Google App Script 実装プロジェクト

「タスク自動管理システム」

礒川恵紀

1. **プロジェクト概要**
2. **実装機能**
    2.1. スプレッドシートへの転記
    2.2. Gmail で通知
    2.3. Slack で通知
    2.4. Slack で ChatGPT を利用してタスク確認
    2.5. ChatGPT を利用してスケジュール作成
    2.6. ChatGPT が作ったスケジュールを Slack で通知
    2.7. ChatGPT が作ったスケジュールを Google カレンダーに追加
3. **改良できる点や取り組んでいる点**

1. プロジェクト概要

本プロジェクトでは、業務報告や現場作業を効率的に集約・管理する支援アプリ「eYACHO」を基盤に、外部データ連携をテーマとしたシステム開発を行いました。Google Apps Script (GAS) を活用し、タスク管理を中心とした各種機能を実装しました。特に、Slack および OpenAI の ChatGPT との連携を実現する API を開発し、REST コネクタを用いて eYACHO との統合を図りました。これにより、タスクの登録、通知、進捗確認、さらには AI を活用したスケジュール生成や Google カレンダーとの同期といった幅広い機能を効率的に提供できるシステムを構築しました。

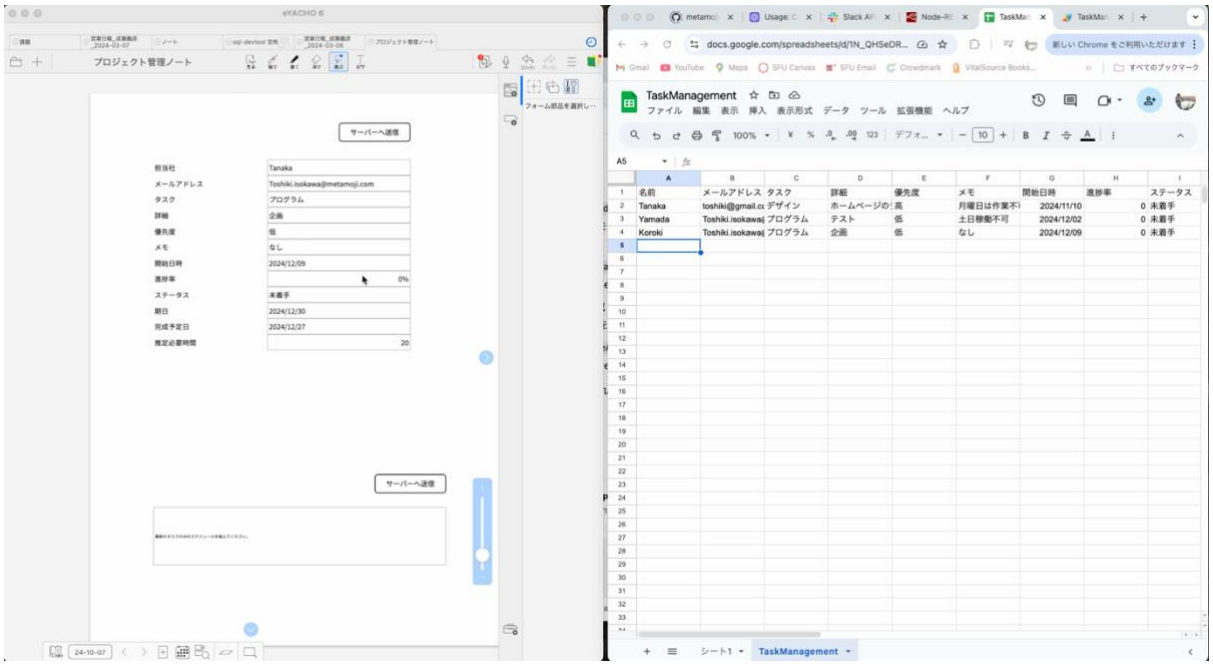
2. 実装機能

2.1. スプレッドシートへの転記

eYACHO 上で設定されるタスクの詳細情報（担当者のメールアドレス、優先度、期日など 12 項目）を Google Apps Script (GAS) を利用してスプレッドシートに転記する機能を実装しました。この機能により、eYACHO で管理される業務データをスプレッドシートで一元的に管理できるようになり、進捗状況の可視化やデータの分析が容易になりました。
