quest" button ensures seamless return

8. User completes the quest and receives rewards

- XP award for full quest completion

- Badge awards for milestone achievements

- Concept mastery progress for related concepts

- Recommendations for next quests to explore

### XP Integration

- XP awarded for starting quests (+5 XP)

- XP awarded for completing quest steps (+10-30 XP per step, based on difficulty)

- XP awarded for completing full quests (+25-100 XP, based on difficulty)

- Challenge nodes provide bonus XP (+5-15 XP)

- XP thresholds unlock advanced quests (e.g., 500 XP for "Advanced Stoic Practices")

- Standard XP visualization shows progress toward unlocks

- Badges for quest completion milestones (5, 25, 50 quests)

### Concepts Integration

- Nodes link to Concepts via clickable hyperlinks

- Tooltips showing concept hierarchy (e.g., "Virtue," "Nature")

- Completing nodes adds to Concept mastery (e.g., 10% to "Stoicism")

- Mini-tree visualization for related concepts

- Seamless navigation between quests and concepts

### Accessibility Features

- High-contrast mode for skill tree visualization

- Screen reader compatibility with ARIA labels

- Voice input for reflections and quiz responses

- Audio narration for quest content

- Adjustable text sizes

- Keyboard navigation for all interactive elements

- Touch-friendly targets (minimum 44x44 pixels)

### Implementation Considerations

- Step Types: Support for various interactive elements (reading, multiple\_choice, reflection, AI interaction)

- Offline Support: Quest content will be available offline after initial download

- Premium Content: Some quests marked as premium for paid subscribers only

- Prerequisites: Some quests may require completion of other quests or specific XP thresholds

- Content Updates: Regular updates with new quests via content management system

- Performance: Progressive loading for large skill trees

- Mobile Optimization: Responsive design with gesture controls for zooming and navigation

Celery Tasks

@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

)

return xp\_earned

@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)

# 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["id"])

# Send notifications for newly unlocked quests

if newly\_unlocked:

notification\_service = NotificationService(db)

for quest\_id in newly\_unlocked:

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

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 newly\_unlocked

@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:

# Increase mastery by 10% for completing a related step

concept\_service.increase\_concept\_mastery(

user\_id=user\_id,

concept\_id=concept\_id,

percentage=10

)

return len(concept\_links)

### Implementation Plan

Phase 1: Core Infrastructure (Weeks 1-2)

- Database schema updates

- Basic API endpoint updates

- XP integration for quest progression

Phase 2: Frontend Components (Weeks 3-4)

- Skill tree visualization

- Quest step refinements

- XP visualization for quest progression

Phase 3: Concepts Integration (Weeks 5-6)

- Concept linking in quest nodes

- Mini-tree visualization

- Mastery progression tracking

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

- Accessibility features implementation

- Performance optimization for skill tree

- Voice input and navigation