# 5.7 Notification System

## 5.7.1 Notification Touchpoint Map and Integration

### Feature Goal

To provide a logical, engaging, and philosophically aligned notification system that enhances user experience, drives meaningful engagement, and supports the app's educational mission through timely, relevant, and personalized communications across all features.

### 5.7.1.1 Definitions and Glossary

#### Notification Touchpoint

A specific moment or user action within the app that triggers a notification, designed to enhance engagement, provide feedback, or guide the user's philosophical journey.

#### Notification Channel

The medium through which notifications are delivered: push notifications, in-app notifications, or email notifications.

#### Notification Priority

The urgency level of a notification (high, medium, low), which determines delivery timing, visual prominence, and channel selection.

#### Quiet Hours

User-defined time periods during which non-urgent notifications are suppressed to respect user focus and rest periods.

#### Engagement Loop

A cycle of user action, notification response, and subsequent user engagement that creates a continuous interaction pattern.

### 5.7.1.2 System Architecture

#### Frontend Components

* **NotificationScreen.tsx**: Dedicated notification inbox
  + Displays all notifications with read/unread status
  + Implements filtering by type, feature, and date
  + Supports batch actions (mark all as read, delete)
  + Renders rich notification content with interactive elements
* **NotificationSettings.tsx**: User preference management
  + Toggles for each notification channel
  + Toggles for each notification category
  + Quiet hours configuration with time pickers
  + Frequency controls for digest notifications
* **NotificationBadge.tsx**: Unread count indicator
  + Displays count of unread notifications
  + Implements subtle animations for new notifications
  + Adapts to different placement contexts (tab bar, header)
  + Supports accessibility features
* **NotificationToast.tsx**: Transient in-app alerts
  + Appears for immediate feedback notifications
  + Supports different visual styles based on notification type
  + Includes action buttons for quick responses
  + Implements smooth entrance/exit animations
* **NotificationPermissionPrompt.tsx**: Permission request UI
  + Explains benefits of enabling notifications
  + Provides clear opt-in/opt-out controls
  + Implements progressive permission strategy
  + Respects previous user choices

#### Backend Components

* **notification\_service.py**: Core notification service
  + Manages notification creation and delivery
  + Implements channel selection logic
  + Handles user preference application
  + Processes notification templates
* **notification\_tasks.py**: Asynchronous notification tasks
  + Handles push notification delivery via Firebase
  + Processes email notifications via SendGrid
  + Manages notification batching and scheduling
  + Implements retry logic for failed deliveries
* **notification\_template\_service.py**: Template management
  + Stores and retrieves notification templates
  + Handles template variable substitution
  + Manages localization of notification content
  + Implements template versioning
* **user\_preference\_service.py**: Preference management
  + Stores and retrieves user notification preferences
  + Handles default preference initialization
  + Manages quiet hours enforcement
  + Processes opt-out requests

### 5.7.1.3 Database Models

-- Stores all notifications for users

CREATE TABLE notifications (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

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

type VARCHAR(50) NOT NULL,

title VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'unread',

priority VARCHAR(10) NOT NULL DEFAULT 'medium',

action\_url VARCHAR(255),

action\_text VARCHAR(100),

image\_url VARCHAR(255),

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

read\_at TIMESTAMP WITH TIME ZONE,

expires\_at TIMESTAMP WITH TIME ZONE,

metadata JSONB DEFAULT '{}'::jsonb,

CONSTRAINT valid\_status CHECK (status IN ('unread', 'read', 'archived')),

CONSTRAINT valid\_priority CHECK (priority IN ('high', 'medium', 'low')),

CONSTRAINT valid\_type CHECK (type IN (

'xp\_award', 'level\_up', 'badge\_earned',

'quest\_reminder', 'quest\_completed', 'quest\_new',

'ask\_insight', 'ask\_continuation',

'journal\_milestone', 'journal\_prompt',

'concept\_discovery', 'concept\_mastery',

'forum\_reply', 'forum\_mention', 'forum\_trending',

'streak\_reminder', 'streak\_milestone',

'system\_announcement', 'feature\_update'