特に、**doPost 関数**を中心に処理を設計しました。eYACHO から POST リクエストを通じて送られるデータと、後述する Slack からのメッセージデータを同じ doPost 関数で処理する必要がありました。この課題を解決するために、リクエストの内容を識別して適切に処理するロジックを組み込みました。具体的には以下のように設計されています：

- **データ識別の実装**
    - doPost 関数内で、リクエストデータを判別するロジックを追加しました。
    - eYACHO から送られてくるデータには特定の項目（例：assignee や dueDate）が含まれるため、それを条件にデータの識別を行います。
- **handleEYACHORequest 関数の作成**
    - eYACHO からのリクエストを処理する専用の関数を用意しました。この関数では、eYACHO から送られたデータをスプレッドシートの適切な列に転記するロジックを実装しています。
    - Slack からのメッセージデータと混同されないように、リクエストを受け取った段階で処理を分岐させています。

これにより、eYACHO から送られるタスクデータと Slack 経由のデータをシームレスに処理できるようになり、スプレッドシートへの正確かつ効率的な転記を実現しました。



2.2. Gmail で通知

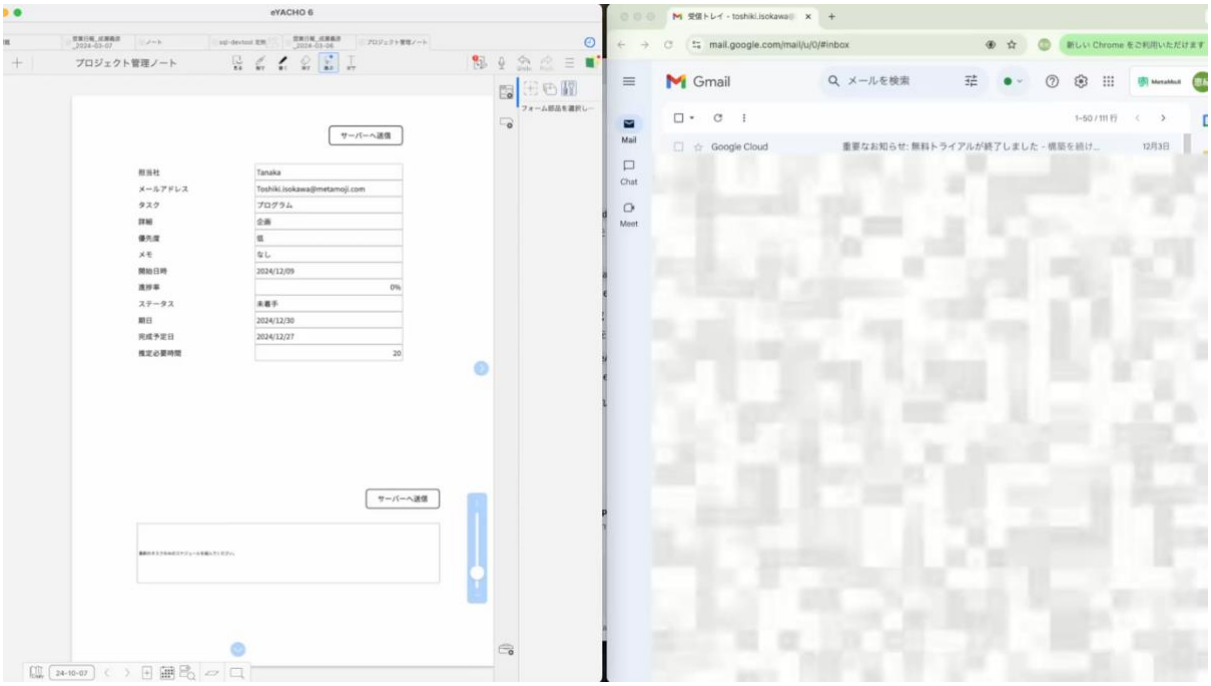
eYACHO から新たに追加されたタスク情報を、スプレッドシートに記載されている全ての担当者のメールアドレスに自動で通知します。これにより、タスクの進捗や詳細が関係者全員にタイムリーに伝達され、迅速な対応が可能となります。
具体的な実装方法としては、Google Apps Script を使用して、スプレッドシートに追加されたタスク情報を取り出し、各担当者のメールアドレスにレポートを送信する仕組みを構築しています。
以下の流れで実装されています：

