Guide d'Implémentation pour le Projet Need 4 Stat

April 2, 2025

1 Introduction

Ce document sert de guide pour intégrer le front-end existant du projet "Need 4 Stat" avec un back-end en utilisant Flask. Il s'inspire des pratiques du projet "Supersite" pour offrir une solution complète et fonctionnelle.

2 Étape 1: Configuration de la Base de Données

2.1 Création de la Table `users`

Pour gérer les utilisateurs, créez une table `users` similaire à celle du projet "Supersite".

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL,
    password TEXT NOT NULL,
    email TEXT NOT NULL,
    role TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3 Étape 2: Définir les Routes Flask

3.1 Route pour l'Inscription et la Connexion

Créez des routes pour gérer l'inscription et la connexion des utilisateurs.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['name']
        password = request.form['password']
```

```
        email = request.form['email']
        db = get_db()
        db.execute('INSERT INTO users (username, password, email, role) VALUES (?, ?, ?,
                   (username, password, email, 'user'))
        db.commit()
        return redirect(url_for('welcome_new_user'))
    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['name']
        password = request.form['password']
        db = get_db()
        user = db.execute('SELECT * FROM users WHERE username = ? AND password = ?', (us
        if user:
            session['user_id'] = user['id']
            return redirect(url_for('profile'))
        else:
            return "Erreur de connexion"
    return render_template('login.html')
```

4 Étape 3: Affichage du Profil Utilisateur

4.1 Route pour le Profil

Après la connexion, affichez un récapitulatif des informations de l'utilisateur connecté.

```
@app.route('/profile')
def profile():
    user_id = session.get('user_id')
    if user_id is None:
        return redirect(url_for('login'))
    db = get_db()
    user = db.execute('SELECT * FROM users WHERE id = ?', (user_id,)).fetchone()
    return render_template('profile.html', user=user)
```

4.2 Template HTML pour le Profil

Créez un fichier `profile.html` pour afficher les informations de l'utilisateur.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>Profil Utilisateur</title>
</head>
<body>
    <h1>Bienvenue, {{ user.username }}!</h1>
    <p>Email: {{ user.email }}</p>
    <a href="{{ url_for('logout') }}">Se déconnecter</a>
</body>
</html>
```

5 Améliorations Proposées

- **Validation Côté Client**: Ajouter des validations pour améliorer l'expérience utilisateur.
- **Hashage des Mots de Passe**: Utiliser une bibliothèque comme bcrypt pour hacher les mots de passe avant de les stocker dans la base de données, renforçant ainsi la sécurité des données utilisateur.