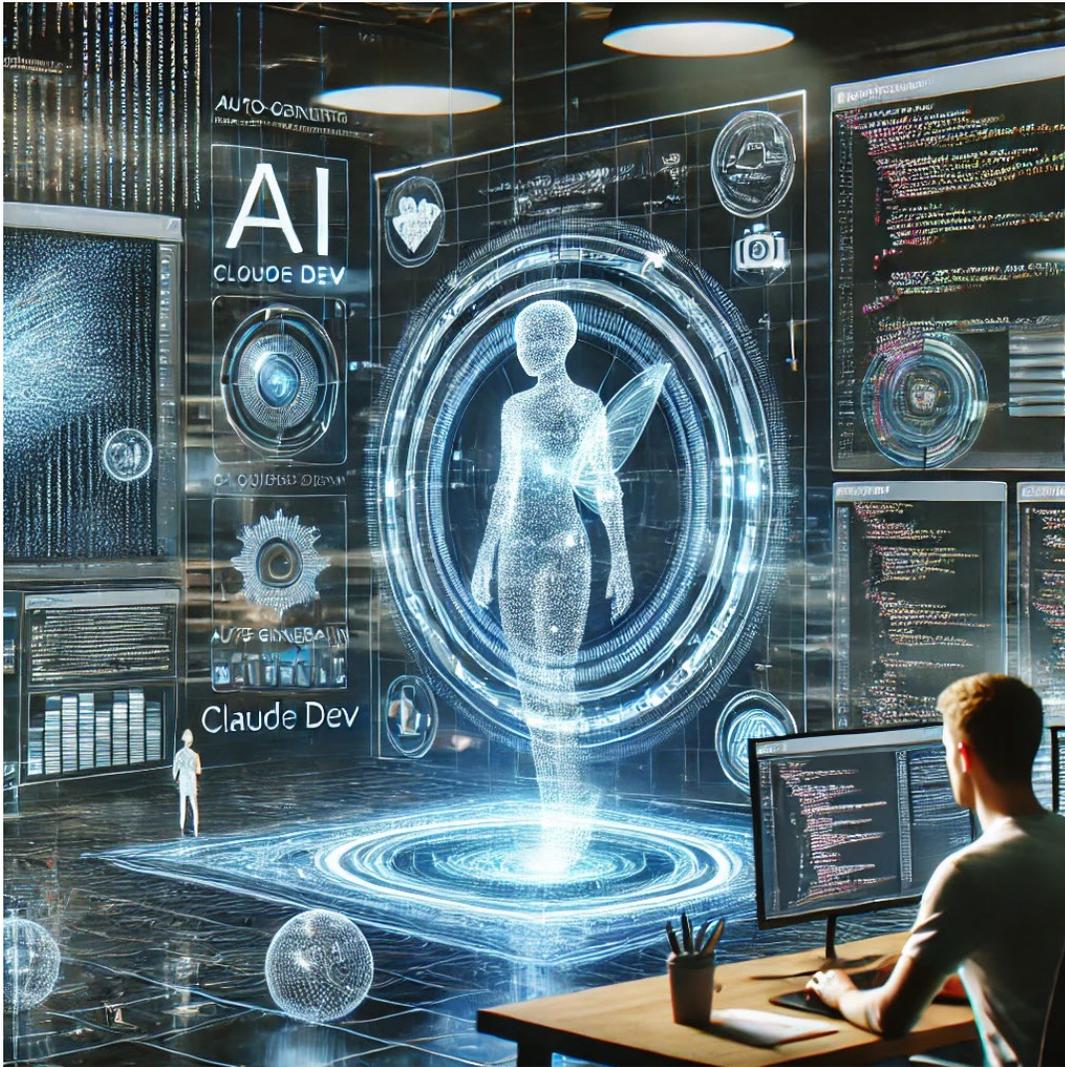


2024.10.8

Vol.4 プロジェクトマネジャーのための
AI DrivenによるPJ管理ツール開発①

～ VS Code + Claude Devで生成AIによるPJ管理用WEBアプリを自動生成 ～

VS Code + Claude Devで生成AIによるWEBアプリを自動生成



※ DALL-E3で生成した『Claude Devによるアプリ自動生成のイメージ』

- 生成AIによるアプリのノーコードによる自動生成の進化には激しいものがあります。
- ついにPromptにより、様々なWebアプリを自動生成する実効的な仕組みが、VS Codeの拡張機能で、Claude Devを利用することで可能になって来ました。
- VS Code + Claude Devの組合せで、その破壊的な効果を確認することが可能です。
- 我々は、この仕組みをPM手法のAI-Drivenによる一つの解法として検討を進めます。
- DifyなどのRAGのワークフローと組み合わせることで、経験知を蓄積しながら、WEB I/FでPJを効率化する仕組み作りを狙います。

目次

1. Claude Devとは？
2. VS Codeのインストールから、Claude Devの拡張機能設定まで
3. Gemini 1.5 FlashからAPIキーの取得
4. Claude Devに対するWebアプリの仕様の入力
5. Webアプリフロントエンドへの適用結果と、修正方法
6. バックエンドへのClaude Code適用の準備：Python Flaskの利用
7. Anthropic Claude 3.5 Sonnetのインストール
8. Anthropic Claude によるバックエンド向けPrompt自動生成
9. バックエンドへのClaude Code適用
10. まとめ
11. おわりに

Claude Devとは？ ✓ 生成AIによる最新のアプリ作成機能により、ノーコードでPJ管理用アプリを作成する

Claude Dev は、Anthropic の Build with Claude 2024 年 6 月ハッカソンの一環としてSaoud Rizwan によって開発された VS Code 拡張機能です。Claude 3.5 Sonnet のエージェント コーディング機能を活用して、複雑なソフトウェア開発タスクを段階的に処理します。AI がファイルを読み書きしたり、プロジェクト全体を最初から作成したり、ターミナル コマンドを実行したり (許可後) できるツールにより、Claude Dev は単純なコード補完やテクニカル サポートを超えた方法でユーザーを支援できます。

Claude Devの概要と利用方法：

- ①MicrosoftのVS Codeの拡張機能に、Claude Devを追加します。
- ②利用したい生成AIを選択して、APIキーを連携させます。
- ③Claude Devのメニューを立上げて、タスク指定欄から、生成AI利用と同じスタイルで、Promptで作成したいWebアプリの仕様を入力します。
- ④ヒューマンインザループの形で、生成された アプリの結果に、修正したい内容をタスク入力します。
- ⑤グラフなどのUIも美しく出力出来ます。
- ⑥一度に生成出来ない場合は、逐次Step by Stepで機能を追加・修正出来ます。

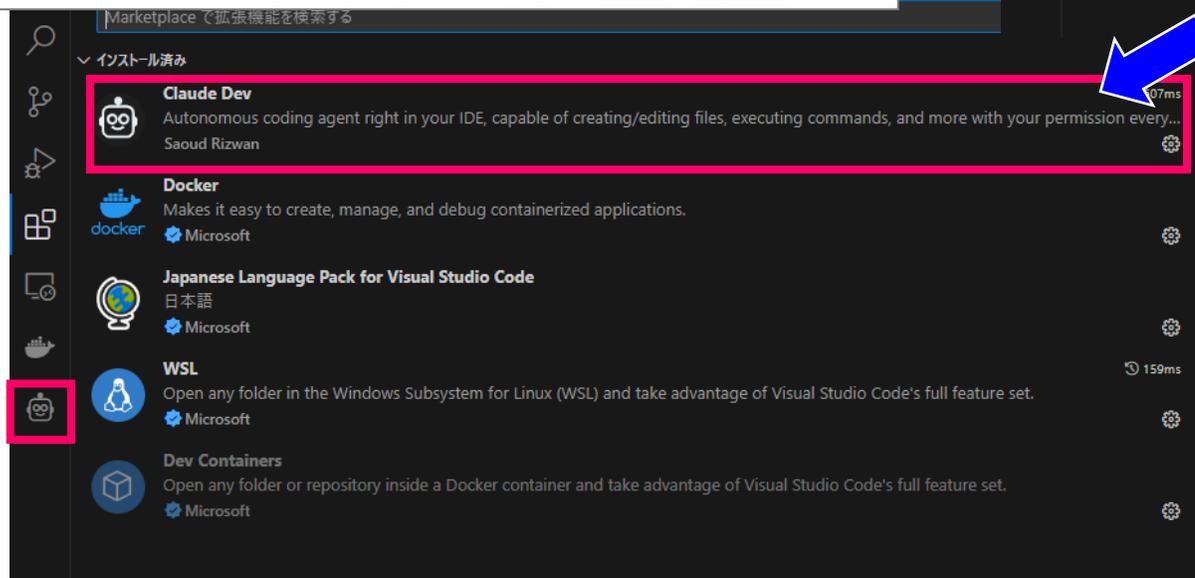
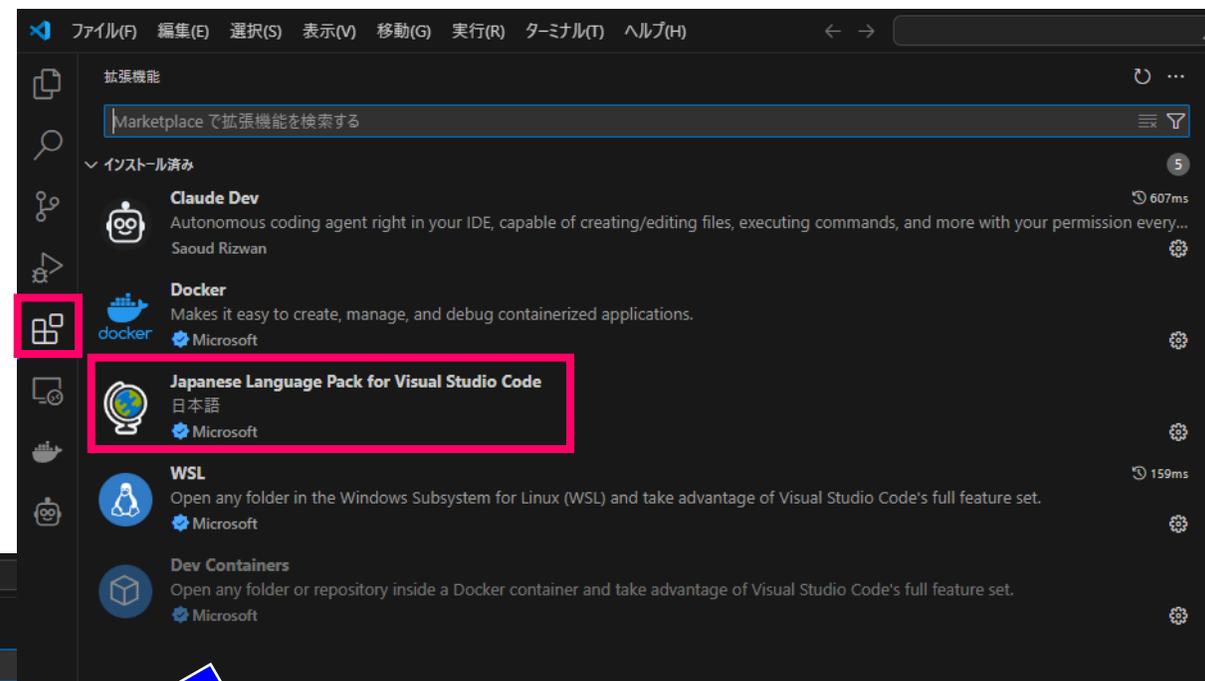
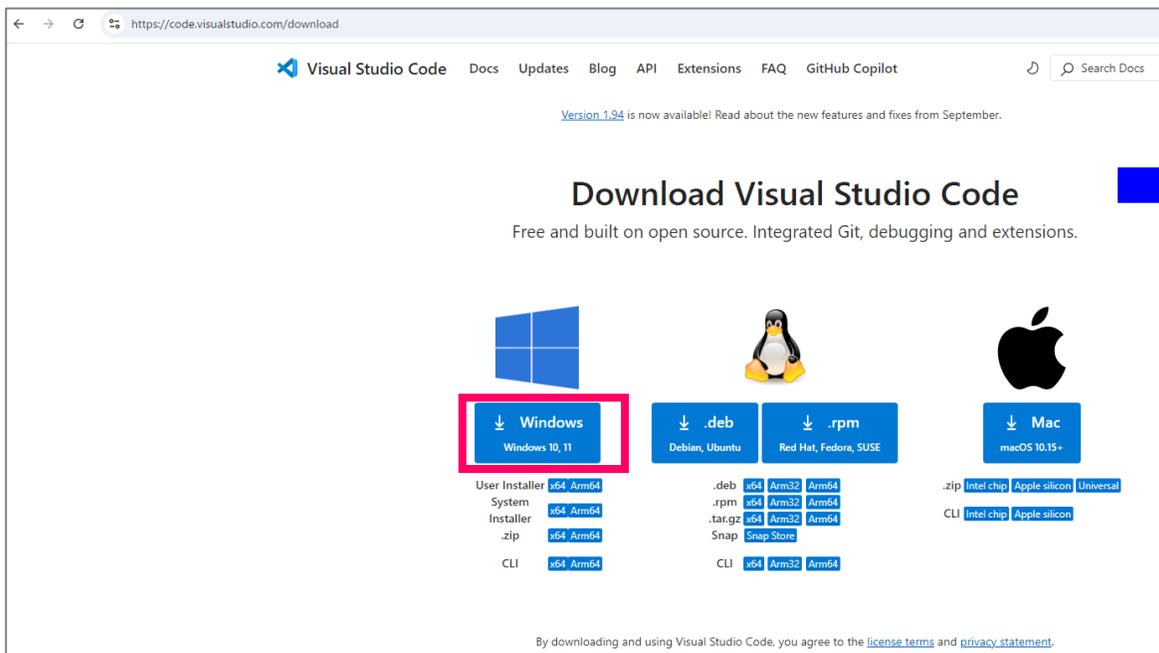
※ 以下の試行は、個人の試行レベルであり、商用環境によるフローではないことにご注意下さい。

