Access Control System

A secure access control system for small organizations, implemented using Flask.

Features

Baseline Features

- User Authentication (login/logout, password hashing)
- Role-Based Access Control (RBAC)
- User Management (Admin only)
- Profile Management
- Security & Session Controls
- Data Layer with SQLAlchemy ORM
- Logging
- Secure Configuration

Enhancements

- Password Policy (complexity, reuse prevention)
- Audit Log Viewer
- Secure Password Reset

Getting Started

1. Clone the repository
2. Create a virtual environment: `python -m venv venv`
3. Activate the virtual environment:
    - Windows: `venv\Scripts\activate`
    - Linux/Mac: `source venv/bin/activate`
4. Install dependencies: `pip install -r requirements.txt`
5. Install Node.js dependencies: `npm install`
6. Copy `env.example` to `.env` and update with your settings
7. Initialize the database: `python setup_db.py`
8. Run the application: `python run.py` (this will automatically build Tailwind CSS)

Project Structure

```
project/
  app/__init__.py
  app/models.py
  app/auth/routes.py
  app/admin/routes.py
  app/profile/routes.py
  app/security/utils.py
  app/templates/   # Jinja2 templates with Tailwind UI
  app/static/
    css/          # Compiled CSS files
    src/          # Source CSS files for Tailwind
    js/           # JavaScript files
    profile_pics/ # User profile images
  instance/       # Contains SQLite database
  node_modules/   # Node.js dependencies
  tests/
```

Testing

Run the test suite: `pytest`

Security Features

- Password hashing with bcrypt
- CSRF protection
- Session timeout
- Role-based access control
- Input validation
- Error handling with no sensitive information leakage

UI Features

- Modern UI built with Tailwind CSS
- Responsive design for mobile and desktop
- Interactive password strength meter
- Real-time session timeout notifications
- Accessible form controls and validation
- Sortable data tables
- Clean, minimalist aesthetic

Development

Tailwind CSS Development

For active development with Tailwind CSS, you can use the watcher to automatically rebuild CSS when changes are made:

```
npm run watch
```

Tailwind CSS Structure

- `app/static/src/input.css` - Source Tailwind CSS file
- `app/static/css/tailwind.css` - Compiled CSS file (generated)
- `tailwind.config.js` - Tailwind configuration
- `postcss.config.js` - PostCSS configuration for Tailwind

Browser Support

The modern UI is optimized for the latest versions of:

- Chrome
- Firefox
- Safari
- Edge