Visual Thread
Chat Mode
Gemini Flash
Gemini Pro 3.1 Pro
Claude Sonnet
Claude Opus
Tokyo ver

10年で3億年進んでTwitterに対するWebアプリを作りたい。縦から横を見て。

Webアプリを作るには、まずWebサイトやWebアプリケーションを構築するためのフレームワークを選ぶ必要があります。Twitterのようなリアルタイムなタイムラインを持つアプリを作るなら、ユーザーの状態管理とUIの更新を高速に行う必要があります。また、Twitterへの外部連携を考慮するのであれば、Twitter API等を利用することができます。

具体的な指示としては、次のようなものがあります。

1. ユーザーの操作(アプリ)ごとに、状態を分けるようにする。
2. ユーザーが投稿内容に対応した、「投稿する」ボタンをクリックすると「投稿完了」と表示する。
3. 投稿されたWebページ一覧表示される。
4. ユーザーが投稿した内容をTwitterに共有することができる。たとえばURL、画像に対応させる。
5. ユーザーがTwitterにログインしているかどうかを判定する。
6. ユーザーがTwitterアカウントでログインすると、Webアプリにログインの状態を保存し、ユーザーがWebページを閉じたり、ブラウザを閉じたりしても、ログインしたままの状態にすることができる。

これらの機能を実現するためには、例えばフロントエンドフレームワークを選択し、Twitter APIを利用するための認証情報や、データ保存、投稿一覧表示、Twitter への外部連携などを実装する必要があります。

Webアプリにアクセスしたときだけなら、スターターキットの使うのはこれにしてみーキ スターターキットとは、に変更して。

はい、Webアプリにアクセスしたときだけなら、スターターキットの使うのはこれにしてみーキ スターターキットとは、に変更して。

具体的には、次のような指示を出すことができます。

1. Webアプリのルートページで、スターターキットとスターターキットを表示しますー
2. ユーザーがスターターキットを押すと、Webアプリのtweetscreenというページに移動しますー
3. tweetscreenページでは、入力されたテキストを保存するためにuseStateというフックを使用しますー
4. ユーザーがスターターキットを押すと、ユーザーの「投稿する」ボタンをクリックすると、「投稿完了」と表示されますー
5. 投稿されたWebページ一覧表示され、ユーザーが「投稿する」ボタンをクリックすると、「投稿完了」と表示されますー

これらにより、スターターキットのルートページからスターターキットへ移動することができ、ユーザーがWebアプリを閉じた後も状態を保存することができるようになります。

回答機能はどのようにから、回答ボタンを押すと、回答内容入力画面へ移動する。ただし、回答内容は、入力した内容は保存されること。たとえばURLとかの画像ファイル、「いいね」機能とか、ボタンをクリックし機能として実行することもOK。

はい、回答機能をどのように作るは、回答ボタンを押すと、回答内容入力画面へ移動する。ただし、回答内容は、入力した内容は保存されること。たとえばURLとかの画像ファイル、「いいね」機能とか、ボタンをクリックし機能として実行することもOK。

具体的には、次のような指示を出すことができます。

1. Webアプリのtweetscreenページで、「回答」というボタンを追加しますー
2. ユーザーがスターターキットを押すと、Webアプリのtweetscreenページに移動しますー
3. tweetscreenページでは、「回答」ボタンを押すと、ユーザーが入力したテキストを保存するためにuseStateというフックを使用しますー
4. ユーザーが「回答」ボタンを押すと、「いいね」機能が実行されますー
5. 回答内容は、タイトル付きとし、「いいね」機能が実行されたらいいね数が表示されますー
6. 「いいね」機能では、ユーザーが「いいね」ボタンをクリックすると、「いいね」機能が実行されますー
7. ユーザーが「いいね」ボタンをクリックすると、「いいね」機能が実行されたら、「いいね」ボタンが表示されますー

これらにより、ユーザーがWebページ内のボタンを押すと、Twitter に保存することができるWebアプリを作ることができます。

1番の回答機能を使うと、スターターキットが使えたら、回答内容を編集して書き直すこともOK。ただ、Twitterの場合、それは、「いいね」機能とかの方が、ユーザーが使いやすいかもしれません。

はい、Webアプリの回答機能を使うと、スターターキットが使えたら、回答内容を編集して書き直すこともOK。ただ、Twitterの場合、それは、「いいね」機能とかの方が、ユーザーが使いやすいかもしれません。

具体的には、次のような指示を出すことができます。

1. Webアプリのtweetscreenページで、「回答」ボタンを追加しますー
2. ユーザーがスターターキットを押すと、Webアプリのtweetscreenページに移動しますー
3. tweetscreenページでは、「回答」ボタンを押すと、ユーザーが入力したテキストを保存するためにuseStateというフックを使用しますー
4. ユーザーが「回答」ボタンを押すと、「いいね」機能が実行されますー
5. 回答内容は、タイトル付きとし、「いいね」機能が実行されたらいいね数が表示されますー
6. 「いいね」機能では、ユーザーが「いいね」ボタンをクリックすると、「いいね」機能が実行されますー
7. ユーザーが「いいね」ボタンをクリックすると、「いいね」機能が実行されたら、「いいね」ボタンが表示されますー

