**AUTOMATIC EMAIL SEND WHEN USER SIGN UP**

**Sprint Goal:** Implement an automatic welcome email feature for new users signing up to "Connect X," a social media engagement application, to improve onboarding and encourage immediate user interaction.

**Feature Overview**

The automatic welcome email feature is a critical component of the "Connect X" onboarding process. Upon successful signup, every new user receives an email that:

* Welcomes them to the platform with a personalized greeting (including their name)
* Explains the core value of "Connect X": connecting users, enhancing engagement, and growing their social presence.
* Includes a call-to-action (CTA) linking to their dashboard to encourage immediate exploration
* Provides a support link for assistance, fostering trust and accessibility.

This feature aims to reduce user drop-off rates post-signup by setting clear expectations and driving engagement from the start. By keeping the email logic independent of the frontend and authentication systems, it remains reusable and adaptable to future changes in "Connect X."

**Objectives**

**Primary**: Automate the delivery of a welcome email to new users after signup

**Secondary**:

* Ensure the solution is modular and independent of frontend or authentication code.
* Use a reliable, easy-to-implement email service to meet sprint deadlines.

**Technical Design and Requirements**

**Requirements**

* **Framework**: Next.js (Pages Router assumed; adaptable to App Router).
* **Email Service**: Resend (API-based email delivery).
* **Inputs**: User email (required) and name (optional) from signup data.
* **Output**: HTML email with "Connect X" branding and CTA links.
* **Environment**: Node.js, hosted on GitHub

**Implementation Process**

**Technology Selection**

I evaluated two options for email delivery:

* **Resend**: A modern email API service with a simple SDK, free tier, and built-in features like retries and analytics.
* **Nodemailer**: A Node.js library using SMTP, offering full control but requiring more setup (e.g., Gmail App Password or custom SMTP).

**Decision**: Chose Resend for 2 main reasons

* Faster implementation
* Managed delivery, reducing maintenance.
* Alignment with Next.js’s serverless architecture.

**Design Consideration**

* **Modularity**: Used a standalone API route to isolate email logic.
* **Scalability**: Selected Resend for its managed delivery and free tier (3,000 emails/month)
* **Security**: Stored sensitive credentials (API key) in environment variables.

**Resend Email Integration in Signup Process**

We set up **Resend** in a Next.js app to send a **welcome email** whenever a new user signs up.

1. **Setup & Configuration:**
   * Installed Resend (npm install resend).
   * Added API key (RESEND\_API\_KEY) and app URL (NEXT\_PUBLIC\_APP\_URL) in .env.local.
   * Ensured .env.local is in .gitignore to keep secrets safe.
2. **Creating the Email API Route (/api/send-welcome-email)**
   * Accepts **POST** requests with email and name.
   * Validates input (email is required).
   * Uses **Resend** to send a welcome email with helpful links.
   * Handles errors and returns appropriate status codes.
3. **Integrating with Signup Process (/api/auth/signup)**
   * After a user signs up, their details are saved to the database.
   * A **fetch request** is made to the email API to send the welcome email.
   * Errors are logged but don’t stop the signup proces

**Challenges, Solutions, and Lessons Learned**

**Challenge 1: Technology Choice**

* **Issue**: Deciding between Resend and Nodemailer.
* **Solution**: Chose Resend for its simplicity and managed features, saving setup time.
* **Lesson**: Prioritizing speed in a sprint can outweigh the flexibility of manual solutions.

**Challenge 2: Ensuring Email Content Relevance**

* **Issue**: Crafting an email that accurately reflected "Connect X"’s value proposition was tricky without direct access to the app’s full branding or user research data.
* **Solution**: Drafted a generic yet focused email highlighting key features (connect, engage, grow) based on the app’s stated purpose, with placeholders for future customization (e.g., dashboard/support URLs).
* **Lesson Learned**: When working with incomplete context, building flexible, editable templates allows for later refinement without major rework.

**Outcomes and Impact**

* **Feature Delivered**: New users now receive a welcome email post-signup.
* **User Experience**: Improved onboarding with a branded, actionable email.
* **Sprint Success**: Completed within timeline, adding measurable value to "Connect X."
* **Team Benefit**: Clear documentation and modular code for future enhancements.

**Conclusion**

The automatic welcome email feature is a great addition to Connect X, making user onboarding smoother and more engaging. By using Next.js and Resend, I built a reliable and flexible solution within the project timeline. This documentation explains the process and results, highlighting my work and paving the way for future improvements.