STEP 22: Notifications & Reminders Engine - Implementation Documentation

Implementation Date: November 30, 2025
Status: ✅ Complete
Author: Fyndr Development Team

Table of Contents

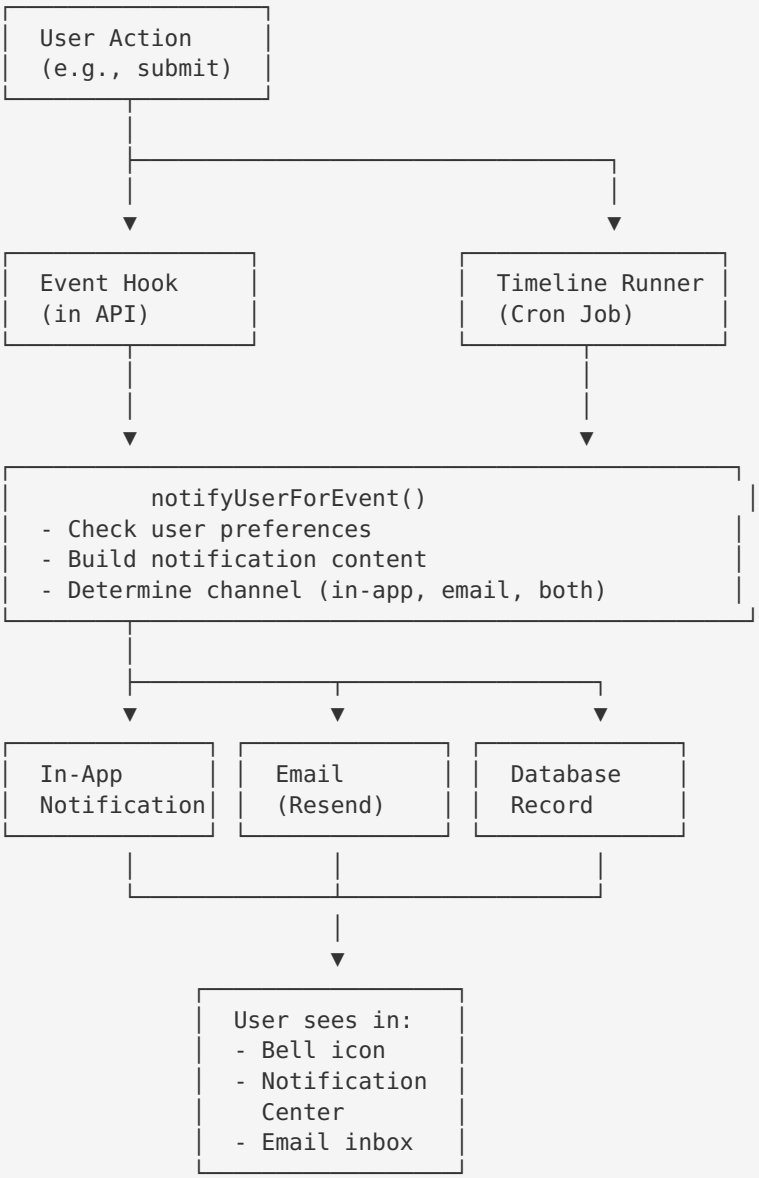
Overview

The Notifications & Reminders Engine is a comprehensive notification system that provides:
- **Event-based notifications** triggered by user actions
- **Timeline-based reminders** for RFP milestones and deadlines
- **Customizable preferences** for both buyers and suppliers
- **Multi-channel delivery** (in-app and email)
- **Real-time unread counts** via bell icon indicators

Key Features:
- ✅ Buyer and supplier notification centers
- ✅ Granular notification preferences
- ✅ Timeline reminder runner for scheduled notifications
- ✅ Event-based hooks integrated into existing APIs
- ✅ Bell icons with unread counts in headers
- ✅ Email integration via Resend
- ✅ Role-based access control

Architecture

High-Level Flow



Files Created/Modified

🆕 New Files Created

Database Schema

- `prisma/schema.prisma` - Updated with `Notification` and `NotificationPreference` models

Core Library Files

- `lib/notification-types.ts` - Notification type constants and TypeScript types
- `lib/notifications.ts` - Core notification functions and timeline runner

