FocusFlow開発チュートリアル (Python & Flask版)

このチュートリアルでは、集中力育成SNS「FocusFlow」の基本的な機能をPythonのWebフレームワークであるFlaskを使用して開発する手順を解説します。 まずはMVP(Minimum Viable Product)として、以下の機能を実装することを目指します。

- ユーザー登録・ログイン機能
- 集中セッションの記録（タスク名、時間）
- 集中履歴の表示ダッシュボード

1. 環境構築

まず、開発に必要な環境を整えます。

Pythonのインストール

お使いのシステムにPython 3がインストールされていることを確認してください。まだの場合は、公式サイトからダウンロードしてインストールします。

仮想環境の作成と有効化

プロジェクトごとに環境を分離するため、仮想環境を作成します。

```
# プロジェクトディレクトリを作成
mkdir focus-flow-app
cd focus-flow-app

# 仮想環境を作成 (Windows)
python -m venv venv

# 仮想環境を有効化 (Windows)
.\venv\Scripts\activate
```

必要なライブラリのインストール

Flaskと、データベースを扱うためのFlask-SQLAlchemy、パスワードをハッシュ化するためのWerkzeugなどをインストールします。

```
pip install Flask Flask-SQLAlchemy Flask-Login Werkzeug
```

2. プロジェクトのセットアップ

Flaskアプリケーションの基本的な構造を作成します。

ディレクトリ構成

以下のようなディレクトリ構成を作成します。

```
focus-flow-app/
├── app/
│   ├── __init__.py
│   ├── models.py
│   ├── routes.py
│   ├── static/
│   └── templates/
│       ├── base.html
│       ├── login.html
│       ├── register.html
│       └── dashboard.html
└── run.py
```

`run.py` の作成

アプリケーションを起動するためのファイルです。

```
from app import create_app

app = create_app()

if __name__ == '__main__':
    app.run(debug=True)
```

`app/__init__.py` の作成

Flaskアプリケーションのインスタンスを作成し、各種設定を行います。

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager

db = SQLAlchemy()
login_manager = LoginManager()

def create_app():
    app = Flask(__name__)
    app.config['SECRET_KEY'] = 'your_secret_key' # 実際には複雑なキーを設定してください
    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///db.sqlite'
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

    db.init_app(app)
    login_manager.init_app(app)
    login_manager.login_view = 'main.login'

    from .models import User
```

```
    @login_manager.user_loader
    def load_user(user_id):
        return User.query.get(int(user_id))

    from .routes import main as main_blueprint
    app.register_blueprint(main_blueprint)

    with app.app_context():
        db.create_all()

    return app
```

3. データベースモデルの定義

`app/models.py` に、ユーザーと集中セッションのデータを保存するためのテーブルを定義します。

```
from flask_login import UserMixin
from . import db
from werkzeug.security import generate_password_hash, check_password_hash

class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(100), unique=True, nullable=False)
    username = db.Column(db.String(100), nullable=False)
    password_hash = db.Column(db.String(128))
    sessions = db.relationship('FocusSession', backref='author', lazy=True)

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

class FocusSession(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    task_name = db.Column(db.String(200), nullable=False)
    duration_minutes = db.Column(db.Integer, nullable=False)
    timestamp = db.Column(db.DateTime, server_default=db.func.now())
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
```

4. ルーティングとビューの作成

`app/routes.py` に、各URLに対応する処理（ビュー関数）を記述します。

```
from flask import Blueprint, render_template, redirect, url_for, request, flash
from flask_login import login_user, logout_user, login_required, current_user
from .models import User, FocusSession
from . import db
```

```python
main = Blueprint('main', __name__)

@main.route('/')
def index():
    return redirect(url_for('main.dashboard'))

@main.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form.get('email')
        username = request.form.get('username')
        password = request.form.get('password')

        user = User.query.filter_by(email=email).first()
        if user:
            flash('このメールアドレスは既に使用されています。')
            return redirect(url_for('main.register'))

        new_user = User(email=email, username=username)
        new_user.set_password(password)

        db.session.add(new_user)
        db.session.commit()

        login_user(new_user)
        return redirect(url_for('main.dashboard'))
    return render_template('register.html')

@main.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')
        remember = True if request.form.get('remember') else False

        user = User.query.filter_by(email=email).first()

        if not user or not user.check_password(password):
            flash('メールアドレスまたはパスワードが正しくありません。')
            return redirect(url_for('main.login'))

        login_user(user, remember=remember)
        return redirect(url_for('main.dashboard'))
    return render_template('login.html')

@main.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('main.login'))

@main.route('/dashboard')
@login_required
def dashboard():
```

```
    sessions =
FocusSession.query.filter_by(user_id=current_user.id).order_by(FocusSession.timest
amp.desc()).all()
    return render_template('dashboard.html', username=current_user.username,
sessions=sessions)