VS Codeのインストール

Vs Codeをインストールして、次に拡張機能



から日本語化する



続いて拡張機能からClaude Devをインストールする
左側にロボット  のメニューが顕れる

これがClaude Devのメニュー選択ボタンになる

Claude Devで利用する生成AIを選んでAPIキー設定
(P.12)

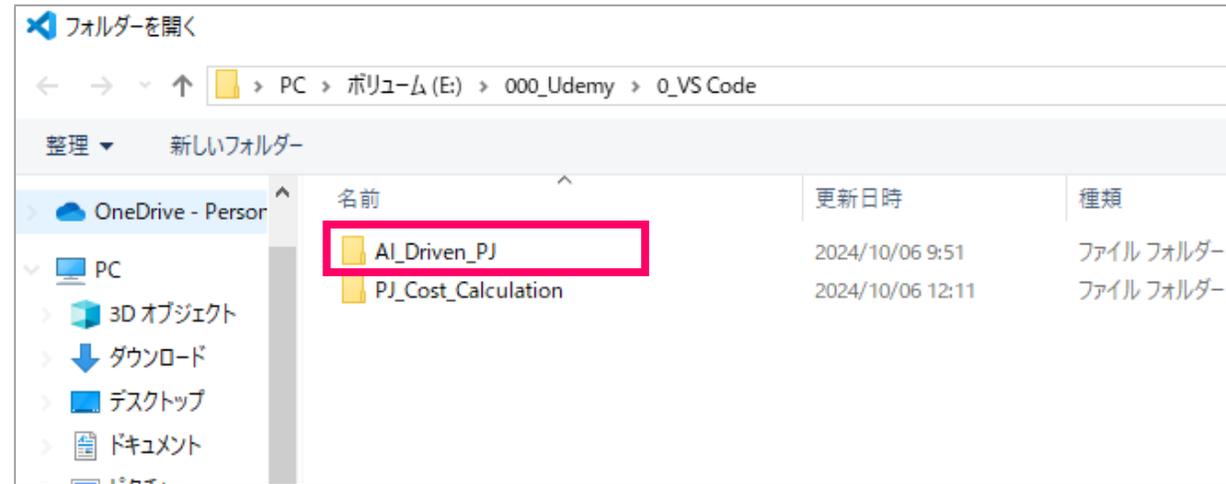
今回はGoogle Geminiを選択
(P.7以降を参照)



VS Codeで新しいフォルダを作成する

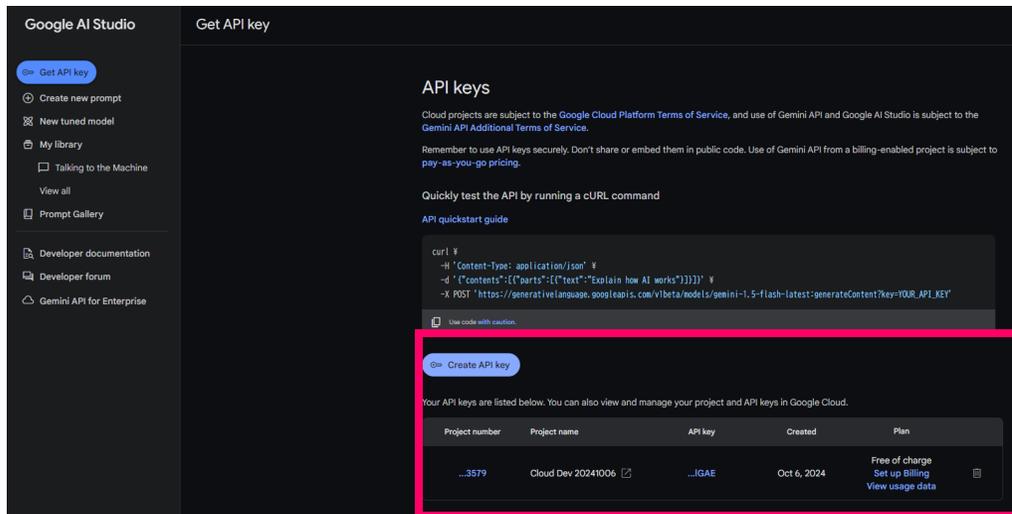
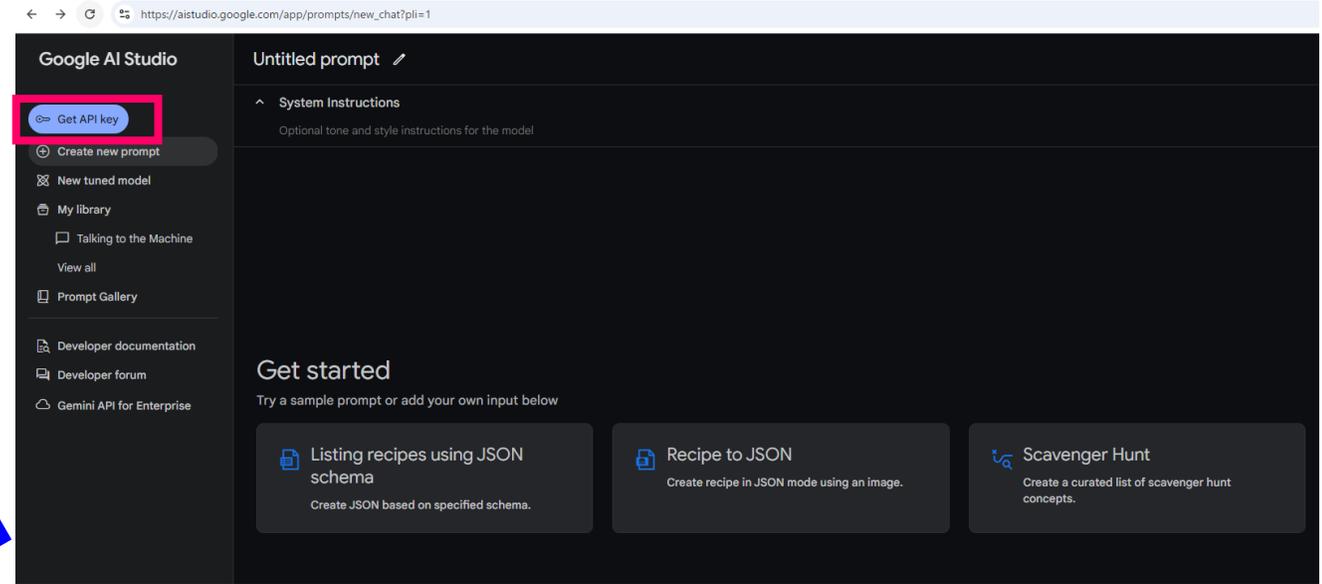


新しいフォルダを作成する



フロントエンドアプリの自動生成

Google Gemini 1.5 FlashでAPIキーを取得する①



ここでプロジェクトを作成していないと、それを要求してくるので、次の画面でプロジェクトを作成する

プロジェクトを作成する①



Google Cloud
https://cloud.google.com > ... > プロジェクトの作成と管理 :

プロジェクトの作成と管理 | Resource Manager Documentation

2024/08/30 — 新しいプロジェクトは、Google Cloud コンソール、Google Cloud CLI または projects.create() メソッドを使用して作成できます。コンソール gcloud ...
プロジェクトの作成 · プロジェクトの識別 · プロジェクトの一覧表示 · プロジェクトの更新

初めて Gemini API を実行する場合は、プロジェクトを作成する必要があります

Resource Manager > Documentation > ガイド

プロジェクトの作成と管理

この情報は役に立ちましたか?

このページでは、Cloud Resource Manager API と Google Cloud コンソールを使用して Google Cloud プロジェクトを作成および管理する方法について説明します。

準備

リソース階層の概要で、プロジェクト リソースについてお読みください。リソース階層の設定については、[Google Cloud ランディング ゾーンのリソース階層を決定する](#)をご覧ください。

プロジェクトの識別には次のものが使用されます。

- **プロジェクト名:** 人が読めるプロジェクト名。
プロジェクト名は Google API では使用されません。プロジェクト名は、プロジェクトの作成中や作成後にいつでも編集できます。プロジェクト名は一意である必要はありません。
- **プロジェクト ID:** グローバルに一意のプロジェクト ID。
プロジェクト ID は、プロジェクトを Google Cloud の他のすべてのプロジェクトと区別するために使用する一意の文字列です。プロジェクト名を入力すると、Google Cloud コンソールは文字、数字、ハイフンの組み合わせで一意のプロジェクト ID を生成します。生成されたプロジェクト ID を使用することをおすすめしますが、この ID はプロジェクトの作成時に編集することもできます。プロジェクトが作成されると、プロジェクト ID は不変になります。
プロジェクト ID には次の要件があります。
 - 6~30 文字にする必要があります。

プロジェクトを作成する②

自分で指定するプロジェクト名を入力する

プロジェクトを選択

プロジェクトとフォルダを検索

最近のプロジェクト

名前	ID
Cloud Dev 20241006	cloud-dev-20241006
My First Project	adroit-particle-361115

キャンセル

Google Cloud

新しいプロジェクト

警告 割り当て内の残りのプロジェクト数は 23 projects 件です。プロジェクトの増加をリクエストするか、プロジェクトを削除してください。 [詳細](#)

[MANAGE QUOTAS](#)

プロジェクト名* AI_Driven_PJ

プロジェクト ID: mystical-app-437804-c8 後で変更することはできません。 [編集](#)

場所* 組織なし [参照](#)

親組織またはフォルダ

作成 キャンセル

Google Gemini Flash 1.5 でAPIキーを生成する②

APIキーを作成の欄で検索すると、先ほど指定したプロジェクト名が表示されるので、それを選択する

The screenshot shows the Google AI Studio interface. On the left, there's a sidebar with navigation options like 'Get API key', 'Create new prompt', etc. The main area is titled 'API キーを取得' and contains instructions and a 'cURL コマンドを実行して API をすばやくテスト' section. A modal window titled 'API キーを作成' is overlaid, asking to select a Google Cloud project. A search bar in the modal contains 'Google Cloud プロジェクトを検索'. Below the search bar, there's a button 'キー 既存のプロジェクトで API キーを作成'. A blue arrow points from this button in the main interface to the modal. Another blue arrow points from the modal's button to the text 'Claude Dev にAPIキーを設定'.

プロジェクト番号	プロジェクト名	API キー	作成日	プラン
...3579	Cloud Dev 20241006	...JGAE	2024/10/06	無料 お支払い情報を設定 使用状況データを表 示

GeminiのAPIキーが生成された

注意) この状態で作成したAPIによる結果は Googleにオープンなので、企業の機密情報、個人プライバシーの情報を利用する用途には使用しないこと!!!

Claude Dev に Gemini Flash 1.5 で取得したAPIキーを設定

CLAUDE DEV

Settings Done

API Provider
Google Gemini

Gemini API Key
.....
This key is stored locally and only used to make API requests from this extension.

Model
gemini-1.5-flash-002

✓ Supports images
Max output: 8,192 tokens
* Free up to 15 requests per minute. After that, billing depends on prompt size. For more info, see pricing details.

Custom Instructions

システムコマンド

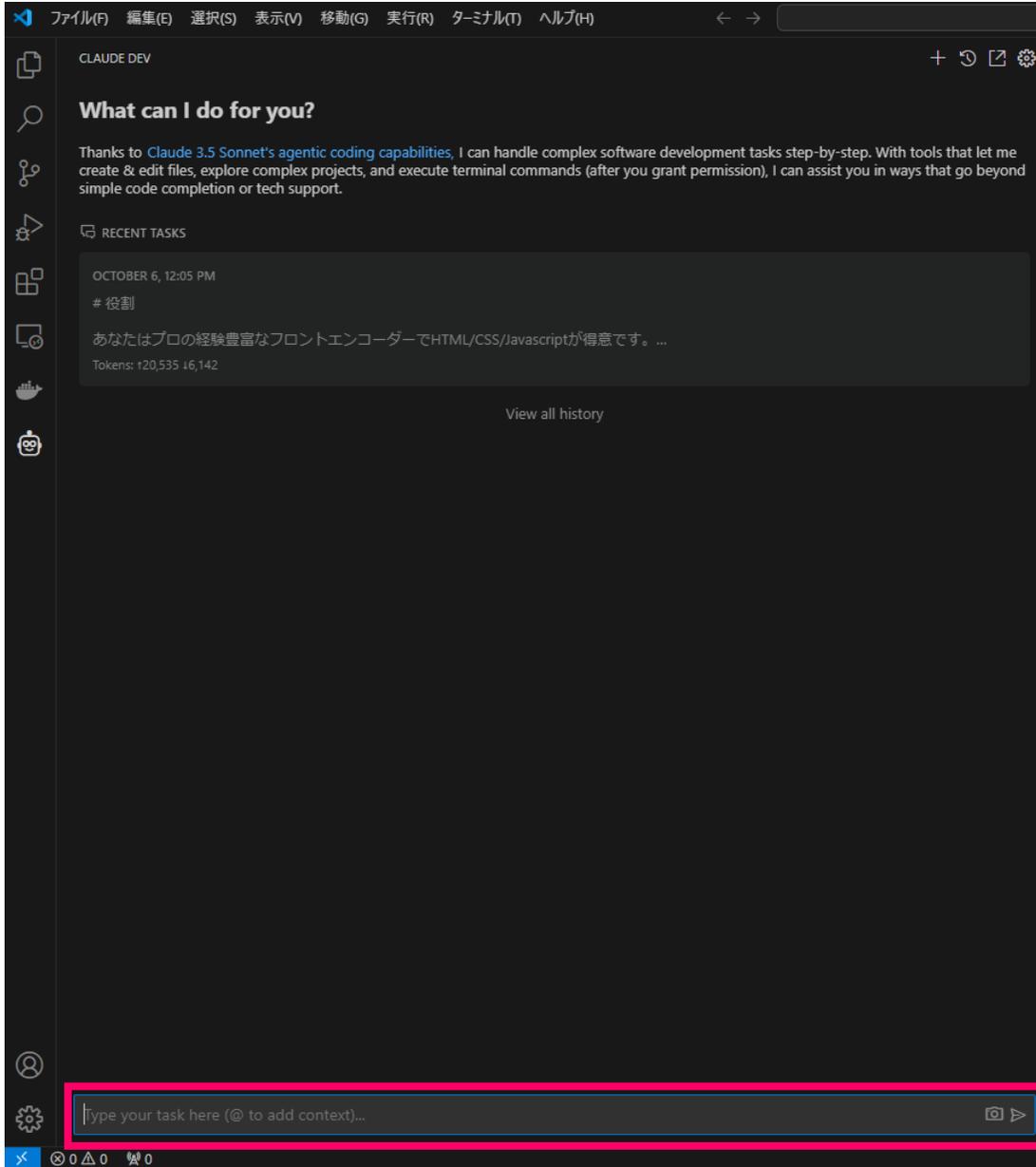
・日本語で必ず応答してください。

These instructions are added to the end of the system prompt sent with every request.