API Endpoints

- `app/api/notifications/run/route.ts` - Notification runner endpoint
- `app/api/notifications/unread-count/route.ts` - Unread count API
- `app/api/notifications/[id]/read/route.ts` - Mark as read API
- `app/api/settings/notifications/route.ts` - Preferences management API

Buyer UI

- `app/dashboard/notifications/page.tsx` - Buyer notification center (server)
- `app/dashboard/notifications/notification-center.tsx` - Notification list (client)
- `app/dashboard/settings/notifications/page.tsx` - Buyer preferences page
- `app/dashboard/bell-icon.tsx` - Buyer bell icon component

Supplier UI

- `app/supplier/notifications/page.tsx` - Supplier notification center (server)
- `app/supplier/notifications/supplier-notification-center.tsx` - Notification list (client)
- `app/supplier/settings/notifications/page.tsx` - Supplier preferences page
- `app/supplier/bell-icon.tsx` - Supplier bell icon component

Documentation

- `STEP_22_TESTING_GUIDE.md` - Comprehensive testing guide
- `STEP_22_IMPLEMENTATION_DOCS.md` - This file

📝 Modified Files

API Event Hooks

- `app/api/supplier/rfps/[rfpId]/response/submit/route.ts` - Added SUPPLIER_RESPONSE_SUBMITTED hook
- `app/api/supplier/rfps/[rfpId]/questions/route.ts` - Added SUPPLIER_QUESTION_CREATED hook
- `app/api/dashboard/rfps/[rfpId]/questions/route.ts` - Added SUPPLIER_QUESTION_ANSWERED and SUPPLIER_BROADCAST_CREATED hooks
- `app/api/dashboard/rfps/[rfpId]/broadcasts/route.ts` - Added SUPPLIER_BROADCAST_CREATED hook
- `app/api/dashboard/rfps/[id]/comparison/readiness/route.ts` - Added READINESS_INDICATOR_UPDATED hook (optional)
- `app/api/rfps/[id]/compare/report/route.ts` - Added COMPARISON_REPORT_READY hook (optional)

Layout Files

- `app/dashboard/dashboard-layout.tsx` - Added `BellIcon` component to header
- `app/supplier/supplier-layout.tsx` - Added `SupplierBellIcon` component to header

Database Schema

Notification Model

```
model Notification {
  id                String    @id @default(cuid())
  userId            String
  rfpId             String?
  supplierResponseId String?
  supplierContactId  String?

  type              String     // Notification type constant
  category          String     // RFP_TIMELINE, SUPPLIER_QA, etc.
  title             String     // Display title
  message           String     // Display message
  channel           String     // IN_APP, EMAIL, or IN_APP_EMAIL
  metadata          Json?      // Additional context data

  readAt            DateTime?  // Null if unread
  createdAt         DateTime   @default(now())

  user              User       @relation(fields: [userId], references: [id],
onDelete: Cascade)

  @@index([userId, createdAt])
  @@index([rfpId])
}
```

NotificationPreference Model

```
model NotificationPreference {
  id                        String   @id @default(cuid())
  userId                    String   @unique
  emailEnabled              Boolean  @default(true)
  inAppEnabled              Boolean  @default(true)

  // Buyer-specific toggles
  buyerRfpTimeline          Boolean  @default(true)
  buyerSupplierResponses    Boolean  @default(true)
  buyerSupplierQuestions    Boolean  @default(true)
  buyerQABroadcasts         Boolean  @default(true)
  buyerReadinessChanges     Boolean  @default(true)

  // Supplier-specific toggles
  supplierQATimeline        Boolean  @default(true)
  supplierSubmissionTimeline Boolean  @default(true)
  supplierBroadcasts        Boolean  @default(true)
  supplierResponseStatus    Boolean  @default(true)

  createdAt                 DateTime @default(now())
  updatedAt                 DateTime @updatedAt

  user                      User     @relation(fields: [userId], references: [id], on
Delete: Cascade)
}
```

API Endpoints

1. Notification Runner

Endpoint: `POST /api/notifications/run`
Auth: Required (any authenticated user)
Purpose: Run timeline-based reminders for all active RFPs

