Admin User Management & Security Features

✅ What Has Been Implemented

1. Fixed Password Reset Functionality

- **Location**: `app/api/auth/request-reset/route.ts`
- **Status**: ✅ COMPLETE
- **Changes**:
- Updated email configuration to use Gmail with proper credentials
- Fixed SMTP settings to work with `GMAIL_USER` and `GMAIL_APP_PASSWORD`
- Email should now be sent successfully

2. Database Schema Updates

- **Location**: `prisma/schema.prisma`
- **Status**: ✅ COMPLETE
- **Added Fields to User Model**:
  ```prisma
  // Phone & SMS fields
  phoneNumber String?
  phoneNumberVerified Boolean @default(false)
  phoneVerifiedAt DateTime?

// Two-Factor Authentication (SMS)
twoFactorEnabled Boolean @default(false)
twoFactorSecret String?
twoFactorExpiry DateTime?
twoFactorBackupCodes Json?
```
- **New Models Created**:
  - SMSVerificationCode - for storing SMS verification codes
  - PhoneResetToken` - for phone-based password reset tokens

3. Twilio SMS Integration Setup

- **Location**: `lib/sms/twilio.ts`
- **Status**: ✅ COMPLETE
- **Features**:
- SMS sending functionality
- 6-digit verification code generation
- Password reset SMS
- 2FA code sending
- Configuration check utility
- **Environment Variables Needed**:
  ```env
  TWILIO_ACCOUNT_SID=your_twilio_account_sid_here
  ```

```
TWILIO_AUTH_TOKEN=your_twilio_auth_token_here
TWILIO_PHONE_NUMBER=your_twilio_phone_number_here
```

4. Admin Dashboard - User Management API Routes

- **Location**: `app/api/admin/users/manage/route.ts`
- **Status**: ✅ COMPLETE
- **Features Implemented**:

Force Password Reset
- Admin can set a temporary password for any user
- If no password provided, generates a random secure password
- Sends email notification with temporary password
- Logs the action in security logs

Usage:
```typescript
POST /api/admin/users/manage
{
  "action": "forcePasswordReset",
  "userId": "user_id_here",
  "temporaryPassword": "optional_temp_password"
}
```

Send Password Reset Email on Behalf of User
- Admin can trigger password reset email for any user
- Generates secure reset token
- Sends reset link via email
- Logs the action

Usage:
```typescript
POST /api/admin/users/manage
{
  "action": "sendPasswordResetEmail",
  "userId": "user_id_here"
}
```

Unlock Account
- Unlocks locked user accounts
- Resets failed login attempts
- Clears lock reason and expiry

Usage:
```typescript
POST /api/admin/users/manage
{
  "action": "unlockAccount",
  "userId": "user_id_here"
}
```

Suspend Account
- Locks user account with reason

- Prevents user from logging in
- Logs suspension with reason

Usage:
```
typescript
  POST /api/admin/users/manage
  {
    "action": "suspendAccount",
    "userId": "user_id_here",
    "reason": "Violation of terms of service"
  }
```

Unsuspend Account
- Removes account suspension
- Clears lock reason

Usage:
```
typescript
  POST /api/admin/users/manage
  {
    "action": "unsuspendAccount",
    "userId": "user_id_here"
  }
```

Delete Account
- Permanently deletes user account
- Cascades to delete all related data (matches, sessions, etc.)
- Logs the deletion

Usage:
```
typescript
  POST /api/admin/users/manage
  {
    "action": "deleteAccount",
    "userId": "user_id_here"
  }
```

5. View User Login History

- **Location**: `app/api/admin/users/login-history/route.ts`
- **Status**: ✅ COMPLETE
- **Features**:
- Fetches user profile information
- Retrieves last 100 security logs (login attempts, locks, etc.)
- Gets last 50 user sessions with details
- Calculates statistics:
  - Total logins
  - Failed login attempts
  - Successful logins
  - Account lock count
  - Unique IP addresses

- Unique devices

Usage:

```
GET /api/admin/users/login-history?userId=user_id_here
```

Response:

```
{
  "user": {
    "id": "...",
    "name": "...",
    "email": "...",
    "loginCount": 45,
    "lastActiveDate": "2025-10-23T...",
    "accountLocked": false
  },
  "statistics": {
    "totalLogins": 45,
    "failedLogins": 3,
    "successfulLogins": 42,
    "accountLocks": 0,
    "uniqueIPs": 2,
    "uniqueDevices": 3
  },
  "loginLogs": [...],
  "sessions": [...]
}
```

🔨 What Still Needs to Be Implemented

1. SMS-Based Password Reset (User-Facing)

Files to Create:
- `app/api/auth/request-sms-reset/route.ts` - Send SMS verification code for password reset
- `app/api/auth/verify-sms-code/route.ts` - Verify SMS code and allow password change
- `components/auth/forgot-password-form.tsx` - Update to add “Reset via SMS” option

Implementation Steps:
1. Add a “Reset via Phone” tab to the forgot password form
2. User enters phone number
3. System generates 6-digit code
4. Save code to `SMSVerificationCode` table
5. Send code via Twilio using `lib/sms/twilio.ts`
6. User enters code on verification page
7. On success, allow password reset

2. Optional 2FA via SMS

Files to Create:
- `app/api/auth/2fa/enable/route.ts` - Enable 2FA for user
- `app/api/auth/2fa/disable/route.ts` - Disable 2FA
- `app/api/auth/2fa/verify/route.ts` - Verify 2FA code during login

