Resend Email Integration for AI Executive Summaries

Overview

This document describes the Resend email integration that allows users to share AI-generated Executive Summaries via email directly from the RFP detail page.

Features Implemented

1. Email Utility (`lib/email.ts`)

- **`sendEmail()`** - Sends emails using Resend API
- Validates API key configuration
- Supports multiple recipients
- Returns detailed success/error responses
- Lazy initialization to prevent build-time errors
- **`generateSummaryEmailHtml()`** - Creates professional HTML email templates
- Clean, responsive design with inline CSS
- Color-coded sections (5 sections: Overview, Goals, Dates, Budget, Risks)
- Mobile-friendly layout
- Professional branding with Fyndr logo
- XSS protection via HTML escaping

2. Share API Route (`app/api/rfps/[id]/share/route.ts`)

- **Endpoint**: `POST /api/rfps/[id]/share`
- **Authentication**: Requires NextAuth session
- **Request Body**:

```
json
  {
    "recipients": ["email1@example.com", "email2@example.com"],
    "summaryHtml": "<html>...</html>"
  }
```

- **Validation**:
- Non-empty recipients array
- Valid email format for each recipient
- API key configuration check
- RFP existence verification
- **Response**:

```
json
  {
    "success": true,
    "message": "Summary sent to X recipient(s)",
    "recipients": ["..."]
  }
```

3. Share Modal Component (`app/dashboard/rfps/[id]/share-summary-modal.tsx`)

- **Features**:
- Email input with Add/Remove functionality
- Real-time email validation
- Email chips/tags display
- Collapsible preview section
- Loading states with spinner
- Success message with auto-close (2 seconds)
- Error handling with retry option
- Keyboard support (Enter to add email)
- **Props**:
- `isOpen` : boolean - Modal visibility state
- `onClose` : () => void - Close handler
- `rfpId` : string - RFP identifier
- `rfpTitle` : string - RFP title for email subject
- `summary` : { overview, goals, dates, budget, risks } - Summary data

4. Updated Components

- **`app/dashboard/rfps/[id]/ai-summary.tsx`**
- Added “Share” button next to “Regenerate” button
- Only visible when summary is generated
- Integrated ShareSummaryModal component
- Added `rfpTitle` prop requirement
- **`app/dashboard/rfps/[id]/page.tsx`**
- Updated to pass `rfpTitle` to AISummary component

Setup Instructions

1. Get Resend API Key

1. Go to https://resend.com (https://resend.com)
2. Sign up or log in
3. Navigate to API Keys (https://resend.com/api-keys)
4. Create a new API key
5. Copy the key (starts with `re_`)

2. Configure Environment Variables

Update your `.env` file:

```
# Resend API Key for email sharing feature
# Get your API key from: https://resend.com/api-keys
RESEND_API_KEY="re_your_actual_api_key_here"
```

3. Email Sender Configuration

Development:
- Uses `onboarding@resend.dev` as sender
- Can only send to verified email addresses
- Perfect for testing

Production:
1. Add your domain at https://resend.com/domains (https://resend.com/domains)
2. Verify domain ownership (add DNS records)
3. Update sender email in `lib/email.ts`:
`typescript`
`   from: 'Fyndr <noreply@yourdomain.com>'`

Usage Guide

For End Users

1. **Generate AI Summary**
   - Navigate to any RFP detail page
   - Click “Generate Executive Summary” button
   - Wait for AI to generate the summary

2. **Share Summary**
   - Once summary appears, click the “Share” button (mail icon)
   - Modal opens with email input field
   - Enter recipient email addresses:

     - Type email and click “Add” or press Enter
     - Add multiple recipients as chips/tags
     - Remove any email by clicking the X on its chip
     - Optional: Click “Show Preview” to see email content
     - Click “Send Summary” button
     - Success message appears and modal auto-closes

For Developers

Testing Locally

1. Set up Resend API key in `.env`
2. Start development server:
   `bash`
   `   npm run dev`
3. Navigate to an RFP detail page
4. Generate AI summary
5. Test email sharing with verified email addresses

Email Template Customization

Edit `lib/email.ts` → `generateSummaryEmailHtml()`:

```
// Change colors, spacing, or structure
// All styling is inline CSS for email compatibility
```