Always allow read-only operations
When enabled, Claude will automatically read files, view directories, and inspect sites without requiring you to click the Allow button.

PromptでClaude Devに生成したいアプリの要求仕様を入力する

PJのメンバー毎の稼働単価と工数を管理する
可視化を含むWebアプリを作成してみる



役割

あなたはプロの経験豊富なフロントエンコーダーでHTML/CSS/Javascriptが得意です。
次の仕様書に従い、ソースコードを書いて下さい。

要求

- ・フロントエンドのみで動作するPJ稼働コスト管理表（日々の稼働メンバ単位での稼働時間、稼働単価、日々のトータル稼働コスト）を開発して下さい。
日々のトータルコスト = \sum （日々のメンバ毎の稼働単価 × 稼働時間）とします。
- ・ソースはHTML/Javascript/CSSで記載して、一つのファイルに纏めて下さい。
- ・稼働時間は、時間（HR）で入力して、稼働単価は円貨で入力します。また日々のトータル稼働コストは、千円単位（K¥）で出力してください。
- ・トータル稼働コストを、メンバ別に色分けした積み上げ棒グラフで日単位・月単位・年単位で可視化してください。
- ・また月単位で、円グラフで数値の多い順に、トータル稼働時間、トータル稼働コストを可視化してください。
- ・サンプルとして擬似データを作成し入力した状態として下さい。

勿論この先には、工数分析や様々な機能の拡張もStep by Stepで可能です。
（様々なUIの追加・改良、グラフの生成なども含めて）

Claude Devを実行した結果

TESTデータを含んだHtmlファイルが生成されている！

The screenshot shows the Claude Dev IDE interface. On the left, a task pane displays the generated HTML code. The main editor shows the rendered HTML file, which includes a table of project cost data. A yellow callout points to the table, and a red box highlights the table content in the code editor.

```
<script>
const data = [
  { date: '2024-07-29', member: 'A', hours: 8, rate: 10000 },
  { date: '2024-07-29', member: 'B', hours: 6, rate: 12000 },
  { date: '2024-07-30', member: 'A', hours: 7, rate: 10000 },
  { date: '2024-07-30', member: 'B', hours: 8, rate: 12000 },
];

let dailyCost = 0;
data.forEach(item => {
  dailyCost += item.hours * item.rate;
});
document.getElementById('total-daily-cost').textContent = Math.round(dailyCost / 1000)

// チャート描画処理は省略 (Chart.jsなどを使用)
</script>

</body>
</html>
```

日付	メンバー	稼働時間 (HR)	稼働単価 (円)	日々の稼働コスト (円)
2024-07-29	A	8	10000	80000
2024-07-29	B	6	12000	72000
2024-07-30	A	7	10000	70000
2024-07-30	B	8	12000	96000