- **スプレッドシートのデータ取得**
  スプレッドシートから、タスク情報（担当者、メールアドレス、タスク内容、優先度、進捗状況、期日など）を取得します。データはヘッダー行を除いた 2 行目以降から取得されます。
- **メール送信処理**
  各担当者のメールアドレスを取得し、メールが有効な場合に通知を送信します。メールには以下の情報が含まれます：
  - 担当者名
  - メールアドレス
  - タスク名
  - 詳細
  - 優先度
  - メモ（任意）
  - 開始日
  - 進捗率
  - ステータス
  - 期日
  - 完成予定日
  - 推定時間
- **メール本文の構成**
  メール本文にはタスクの詳細情報がフォーマット化されて含まれ、担当者がタスクの進捗や状況を把握しやすいように整えられています。日時の情報

（開始日、期日、完成予定日）は日付形式に変換されて表示され、メモ欄には空欄の場合に「なし」と表示されます。

- **通知の自動化**

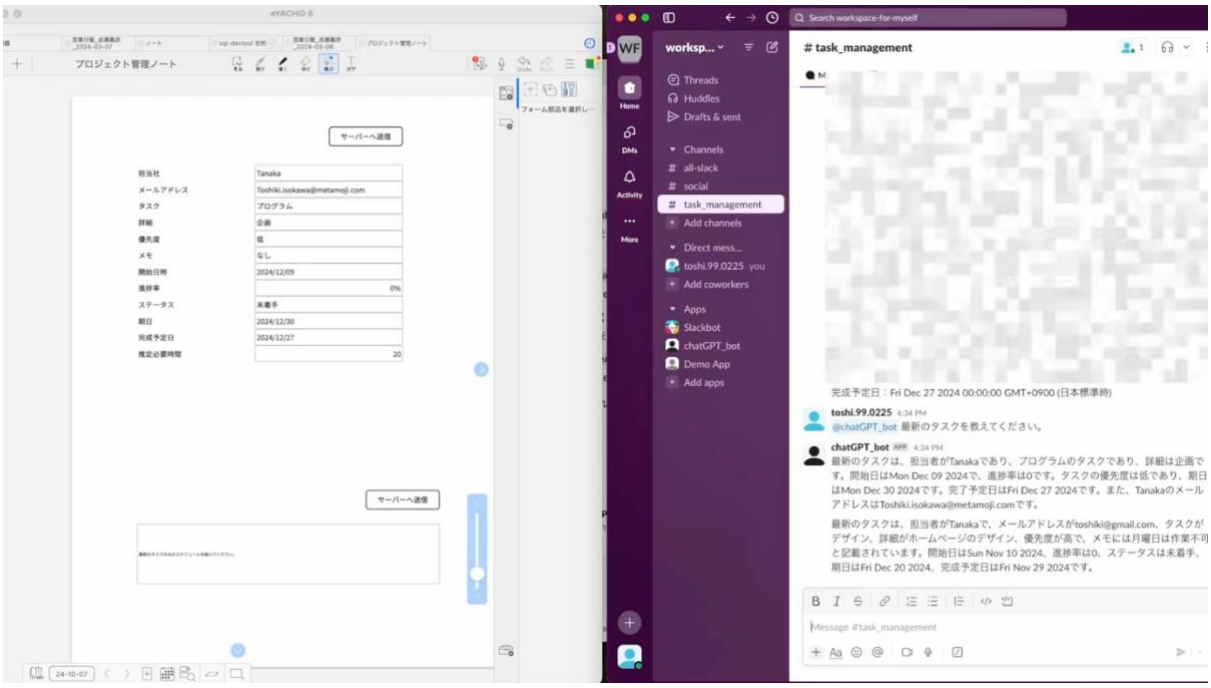
  メール送信は自動的に実行され、スプレッドシート内のタスク情報が更新されるたびに、関連する担当者全員に最新のタスク情報が通知されます。これにより、タスク管理の効率化が実現されます。