))

);

-- Indexes for performance

CREATE INDEX idx\_notifications\_user ON notifications(user\_id);

CREATE INDEX idx\_notifications\_status ON notifications(status);

CREATE INDEX idx\_notifications\_created ON notifications(created\_at);

CREATE INDEX idx\_notifications\_type ON notifications(type);

-- Stores user notification preferences

CREATE TABLE notification\_preferences (

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

push\_enabled BOOLEAN DEFAULT TRUE,

email\_enabled BOOLEAN DEFAULT TRUE,

ask\_notifications BOOLEAN DEFAULT TRUE,

quest\_notifications BOOLEAN DEFAULT TRUE,

forum\_notifications BOOLEAN DEFAULT TRUE,

journal\_notifications BOOLEAN DEFAULT TRUE,

xp\_notifications BOOLEAN DEFAULT TRUE,

concept\_notifications BOOLEAN DEFAULT TRUE,

streak\_notifications BOOLEAN DEFAULT TRUE,

system\_notifications BOOLEAN DEFAULT TRUE,

quiet\_hours\_start TIME DEFAULT '22:00',

quiet\_hours\_end TIME DEFAULT '07:00',

quiet\_hours\_enabled BOOLEAN DEFAULT TRUE,

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

updated\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()

);

-- Stores notification delivery attempts and status

CREATE TABLE notification\_deliveries (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

notification\_id UUID NOT NULL REFERENCES notifications(id) ON DELETE CASCADE,

channel VARCHAR(20) NOT NULL,

status VARCHAR(20) NOT NULL DEFAULT 'pending',

attempt\_count INTEGER DEFAULT 0,

last\_attempt\_at TIMESTAMP WITH TIME ZONE,

delivered\_at TIMESTAMP WITH TIME ZONE,

error\_message TEXT,

device\_info JSONB,

CONSTRAINT valid\_channel CHECK (channel IN ('push', 'in\_app', 'email')),

CONSTRAINT valid\_delivery\_status CHECK (status IN ('pending', 'delivered', 'failed', 'cancelled'))

);

-- Indexes for performance

CREATE INDEX idx\_notification\_deliveries\_notification ON notification\_deliveries(notification\_id);

CREATE INDEX idx\_notification\_deliveries\_status ON notification\_deliveries(status);

CREATE INDEX idx\_notification\_deliveries\_channel ON notification\_deliveries(channel);

-- Stores device tokens for push notifications

CREATE TABLE device\_tokens (

id UUID PRIMARY KEY DEFAULT gen\_random\_uuid(),

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

token TEXT NOT NULL,

device\_type VARCHAR(20) NOT NULL,

app\_version VARCHAR(20),

os\_version VARCHAR(20),

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

last\_used\_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

is\_active BOOLEAN DEFAULT TRUE,

CONSTRAINT valid\_device\_type CHECK (device\_type IN ('ios', 'android', 'web'))

);

-- Unique constraint to prevent duplicate tokens

CREATE UNIQUE INDEX idx\_device\_tokens\_unique ON device\_tokens(user\_id, token) WHERE is\_active = TRUE;

### 5.7.1.4 Comprehensive Notification Touchpoint Map

#### 5.7.1.4.1 Ask Feature Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| ASK-01 | First question asked | xp\_award | medium | in-app | "Philosophical journey begun! You earned {xp} XP for your first question." | View XP progress |
| ASK-02 | New philosophical tone unlocked | feature\_update | medium | in-app, push | "New perspective unlocked: {tone\_name}! Explore a new philosophical voice." | Try new tone |
| ASK-03 | 24 hours since saving insight without revisiting | ask\_insight | low | push | "Your insight on '{question\_snippet}' awaits deeper reflection." | Revisit insight |
| ASK-04 | Concept badge earned through questions | badge\_earned | medium | in-app | "You've earned the {concept\_name} Concept Badge through thoughtful questioning!" | Explore concept |
| ASK-05 | AI suggests follow-up question | ask\_continuation | low | in-app | "Philosophical thread detected: Continue your exploration of {concept\_name}?" | Ask follow-up |
| ASK-06 | Question asked about advanced concept | concept\_discovery | medium | in-app | "You're exploring advanced territory with {concept\_name}. Dive deeper?" | Explore concept |
| ASK-07 | 5 questions asked on related topics | streak\_milestone | medium | in-app | "Your philosophical focus is deepening! 5 questions on {concept\_name}." | Continue streak |