Request:

```
POST /api/notifications/run
```

Response:

```
{
  "success": true,
  "processedRfps": 12,
  "notificationsCreated": 8,
  "timestamp": "2025-11-30T12:00:00Z",
  "message": "Processed 12 RFPs and created 8 notifications"
}
```

Usage:
- Should be called by external cron job (hourly or daily)
- Checks all non-ARCHIVED RFPs for timeline events
- Creates notifications for matching events
- Prevents duplicates using `checkIfNotificationSentToday()`

---

2. Unread Count

Endpoint: `GET /api/notifications/unread-count`
Auth: Required
Purpose: Get unread notification count for current user

Request:

```
GET /api/notifications/unread-count
```

Response:

```
{
  "count": 5
}
```

Usage:
- Called by bell icon every 30 seconds
- Returns count where `readAt` is null
- User-scoped (only shows current user’s count)

---

3. Mark as Read

Endpoint: `PATCH /api/notifications/[id]/read`
Auth: Required
Purpose: Mark a specific notification as read

Request:

```
PATCH /api/notifications/abc123/read
```

Response:

```
{
  "success": true,
  "notification": {
    "id": "abc123",
    "readAt": "2025-11-30T12:05:00Z",
    ...
  }
}
```

Security:
- Verifies notification belongs to current user
- Returns 403 if ownership check fails

---

4. Notification Preferences

Endpoints:
- `GET /api/settings/notifications` - Get preferences
- `POST /api/settings/notifications` - Update preferences

Auth: Required

GET Response:

```
{
  "id": "pref123",
  "userId": "user123",
  "emailEnabled": true,
  "inAppEnabled": true,
  "buyerRfpTimeline": true,
  "buyerSupplierResponses": true,
  ...
}
```

POST Request:

```
{
  "emailEnabled": false,
  "buyerRfpTimeline": false
}
```

POST Response:

```
{
  "success": true,
  "preferences": {
    "emailEnabled": false,
    "buyerRfpTimeline": false,
    ...
  }
}
```

Features:
- Auto-creates preferences on first GET
- Validates field names
- Only updates provided fields (partial update)

Notification Types

Event-Based Types

Type	Category	Triggered By	Recipients
`RFP_TIMELINE_QA_WINDOW_OPEN`	RFP_TIMELINE	Timeline runner	All suppliers
`RFP_TIMELINE_QA_WINDOW_CLOSING_SOON`	RFP_TIMELINE	Timeline runner	All suppliers
`RFP_TIMELINE_SUBMISSION_WINDOW_OPEN`	RFP_TIMELINE	Timeline runner	All suppliers
`RFP_TIMELINE_SUBMISSION_DEADLINE_SOON`	RFP_TIMELINE	Timeline runner	All suppliers
`RFP_TIMELINE_SUBMISSION_DEADLINE_PASSED`	RFP_TIMELINE	Timeline runner	Buyer
`RFP_TIMELINE_DEMO_WINDOW_OPEN`	RFP_TIMELINE	Timeline runner	Buyer
`RFP_TIMELINE_AWARD_DATE_SOON`	RFP_TIMELINE	Timeline runner	Buyer
`RFP_TIMELINE_AWARD_DATE_PASSED`	RFP_TIMELINE	Timeline runner	Buyer
`SUPPLIER_QUESTION_CREATED`	SUPPLIER_QA	Supplier submits question	Buyer
`SUPPLIER_QUESTION_ANSWERED`	SUPPLIER_QA	Buyer answers question	Specific supplier
`SUPPLIER_BROADCAST_CREATED`	SUPPLIER_QA	Buyer creates broadcast	All suppliers
`SUPPLIER_RESPONSE_SUBMITTED`	SUPPLIER_RESPONSE	Supplier submits response	Buyer
`READINESS_INDICATOR_UPDATED`	READINESS	Buyer runs readiness calc	Buyer
`COMPARISON_REPORT_READY`	SYSTEM	Report generation complete	Buyer

Event Hooks

Hook 1: Supplier Response Submission

File: `app/api/supplier/rfps/[rfpId]/response/submit/route.ts`

Code Location: After `prisma.supplierResponse.update()`

Notification:

```
await notifyUserForEvent(SUPPLIER_RESPONSE_SUBMITTED, rfp.user, {
  rfpId: rfp.id,
  rfpTitle: rfp.title,
  supplierName: supplierContact.name,
  supplierContactId: supplierContact.id,
  supplierResponseId: submittedResponse.id,
});
```