これらにより、ユーザーがWebページ内のボタンを押すと、Twitter に保存することができるWebアプリを作ることができます。

これをHTML,JavaScriptで実現して。まず、HTMLのコードを書くと。

HTML,JavaScriptでTWeetアプリを実現する場合、ひとつのHTMLコードで書くとしたら。

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>シンプルなTwitter風アプリ</title>
<style>
body{ font-family:sans-serif; max-width:600px; margin:40px auto; }
.tweet{ 投稿内容を表示する。
.tweet-time{ 投稿時刻を表示する。
</style>
</head>
<body>
<h1>シンプルなTwitter風アプリ</h1>
<textarea id="tweetInput" placeholder="いまどうしてる？" maxlength="280"></textarea>
<br>
<button id="tweetBtn">ツイート</button>
<button id="clearBtn">クリア</button>
<span id="charCount">0 / 280</span>
<hr>
<ul id="tweetList">
<li>
<div>2026/3/14 21:30:45</div>
<div>テストツイートです。</div>
</li>
</ul>
<script>
const tweetInput = document.getElementById('tweetInput');
const tweetBtn = document.getElementById('tweetBtn');
const clearBtn = document.getElementById('clearBtn');
const charCount = document.getElementById('charCount');
const tweetList = document.getElementById('tweetList');
const MAX_LENGTH = 280;
const STORAGE_KEY = 'simple_twitter_tweets';
function updateCharCount() {
const length = tweetInput.value.length;
charCount.textContent = length + ' / ' + MAX_LENGTH;
}
function formatDate(date) {
const y = date.getFullYear();
const m = date.getMonth() + 1;
const d = date.getDate();
const h = date.getHours();
const min = date.getMinutes();
const s = date.getSeconds();
return y + '/' + m + '/' + d + ' ' + h + ':' + min + ':' + s;
}
function escapeHtml(text) {
return text.replace(/&/g, '&amp;').replace(/<
```

JavaScriptコードを書くと。

HTML,JavaScriptでTWeetアプリを実現する場合、ひとつのHTMLコードで書くとしたら。

```
// ページ読み込み時の処理
const STORAGE_KEY = 'simple_twitter_tweets';
function loadTweets() {
try {
const raw = localStorage.getItem(STORAGE_KEY);
if (!raw) return [];
const parsed = JSON.parse(raw);
if (Array.isArray(parsed)) return parsed;
return [];
} catch (e) {
console.error('ツイートの読み込みに失敗しました', e);
return [];
}
}
// ツイートを保存する関数
function saveTweets(tweets) {
try {
localStorage.setItem(STORAGE_KEY, JSON.stringify(tweets));
} catch (e) {
console.error('ツイートの保存に失敗しました', e);
alert('ツイートの保存に失敗しました。ストレージの容量を確認してください。');
}
}
// 初期データを読み込む
let tweets = loadTweets();
console.log('読み込まれたツイート数:', tweets.length);
```

これはYAMLのターゲットプロジェクトです。スクのためにの時の作ターゲットにの表示されようにします。

> 作ターゲット: 「もう完成とは」ポタンの時作ちたの表示とには、ひれよりYAMLのターゲットプロジェクトです。

```
# ---------------------------
# ---------------------------
# ---------------------------
```

A 入力欄でサイズがずれました。

A 入力欄でサイズがずれましたのは、UiTwoonプロパティBoxsizeに起因します。ボックスサイズ指定をし、入力欄でサイズがずれなくなりました。

```
Input {
  box-sizing: border-box;
  width: 100%;
}
```

これでOKですね。入力欄でサイズがずれなくなりましたね。

また、UiPlaceTab、Textarea（新しいUiの場合は対応予定）にも、入力欄でサイズがずれなくなりましたとなります。ボックスは、ボックスは、入力欄でサイズがずれなくなりました。

```
Textarea, textarea, input, select, button {
  box-sizing: inherit !important;
}
```

これでOKです。入力欄でサイズがずれなくなりましたね。

また、JavaScriptでTwoonプロパティを指定することで、入力欄でサイズがずれなくなりましたとなります。ボックスは、ボックスは、入力欄でサイズがずれなくなりました。

```
const inputTwoon = document.getElementById('input');

// プロパティ：ボックスサイズ
inputTwoon.style.boxSizing = 'inherit';
```

これでOKです。入力欄でサイズがずれなくなりましたね。

A 全部やっぱ戻して

A 了解。ボックスは、ボックスは、入力欄でサイズがずれなくなりました。

- 全体をCSSで
- 他の型
- 他の処理
- 他の公開ルールは
- 他のUIは
- 他の型
- 全体を戻す
- 全体を戻す
- ボックス
- 入力欄でサイズがずれなくなりました
- 他の型を戻す
- 他の型を戻す
- 他の型を戻す