🚀 QUICK EMAIL INTEGRATION GUIDE

How to Add Emails to Your Workflows

All email sending now goes through the **UnifiedEmailService**. Here's how to add emails to your app:

📦 Import

```
import { EmailSender } from '@/lib/email/unified-email-service';
```

✅ Ready-to-Use Email Functions

1️⃣ User Registration

Where: After creating new user account

```
await EmailSender.welcome(user.id, user.email, user.name);
```

2️⃣ Password Reset

Where: When user requests password reset

```
const resetUrl = `https://mindfulchampion.com/reset-password?token=${resetToken}`;
await EmailSender.passwordReset(user.id, user.email, user.name, resetToken, resetUrl);
```

3️⃣ Subscription Confirmed

Where: After successful payment/subscription

```
await EmailSender.subscriptionConfirmed(
  user.id,
  user.email,
  user.name,
  'Pro Plan',
  29.99,
  'January 12, 2026'
);
```

4️⃣ Subscription Expiring

Where: Scheduled job (cron) X days before expiry

```
await EmailSender.subscriptionExpiring(
  user.id,
  user.email,
  user.name,
  'Pro Plan',
  'December 20, 2025',
  7  // days left
);
```

5️⃣ Payment Receipt

Where: After successful payment

```
await EmailSender.paymentReceipt(
  user.id,
  user.email,
  user.name,
  29.99,
  'Pro Plan',
  'txn_abc123',
  'December 12, 2025'
);
```

6️⃣ Video Analysis Failed

Where: When video processing encounters error

```
await EmailSender.analysisFailed(
  user.id,
  user.email,
  user.name,
  'My Match Video.mp4',
  'Video format not supported. Please use MP4, MOV, or AVI.'
);
```

7️⃣ Tournament Registration

Where: After user registers for tournament

```
await EmailSender.tournamentRegistration(
  user.id,
  user.email,
  user.name,
  'Holiday Championship',
  'December 20, 2025',
  'Downtown Sports Complex'
);
```

8️⃣ Tournament Reminder

Where: Scheduled job before tournament starts

```
await EmailSender.tournamentReminder(
  user.id,
  user.email,
  user.name,
  'Holiday Championship',
  2,  // hours until
  'Downtown Sports Complex'
);
```

9️⃣ Redemption Request

Where: When user redeems reward points

```
await EmailSender.redemptionRequest(
  user.id,
  user.email,
  user.name,
  'Pro Paddle Upgrade',
  500,  // points cost
  'req_abc123'  // request ID
);
```

🔟 Redemption Approved

Where: When sponsor approves redemption

```
await EmailSender.redemptionApproved(
  user.id,
  user.email,
  user.name,
  'Pro Paddle Upgrade',
  'Paddle Pro Shop',
  'Visit our store at 123 Main St with this email to claim your reward.'
);
```

🛡️ Error Handling Best Practice

```
// ✅ GOOD: Don't let email failures block critical workflows
const emailResult = await EmailSender.welcome(user.id, user.email, user.name);
if (!emailResult.success) {
  console.error('Failed to send welcome email:', emailResult.error);
  // Log to monitoring, but don't fail user registration
}

// ❌ BAD: Blocking critical flow
try {
  await EmailSender.welcome(user.id, user.email, user.name);
  // If email fails, user registration fails too!
} catch (error) {
  throw error; // Don't do this!
}
```

🎯 Common Integration Points

File: `app/api/auth/[...nextauth]/route.ts`

```
// After successful signup:
await EmailSender.welcome(newUser.id, newUser.email, newUser.name);
await EmailSender.adminNewUser(newUser.name, newUser.email, newUser.id, new Date().toI
SOString());
```

File: `app/api/subscription/webhook/route.ts`

```
// Stripe webhook handler:
switch (event.type) {
  case 'checkout.session.completed':
    await EmailSender.subscriptionConfirmed(...);
    await EmailSender.paymentReceipt(...);
    await EmailSender.adminPayment(...);
    break;
}
```

File: `app/api/video/analyze/route.ts`

```
// After video analysis:
if (analysis.success) {
  // Existing video-analysis-complete email already integrated
} else {
  await EmailSender.analysisFailed(userId, userEmail, userName, videoTitle, er-
ror.message);
}
```

File: `app/api/tournaments/register/route.ts`

```
// After successful registration:
await EmailSender.tournamentRegistration(
  user.id,
  user.email,
  user.name,
  tournament.name,
  tournament.date,
  tournament.location
);
```

🧪 Testing

Send Test Email

```
import { UnifiedEmailService } from '@/lib/email/unified-email-service';

await UnifiedEmailService.sendTestEmail('your-email@example.com');
```

Check Email Logs

Go to: `/admin/emails`
- View all sent emails
- Filter by status (SENT, FAILED)
- Resend failed emails
- See error details

🎨 Custom Emails

For custom one-off emails:

```
import { UnifiedEmailService } from '@/lib/email/unified-email-service';

await UnifiedEmailService.sendEmail({
  type: 'CUSTOM',
  recipientEmail: 'user@example.com',
  recipientName: 'John Doe',
  subject: 'Special Announcement',
  html: '<h1>Hello!</h1><p>This is a custom email.</p>',
  text: 'Hello! This is a custom email.',
  userId: 'optional-user-id',
});
```

✅ Complete List of Available Functions

```
EmailSender.welcome()
EmailSender.passwordReset()
EmailSender.subscriptionConfirmed()
EmailSender.subscriptionExpiring()
EmailSender.subscriptionCancelled()
EmailSender.paymentReceipt()
EmailSender.tournamentRegistration()
EmailSender.tournamentReminder()
EmailSender.tournamentResults()
EmailSender.analysisFailed()
EmailSender.redemptionRequest()
EmailSender.redemptionApproved()
EmailSender.redemptionShipped()
EmailSender.adminNewUser()
EmailSender.adminPayment()
EmailSender.adminError()
```

That's it! Simple one-line email sending with automatic logging and error handling. 🎉

For more details, see: `/home/ubuntu/EMAIL_SYSTEM_COMPLETE_OVERHAUL.md`