WCA LOGIN SYSTEM

The WCA Login System is a modern authentication platform built with Next.js, Prisma, and MySQL, designed for secure and user-friendly account management. It features both traditional login (username/email + password) and Google OAuth login via NextAuth.js. The system includes email verification, password reset, and a clean user interface styled with Bootstrap and custom CSS.

Features

- User Registration with Full Name, Username, Email, and Password
- Email Verification using unique verification tokens
- Secure Password Hashing with bcryptjs
- Login Authentication with JWT (JSON Web Tokens)
- Google Sign-In using OAuth2 via NextAuth.js
- Password Reset via Email with Expiry Token
- MySQL Database Integration via Prisma ORM
- Responsive UI with Bootstrap 5
- Clean and minimalist form design
- ICIPE branding with centered logo

Tech Stack

- **Frontend**: Next.js 15, React
- **Backend**: Node.js (API routes in Next.js)
- **Database**: MySQL (via Prisma ORM)
- **Email**: Nodemailer with Gmail SMTP
- **Authentication**: bcryptjs + JWT + NextAuth.js (Google Provider)

How It Works

1. Registration Flow

User fills out the registration form
A unique verification token is generated, and the password is hashed
User receives a verification email with a link
Upon clicking the link, their account is verified
They are redirected to the login page

2. Login Flow (Two Options)

Option 1: Login with username/email and password
  If account is verified and credentials are valid, a JWT is issued
Option 2: Sign in with Google (NextAuth.js)
  Only works if the user has been verified. Follow the Google Prompt till you succeed.

3. Forgot Password Flow

User clicks "Forgot Password?" and enters their email
If a matching user exists, a reset token is generated with a 30-minute expiry
The user receives an email with a reset link
User enters a new password on the reset page
After a successful reset, they are redirected to the login page

Project Structure

```
/pages
  index.tsx           → Registration Page
  login.tsx           → Login Page (Google + Manual Login)
  verify.tsx          → Email Verification Handler
  forgot-password.tsx  → Request Reset Page
  reset-password.tsx   → New Password Input Page
  /api
    register.ts         → API route for registration
    login.ts            → API route for login
    verify.ts           → API route for email verification
    forgot-password.ts   → API for initiating password reset
    reset-password.ts    → API for applying new password
    auth/[...nextauth].ts → NextAuth handler for Google Sign-In

/public
  icipe-logo.png       → ICIPE Logo

/prisma
  schema.prisma        → Prisma Database Schema

/utils
  sendEmail.ts         → Verification Email Logic
  sendResetEmail.ts    → Password Reset Email Logic

/styles
  globals.css          → Custom global styles
```

Getting Started

1. Clone the Repository

```
git clone https://github.com/LouisNderitu20/auth-system-project.git
cd auth-system-project
```

2. Install Dependencies

```
npm install
```

3. Configure Environment Variables (.env)

```
DATABASE_URL="mysql://root:yourpassword@localhost:3306/auth_system"
JWT_SECRET="your_jwt_secret"
EMAIL_USER="your_email@gmail.com"
EMAIL_PASS="your_gmail_app_password"
GOOGLE_CLIENT_ID="your_google_client_id"
GOOGLE_CLIENT_SECRET="your_google_client_secret"
NEXTAUTH_SECRET="your_nextauth_secret"
NEXTAUTH_URL="http://localhost:3000"
```

4. Set Up the Database

```
npx prisma migrate dev --name init
```

5. Run the Development Server

```
npm run dev
```

Notes

System supports login via username + password or Google OAuth
Verification and reset emails use your chosen email (e.g. Gmail via Nodemailer)
Google Sign-In is blocked if the user hasn't verified via email
Password reset tokens expire after 30 minutes for security

Future Improvements

User dashboard after login
Admin panel for user management
Loading indicators and toast notifications
Role-based access control

License