これを一旦、Saveする

Saveした後の画面

The screenshot shows a code editor with the following content:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>PJ稼働コスト管理表</title>
5 <style>
6 body { font-family: sans-serif; }
7 table { width: 100%; border-collapse: collapse; }
8 th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
9 th { background-color: #f2f2f2; }
10 canvas { width: 800px; height: 600px; }
11 </style>
12 </head>
13 <body>
14
15 <h1>PJ稼働コスト管理表</h1>
16
17 <table>
18 <thead>
19 <tr>
20 <th>日付</th>
21 <th>メンバー</th>
22 <th>稼働時間(HR)</th>
23 <th>稼働単価(円)</th>
24 <th>日々の稼働コスト(円)</th>
25 </tr>
26 </thead>
27 <tbody>
28 <tr>
29 <td>2024-07-29</td>
30 <td>A</td>
31 <td>8</td>
32 <td>10000</td>
33 <td>80000</td>
34 </tr>
35 <tr>
36 <td>2024-07-29</td>
37 <td>B</td>
38 <td>6</td>
39 <td>12000</td>
40 <td>72000</td>
41 </tr>
42 <tr>
43 <td>2024-07-30</td>
44 <td>A</td>
45 <td>7</td>
46 <td>10000</td>
47 <td>70000</td>
48 </tr>
49 <tr>
```

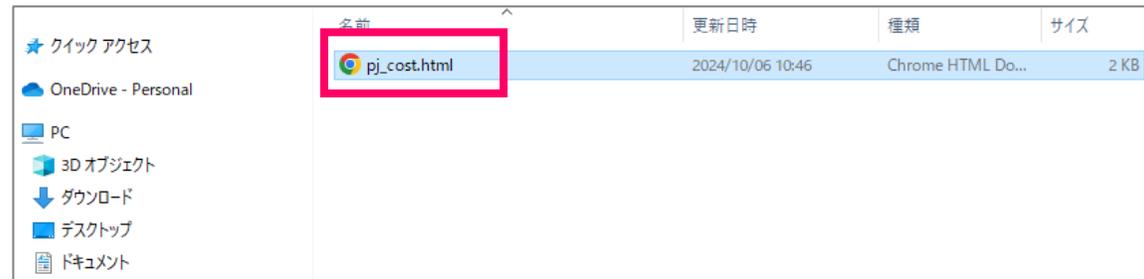
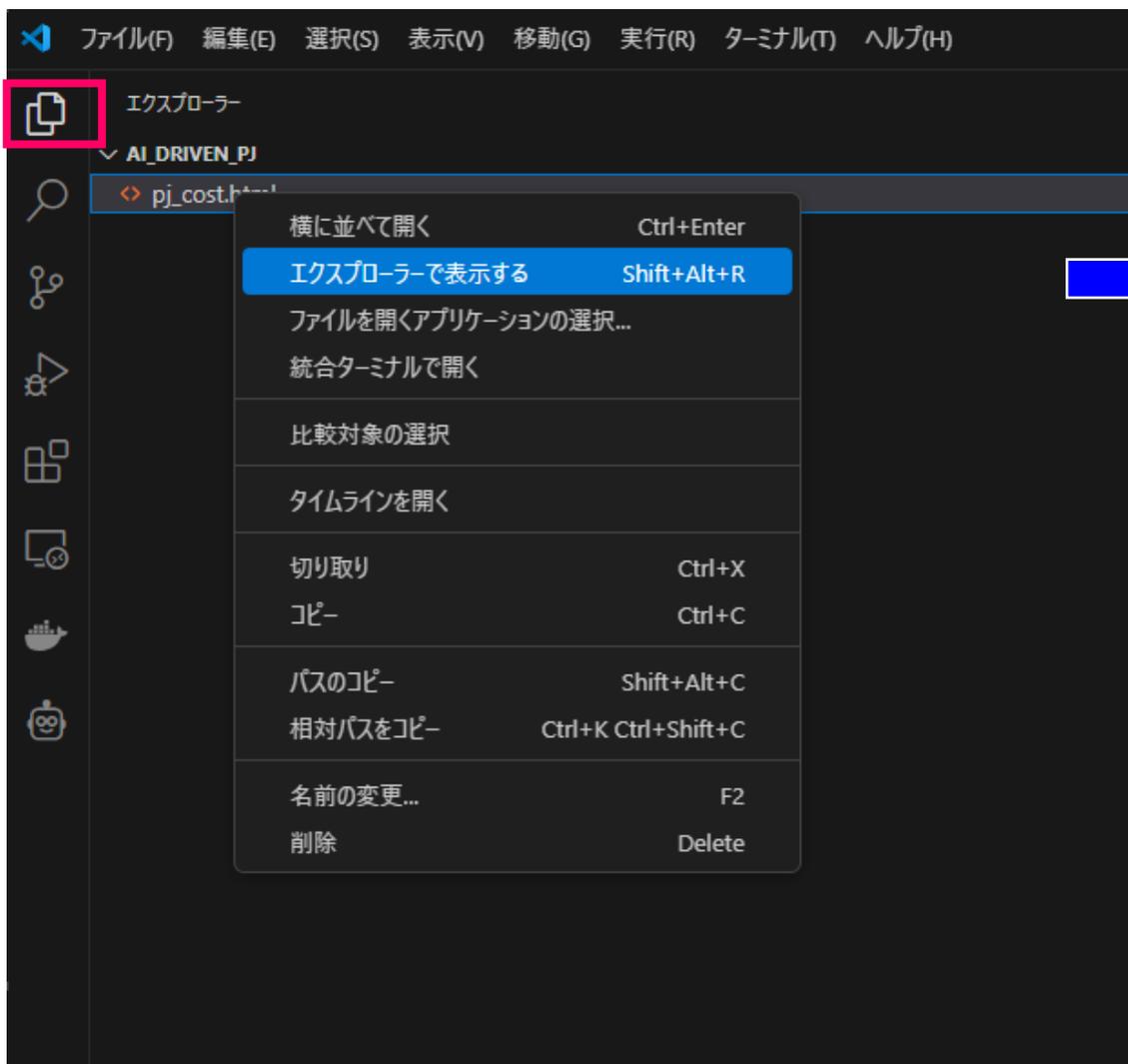
The rendered output shows a table with the following data:

日付	メンバー	稼働時間(HR)	稼働単価(円)	日々の稼働コスト(円)
2024-07-29	A	8	10000	80000
2024-07-29	B	6	12000	72000
2024-07-30	A	7	10000	70000
2024-07-30	B	8	12000	96000

The editor also shows the JavaScript code for calculating the daily cost and rendering the table. The task is completed, and the file 'pj_cost.html' has been saved.

作成したWEBアプリを動作させてみる

エクスプローラで開くと、作成したhtmlファイルが出来ている



作成したhtmlファイルを開くと⇒以下：

Web browser displaying the 'PJ稼働コスト管理表' (PJ Work Cost Management Table). A blue arrow points from the text above to the browser window.

日付	メンバー	稼働時間(HR)	稼働単価(円)	日々の稼働コスト(円)
2024-07-29	A	8	10000	80000
2024-07-29	B	6	12000	72000
2024-07-30	A	7	10000	70000
2024-07-30	B	8	12000	96000

日々の総稼働コスト: 318 K¥

WEB画面に、取り込んだサンプルデータが計算されて表示される！

作成したWEBアプリに入力機能を追加する



のアイコンからClaude Devに追加タスクをINPUTする

The screenshot shows the Claude Dev interface in VS Code. On the left, a task completion message is displayed, indicating that the project cost management table has been created. The message includes instructions on how to use the table and how to add input functionality. The main editor shows the HTML code for the table, which is currently empty. The table structure is as follows:

日付	メンバー	稼働時間 (HR)	稼働単価 (円)	日々の稼働コスト (円)
2024-07-29	A	8	10000	80000
2024-07-29	B	6	12000	72000
2024-07-30	A	7	10000	70000

Claude Devの実行結果

結果を確認しながら、Step By Step、ヒューマンインザループでアプリのブラッシュアップが出来る！

The screenshot shows the Claude Dev interface with the following components:

- Task Panel (Left):** Displays the task "# 役割" and provides a summary of the user's role as a front-end developer. It shows token usage (11,345 up, 4,040 down) and offers a "Save" button (highlighted in red) and a "Reject" button.
- Code Editor (Center):** Shows the original code for `pj_cost.html`. The code includes HTML structure, CSS styles, and JavaScript logic for calculating and saving project costs. A red box highlights the table data in the `<tbody>` section, showing the following rows:

日付	メンバー	稼働時間 (HR)	稼働単価 (円)	日々の稼働コスト (円)
2024-07-29	A	8	10000	80000
2024-07-29	B	6	12000	72000
2024-07-30	A	7	10000	70000
- Diff View (Right):** Shows the changes made by Claude. The diff highlights the addition of a new table row and the update of the table's `tbody` id to "data-table".

またSaveする

追加した機能の部分

バックエンドアプリの自動生成

Python Flaskとは？

Flask（フラスク）は、プログラミング言語Python用の、軽量なウェブアプリケーションフレームワークである。標準で提供する機能を最小限に保っているため、自身を「マイクロフレームワーク」と呼んでいる。Werkzeug WSGIツールキットとJinja2テンプレートエンジンを基に作られています。

フレームワークとは、動的なウェブサイト、ウェブアプリケーション、Webサービスの開発をサポートするために設計されたデータベースへのアクセスのためのライブラリやテンプレートエンジン（Webテンプレートも参照）、セッション管理を提供し、コードの再利用を促進させる仕組みから構成されます。

大規模開発向けにはDjango、小規模で軽量な開発ではFlaskがメジャーであり、開発の効率化に寄与します。

バックエンドアプリの自動生成では、Python Flaskのフレームワークを活用します。

Pythonのインストール

Windowsの環境変数に、Pythonのexeファイルを対応させて、コマンドプロンプトからすぐにPythonが実行出来て、また特権モードが使用できるようにして、PIPコマンド※が使えるようにする
(※様々な機能追加用コマンド)

Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-07 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	security	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 619
3.8	security	2019-10-14	2024-10	PEP 619

Python 3.12.7 (64-bit) Setup

Install Python 3.12.7 (64-bit)

Select **Install Now** to install Python with default settings, or choose **Customize** to enable or disable features.

Install Now

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ **Customize installation**
Choose location and features

Use admin privileges when installing py.exe
 Add python.exe to PATH

Cancel

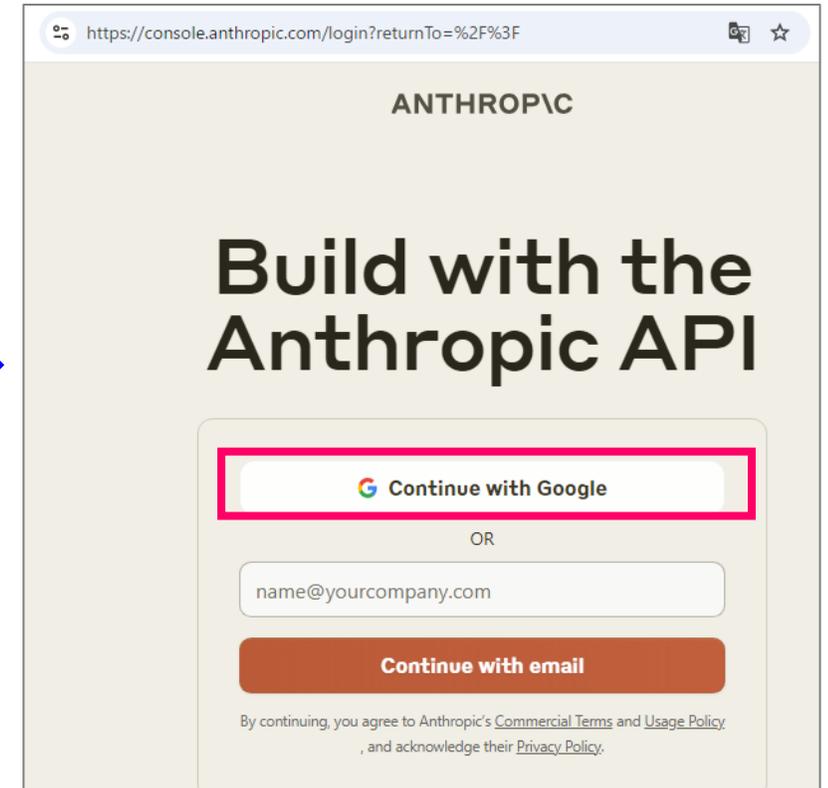
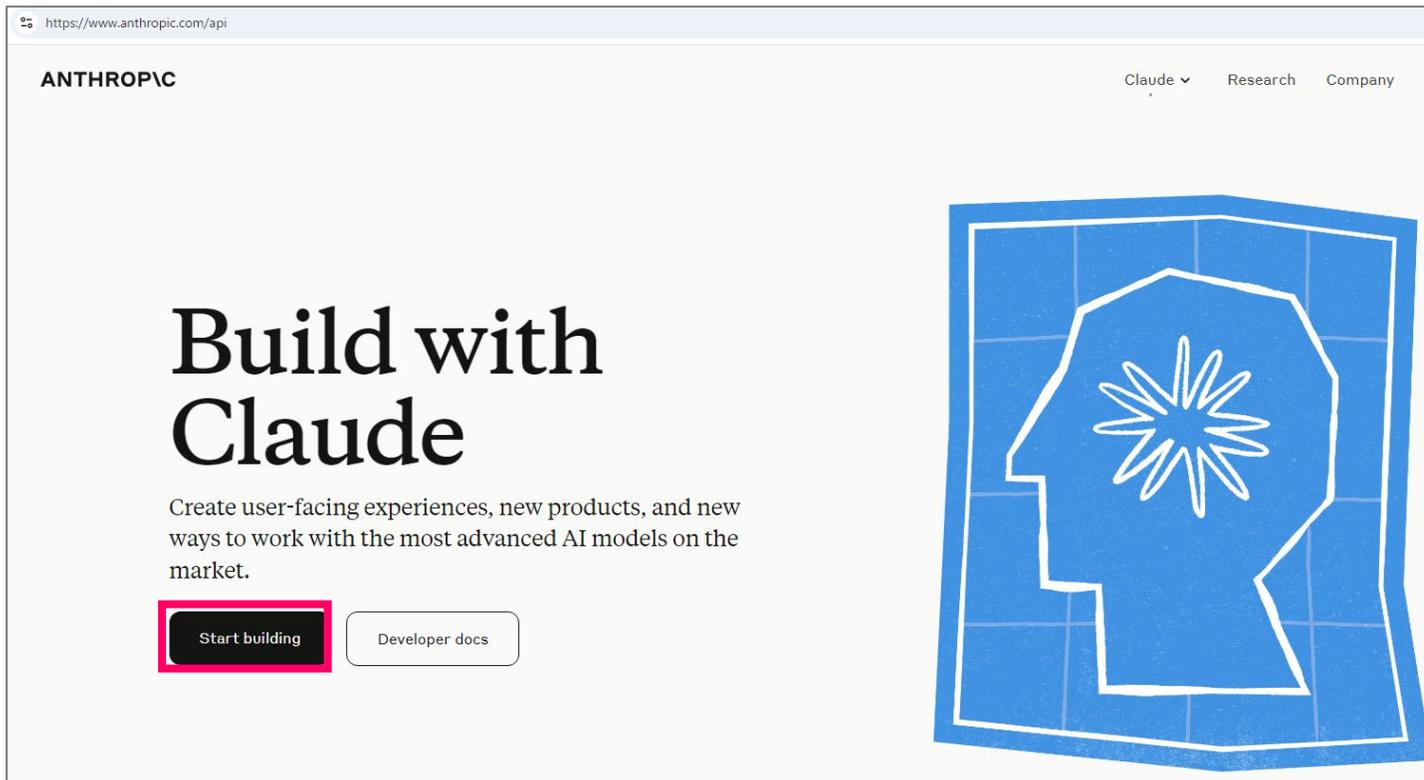
```
コマンドプロンプト - python
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>python
Python 3.12.7 (tags/v3.12.7:0b05ead, Oct 1 2024, 03:06:41) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more.
>>> print("Hello")
Hello
>>>
```

Anthropic Claude 3.5 Sonnetのインストール

フロントエンドの設計に続いて、バックエンドの設計も必要になる。ここではGemini Flash 1.5より高性能の生成AIが必要になるので、Anthropic Claude 3.5 Sonnetを利用します。APIキー取得の手順も示します。

<https://www.anthropic.com/api>

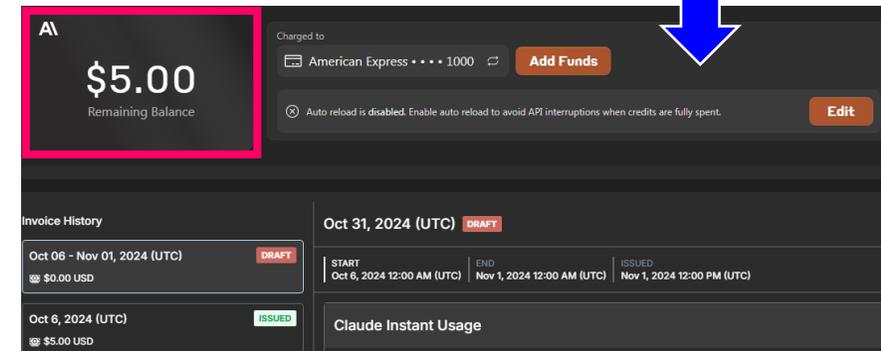
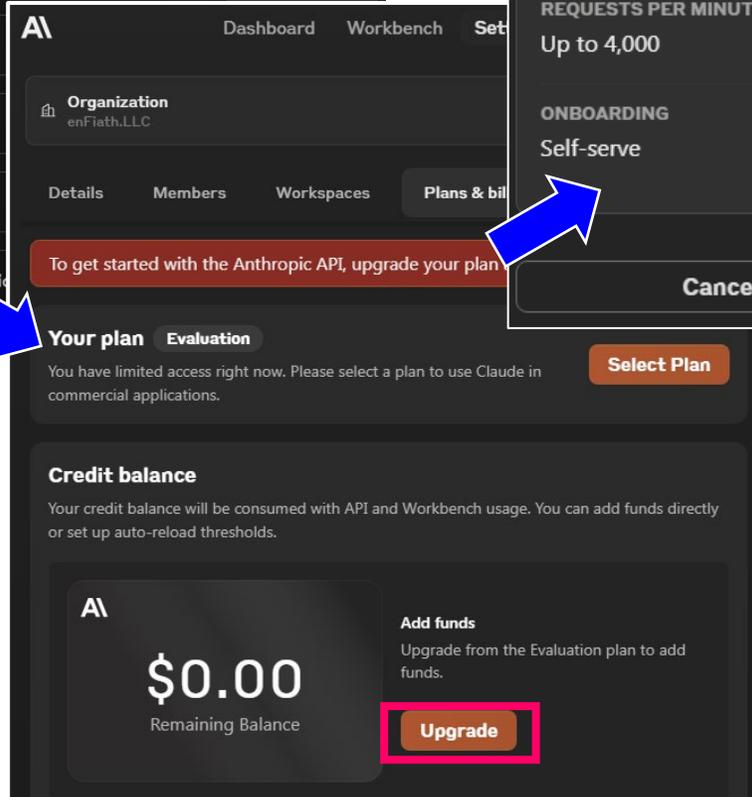
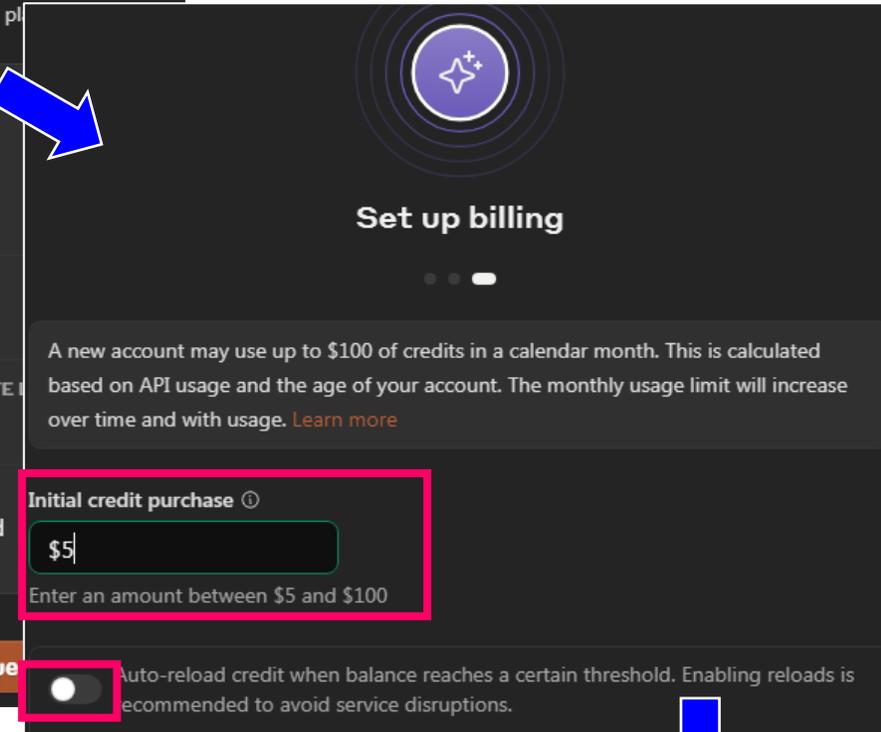
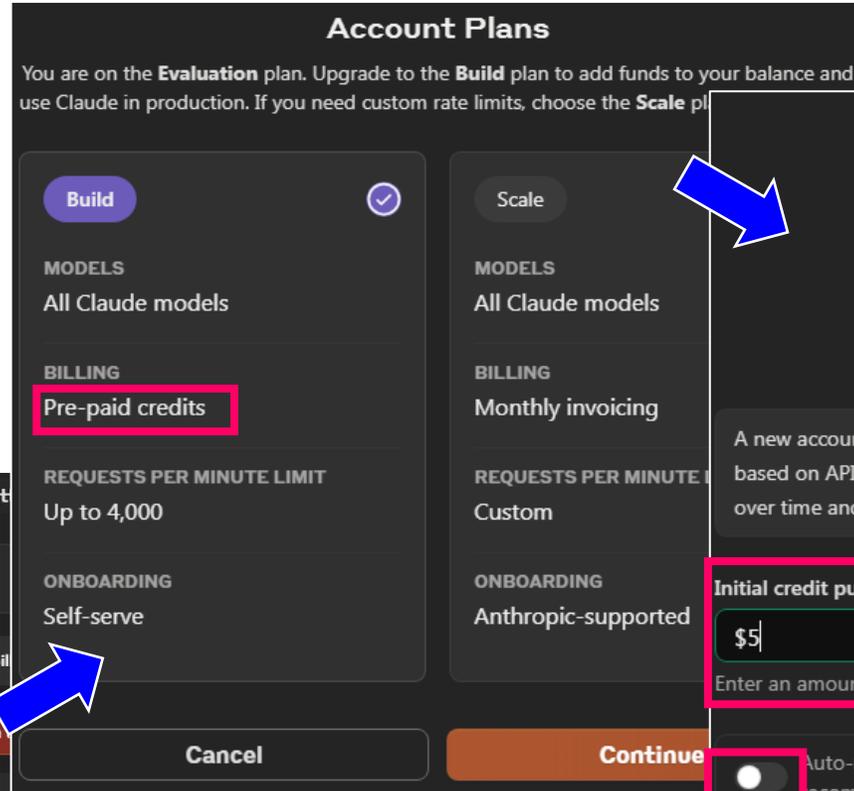
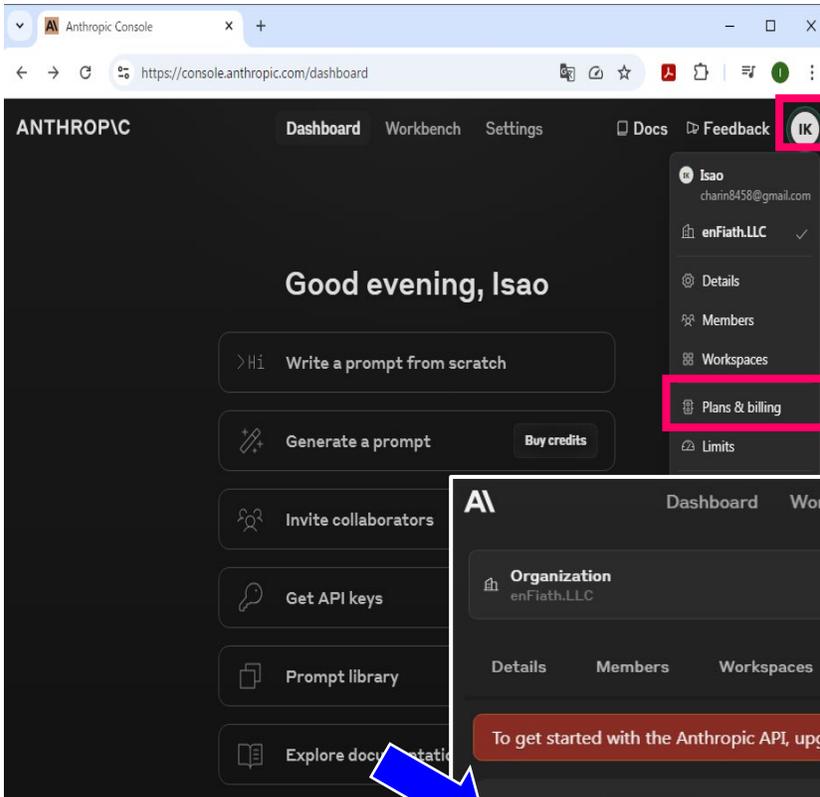


Googleで認証

Anthropic Claude 3.5 SonnetでのAPIキー取得①

プリペイドでチャージ

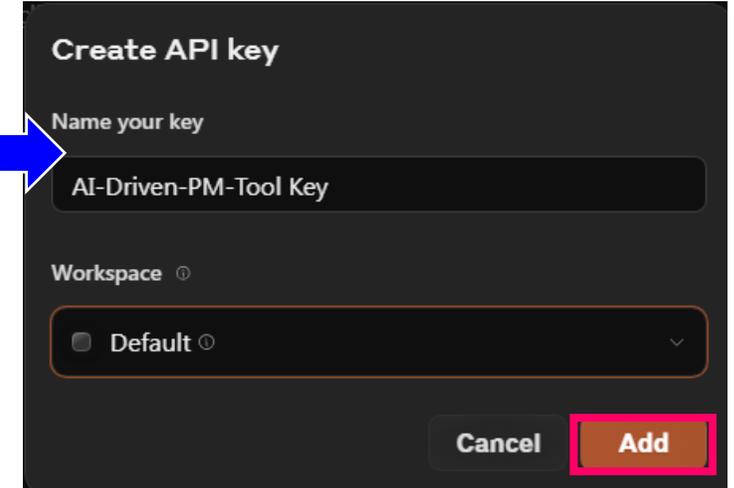
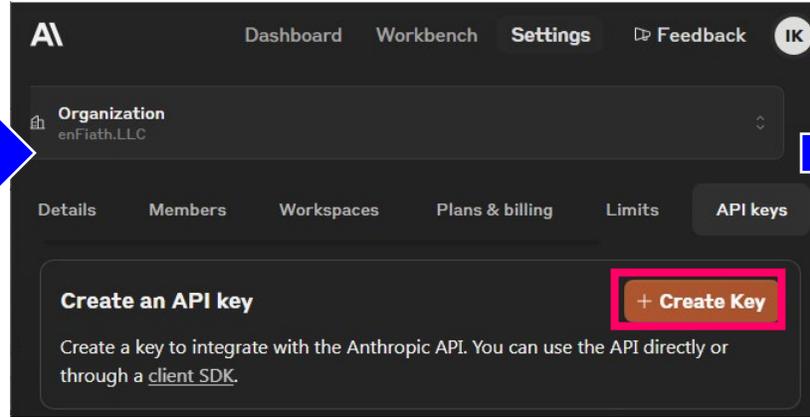
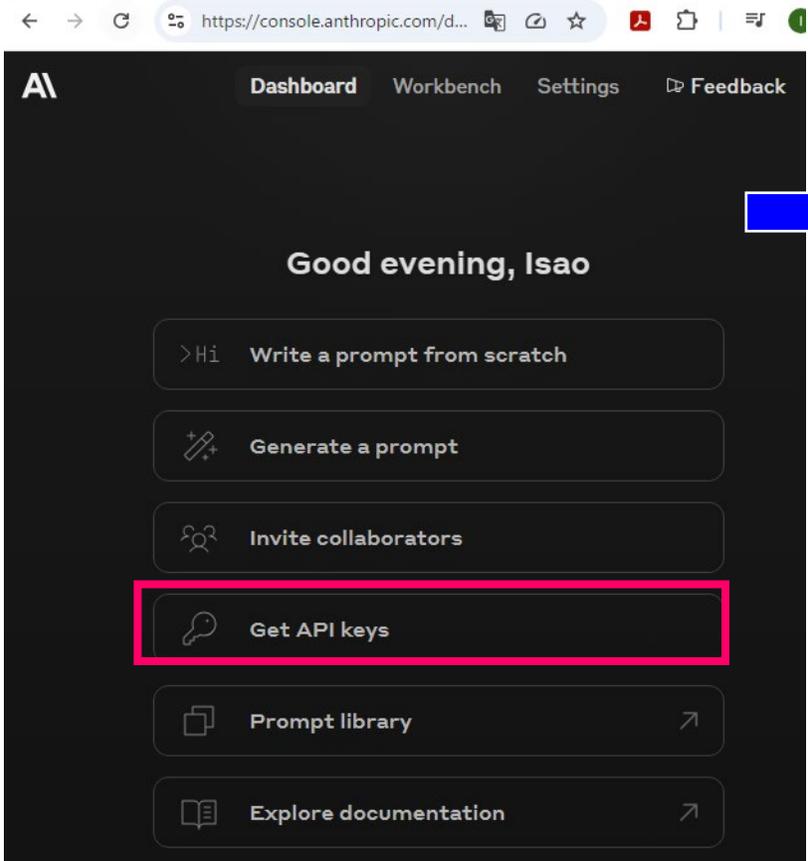
\$5 : この例では1回きりのプリペイド課金を選択



増額するならこの画面から



Anthropic Claude 3.5 SonnetでのAPIキー取得②



APIキー取得

Anthropic Claude 3.5 Sonnetで取得したAPIキーを設定

CLAUDE DEV

Settings

API Provider
Anthropic

Anthropic API Key
.....

Use custom base URL
This key is stored locally and only used to make API requests from this extension.

Model
claude-3-5-sonnet-20240620

✓ Supports images
✓ Supports prompt caching
Max output: 8,192 tokens
Input price: \$3.00/million tokens
Cache writes price: \$3.75/million tokens
Cache reads price: \$0.30/million tokens
Output price: \$15.00/million tokens

Custom Instructions

システムコマンド

・日本語で必ず応答してください。

These instructions are added to the end of the system prompt sent with every request.

Always allow read-only operations
When enabled, Claude will automatically read files, view directories, and inspect sites without requiring you to click the Allow button.

Done

VS CodeのClaude Devの画面でバックエンドタスクを割り当てる

The screenshot shows the VS Code interface with the Claude Dev extension. The left sidebar contains the Claude Dev chat window, which displays a welcome message and recent tasks. The main editor area shows the code for an application, including HTML and JavaScript. The code defines a data array and a function to render a table of costs.

Chat Window Content:

What can I do for you?

Thanks to Claude 3.5 Sonnet's agentic coding capabilities, I can handle complex software development tasks step-by-step. With tools that let me create & edit files, explore complex projects, and execute terminal commands (after you grant permission), I can assist you in ways that go beyond simple code completion or tech support.

RECENT TASKS

OCTOBER 6, 12:34 PM
役割
あなたはプロの経験豊富なフロントエンコーダーでHTML/CSS/Javascriptが得意です。...
Tokens: 113,189 / 47,771

OCTOBER 6, 12:05 PM
役割
あなたはプロの経験豊富なフロントエンコーダーでHTML/CSS/Javascriptが得意です。...
Tokens: 120,535 / 46,142

View all history

Type your task here (@ to add context)...

Code Editor Content (index.html):