When: Supplier finalizes their response

Hook 2: Supplier Question Created

File: `app/api/supplier/rfps/[rfpId]/questions/route.ts`

Code Location: After `prisma.supplierQuestion.create()`

Notification:

```
await notifyUserForEvent(SUPPLIER_QUESTION_CREATED, rfp.user, {
  rfpId: rfp.id,
  rfpTitle: rfp.title,
  questionId: newQuestion.id,
});
```

When: Supplier submits a question during Q&A window

Hook 3: Supplier Question Answered

File: `app/api/dashboard/rfps/[rfpId]/questions/route.ts`

Code Location: After `prisma.supplierQuestion.update()`

Notification (Non-Broadcast):

```
await notifyUserForEvent(SUPPLIER_QUESTION_ANSWERED, supplierContact.portalUser, {
  rfpId: rfp.id,
  rfpTitle: rfp.title,
  questionId: updatedQuestion.id,
});
```

When: Buyer answers a question without broadcasting

Hook 4: Broadcast Created

Files:
- `app/api/dashboard/rfps/[rfpId]/questions/route.ts` (when broadcast=true)
- `app/api/dashboard/rfps/[rfpId]/broadcasts/route.ts`

Code Location: After creating broadcast message

Notification (All Suppliers):

```
const allSupplierContacts = await prisma.supplierContact.findMany({
  where: { rfpId, portalUserId: { not: null } },
  include: { portalUser: true }
});

for (const contact of allSupplierContacts) {
  await notifyUserForEvent(SUPPLIER_BROADCAST_CREATED, contact.portalUser, {
    rfpId: rfp.id,
    rfpTitle: rfp.title,
    broadcastId: broadcast.id,
  });
}
```

When: Buyer broadcasts an answer or announcement

Hook 5: Readiness Indicator Updated (Optional)

File: `app/api/dashboard/rfps/[id]/comparison/readiness/route.ts`

Code Location: Before response return

Notification:

```
await notifyUserForEvent(READINESS_INDICATOR_UPDATED, buyer, {
  rfpId: rfp.id,
  rfpTitle: rfp.title,
});
```

When: Buyer runs readiness calculation

Hook 6: Comparison Report Ready (Optional)

File: `app/api/rfps/[id]/compare/report/route.ts`

Code Location: After saving `comparisonReportUrl`

Notification:

```
await notifyUserForEvent(COMPARISON_REPORT_READY, buyer, {
  rfpId: rfp.id,
  rfpTitle: rfp.title,
});
```

When: Board-ready report is generated

Timeline Reminders

Function: `runTimelineReminders()`

Location: `lib/notifications.ts`

Purpose: Process timeline-based reminders for all active RFPs

Logic Flow:
1. Fetch all non-ARCHIVED RFPs
2. For each RFP:
- Check timeline fields (askQuestionsStart, submissionEnd, etc.)
- Compare against current date
- Create notifications for matching events
3. Prevent duplicates using `checkIfNotificationSentToday()`

Timeline Checks:

Field	Condition	Notification	Recipients
`askQuestionsStart`	equals today	Q&A Window Open	All suppliers
`askQuestionsEnd`	3 days from today	Q&A Window Closing Soon	All suppliers
`submissionStart`	equals today	Submission Window Open	All suppliers
`submissionEnd`	3 days from today	Submission Deadline Soon	All suppliers
`submissionEnd`	1 day after	Submission Deadline Passed	Buyer
`demoWindowStart`	equals today	Demo Window Open	Buyer
`awardDate`	3 days from today	Award Date Soon	Buyer
`awardDate`	equals today	Award Date Reached	Buyer

Duplicate Prevention:

```
async function checkIfNotificationSentToday(
  userId: string,
  rfpId: string,
  type: NotificationType
): Promise<boolean> {
  const startOfDay = new Date();
  startOfDay.setHours(0, 0, 0, 0);

  const count = await prisma.notification.count({
    where: {
      userId,
      rfpId,
      type,
      createdAt: { gte: startOfDay },
    },
  });

  return count > 0;
}
```