#### 5.7.1.4.2 Quest Feature Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| QUEST-01 | New quest available | quest\_new | medium | in-app, push | "New philosophical journey awaits: '{quest\_name}' has been unlocked!" | Start quest |
| QUEST-02 | Quest step completed | xp\_award | medium | in-app | "Step completed! You earned {xp} XP on your '{quest\_name}' journey." | Continue quest |
| QUEST-03 | Quest completed | quest\_completed | high | in-app, push | "Philosophical milestone achieved! You've completed '{quest\_name}'." | View rewards |
| QUEST-04 | 48 hours of quest inactivity | quest\_reminder | low | push | "Your journey through '{quest\_name}' awaits your return." | Resume quest |
| QUEST-05 | Challenge node unlocked | quest\_new | medium | in-app | "A philosophical challenge appears in '{quest\_name}'! Test your understanding." | Attempt challenge |
| QUEST-06 | Quest prerequisite nearly met | quest\_reminder | low | in-app | "You're almost ready to begin '{quest\_name}'. Complete {prerequisite\_name} first." | Complete prerequisite |
| QUEST-07 | Weekly quest digest | quest\_reminder | low | email | "Your philosophical journeys this week: {active\_quests\_summary}" | Review quests |

#### 5.7.1.4.3 Explore Feature Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| EXPLORE-01 | New concept discovered | concept\_discovery | medium | in-app | "New philosophical territory: You've discovered {concept\_name}!" | Explore concept |
| EXPLORE-02 | Concept mastery level increased | concept\_mastery | medium | in-app | "Your understanding of {concept\_name} has deepened to level {level}!" | View mastery |
| EXPLORE-03 | Related concept suggestion | concept\_discovery | low | in-app | "If {concept\_name} interests you, you might explore {related\_concept}." | Explore related |
| EXPLORE-04 | Concept constellation completed | badge\_earned | medium | in-app, push | "Constellation mastery! You've explored all concepts in {constellation\_name}." | View badge |
| EXPLORE-05 | New pathway unlocked | feature\_update | medium | in-app | "New learning pathway unlocked: '{pathway\_name}' is now available." | Start pathway |
| EXPLORE-06 | 10 concepts explored | streak\_milestone | medium | in-app | "Your philosophical map expands! You've explored 10 concepts." | Continue exploring |
| EXPLORE-07 | Monthly concept digest | concept\_discovery | low | email | "Expand your philosophical horizons with these concepts: {recommended\_concepts}" | Explore concepts |

#### 5.7.1.4.4 Journal Feature Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| JOURNAL-01 | First journal entry created | xp\_award | medium | in-app | "Reflection begun! You earned {xp} XP for your first journal entry." | Continue journaling |
| JOURNAL-02 | Journal streak milestone | streak\_milestone | medium | in-app, push | "Consistent reflection! You've journaled for {days} consecutive days." | Continue streak |
| JOURNAL-03 | Weekly journal prompt | journal\_prompt | low | push, email | "Philosophical prompt of the week: '{prompt\_text}'" | Write entry |
| JOURNAL-04 | Journal entry with multiple concept tags | journal\_milestone | medium | in-app | "Your entry connects {count} philosophical concepts! Interdisciplinary thinking unlocked." | View connections |
| JOURNAL-05 | 10th journal entry milestone | journal\_milestone | medium | in-app | "Reflection milestone! You've created 10 philosophical journal entries." | Review journey |
| JOURNAL-06 | 7 days without journaling | journal\_prompt | low | push | "Your philosophical journal awaits new reflections." | Write entry |
| JOURNAL-07 | Seasonal reflection prompt | journal\_prompt | low | email | "Seasonal philosophical reflection: How has your thinking on {concept\_name} evolved?" | Write reflection |