API Error Handling

The share API returns detailed errors:

- **401**: User not authenticated
- **400**: Invalid recipients or email format
- **404**: RFP not found
- **500**: Missing API key or send failure

Adding More Sections

To add more summary sections:

1. Update `AISummary` interface in `ai-summary.tsx`
2. Update API summary generation in `app/api/rfps/[id]/summary/route.ts`
3. Update email template in `lib/email.ts`
4. Update modal preview in `share-summary-modal.tsx`

File Structure

```
nextjs_space/
├── lib/
│   └── email.ts                        # Email utility functions
├── app/
│   ├── api/
│   │   └── rfps/
│   │       └── [id]/
│   │           └── share/
│   │               └── route.ts        # Share API endpoint
│   └── dashboard/
│       └── rfps/
│           └── [id]/
│               ├── page.tsx            # RFP detail page (updated)
│               ├── ai-summary.tsx      # AI Summary component (updated)
│               └── share-summary-modal.tsx # Share modal component
└── .env                                # Environment configuration
```

Dependencies

New Packages Installed

```
{
  "resend": "^4.0.1",
  "@react-email/render": "^1.0.1"
}
```

Why These Packages?

- **resend**: Official Resend SDK for Node.js/Next.js
- **@react-email/render**: Required dependency for Resend email rendering

Security Considerations

1. Authentication

- All email operations require valid NextAuth session
- API route validates session before processing

2. Input Validation

- Email format validation using regex
- HTML escaping to prevent XSS attacks
- API key existence checks

3. Rate Limiting

Consider adding rate limiting for production:
- Limit emails per user per hour
- Implement in API route middleware

4. Environment Variables

- Never commit `.env` to version control
- API key is in `.gitignore`
- Use different keys for dev/staging/production

Troubleshooting

Build Error: "Missing API key"

Solution: API key is optional at build time. The code now uses lazy initialization.

Email Not Sending

Checks:
1. Verify `RESEND_API_KEY` is set in `.env`
2. Restart Next.js dev server after adding key
3. Check Resend dashboard for verified emails
4. Verify recipient email is verified (in development mode)

Modal Not Opening

Checks:
1. Ensure summary is generated first
2. Check browser console for errors
3. Verify Share button appears after summary generation

Email Template Not Rendering

Checks:
1. Verify HTML template syntax in `lib/email.ts`
2. Check inline CSS (email clients don't support external CSS)
3. Test with different email clients

Testing Checklist

- [ ] API key configured correctly

- [ ] Share button appears after summary generation
- [ ] Modal opens/closes properly
- [ ] Email input validation works
- [ ] Can add/remove multiple emails
- [ ] Preview section displays correctly
- [ ] Success message appears after sending
- [ ] Error handling works (invalid emails, missing API key)
- [ ] Email received with proper formatting
- [ ] Mobile responsive design

Future Enhancements

Potential Improvements

1. **Batch Sending**
   - Queue emails for large recipient lists
   - Background job processing

2. **Email Templates**
   - Multiple template options
   - Custom branding per organization

3. **Delivery Tracking**
   - Track email opens and clicks
   - Resend webhooks integration

4. **Scheduled Sending**
   - Send emails at specific times
   - Recurring summary emails

5. **Attachment Support**
   - PDF generation of summary
   - Attach original RFP documents

API Reference

sendEmail()

```
async function sendEmail(
  to: string[],
  subject: string,
  html: string
): Promise<{
  success: boolean;
  message?: string;
  error?: string;
}>
```

Parameters:
- `to` : Array of recipient email addresses
- `subject` : Email subject line
- `html` : HTML content of email

Returns: Promise with success status and message/error

generateSummaryEmailHtml()

```
function generateSummaryEmailHtml(
  rfpTitle: string,
  sections: {
    overview: string;
    goals: string;
    dates: string;
    budget: string;
    risks: string;
  }
): string
```

Parameters:
- `rfpTitle`: Title of the RFP
- `sections`: Object with 5 summary sections

Returns: HTML string for email

Support

For issues or questions:
1. Check this documentation
2. Review error messages in browser console
3. Check Resend dashboard logs
4. Verify environment configuration

License

This integration follows the same license as the main Fyndr application.

Last Updated: November 2025
Version: 1.0.0
Author: Fyndr Development Team