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 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)
- ❖ 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

How It Works

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.

Login Flow

- ✓ User logs in with their username and password.
- ✓ If the account is verified and credentials are valid, the user is authenticated.

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
  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
/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**

❖ Create a .env file and add:

1. DATABASE_URL="mysql://user:password@localhost:3306/auth_system"
2. JWT_SECRET="your_jwt_secret_here"
3. EMAIL_USER="your_email@gmail.com"
4. EMAIL_PASS="your_app_password"

4. **Set Up the Database**
   - ✓ npx prisma migrate dev --name init
   - ✓ (If issues arise: npx prisma migrate reset)
5. **Run the Development Server**
   - ✓ npm run dev

Notes

- ❖ The system supports secure login using username and password, not email.
- ❖ Verification and reset emails are personalized using the provided username.
- ❖ Password reset links expire after 30 minutes for security.
- ❖ This app uses Gmail SMTP — App Passwords are recommended.

Future Improvements

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

License

This project is for educational and demonstration purposes.