Date Comparison:

```
function isSameDay(date1: Date, date2: Date): boolean {
  return (
    date1.getFullYear() === date2.getFullYear() &&
    date1.getMonth() === date2.getMonth() &&
    date1.getDate() === date2.getDate()
  );
}
```

User Interface

Buyer Notification Center

Route: `/dashboard/notifications`

Features:
- Displays last 50 notifications
- Reverse-chronological order
- Unread notifications have blue background + dot
- Relative timestamps (“2 hours ago”)
- Category badges with color coding
- Click to navigate to RFP
- Mark as read button

Empty State:
- Displays bell icon
- “No notifications yet” message
- Link back to RFPs

Supplier Notification Center

Route: `/supplier/notifications`

Features:
- Same as buyer, but scoped to supplier notifications
- Navigates to `/supplier/rfps/[id]`
- Link back to supplier portal

Notification Preferences (Buyer)

Route: `/dashboard/settings/notifications`

Features:
- Global toggles: Email, In-App
- Category toggles:
- Timeline Reminders
- Supplier Responses
- Supplier Questions
- Broadcast Announcements
- Readiness & Comparison Updates
- Save button with loading state
- Success/error messages

Notification Preferences (Supplier)

Route: `/supplier/settings/notifications`

Features:
- Global toggles: Email, In-App
- Category toggles:
- Q&A Timeline
- Submission Timeline
- Buyer Broadcasts
- Response Status Updates
- Save button with loading state
- Success/error messages

Bell Icon Components

Buyer: `app/dashboard/bell-icon.tsx`
Supplier: `app/supplier/bell-icon.tsx`

Features:
- Fetches unread count on mount
- Polls every 30 seconds
- Shows `BellDot` icon when unread > 0
- Displays red badge with count (max 99+)

- Links to notification center
- Hover effects

Security & Permissions

Authentication

- All API endpoints require `getServerSession()`
- Unauthorized users receive 401 status

Authorization

- Notification ownership verified before mark-as-read
- Preferences API auto-scopes to current user
- Suppliers cannot access buyer routes
- Buyers cannot access supplier routes

Data Scoping

- Notifications filtered by `userId`
- No cross-user data leakage
- RFP ownership verified in event hooks

Input Validation

- Preferences API validates field names
- Preferences API validates boolean types
- Invalid requests return 400 status

Deployment Instructions

Step 1: Database Migration

```
cd /home/ubuntu/fyndr/nextjs_space
npx prisma generate
npx prisma db push
```

Verify:

```
SELECT * FROM "Notification" LIMIT 1;
SELECT * FROM "NotificationPreference" LIMIT 1;
```

Step 2: Environment Variables

Ensure `.env` contains:

```
DATABASE_URL="postgresql://..."
NEXTAUTH_URL="https://your-domain.com"
RESEND_API_KEY="re_..."
```

Step 3: Build & Deploy

```
npm run build
pm2 restart fyndr
```

Step 4: Setup Cron Job

Option A: Server Cron

```
crontab -e

# Add this line (run every hour at :00):
0 * * * * curl -X POST https://your-domain.com/api/notifications/run -H "Cookie: $
(cat ~/session-cookie.txt)"
```

Option B: External Service (Recommended)
- Use services like:
- Vercel Cron Jobs
- AWS EventBridge
- GitHub Actions
- Render Cron Jobs
- Cron-job.org

Example GitHub Action:

```yaml
name: Notification Runner
on:
  schedule:
    - cron: '0 * * * *'  # Every hour
jobs:
  run:
    runs-on: ubuntu-latest
    steps:
      - name: Call notification runner
        run: |
          curl -X POST https://your-domain.com/api/notifications/run \
            -H "Authorization: Bearer ${{ secrets.API_TOKEN }}"
```

Step 5: Test Deployment

1. **Create test notification:**
   - Log in as supplier
   - Submit a response

- Log in as buyer
- Check bell icon and notification center

2. **Test timeline reminders:**
   - Set RFP timeline fields to today/near future
   - Call `/api/notifications/run` manually
   - Verify notifications created

3. **Test preferences:**
   - Disable email notifications
   - Trigger an event
   - Verify no email sent but in-app notification created