```
1 <html>
2 <body>
32 <div>
33   <h2>日別コスト</h2>
34   <canvas id="daily-chart"></canvas>
35 </div>
36
37 <div>
38   <h2>月別コスト</h2>
39   <canvas id="monthly-chart"></canvas>
40 </div>
41
42 <div>
43   <h2>年別コスト</h2>
44   <canvas id="yearly-chart"></canvas>
45 </div>
46
47 <div>
48   <h2>月別コスト内訳 (円グラフ)</h2>
49   <canvas id="monthly-pie-chart-cost"></canvas>
50   <canvas id="monthly-pie-chart-hour"></canvas>
51 </div>
52
53 <script>
54 const data = [
55   { date: '2024-03-01', member: 'A', hours: 8, rate: 10000 },
56   { date: '2024-03-01', member: 'B', hours: 6, rate: 12000 },
57   { date: '2024-03-02', member: 'A', hours: 7, rate: 10000 },
58   { date: '2024-03-02', member: 'C', hours: 8, rate: 15000 },
59   { date: '2024-03-03', member: 'B', hours: 4, rate: 12000 },
60   { date: '2024-03-03', member: 'C', hours: 10, rate: 15000 },
61   { date: '2024-03-04', member: 'A', hours: 9, rate: 10000 },
62   { date: '2024-03-04', member: 'B', hours: 7, rate: 12000 },
63   { date: '2024-03-05', member: 'A', hours: 6, rate: 10000 },
64   { date: '2024-03-05', member: 'C', hours: 5, rate: 15000 },
65 ];
66
67 function renderTable(data) {
68   const tableBody = document.getElementById('data-table');
69   tableBody.innerHTML = ''; //既存データのクリア
70   data.forEach(item => {
71     const row = tableBody.insertRow();
72     row.insertCell().textContent = item.date;
73     row.insertCell().textContent = item.member;
74     row.insertCell().textContent = item.hours;
75     row.insertCell().textContent = item.rate;
76     row.insertCell().textContent = item.hours * item.rate;
77   });
78 }
```

バックエンドアプリ生成用のタスクのPrompt

バックエンドアプリを生成するPromptは、機能コンポーネント毎に丁寧に他との関係性を整合とる必要があるため、正確性が要求されてかつ複雑になる。これをAnthropicのPrompt生成機能により実現する。以下は、その前段階として、AnthropicのPrompt生成機能に要求するタスクを伝えるためのPrompt。

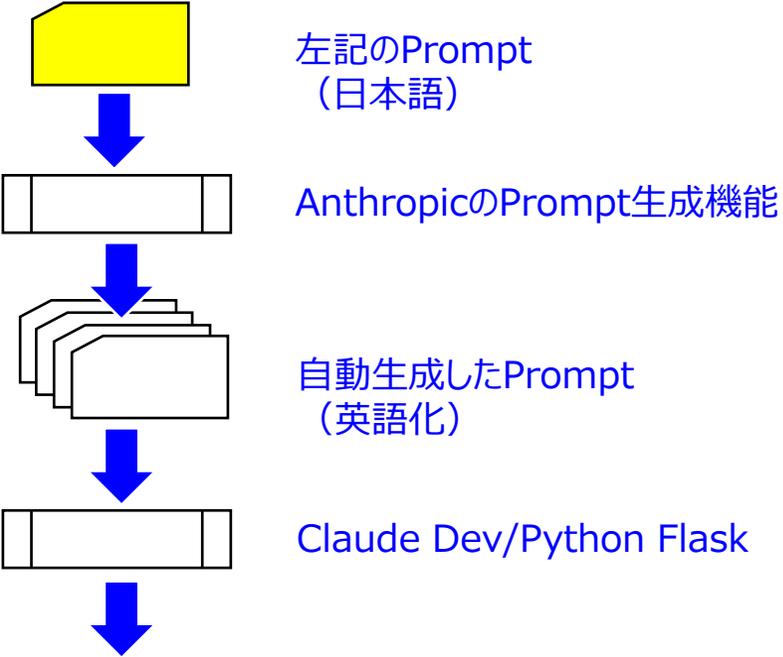
```
@index.html
これからindex.htmlをpython flaskとsqliteを利用してWebアプリケーション化します。

/アクセスで、index.htmlを表示し、index.htmlテーブル(日付(Date)・メンバ(Str)・稼働
単価(int)・稼働工数(int) ) の内容をsqliteを利用して編集・保存できるようにしてください。