#### 5.7.1.4.5 Forum Feature Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| FORUM-01 | Reply to user's post | forum\_reply | high | in-app, push | "{username} replied to your post: '{reply\_snippet}'" | View reply |
| FORUM-02 | User mentioned in post | forum\_mention | high | in-app, push | "{username} mentioned you in '{thread\_title}'" | View mention |
| FORUM-03 | User's thread reaches engagement milestone | forum\_trending | medium | in-app | "Your discussion on '{thread\_title}' is gaining traction!" | View thread |
| FORUM-04 | New thread in followed category | forum\_trending | low | in-app | "New discussion in {category\_name}: '{thread\_title}'" | View thread |
| FORUM-05 | First forum contribution | xp\_award | medium | in-app | "Community philosopher! You earned {xp} XP for your first forum contribution." | Continue engaging |
| FORUM-06 | Weekly forum digest | forum\_trending | low | email | "Philosophical discussions you might enjoy this week: {thread\_recommendations}" | Explore threads |
| FORUM-07 | Thread related to user's recent concepts | forum\_trending | medium | in-app | "Discussion related to {concept\_name}: '{thread\_title}'" | Join discussion |

#### 5.7.1.4.6 Wisdom XP System Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| XP-01 | Level up | level\_up | high | in-app, push | "Philosophical growth! You've reached Wisdom Level {level}." | View rewards |
| XP-02 | Badge earned | badge\_earned | medium | in-app | "Achievement unlocked: {badge\_name}! {badge\_description}" | View badge |
| XP-03 | Daily streak continued | streak\_milestone | medium | in-app | "Philosophical consistency! Day {count} of your wisdom streak." | Continue streak |
| XP-04 | Streak at risk (no activity for 20+ hours) | streak\_reminder | medium | push | "Your {count}-day philosophical streak ends in {hours} hours. Don't lose momentum!" | Take action |
| XP-05 | Weekly XP summary | xp\_award | low | in-app, email | "Your philosophical growth this week: {xp} XP earned across {activities} activities." | Review progress |
| XP-06 | Approaching next level | xp\_award | medium | in-app | "Almost there! Just {xp\_needed} XP until Wisdom Level {next\_level}." | Take action |
| XP-07 | Milestone XP earned (1000, 5000, etc.) | xp\_award | medium | in-app | "Philosophical milestone! You've earned {total\_xp} total XP on your journey." | Review journey |

#### 5.7.1.4.7 Profile and System Touchpoints

| **Touchpoint ID** | **Trigger Event** | **Notification Type** | **Priority** | **Channels** | **Content Template** | **User Action** |
| --- | --- | --- | --- | --- | --- | --- |
| SYS-01 | New app feature released | system\_announcement | medium | in-app, push, email | "New feature: {feature\_name} is now available! {feature\_description}" | Explore feature |
| SYS-02 | Subscription renewal upcoming | system\_announcement | high | in-app, push, email | "Your {tier\_name} subscription renews in {days} days." | Manage subscription |
| SYS-03 | Account milestone (30 days, etc.) | system\_announcement | low | in-app | "Philosophical journey milestone! You've been with us for {days} days." | View journey |
| SYS-04 | Personalized feature suggestion | feature\_update | low | in-app | "Based on your interests in {concept\_name}, you might enjoy {feature\_name}." | Try feature |
| SYS-05 | App update available | system\_announcement | medium | in-app | "Update available: Version {version} with {feature\_highlights}." | Update app |
| SYS-06 | First week summary for new users | system\_announcement | medium | email | "Your first week of philosophical exploration: {activity\_summary}" | Continue journey |
| SYS-07 | Monthly engagement summary | system\_announcement | low | email | "Your philosophical journey this month: {monthly\_summary}" | Review activity |