@main.route('/start_session', methods=['POST'])
@login_required
def start_session():
    task_name = request.form.get('task_name')
    duration_minutes = request.form.get('duration_minutes')

    if not task_name or not duration_minutes:
        flash('タスク名と時間を入力してください。')
        return redirect(url_for('main.dashboard'))

    new_session = FocusSession(
        task_name=task_name,
        duration_minutes=int(duration_minutes),
        author=current_user
    )
    db.session.add(new_session)
    db.session.commit()

    return redirect(url_for('main.dashboard'))
```

5. HTMLテンプレートの作成

ユーザーがブラウザで見る画面を作成します。

`app/templates/base.html` (共通レイアウト)

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}FocusFlow{% endblock %}</title>
    <style>
        /* 簡単なスタイリング */
        body { font-family: sans-serif; margin: 2em; }
        nav { margin-bottom: 1em; }
        .flash { background: #cee5f5; padding: 0.5em; border: 1px solid #aacbe2; }
    </style>
</head>
<body>
    <nav>
        <h1>FocusFlow</h1>
        {% if current_user.is_authenticated %}
            <a href="{{ url_for('main.dashboard') }}">ダッシュボード</a>
            <a href="{{ url_for('main.logout') }}">ログアウト</a>
```

```
        {% else %}
            <a href="{{ url_for('main.login') }}">ログイン</a>
            <a href="{{ url_for('main.register') }}">登録</a>
        {% endif %}
    </nav>
    <hr>
    {% with messages = get_flashed_messages() %}
    {% if messages %}
        <ul class=flash>
        {% for message in messages %}
            <li>{{ message }}</li>
        {% endfor %}
        </ul>
    {% endif %}
    {% endwith %}
    {% block content %}{% endblock %}
</body>
</html>
```

app/templates/register.html

```
{% extends "base.html" %}

{% block title %}ユーザー登録{% endblock %}

{% block content %}
    <h2>ユーザー登録</h2>
    <form method="POST" action="/register">
        <p><input type="email" name="email" placeholder="メールアドレス" required></p>
        <p><input type="text" name="username" placeholder="ユーザー名" required></p>
        <p><input type="password" name="password" placeholder="パスワード" required>
</p>
        <p><button type="submit">登録</button></p>
    </form>
{% endblock %}
```

app/templates/login.html

```
{% extends "base.html" %}

{% block title %}ログイン{% endblock %}

{% block content %}
    <h2>ログイン</h2>
    <form method="POST" action="/login">
        <p><input type="email" name="email" placeholder="メールアドレス" required></p>
        <p><input type="password" name="password" placeholder="パスワード" required>
</p>
        <p><input type="checkbox" name="remember"> ログイン状態を維持する</p>
```

```
        <p><button type="submit">ログイン</button></p>
    </form>
{% endblock %}
```

app/templates/dashboard.html

```
{% extends "base.html" %}

{% block title %}ダッシュボード{% endblock %}

{% block content %}
    <h2>ようこそ, {{ username }} さん!</h2>

    <h3>新しい集中セッションを開始</h3>
    <form method="POST" action="/start_session">
        <p><input type="text" name="task_name" placeholder="タスク名" required></p>
        <p><input type="number" name="duration_minutes" placeholder="集中時間
（分）" required></p>
        <p><button type="submit">開始</button></p>
    </form>

    <h3>集中履歴</h3>
    <table border="1">
        <thead>
            <tr>
                <th>タスク名</th>
                <th>集中時間（分）</th>
                <th>日時</th>
            </tr>
        </thead>
        <tbody>
            {% for session in sessions %}
            <tr>
                <td>{{ session.task_name }}</td>
                <td>{{ session.duration_minutes }}</td>
                <td>{{ session.timestamp.strftime('%Y-%m-%d %H:%M') }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

6. アプリケーションの実行

すべてのファイルを作成したら、ターミナルで以下のコマンドを実行してアプリケーションを起動します。

```
# focus-flow-app ディレクトリにいることを確認
python run.py
```

ブラウザで `http://127.0.0.1:5000` にアクセスすると、アプリケーションが表示されます。

7. 今後の展望

このチュートリアルで作成したMVPをベースに、「フォーカスフロー仕様.md」にあるさらに高度な機能を実装していくことができます。

- **集中ゲージの実装**: JavaScriptを使い、バックエンドと非同期通信（WebSocketやAjax）を行ってリアルタイムにゲージを更新します。
- **デスクトップアプリ連携**: Electronなどでデスクトップアプリ化し、`psutil`などのライブラリでアクティブなアプリケーションを検知して集中度を判定します。
- **ソーシャル機能**: ユーザー同士で「応援」を送る機能や、バーチャルな「集中ルーム」を作成します。
- **AIによる分析**: 蓄積された集中履歴データを分析し、ユーザーにフィードバックを返すAIコーチング機能を実装します。

このチュートリアルが、あなたの「FocusFlow」開発の第一歩となることを願っています。