構成は以下のように修正してください。
app.py
templates/index.html

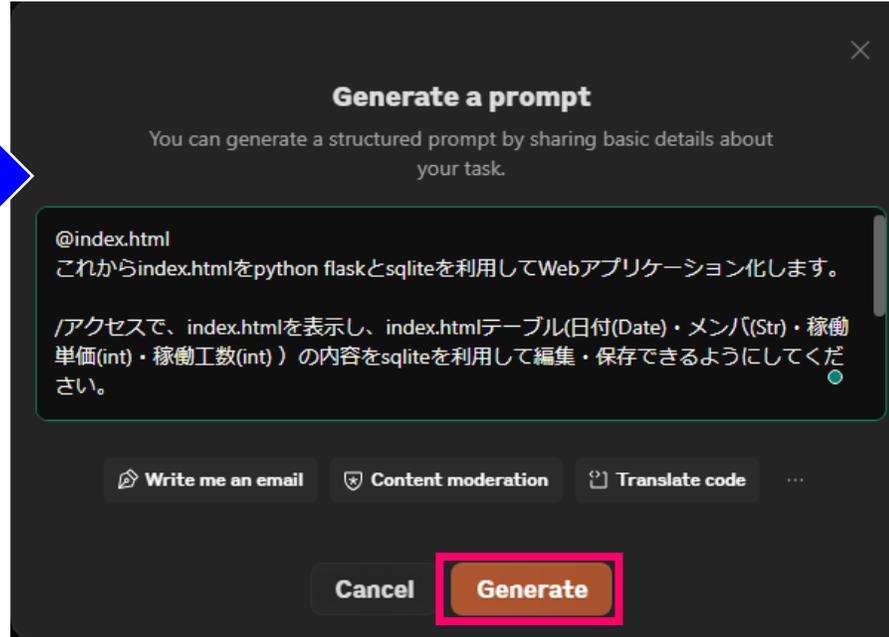
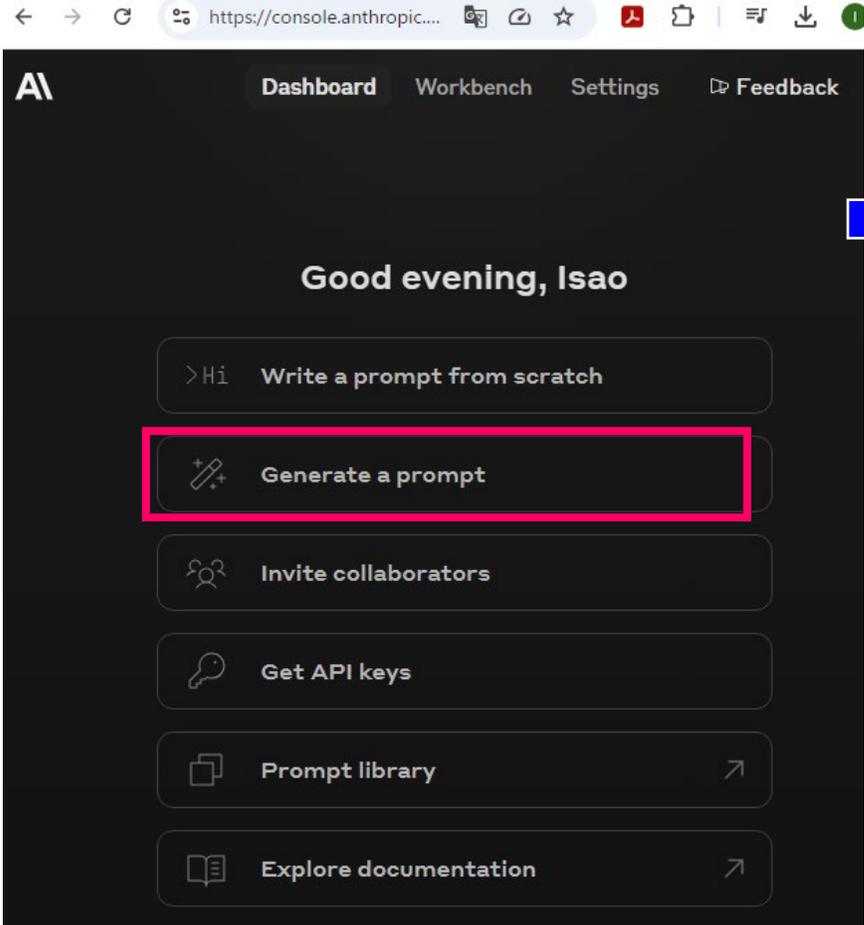
必要なモジュールのセットアップ方法も提示してください。

-----
```



フロントエンドに合わせて、Python Flaskのフレームワークを利用したバックエンドアプリ (DBを含む)

Anthropic ClaudeでPromptを自動生成する



Anthropic Claudeで自動生成されたPrompt①

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The application should display the content of index.html when accessed at the root URL ("/") and allow editing and saving of the table data using SQLite.

Here's the content of the original index.html file:

```
<index_html>
{{INDEX_HTML}}
</index_html>
```

To create the Flask application with SQLite integration, follow these steps:

1. Create a new directory for your project and set up the following structure:
 - app.py
 - templates/
 - index.html
 - instance/
 - database.db (This will be created automatically by Flask)
2. Set up the SQLite database:
 - In app.py, use Flask-SQLAlchemy to define a model for the table data
 - Create functions to initialize the database and add sample data
3. Modify app.py to include the following:
 - Import necessary modules (Flask, Flask-SQLAlchemy, etc.)
 - Configure the Flask application and SQLAlchemy
 - Define routes for displaying, adding, editing, and deleting data
 - Implement CRUD operations using SQLAlchemy

Anthropic Claudeで自動生成されたPrompt②

4. Modify templates/index.html:
 - Add form elements for adding and editing data
 - Include JavaScript for handling AJAX requests to update the table dynamically

5. Here's a basic structure for app.py:

```
```python
from flask import Flask, render_template, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
db = SQLAlchemy(app)

class TableData(db.Model):
 id = db.Column(db.Integer, primary_key=True)
 date = db.Column(db.Date, nullable=False)
 member = db.Column(db.String(100), nullable=False)
 unit_price = db.Column(db.Integer, nullable=False)
 work_hours = db.Column(db.Integer, nullable=False)

@app.route('/')
def index():
 data = TableData.query.all()
 return render_template('index.html', data=data)
```

# Anthropic Claudeで自動生成されたPrompt③

```
@app.route('/add', methods=['POST'])
def add_data():
 # Implement adding new data
 pass

@app.route('/edit', methods=['POST'])
def edit_data():
 # Implement editing existing data
 pass

@app.route('/delete', methods=['POST'])
def delete_data():
 # Implement deleting data
 pass

if __name__ == '__main__':
 with app.app_context():
 db.create_all()
 app.run(debug=True)
` ``
```

# Anthropic Claudeで自動生成されたPrompt④

6. Modify templates/index.html to include forms and JavaScript for AJAX requests. Here's a basic structure:

```
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Data Table</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <h1>Data Table</h1>
  <table id="data-table">
    <!-- Table headers and data rows -->
  </table>

  <form id="add-form">
    <!-- Form fields for adding new data -->
  </form>

  <script>
    // JavaScript for handling AJAX requests
  </script>
</body>
</html>
```
```

## Anthropic Claudeで自動生成されたPrompt⑤

7. To set up the required modules, run the following commands in your terminal:

```
```\n\npip install flask flask-sqlalchemy\n```\n
```

8. To run the application:

- Open a terminal in your project directory
- Run the command: `python app.py`
- Open a web browser and navigate to `http://localhost:5000`

Remember to implement proper error handling, input validation, and security measures in your final application. This basic structure provides a starting point for creating a Flask web application with SQLite integration based on the given index.html file.

Claude Devの画面でバックエンドタスクを割り当てる①

The screenshot shows the Claude Dev interface. On the left, there's a sidebar with navigation icons and a chat window. The chat window contains the following text:

What can I do for you?

Thanks to Claude 3.5 Sonnet's agentic coding capabilities, I can handle complex software development tasks step-by-step. With tools that let me create & edit files, explore complex projects, and execute terminal commands (after you grant permission), I can assist you in ways that go beyond simple code completion or tech support.

Anthropic Claudeで自動生成したPromptを貼る
ただし冒頭に、フロントエンジンで作成した @index.html
をメンション (@を付けて) 追加しておくこと!

@index.html

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The application should display the content of index.html when accessed at the root URL ("/") and allow editing and saving of the table data using SQLite.

Here's the content of the original index.html file:

```
<index_html>
{{INDEX_HTML}}
</index_html>
```

つづく……

The code editor on the right shows the content of index.html:

```
<html>
<body>
<div>
<h2>日別コスト</h2>
<canvas id="daily-chart"></canvas>
</div>
<div>
<h2>月別コスト</h2>
<canvas id="monthly-chart"></canvas>
</div>
<div>
<h2>年別コスト</h2>
<canvas id="yearly-chart"></canvas>
</div>
<div>
<h2>月別コスト内訳(円グラフ)</h2>
<canvas id="monthly-pie-chart-cost"></canvas>
<canvas id="monthly-pie-chart-hour"></canvas>
</div>
<script>
const data = [
  { date: '2024-03-01', member: 'A', hours: 8, rate: 10000 },
  { date: '2024-03-01', member: 'B', hours: 6, rate: 12000 },
  { date: '2024-03-02', member: 'A', hours: 7, rate: 10000 },
  { date: '2024-03-02', member: 'C', hours: 8, rate: 15000 },
  { date: '2024-03-03', member: 'B', hours: 4, rate: 12000 },
  { date: '2024-03-03', member: 'C', hours: 10, rate: 15000 },
  { date: '2024-03-04', member: 'A', hours: 9, rate: 10000 },
  { date: '2024-03-04', member: 'B', hours: 7, rate: 12000 },
  { date: '2024-03-05', member: 'A', hours: 6, rate: 10000 },
  { date: '2024-03-05', member: 'C', hours: 5, rate: 15000 },
];

function renderTable(data) {
  const tableBody = document.getElementById('data-table');
  tableBody.innerHTML = ''; // 既存データのクリア
  data.forEach(item => {
    const row = tableBody.insertRow();
    row.insertCell().textContent = item.date;
    row.insertCell().textContent = item.member;
    row.insertCell().textContent = item.hours;
    row.insertCell().textContent = item.rate;
    row.insertCell().textContent = item.hours * item.rate;
  });
}
```

Claude Devの画面でバックエンドタスクを割り当てる②

CLAUDE DEV

Task
@index.html

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica: [See more](#)

Tokens: ↑ 4 ↓ 219
Cache: 🗑 +6,011 → 0
API Cost: \$0.0258

EXPORT

✓ API Request \$0.0258

タスクを分析し、必要な手順を整理します：

- プロジェクト構造の作成
- app.pyの作成とSQLiteデータベースのセットアップ
- templates/index.htmlの修正
- 必要なモジュールのインストール

まず、プロジェクト構造を作成するために、list_filesツールを使用して現在のディレクトリ構造を確認し、必要なディレクトリとファイルを作成します。

📁 Claude wants to recursively view all files in this directory:

PJ_Cost_Calculation

タスクを分析し、必要な手順を整理します：

プロジェクト構造の作成
app.pyの作成とSQLiteデータベースのセットアップ
templates/index.htmlの修正
必要なモジュールのインストール

まず、プロジェクト構造を作成するために、list_filesツールを使用して現在のディレクトリ構造を確認し、必要なディレクトリとファイルを作成します。

Approve

Reject

Type a message (@ to add context)...



CLAUDE DEV

Task
@index.html

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica: [See more](#)

Tokens: ↑ 8 ↓ 424
Cache: 🗑 +6,283 → 6,011
API Cost: \$0.0317

EXPORT

✓ API Request \$0.0258

タスクを分析し、必要な手順を整理します：

- プロジェクト構造の作成
- app.pyの作成とSQLiteデータベースのセットアップ
- templates/index.htmlの修正
- 必要なモジュールのインストール

まず、プロジェクト構造を作成するために、list_filesツールを使用して現在のディレクトリ構造を確認し、必要なディレクトリとファイルを作成します。

📁 Claude wants to recursively view all files in this directory:

PJ_Cost_Calculation

✓ API Request \$0.0059

現在のディレクトリ構造を確認しました。必要なディレクトリとファイルを作成する必要があります。以下の手順で進めます：

- templatesディレクトリを作成し、その中にindex.htmlを移動
- app.pyファイルを作成
- app.pyの内容を実装
- templates/index.htmlの内容を修正

まず、templatesディレクトリを作成し、index.htmlを移動させます。

📄 Claude wants to execute this command:

```
mkdir templates && move index.html templates\
```

Run Command

Reject

Type a message (@ to add context)...

Claude Devの画面でバックエンドタスクを割り当てる③

エラー発生:

Task
@index.html

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica [See more](#)

Tokens: ↑ 12 ↓ 553
Cache: 0 +6,708 → 12,294
API Cost: \$0.0372

EXPORT

- プロジェクト構造の作成
- app.pyの作成とSQLiteデータベースのセットアップ
- templates/index.htmlの修正
- 必要なモジュールのインストール

まず、プロジェクト構造を作成するために、list_filesツールを使用して現在のディレクトリ構造を確認し、必要なディレクトリとファイルを作成します。

📁 Claude wants to recursively view all files in this directory:

PJ_Cost_Calculation

✓ API Request \$0.0059

現在のディレクトリ構造を確認しました。必要なディレクトリとファイルを作成する必要があります。以下の手順に進めます:

- templatesディレクトリを作成し、その中にindex.htmlを移動
- app.pyファイルを作成
- app.pyの内容を実装
- templates/index.htmlの内容を修正

まず、templatesディレクトリを作成し、index.htmlを移動させます。

📄 Claude wants to execute this command:

```
mkdir templates && move index.html templates\
```

> Command Output

✓ API Request \$0.0054

申し訳ありません。Windows環境でのコマンド実行に問題がありました。コマンドを分割して実行し直します。まず、templatesディレクトリを作成し、次にindex.htmlを移動させます。

📄 Claude wants to execute this command:

```
mkdir templates
```

Run Command

Reject

Type a message (@ to add context)...

```
index.html X
index.html > html > body > script
2 <html>
14 <body>
32 <div>
33 <h2>日別コスト</h2>
34 <canvas id="daily-chart"></canvas>
35 </div>
36
37 <div>
38 <h2>月別コスト</h2>
39 <canvas id="monthly-chart"></canvas>
40 </div>
41
42 <div>
43 <h2>年別コスト</h2>
44 <canvas id="yearly-chart"></canvas>
45 </div>
46
47 <div>
48 <h2>月別コスト内訳(円グラフ)</h2>
49 <canvas id="monthly-pie-chart-cost"></canvas>
50 <canvas id="monthly-pie-chart-hour"></canvas>
51 </div>
52
53 <script>
54 const data = [
55   { date: '2024-03-01', member: 'A', hours: 8, rate: 10000 },
56   { date: '2024-03-01', member: 'B', hours: 6, rate: 12000 },
57   { date: '2024-03-02', member: 'A', hours: 7, rate: 10000 },
58   { date: '2024-03-02', member: 'C', hours: 8, rate: 15000 },
59   { date: '2024-03-03', member: 'B', hours: 4, rate: 12000 },
60   { date: '2024-03-03', member: 'C', hours: 10, rate: 15000 },
61   { date: '2024-03-04', member: 'A', hours: 9, rate: 10000 },
62   { date: '2024-03-04', member: 'B', hours: 7, rate: 12000 },
63   { date: '2024-03-05', member: 'A', hours: 6, rate: 10000 },
64   { date: '2024-03-05', member: 'C', hours: 5, rate: 15000 },

