DSFG Flask Backend Starter

A comprehensive Flask backend starter template with a well-organized structure, authentication system, and database integration.

Project Structure

```
├── app.py                 # Main application entry point
├── .env.sample            # Sample environment variables
├── requirements.txt       # Project dependencies
├── models/                # Database models
│   ├── __init__.py
│   └── user.py          # User model for authentication
├── routes/                # API routes
│   ├── __init__.py
│   ├── auth.py          # Authentication routes
│   └── main.py          # Main application routes
└── utils/                 # Utility functions
    ├── __init__.py
    └── db.py            # Database utilities
```

Setup Instructions

1. Clone the repository
2. Create a virtual environment

   ```
   python -m venv venv
   ```

3. Activate the virtual environment
   - Windows: `venv\Scripts\activate`
   - Unix/MacOS: `source venv/bin/activate`
4. Install dependencies

   ```
   pip install -r requirements.txt
   ```

5. Create a `.env` file based on `.env.sample`

   ```
   cp .env.sample .env
   ```

6. Run the application

   ```
   python app.py
   ```

API Endpoints

Authentication

- `POST /api/auth/register` - Register a new user
  - Request body: `{"username": "user", "email": "user@example.com", "password": "password"}`
- `POST /api/auth/login` - Login a user
  - Request body: `{"username": "user", "password": "password"}`
- `POST /api/auth/logout` - Logout a user (requires authentication)
- `GET /api/auth/me` - Get current user profile (requires authentication)

Main

- `GET /api/` - API status check
- `GET /api/protected` - Protected route example (requires authentication)

Authentication

This starter uses JWT (JSON Web Tokens) for authentication. To access protected routes, include the token in the Authorization header:

```
Authorization: Bearer <your_token>
```

Database

This project uses PostgreSQL as the database. To set up:

1. Install PostgreSQL on your system if not already installed
2. Create a database named `dsfg_db`

   ```
   createdb dsfg_db
   ```

   or use pgAdmin to create the database
3. Update the `DATABASE_URI` in your `.env` file if needed with your PostgreSQL credentials

```
DATABASE_URI=postgresql://username:password@localhost:5432/dsfg_db
```

Testing

Run tests using pytest:

```
pytest
```