- `app/api/auth/2fa/generate-backup-codes/route.ts` - Generate backup codes
- `components/settings/2fa-settings.tsx` - UI for 2FA management

Implementation Steps:
1. Add 2FA settings section in user profile
2. When user enables 2FA:
- Verify phone number with SMS code
- Generate 10 backup codes
- Store in `twoFactorBackupCodes` field
3. During login:
- After password verification, if 2FA enabled, send SMS code
- Require code verification before completing login
- Allow use of backup code if SMS fails
4. Add ability to disable 2FA (requires password confirmation)

3. Phone Number in User Profile

Files to Update:
- `components/settings/profile-settings.tsx` - Add phone number field
- `app/api/user/profile/route.ts` - Add phone number update endpoint
- `app/api/auth/verify-phone/route.ts` - Phone verification endpoint

Implementation Steps:
1. Add phone number input field in profile settings
2. When user adds/changes phone:
- Send verification code via SMS
- Store unverified number temporarily
- On code verification, mark as verified
- Update `phoneNumber`, `phoneNumberVerified`, `phoneVerifiedAt`

4. Admin Dashboard UI Updates

Files to Update:
- `components/admin/sections/enhanced-users-section.tsx` - Add action buttons
- `components/admin/user-detail-view.tsx` - Add management controls
- `components/admin/login-history-modal.tsx` - NEW: Display login history

UI Components Needed:

```
// Add these buttons to user detail view:
- "Force Password Reset" button → Opens dialog
- "Send Reset Email" button → Confirms and sends
- "Unlock Account" button → Shows when account is locked
- "Suspend Account" button → Opens dialog with reason input
- "Delete Account" button → Requires confirmation
- "View Login History" button → Opens modal with history
```

🔐 Security Considerations

1. Admin Authentication

⚠️ **CRITICAL**: Add admin role checking to all admin API routes:

```
// Add this to beginning of each admin API route
import { getServerSession } from "next-auth";
import { authOptions } from "@/app/api/auth/[...nextauth]/route";

// In each route handler:
const session = await getServerSession(authOptions);

if (!session || session.user.role !== 'ADMIN') {
  return NextResponse.json(
    { error: 'Unauthorized' },
    { status: 403 }
  );
}
```

2. Rate Limiting

Consider adding rate limiting for:
- SMS sending (prevent abuse)
- Password reset attempts
- Login attempts

3. Audit Logging

All admin actions are logged to `SecurityLog` table. Consider:
- Adding admin user ID to logs
- Creating separate admin audit log table
- Email notifications to super-admins for critical actions

📦 Required Environment Variables

Make sure these are set in `.env`:

```
# Gmail (Already configured)
GMAIL_USER=deansnow59@gmail.com
GMAIL_APP_PASSWORD=mtjlwqxxgxpzuqwe

# Twilio (YOU NEED TO SET THESE)
TWILIO_ACCOUNT_SID=your_twilio_account_sid_here
TWILIO_AUTH_TOKEN=your_twilio_auth_token_here
TWILIO_PHONE_NUMBER=your_twilio_phone_number_here

# Next Auth
NEXTAUTH_URL=https://mindful-champion-e8mkk8.abacusai.app
NEXTAUTH_SECRET=zRLB8vJYT6iqfo0jb2lZTwbbkLrck6uR
```

🧪 Testing Checklist

Email Password Reset

- [ ] Request password reset via email
- [ ] Receive email with reset link
- [ ] Click link and reset password successfully

- [ ] Verify old password no longer works
- [ ] Verify new password works

Admin Dashboard Features

- [ ] Force password reset as admin
- [ ] User receives temporary password email
- [ ] Send reset email on behalf of user
- [ ] View user login history
- [ ] Unlock a locked account
- [ ] Suspend an account (verify user can't login)
- [ ] Unsuspend an account
- [ ] Delete an account (verify data is removed)

Database

- [ ] Verify phone number fields exist in User table
- [ ] Verify SMSVerificationCode table exists
- [ ] Verify PhoneResetToken table exists
- [ ] Check that migrations have been applied

📝 Next Steps

1. **Get Twilio Credentials**:
   - Sign up at https://www.twilio.com
   - Get your Account SID, Auth Token, and Phone Number
   - Add them to `.env`

2. **Implement Remaining Features**:
   - SMS-based password reset UI and API
   - 2FA enable/disable/verify functionality
   - Phone number verification in profile settings
   - Admin UI components for the new API endpoints

3. **Add Admin Role Check**:
   - Implement admin authentication in all admin API routes
   - Consider using middleware for cleaner code

4. **Test Everything**:
   - Test email password reset
   - Test each admin action
   - Test with different user accounts
   - Check security logs are being created

5. **Deploy and Monitor**:
   - Deploy the changes
   - Monitor for errors in production
   - Check email delivery rates
   - Monitor SMS costs

🆘 Support & Documentation

Twilio Documentation

- https://www.twilio.com/docs/sms/quickstart/node
- https://www.twilio.com/docs/usage/tutorials/how-to-use-twilio-nodejs

Next.js API Routes

- https://nextjs.org/docs/app/building-your-application/routing/route-handlers

Prisma

- https://www.prisma.io/docs/

Last Updated: October 23, 2025