# \*\*Jumble Frontend\*\*

This is the frontend portion of the \*\*Jumble\*\* app, built with Python and a combination of HTML, CSS, and JavaScript. It is designed to work seamlessly with Python frameworks such as \*\*Flask\*\* or \*\*Django\*\*.

---

## \*\*Folder Structure\*\*

```

/frontend

├── /static

│ ├── /css # Styling files

│ ├── /js # JavaScript files

│ ├── /images # Image assets

│ ├── /icons # SVG or PNG icons

│ ├── /fonts # Custom font files

├── /templates

│ ├── /layouts # Base HTML layouts

│ ├── /components # Reusable components like navbar, footer

│ ├── /pages # Individual pages (home, explore, etc.)

├── /context # Global state management (optional)

├── /hooks # Custom JavaScript hooks (if applicable)

├── /utils # Utility functions for frontend logic

```

---

## \*\*Folder Descriptions\*\*

### \*\*1. `/static`\*\*

Holds all static files that are served by the backend.

- \*\*Subfolders:\*\*

- \*\*`/css`\*\*: Global and component-specific stylesheets.

- Example: `styles.css`, `variables.css`.

- \*\*`/js`\*\*: JavaScript files for dynamic interactions.

- Example: `scripts.js`, `api.js`.

- \*\*`/images`\*\*: Logos, backgrounds, and placeholders.

- Example: `logo.png`, `background.jpg`.

- \*\*`/icons`\*\*: Scalable icons for buttons and other UI elements.

- Example: `search-icon.svg`, `menu-icon.png`.

- \*\*`/fonts`\*\*: Custom fonts used in the application.

- Example: `Roboto-Regular.ttf`.

---

### \*\*2. `/templates`\*\*

Contains all HTML files rendered by the backend.

- \*\*Subfolders:\*\*

- \*\*`/layouts`\*\*: Base templates for consistent layouts.

- Example: `base.html` (includes header, footer, etc.).

- \*\*`/components`\*\*: Reusable HTML components.

- Example: `navbar.html`, `footer.html`.

- \*\*`/pages`\*\*: Individual pages for the app.

- Example: `home.html`, `explore.html`, `profile.html`.

---

### \*\*3. `/context`\*\*

Manages global state using Python-based logic or integrations with frontend libraries.

- \*\*Files:\*\*

- `user\_context.py`: Manages user authentication and preferences.

- `theme\_context.py`: Handles light/dark mode state.

---

### \*\*4. `/hooks`\*\*

Custom JavaScript hooks for managing shared logic across components (optional if integrating a JavaScript framework).

- \*\*Files:\*\*

- `useAuth.js`: Handles user authentication state.

- `useFetch.js`: Centralized logic for API calls.

---

### \*\*5. `/utils`\*\*

Helper functions and scripts to streamline frontend operations.

- \*\*Files:\*\*

- `validation.js`: Input validation functions for forms.

- `dateFormatter.js`: Utilities for formatting dates.

- `api.py`: Python utility for backend communication.

---

## \*\*How to Set Up\*\*

1. \*\*Clone the Repository:\*\*

```bash

git clone https://github.com/your-username/jumble-frontend.git

cd jumble-frontend

```

2. \*\*Install Dependencies:\*\*

If using Flask or Django, make sure your backend dependencies are installed. For example:

```bash

pip install flask

```

3. \*\*Set Up Static Files:\*\*

Place your CSS, JS, and images in the `/static` folder.

4. \*\*Organize Templates:\*\*

Add reusable components to `/templates/components` and pages to `/templates/pages`.

5. \*\*Run the Backend to Serve Frontend:\*\*

- \*\*Flask Example:\*\*

```bash

python run.py

```

- \*\*Django Example:\*\*

```bash

python manage.py runserver

```

---

## \*\*Best Practices\*\*

1. \*\*Organize Components\*\*: Break down reusable HTML components (e.g., navbar, modals) into `/components`.

2. \*\*Use Layouts\*\*: Define a consistent layout in `/layouts/base.html` to avoid duplication.

3. \*\*Optimize Static Files\*\*: Minify CSS and JavaScript files for production use.

4. \*\*Leverage Context\*\*: Use Python-based context for managing global app state.

5. \*\*Validate Inputs\*\*: Include client-side validation scripts from `/utils/validation.js`.

---

## \*\*Example Usage\*\*

### \*\*Base Layout (layouts/base.html)\*\*

```html

<!DOCTYPE html>

<html lang="en">

<head>

<title>{% block title %}Jumble{% endblock %}</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/styles.css') }}">

</head>

<body>

{% include 'components/navbar.html' %}

<main>

{% block content %}{% endblock %}

</main>

{% include 'components/footer.html' %}

</body>

</html>

```

### \*\*Home Page (pages/home.html)\*\*

```html

{% extends 'layouts/base.html' %}

{% block title %}Home | Jumble{% endblock %}

{% block content %}

<h1>Welcome to Jumble</h1>

<div class="event-list">

<p>Discover events tailored for you!</p>

</div>

{% endblock %}

```

### \*\*CSS Styling (static/css/styles.css)\*\*

```css

body {

font-family: 'Arial', sans-serif;

margin: 0;

padding: 0;

background-color: #f9f9f9;

}

h1 {

color: #663399;

text-align: center;

}

```

---

## \*\*How to Run Tests\*\*

1. \*\*Install Testing Dependencies (if applicable):\*\*

```bash

pip install pytest flask-testing

```

2. \*\*Run Tests:\*\*

```bash

pytest

```

---

## \*\*Technologies Used\*\*

- \*\*Frontend\*\*: HTML, CSS, JavaScript

- \*\*Backend Frameworks\*\*: Flask/Django

- \*\*Static File Management\*\*: Flask's `url\_for` or Django's `{% static %}`

- \*\*State Management\*\*: Python-based context or JavaScript hooks (optional)

---

## \*\*Contributing\*\*

1. Fork the repository.

2. Create a feature branch.

3. Commit your changes.

4. Push to your fork and submit a pull request.

---

Feel free to extend this README as the project evolves. For questions, contact the development team.