### 5.7.1.5 Notification Workflows

#### 5.7.1.5.1 Standard Notification Flow

1. **Event Trigger**
   * System detects notification-worthy event
   * Event details captured with context data
   * Notification record created in database
2. **Eligibility Check**
   * User preferences checked for opt-out
   * Quiet hours status evaluated
   * Frequency caps applied to prevent notification fatigue
3. **Channel Selection**
   * Priority level determines eligible channels
   * User preferences filter available channels
   * Context (app state, time of day) influences selection
4. **Content Generation**
   * Template selected based on notification type
   * Dynamic content inserted from context data
   * Personalization applied based on user history
   * Localization applied if multiple languages supported
5. **Delivery Execution**
   * In-app notifications delivered immediately
   * Push notifications sent via Firebase Cloud Messaging
   * Email notifications sent via SendGrid
   * Delivery attempts recorded for analytics
6. **User Interaction**
   * User receives and views notification
   * Notification marked as read when viewed
   * Click-through actions tracked for engagement metrics
   * Dismissal or action completion recorded

#### 5.7.1.5.2 Batch Processing Flow

1. **Aggregation Phase**
   * Low-priority notifications collected throughout day
   * Similar notifications grouped by type and context
   * User-specific digests prepared based on activity
2. **Optimal Timing Calculation**
   * Historical engagement patterns analyzed
   * Time zone and quiet hours considered
   * App usage patterns factored into timing
3. **Digest Compilation**
   * Related notifications summarized into digest format
   * Personalized recommendations added
   * Clear call-to-action for each notification group
4. **Scheduled Delivery**
   * Digests delivered during peak engagement windows
   * Delivery spread to avoid system load spikes
   * Immediate delivery canceled if user opens app before scheduled time

### 5.7.1.6 Notification Design Principles

#### 5.7.1.6.1 Philosophical Alignment

All notifications should reflect the app's philosophical nature through:

* **Thoughtful Language**: Using contemplative, nuanced phrasing rather than urgent or marketing-focused language
* **Growth Mindset**: Emphasizing learning and development over competition or consumption
* **Meaningful Engagement**: Encouraging depth of interaction rather than frequency
* **Respectful Tone**: Addressing the user as a thoughtful individual on a philosophical journey

#### 5.7.1.6.2 Progressive Disclosure

Notifications should implement progressive disclosure by:

* **Starting Simple**: New users receive only essential notifications
* **Gradual Introduction**: New notification types introduced as users engage with related features
* **Complexity Matching**: Notification sophistication increases with user progression
* **Context Sensitivity**: Providing more detailed information for advanced users

#### 5.7.1.6.3 Engagement Loops

Notifications should create virtuous engagement loops by:

* **Meaningful Triggers**: Sending notifications only for genuinely valuable interactions
* **Clear Next Steps**: Always providing a logical next action
* **Value Delivery**: Ensuring each notification delivers immediate value
* **Journey Continuation**: Connecting current achievements to future possibilities

### 5.7.1.7 Implementation Considerations

#### 5.7.1.7.1 Technical Implementation

# src/services/notification\_service.py

class NotificationService:

def \_\_init\_\_(self, db, user\_service, preference\_service):

self.db = db

self.user\_service = user\_service

self.preference\_service = preference\_service

self.template\_service = NotificationTemplateService()

async def create\_notification(self, user\_id, notification\_type, context\_data, priority="medium"):

"""Create a new notification record."""

# Check user eligibility

if not await self.\_is\_eligible(user\_id, notification\_type, priority):

return None

# Generate notification content

template = self.template\_service.get\_template(notification\_type)

title = template.render\_title(context\_data)

content = template.render\_content(context\_data)

# Create notification record

notification\_id = await self.db.notifications.insert\_one({

"user\_id": user\_id,

"type": notification\_type,

"title": title,

"content": content,

"priority": priority,

"status": "unread",

"created\_at": datetime.now(),

"metadata": context\_data

})

# Determine delivery channels

channels = await self.\_select\_channels(user\_id, notification\_type, priority)

# Schedule deliveries

for channel in channels:

await self.\_schedule\_delivery(notification\_id, channel)

return notification\_id

async def \_is\_eligible(self, user\_id, notification\_type, priority):

"""Check if user is eligible to receive this notification."""