Troubleshooting

Issue: No notifications appearing

Diagnosis:
1. Check user preferences (may be disabled globally)
2. Verify authentication session
3. Check browser console for API errors
4. Query database directly

Solution:

```
-- Check if preferences exist and are enabled
SELECT * FROM "NotificationPreference" WHERE userId = '[user-id]';

-- Check if notifications are being created
SELECT * FROM "Notification" WHERE userId = '[user-id]' ORDER BY createdAt DESC LIMIT
5;
```

Issue: Duplicate notifications

Diagnosis:
- Check if notification runner called multiple times
- Verify `checkIfNotificationSentToday()` logic
- Check date comparison

Solution:

```
-- Find duplicates
SELECT userId, rfpId, type, DATE(createdAt), COUNT(*)
FROM "Notification"
GROUP BY userId, rfpId, type, DATE(createdAt)
HAVING COUNT(*) > 1;
```

Fix:
- Ensure cron job runs only once per hour
- Check `isSameDay()` function logic

Issue: Bell icon not updating

Diagnosis:
1. Check network tab for `/api/notifications/unread-count` calls
2. Verify 30-second poll interval
3. Check session validity

Solution:
- Hard refresh browser (Cmd+Shift+R)
- Clear cookies and re-login
- Check console for JavaScript errors

Issue: Emails not sending

Diagnosis:
1. Verify `RESEND_API_KEY` in `.env`
2. Check `lib/email.ts` sendEmail function
3. Test Resend API directly

Solution:

```
// Test email function
import { sendEmail } from '@/lib/email';

await sendEmail({
  to: 'test@example.com',
  subject: 'Test',
  html: '<p>Test email</p>',
});
```

Check Resend Dashboard:
- View sent emails
- Check for errors
- Verify API key permissions

Issue: Timeline reminders not running

Diagnosis:
1. Check if cron job is configured
2. Manually call `/api/notifications/run`
3. Check RFP timeline fields

Solution:

```
# Manual test
curl -X POST https://your-domain.com/api/notifications/run \
  -H "Cookie: [session-cookie]"

# Check response
{
  "processedRfps": 0,  # If 0, no active RFPs
  "notificationsCreated": 0
}
```

Verify RFP data:

```
SELECT id, title, stage, askQuestionsStart, submissionEnd, awardDate
FROM "RFP"
WHERE stage != 'ARCHIVED'
LIMIT 10;
```

Performance Considerations

Database Indexes

- Index on `(userId, createdAt)` for fast notification queries
- Index on `rfpId` for RFP-related lookups

Query Optimization

- Fetch only last 50 notifications (not all)
- Use `select` to limit fields in includes
- Eager load relations in single query

Polling Frequency

- Bell icon polls every 30 seconds
- Balance between real-time updates and server load
- Consider WebSockets for real-time (future enhancement)

Email Rate Limiting

- Resend has rate limits (check plan)
- Consider batching emails
- Implement exponential backoff on failures

Future Enhancements

Phase 2 Features

- [ ] Push notifications (mobile)
- [ ] WebSocket real-time updates
- [ ] Notification grouping/batching
- [ ] Custom notification sounds

- [ ] Notification digests (daily/weekly summary)
- [ ] SMS notifications (via Twilio)
- [ ] Slack/Teams integration

Analytics

- [ ] Track notification open rates
- [ ] Track click-through rates
- [ ] A/B test notification content
- [ ] User engagement metrics

Maintenance

Regular Tasks

Daily:
- Monitor notification delivery rates
- Check for failed email sends
- Review error logs

Weekly:
- Analyze notification preferences usage
- Check database growth
- Archive old notifications (>90 days)

Monthly:
- Review and optimize queries
- Update notification templates
- Collect user feedback

Support & Resources

Key Files

- Core logic: `lib/notifications.ts`
- Type definitions: `lib/notification-types.ts`
- Testing guide: `STEP_22_TESTING_GUIDE.md`

Documentation

- Prisma Docs: https://www.prisma.io/docs
- Next.js API Routes: https://nextjs.org/docs/app/building-your-application/routing/route-handlers
- Resend API: https://resend.com/docs

Team Contacts

- Backend: [Email]
- Frontend: [Email]
- DevOps: [Email]

Last Updated: November 30, 2025
Version: 1.0.0
Status: Production Ready ✅