```

問題 出力 デバッグコンソール ターミナル ポート

```
PS E:\000_Udemy\0_VS_Code\PJ_Cost_Calculation> mkdir templates && move index.html templates\
発生場所 行:1 文字:17
+ mkdir templates && move index.html templates\
+ ~~~
トークン '&&' は、このバージョンでは有効なステートメント区切りではありません。
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : InvalidEndOfLine
```

PS E:\000_Udemy\0_VS_Code\PJ_Cost_Calculation> |

行 184, 列 10 スペース:4 UTF-8 CRLF HTML

Claude Devの画面でバックエンドタスクを割り当てる④

CLAUDE DEV

Task
@index.html

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica [See more](#)

Tokens: ↑ 16 ↓ 660
Cache: 🗑️ +7,016 → 19,002
API Cost: \$0.0420 EXPORT

✓ API Request \$0.0059

現在のディレクトリ構造を確認しました。必要なディレクトリとファイルを作成する必要があります。以下の手順で進めます：

1. templatesディレクトリを作成し、その中にindex.htmlを移動
2. app.pyファイルを作成
3. app.pyの内容を実装
4. templates/index.htmlの内容を修正

まず、templatesディレクトリを作成し、index.htmlを移動させます。

☒ Claude wants to execute this command:

```
mkdir templates && move index.html templates\
```

> Command Output

✓ API Request \$0.0054

申し訳ありません。Windows環境でのコマンド実行に問題がありました。コマンドを分割して実行し直します。まず、templatesディレクトリを作成し、次にindex.htmlを移動させます。

☒ Claude wants to execute this command:

```
mkdir templates
```

> Command Output

✓ API Request \$0.0048

templatesディレクトリが正常に作成されました。次に、index.htmlファイルをtemplatesディレクトリに移動させます。

☒ Claude wants to execute this command:

```
move index.html templates\
```

Run Command Reject

Type a message (@ to add context)...

Claude Devの画面でバックエンドタスクを割り当てる⑤

The screenshot displays the Claude Dev interface with a task pane on the left and a code editor on the right. The task pane shows a sequence of API requests and commands:

- Task:** @index.html. Description: "You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica See more". Tokens: 20 ↓, 1,505. Cache: +14,217 → 19,002. API Cost: \$0.0816.
- API Request \$0.0054:** Command: `mkdir templates && move index.html templates\`. Message: "申し訳ありません。Windows環境でのコマンド実行に問題がありました。コマンドを分割して実行し直します。まず、templatesディレクトリを作成し、次にindex.htmlを移動させます。"
- API Request \$0.0048:** Command: `mkdir templates`. Message: "templatesディレクトリが正常に作成されました。次に、index.htmlファイルをtemplatesディレクトリに移動させます。"
- API Request \$0.0397:** Command: `move index.html templates\`. Message: "index.htmlファイルがtemplatesディレクトリに正常に移動されたようです。次に、app.pyファイルを作成し、必要なコードを実装します。その後、templates/index.htmlの内容を修正します。まず、app.pyファイルを作成し、基本的な構造を実装します。"

The code editor shows the following Python code for `app.py`:

```
1+ from flask import Flask, render_template, request, json
2+ from flask_sqlalchemy import SQLAlchemy
3+ from datetime import datetime
4+
5+ app = Flask(__name__)
6+ app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data
7+ db = SQLAlchemy(app)
8+
9+ class TableData(db.Model):
10+     id = db.Column(db.Integer, primary_key=True)
11+     date = db.Column(db.Date, nullable=False)
12+     member = db.Column(db.String(100), nullable=False)
13+     unit_price = db.Column(db.Integer, nullable=False)
14+     work_hours = db.Column(db.Integer, nullable=False)
15+
16+ @app.route('/')
17+ def index():
18+     data = TableData.query.all()
19+     return render_template('index.html', data=data)
20+
21+ @app.route('/add', methods=['POST'])
22+ def add_data():
23+     date = datetime.strptime(request.form['date'], '%Y-%m-%d')
24+     member = request.form['member']
25+     unit_price = int(request.form['unit_price'])
26+     work_hours = int(request.form['work_hours'])
27+
28+     new_data = TableData(date=date, member=member, unit
29+ db.session.add(new_data)
30+ db.session.commit()
31+
32+     return jsonify({'success': True, 'id': new_data.id})
33+
34+ @app.route('/edit', methods=['POST'])
35+ def edit_data():
```

The terminal at the bottom shows the execution of the commands:

```
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> mkdir templates
ディレクトリ: E:\000_Udemy\0_VS Code\PJ_Cost_Calculation
Mode                LastWriteTime         Length Name
----                -
d-----          2024/10/07   0:35         templates
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> move index.html templates\
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> []
```

A notification at the bottom right prompts to install the Python language extension: "Python 言語 用のおすすめ拡張機能 'Python' 拡張機能 提供元: Microsoft をインストールしますか? インストール 推奨事項の表示".

Claude Devの画面でバックエンドタスクを割り当てる⑥

Task
@index.html
You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica See more
Tokens: ↑ 24 ↓ 1,650
Cache: 🗑 +22,323 → 19,002
API Cost: \$0.1142 EXPORT

Claude wants to execute this command:
mkdir templates
> Command Output

✓ **API Request** \$0.0048
templatesディレクトリが正常に作成されました。次に、index.htmlファイルをtemplatesディレクトリに移動させます。

Claude wants to execute this command:
move index.html templates\
> Command Output

✓ **API Request** \$0.0397
index.htmlファイルがtemplatesディレクトリに正常に移動されたようです。次に、app.pyファイルを作成し、必要なコードを実装します。その後、templates/index.htmlの内容を修正します。
まず、app.pyファイルを作成し、基本的な構造を実装します。

Claude wants to create a new file:
app.py

✓ **API Request** \$0.0326
app.pyファイルが正常に作成されました。次に、templates/index.htmlファイルの内容を修正して、Flaskアプリケーションと連携できるようにします。まず、現在のindex.htmlの内容を確認し、必要な修正を加えます。

Claude wants to read this file:
templates/index.html

Approve Reject

Type a message (@ to add context)...

```
app.py > ...
1 from flask import Flask, render_template, request, jsonify
2 from flask_sqlalchemy import SQLAlchemy
3 from datetime import datetime
4
5 app = Flask(__name__)
6 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
7 db = SQLAlchemy(app)
8
9 class TableData(db.Model):
10     id = db.Column(db.Integer, primary_key=True)
11     date = db.Column(db.Date, nullable=False)
12     member = db.Column(db.String(100), nullable=False)
13     unit_price = db.Column(db.Integer, nullable=False)
14     work_hours = db.Column(db.Integer, nullable=False)
15
16 @app.route('/')
17 def index():
18     data = TableData.query.all()
19     return render_template('index.html', data=data)
20
21 @app.route('/add', methods=['POST'])
22 def add_data():
23     date = datetime.strptime(request.form['date'], '%Y-%m-%d').date()
24     member = request.form['member']
25     unit_price = int(request.form['unit_price'])
26     work_hours = int(request.form['work_hours'])
27
28     new_data = TableData(date=date, member=member, unit_price=unit_price, work_hours=work_hours)
29     db.session.add(new_data)
30     db.session.commit()
31
32     return jsonify({'success': True, 'id': new_data.id})
33
34 @app.route('/edit', methods=['POST'])
35 def edit_data():
```

問題 2 出力 デバッグコンソール ターミナル ポート Claude Dev + ▾ □ 🗑 ... ^ ▾ ×

```
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> mkdir templates

ディレクトリ: E:\000_Udemy\0_VS Code\PJ_Cost_Calculation

Mode                LastWriteTime         Length Name
----                -
d-----          2024/10/07    0:35             templates

PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> move index.html templates\
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation>
```

行 1、列 1 スペース: 4 UTF-8 CRLF Python 3.12.7 64-bit

Claude Devの画面でバックエンドタスクを割り当てる⑦

CLAUDE DEV

Task

@index.html

You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica See more

Tokens: ↑ 28 ↓ 4,490

Cache: 0 +24,728 → 27,108

API Cost: \$0.1683

EXPORT

Command Output

API Request \$0.0397

index.htmlファイルがtemplatesディレクトリに正常に移動されたようです。次に、app.pyファイルを作成し、必要なコードを実装します。その後、templates/index.htmlの内容を修正します。

まず、app.pyファイルを作成し、基本的な構造を実装します。

Claude wants to create a new file:

app.py

API Request \$0.0326

app.pyファイルが正常に作成されました。次に、templates/index.htmlファイルの内容を修正して、Flaskアプリケーションと連携できるようにします。まず、現在のindex.htmlの内容を確認し、必要な修正を加えます。

Claude wants to read this file:

templates/index.html

API Request \$0.0541

index.htmlファイルの内容を確認しました。Flaskアプリケーションと連携するために、以下の修正が必要です:

- データをサーバーサイドから取得するように変更
- データの追加、編集、削除機能を実装
- AJAXリクエストを使用してサーバーと通信
- チャートの描画をサーバーから取得したデータで行うように修正

これらの変更を加えたindex.htmlファイルを作成します。

Claude wants to edit this file:

templates/index.html

Save

Reject

Type a message (@ to add context)...

```
templates > index.html > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>PJ稼働コスト管理表</title>
5 <script src="https://cdn.jsdelivr.net/npm/chart.js" ></script>
6 </head>
7 <style>
8 body { font-family: sans-serif; }
9 table { width: 100%; border-collapse: collapse; }
10 th, td { border: 1px solid #ddd; padding: 8px; text-align: center; }
11 th { background-color: #f2f2f2; }
12 canvas { max-width: 100%; }
13 </style>
14 </head>
15 <body>
16 <h1>PJ稼働コスト管理表</h1>
17
18 <table>
19 <thead>
20 <tr>
21 <th>日付</th>
22 <th>メンバー</th>
23 <th>稼働時間(HR)</th>
24 <th>稼働単価(円)</th>
25 <th>日別コスト(円)</th>
26 </tr>
27 </thead>
28 <tbody id="data-table">
29 <!-- データはJavaScriptで動的に追加 -->
30 </tbody>
31 </table>
32
33
34 <h2>データ追加</h2>
35 <form id="add-form">
```

問題 2 出力 デバッグコンソール ターミナル ポート

PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> mkdir templates

ディレクトリ: E:\000_Udemy\0_VS Code\PJ_Cost_Calculation

| Mode | LastWriteTime | Length | Name |
|-------|-----------------|--------|-----------|
| d---- | 2024/10/07 0:35 | | templates |

PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> move index.html templates\
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation>

行 1、列 1 スペース: 2 UTF-8 CRLF HTML

Claude Devの画面でバックエンドタスクを割り当てる⑧

The screenshot displays the Claude Dev interface with a task completion notification on the left and a code editor on the right.

Task Completion Notification:

- Task:** @index.html
- Description:** You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica. [See more](#)
- Tokens:** ↑ 32 ↓ 5,524
- Cache:** 🗑️ +27,638 → 37,619
- API Cost:** \$0.1979
- EXPORT** button

Task Completed:

Flask Web アプリケーションの作成が完了しました。以下の手順でアプリケーションを実行できます:

- 必要なモジュールをインストール:
`pip install flask flask-sqlalchemy`
- アプリケーションを実行:
`python app.py`
- ブラウザで `http://localhost:5000` にアクセスしてアプリケーションを確認してください。

アプリケーションには以下の機能があります:

- データの表示、追加、編集、削除
- 日別、月別、年別のコストグラフ
- 月別コスト内訳の円グラフ

初回実行時はデータベースが空なので、サンプルデータを追加してグラフの動作を確認してください。エラーが発生した場合は、ターミナルのエラーメッセージを確認し、必要に応じて修正を行ってください。

Claude wants to execute this command:

```
python app.py
```

Run Command button is highlighted with a red box.

Code Editor:

```
templates > index.html > ...
1 |!DOCTYPE html|
2 |<html|
3 |<head|
4 |<title>PJ稼働コスト管理表</title>
5 |<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
6 |<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
7 |<style>
8 |body { font-family: sans-serif; }
9 |table { width: 100%; border-collapse: collapse; }
10|th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
11|th { background-color: #f2f2f2; }
12|canvas { max-width: 100%; }
13|</style>
14|</head>
15|<body>
16|<h1>PJ稼働コスト管理表</h1>
17|
18|<table>
19|  <thead>
20|    <tr>
21|      <th>日付</th>
22|      <th>メンバー</th>
23|      <th>稼働時間(HR)</th>
24|      <th>稼働単価(円)</th>
25|      <th>日別コスト(円)</th>
26|      <th>操作</th>
27|    </tr>
28|  </thead>
29|  <tbody id="data-table">
30|    <!-- データはJavaScriptで動的に追加 -->
31|  </tbody>
32|</table>
33|
34|<h2>データ追加</h2>
35|<form id="add-form">
```

Terminal:

```
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> mkdir templates

ディレクトリ: E:\000_Udemy\0_VS Code\PJ_Cost_Calculation

Mode                LastWriteTime         Length Name
----                -
d-----            2024/10/07   0:35         templates

PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> move index.html templates\
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> |
```

Claude Devの画面でバックエンドタスクを割り当てる⑨

The screenshot displays the Claude Dev interface with a task completion notification on the left, a code editor in the center, and a terminal window at the bottom.

Task Completion Notification:

- Task:** @index.html
- Description:** You are tasked with converting an HTML file (index.html) into a Flask web application using SQLite for database operations. The applica: [See more](#)
- Tokens:** ↑ 32 ↓ 5,524
- Cache:** 0 +27,638 → 37,619
- API Cost:** \$0.1979
- EXPORT** button

Instructions:

- データの追加: フォームに新しいデータを入力し、「追加」ボタンをクリックすることでデータを追加できます。
- データの編集: 各行の「編集」ボタンをクリックすると、データを編集できます。
- データの削除: 各行の「削除」ボタンをクリックすると、データを削除できます。
- グラフの表示: 日別、月別、年別のコストグラフと、月別コスト内訳の円グラフが表示されます。

注意事項:

- 初回実行時は、データベースが空の状態が始まります。いくつかのサンプルデータを追加して、グラフの動作を確認してください。
- エラーが発生した場合は、ターミナルのエラーメッセージを確認し、必要に応じて修正を行ってください。
- アプリケーションの改善や機能の追加が必要な場合は、app.pyやtemplates/index.htmlを適宜修正してください。

これで、Flask Web アプリケーションのセットアップと実行手順の説明が完了しました。アプリケーションを実行し、正常に動作することを確認してください。

タスクが完了したので、attempt_completionツールを使用して結果を提示します。

✓ **Task Completed**

Flask Web アプリケーションの作成が完了しました。以下の手順でアプリケーションを実行できます:

- 必要なモジュールをインストール:
`pip install flask flask-sqlalchemy`
- アプリケーションを実行:
`python app.py`
- ブラウザで `http://localhost:5000` にアクセスしてアプリケーションを確認してください。

アプリケーションには以下の機能があります:

- データの表示、追加、編集、削除
- 日別、月別、年別のコストグラフ
- 月別コスト内訳の円グラフ

初回実行時はデータベースが空なので、サンプルデータを追加してグラフの動作を確認してください。エラーが発生した場合は、ターミナルのエラーメッセージを確認し、必要に応じて修正を行ってください。

Claude wants to execute this command:

```
python app.py
```

Start New Task button

Code Editor: index.html templates > ...

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>PJ稼働コスト管理表</title>
5 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
6 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
7 <style>
8 body { font-family: sans-serif; }
9 table { width: 100%; border-collapse: collapse; }
10 th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
11 th { background-color: #f2f2f2; }
12 canvas { max-width: 100%; }
13 </style>
14 </head>
15 <body>
16 <h1>PJ稼働コスト管理表</h1>
17
18 <table>
19 <thead>
20 <tr>
21 <th>日付</th>
22 <th>メンバー</th>
23 <th>稼働時間(HR)</th>
24 <th>稼働単価(円)</th>
25 <th>日別コスト(円)</th>
26 <th>操作</th>
27 </tr>
28 </thead>
29 <tbody id="data-table">
30 <!-- データはJavaScriptで動的に追加 -->
31 </tbody>
32 </table>
33
34 <h2>データ追加</h2>
35 <form id="add-form">
```

Terminal:

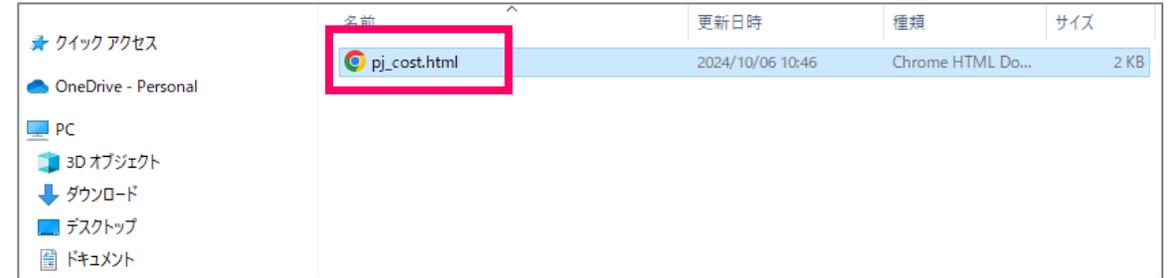
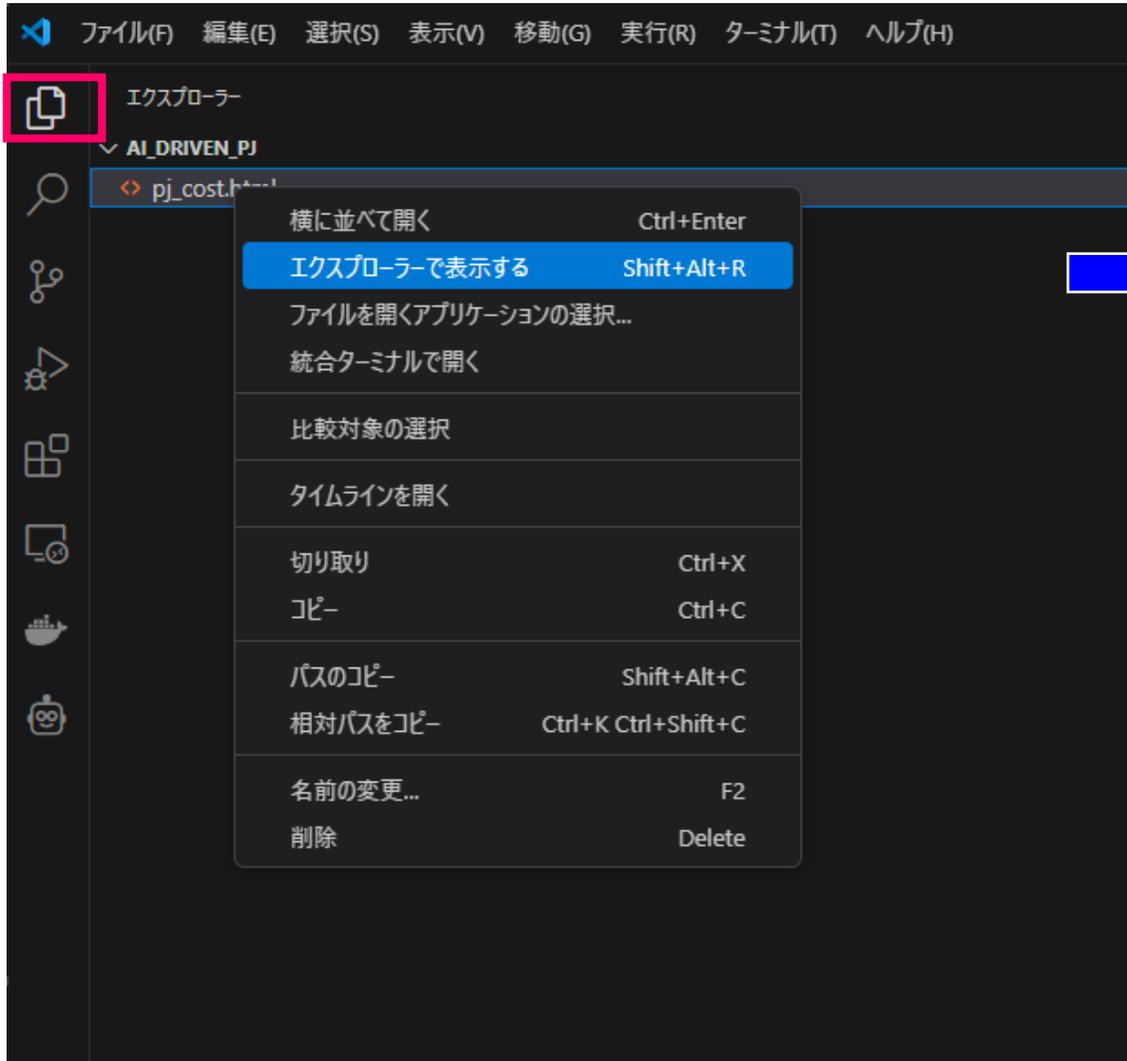
```
Mode                               LastwriteTime                       Length Name
----                               -
d-----                            2024/10/07 0:35                      templates

PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> move index.html templates\
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation> python app.py
Traceback (most recent call last):
  File "E:\000_Udemy\0_VS Code\PJ_Cost_Calculation\app.py", line 1, in <module>
    from flask import Flask, render_template, request, jsonify
ModuleNotFoundError: No module named 'flask'
PS E:\000_Udemy\0_VS Code\PJ_Cost_Calculation>
```

作成したWEBアプリを動作させてみる①

フロントエンドとバックエンドの連携版

エクスプローラで開くと、作成したhtmlファイルが出来ている



作成したhtmlファイルを開くと⇒次ページ





PJ稼働コスト管理表

| 日付 | メンバー | 稼働時間(HR) | 稼働単価(円) | 日別コスト(円) | 操作 |
|----|------|----------|---------|----------|----|
|----|------|----------|---------|----------|----|

データ追加

日別コスト

グラフ機能を実装しているが、まだ上手く表示されないなので、この部分の追加タスク指定が必要！
ヒューマンインザループで、設計者が確認しながら、フロントエンドとバックエンドの仕様をブラッシュアップしていく

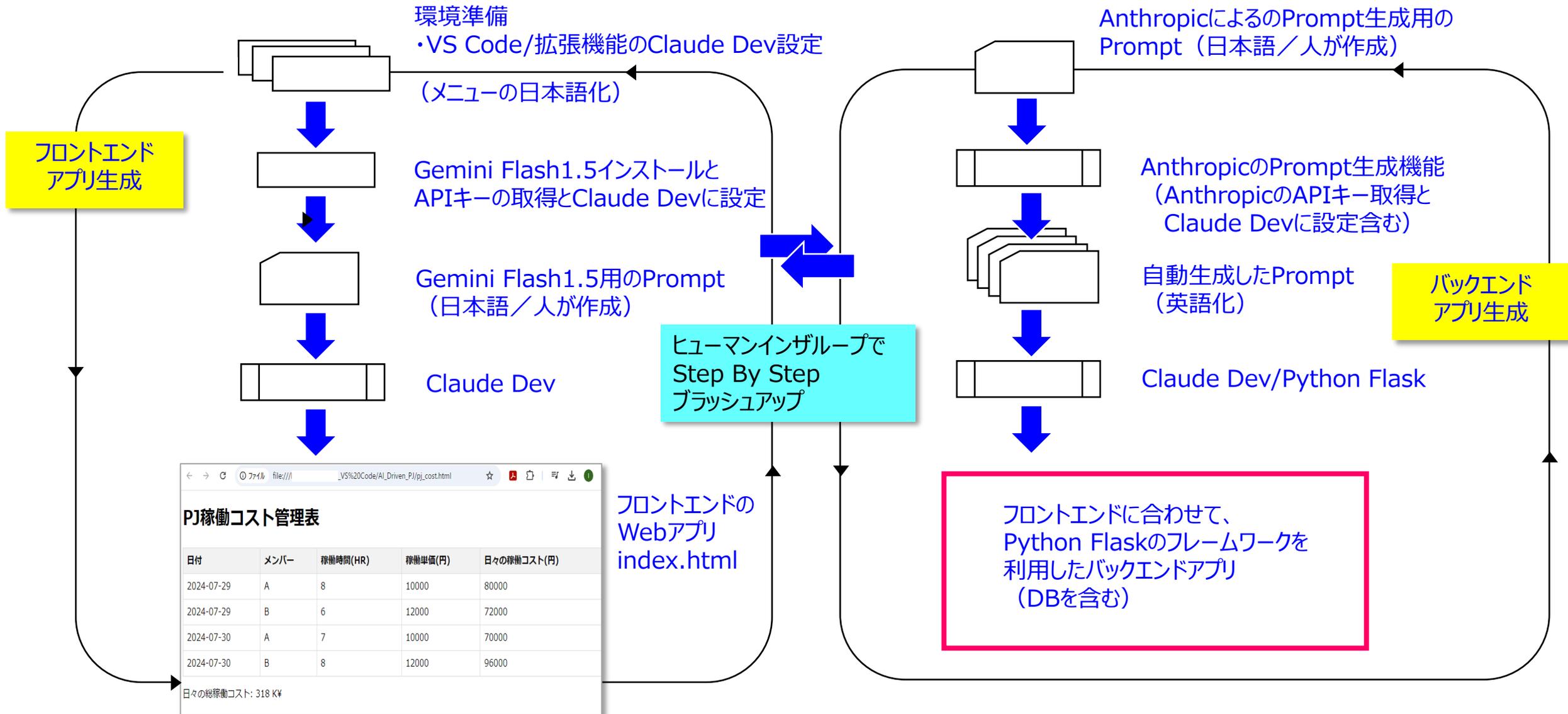
月別コスト

年別コスト

Claude DevによるWebアプリ構築の流れ (纏め)

Claude DevによるWebアプリ構築の流れ

※ フロントエンドは、軽量・効率的なGemini Flash1.5を利用
 バックエンドは、性能が要求されるので、Anthropic Claude 3.5 Sonnetを利用



ブラウザ画面のスクリーンショット: PJ稼働コスト管理表

| 日付 | メンバー | 稼働時間(HR) | 稼働単価(円) | 日々の稼働コスト(円) |
|------------|------|----------|---------|-------------|
| 2024-07-29 | A | 8 | 10000 | 80000 |
| 2024-07-29 | B | 6 | 12000 | 72000 |
| 2024-07-30 | A | 7 | 10000 | 70000 |
| 2024-07-30 | B | 8 | 12000 | 96000 |

日々の総稼働コスト: 318 K¥



まとめ

- VS Code + Claude Devにより、生成AIを活用して、PJを効率的に管理するWebアプリの自動生成の試行を行った。
- Web上のUIを実現するフロントエンド機能、SQLiteをDBとしてデータの蓄積・再利用に利用するバックエンド機能の両方を、Prompt入力により自動生成出来ることを確認した。
- バックエンド機能の実現には、Anthropic ClaudeによるPrompt自動生成機能を利用した。これはバックエンドの各機能要素を、設計者の確認を得ながら逐次自動生成出来る。
- PJ管理を支援するWebアプリケーションは、情報の共有、オープン化で、PJに関係する全てのステークホルダ間のCommunity力を強化することに役立つ。
- 今回の試行は、完全なアプリケーションを作ったものではないが、生成AIの進化により、今後のPJ管理の効率化に大きく貢献出来そうな感触を得た。
- PJの各種実績をRAGやSQLiteなどのDBに蓄積しつつ、PJのコスト予想・リスク分析、ポートフォリオ選択に繋げるよう、生成AIの進化をWatchしつつ取り込んで行きたい。
- ただし「自動生成」による効率化は大きいですが、PMや設計者がPJの方向を定めるために、正しいINPUTを生成AIにStep By Stepで与えることは益々重要であり、PM自身もこの動きを常に学んで取り込んでいく必要がある。

おわりに

本資料の作成グループ：AI@Work WG1 D0チームでは、プロジェクトマネジメントの観点で、AIを含むプロジェクトを成功させて、価値を実現するための手法を研究しています。研究の成果を、コンフェレンスでの発表、ブログ記事、スライドなど様々な形式で発信していきます。

我々と一緒に研究したいという仲間の方を常時募集しています。
またお気づきの点、アイデアとか、前向きに取り組みを進めるためのご意見をお待ちしております。
(AI@Workまで)