# Get user preferences

prefs = await self.preference\_service.get\_preferences(user\_id)

# Check if notification type is enabled

type\_enabled = getattr(prefs, f"{notification\_type.split('\_')[0]}\_notifications", True)

if not type\_enabled:

return False

# Check quiet hours for non-high priority

if priority != "high" and prefs.quiet\_hours\_enabled:

current\_time = datetime.now().time()

if self.\_is\_in\_quiet\_hours(current\_time, prefs.quiet\_hours\_start, prefs.quiet\_hours\_end):

return False

# Check frequency caps

if await self.\_is\_frequency\_capped(user\_id, notification\_type):

return False

return True

async def \_select\_channels(self, user\_id, notification\_type, priority):

"""Select appropriate delivery channels based on type and preferences."""

channels = []

prefs = await self.preference\_service.get\_preferences(user\_id)

# In-app notifications for all except lowest priority

if priority != "low":

channels.append("in\_app")

# Push notifications based on preference and priority

if prefs.push\_enabled and (priority == "high" or

(priority == "medium" and notification\_type not in ["xp\_award", "concept\_discovery"])):

channels.append("push")

# Email for digests and high-priority system announcements

if prefs.email\_enabled and (notification\_type in ["system\_announcement", "quest\_reminder", "journal\_prompt"] or

"\_digest" in notification\_type):

channels.append("email")

return channels

async def \_schedule\_delivery(self, notification\_id, channel):

"""Schedule notification delivery through specified channel."""

if channel == "in\_app":

# In-app notifications are delivered immediately by being in the database

return

# Create delivery record

delivery\_id = await self.db.notification\_deliveries.insert\_one({

"notification\_id": notification\_id,

"channel": channel,

"status": "pending",

"attempt\_count": 0

})

# Schedule task based on channel

if channel == "push":

send\_push\_notification.delay(delivery\_id)

elif channel == "email":

send\_email\_notification.delay(delivery\_id)

# src/tasks/notification\_tasks.py

@celery\_app.task

def send\_push\_notification(delivery\_id):

"""Send a push notification via Firebase."""

try:

# Get delivery record

delivery = db.notification\_deliveries.find\_one({"\_id": delivery\_id})

if not delivery or delivery["status"] != "pending":

return

# Get notification

notification = db.notifications.find\_one({"\_id": delivery["notification\_id"]})

if not notification:

return

# Get user's device tokens

tokens = db.device\_tokens.find({"user\_id": notification["user\_id"], "is\_active": True})

if not tokens:

mark\_delivery\_failed(delivery\_id, "No active device tokens")

return

# Prepare message

message = {

"notification": {

"title": notification["title"],

"body": notification["content"]

},

"data": {

"notification\_id": str(notification["\_id"]),

"type": notification["type"],

"action\_url": notification.get("action\_url", "")

}

}

# Send to each device

success = False

for token\_doc in tokens:

try:

firebase\_admin.messaging.send(

message,

token=token\_doc["token"]

)

success = True

except firebase\_admin.messaging.exceptions.FirebaseError as e:

# If token is invalid, mark it as inactive

if "invalid-registration-token" in str(e):

db.device\_tokens.update\_one(

{"\_id": token\_doc["\_id"]},

{"$set": {"is\_active": False}}

)

continue

# Update delivery status

if success:

db.notification\_deliveries.update\_one(

{"\_id": delivery\_id},

{

"$set": {

"status": "delivered",

"delivered\_at": datetime.now()

},

"$inc": {"attempt\_count": 1}

}

)

else:

mark\_delivery\_failed(delivery\_id, "Failed to deliver to any device")

except Exception as e:

mark\_delivery\_failed(delivery\_id, str(e))

# Retry for transient errors

if "retry" in str(e).lower() or isinstance(e, (ConnectionError, Timeout)):

raise self.retry(exc=e, countdown=60, max\_retries=3)

#### 5.7.1.7.2 Performance Considerations

* **Batch Processing**: Group low-priority notifications for efficient delivery
* **Caching Strategy**: Cache notification templates and user preferences in Redis
* **Asynchronous Processing**: Use Celery tasks for all external notification delivery
* **Rate Limiting**: Implement per-user rate limits to prevent notification fatigue
* **Delivery Windowing**: Schedule notifications during optimal engagement periods
* **Payload Optimization**: Keep push notification payloads under 4KB
* **Database Indexing**: Ensure proper indexes on notification queries
* **Archiving Strategy**: Move read notifications to archive table after 30 days

#### 5.7.1.7.3 Testing Strategy

* **A/B Testing**: Test different notification phrasings for engagement
* **User Cohort Analysis**: Compare engagement between different notification strategies
* **Notification Fatigue Monitoring**: Track uninstall rates after notification spikes
* **Quiet Period Testing**: Measure re-engagement after deliberate notification pauses
* **Channel Effectiveness**: Compare open rates across different delivery channels
* **Content Testing**: Test philosophical vs. direct language in notifications
* **Timing Optimization**: Test different delivery times for optimal engagement

### 5.7.1.8 Cross-Feature Integration

#### 5.7.1.8.1 Ask Feature Integration

* **Notification-to-Question Flow**: Notifications about concepts include "Ask about this" action
* **Response Quality Feedback**: Notification after rating AI response with "Tell us more" option
* **Philosophical Tone Suggestions**: Notifications suggesting new tones based on question history
* **Conversation Continuation**: Gentle reminders for abandoned philosophical threads

#### 5.7.1.8.2 Quest Feature Integration

* **Quest Progression Visualization**: Notifications include mini skill tree visualization
* **Contextual Reminders**: Notifications reference user's current position in quest journey
* **Challenge Unlocks**: Special notification styling for challenge node availability
* **Quest Recommendations**: Notifications suggest quests related to recent user activity

#### 5.7.1.8.3 Explore Feature Integration

* **Concept Relationship Notifications**: Highlight newly discovered connections between concepts
* **Visualization Previews**: Include mini concept map visualizations in notifications
* **Mastery Progress**: Show mastery progress bars in level-up notifications
* **Exploration Path Suggestions**: Recommend next concepts based on exploration history

#### 5.7.1.8.4 Journal Feature Integration

* **Reflection Prompts**: Notifications include philosophical prompts based on recent activity
* **Milestone Visualizations**: Include journal streak calendars in milestone notifications
* **Entry Suggestions**: Recommend journal topics based on recent concept exploration
* **Memory Surfacing**: Periodically resurface past journal entries for reflection

#### 5.7.1.8.5 Forum Feature Integration

* **Discussion Discovery**: Notifications highlight discussions related to user's interests
* **Contribution Recognition**: Special notifications for valuable community contributions
* **Philosophical Discourse**: Notifications frame discussions in philosophical context
* **Community Milestones**: Celebrate collective learning achievements

### 5.7.1.9 Analytics Integration

* **Notification Effectiveness Metrics**:
  + Open rate by notification type, channel, and time of day
  + Action completion rate by notification type
  + Time-to-open by priority and channel
  + Opt-out rate by notification type
* **Engagement Impact Analysis**:
  + Session initiation from notifications
  + Feature usage following notifications
  + Retention correlation with notification engagement
  + Learning outcomes correlated with notification patterns
* **User Preference Insights**:
  + Channel preference patterns by user segment
  + Notification type preferences by philosophical interests
  + Quiet hours patterns across user base
  + Notification tolerance thresholds

### 5.7.1.10 Implementation Phases

#### Phase 1: Core Notification System (Weeks 1-2)

* Implement basic notification database models
* Set up Firebase Cloud Messaging integration
* Create essential notification components
* Implement core notification service

#### Phase 2: Feature-Specific Notifications (Weeks 3-4)

* Implement Ask feature notifications
* Implement Quest feature notifications
* Implement XP system notifications
* Create notification preference management

#### Phase 3: Advanced Notifications (Weeks 5-6)

* Implement batch processing and digests
* Create email notification templates
* Implement quiet hours and frequency capping
* Add rich notification content with images

#### Phase 4: Optimization & Analytics (Weeks 7-8)

* Implement notification analytics
* Create A/B testing framework
* Optimize delivery timing algorithms
* Fine-tune notification content and frequency