2.3. Slack で通知

eYACHO から新たに追加されたタスク情報を Slack に通知するために、Webhook を利用してメッセージを送信しています。これにより、タスクが追加された際にチームメンバーが即座に情報を受け取ることができ、迅速な対応を促進します。
具体的な実装は以下のように構成されています：

- **Webhook URL の設定**
  Slack では「Incoming Webhooks」を使用して外部アプリケーションからメッセージを受信することができます。sendSlackNotification 関数では、この Webhook の URL（webhookUrl）を指定し、そこにメッセージを送信する形式となっています。
- **メッセージ構築**
  notifyNewTaskToSlack 関数では、タスク情報（担当者、タスク名、詳細、優先度、期日、進捗率など）を Slack メッセージ形式で構築しています。この情報は、data オブジェクトから動的に取得され、タスクの詳細をわかりやすく表示します。
- **Slack にメッセージ送信**
  メッセージが構築された後、sendSlackNotification 関数が呼び出され、Slack の Webhook URL に対して POST リクエストを送信します。このリクエストには、構築したメッセージが JSON 形式でペイロードとして含まれます。UrlFetchApp.fetch メソッドを使用して、指定した URL にリクエストが送られ、Slack に通知が届く仕組みです。
- **メッセージの内容**
  メッセージには、タスクの各詳細情報が整然と含まれており、担当者名、メールアドレス、タスク名、詳細、優先度、メモ、開始日、進捗率、ステータス、期日、完成予定日、推定時間が表示されます。これにより、タスクの進捗状況を Slack 内で迅速に確認できます。
- **通知の呼び出し**
  notifyNewTaskToSlack 関数は、eYACHO から送信されたタスク情報を処理する handleEYACHORequest 関数内で呼び出され、タスクが追加されるたびに Slack に自動的に通知が送られるようになっています。


プロジェクト管理ノート
サーバーへ送信
担当社
Tanaka
メールアドレス
Toshiki.isokawa@metamoji.com
タスク
プログラム
詳細
企画
優先度
低
メモ
なし
開始日時
2024/12/09
進捗率
0%
ステータス
未着手
期日
2024/12/30
完成予定日
2024/12/27
推定必要時間
20
サーバーへ送信
worksp...
Threads
Huddles
Drafts & sent
Channels
all-slack
social
task_management
Add channels
Direct mess...
toshi.99.0225 you
Add coworkers
Apps
Slackbot
chatGPT_bot
Demo App
Add apps
task_management
完成予定日：Fri Dec 27 2024 00:00:00 GMT+0900 (日本標準時)
toshi.99.0225 4:34 PM
@chatGPT_bot 最新のタスクを教えてください。
chatGPT_bot APP 4:34 PM
最新のタスクは、担当者がTanakaであり、プログラムのタスクであり、詳細は企画です。開始日はMon Dec 09 2024で、進捗率は0です。タスクの優先度は低であり、期日はMon Dec 30 2024です。完了予定日はFri Dec 27 2024です。また、Tanakaのメールアドレスは Toshiki.isokawa@metamoji.com です。
最新のタスクは、担当者がTanakaで、メールアドレスがtoshiki@gmail.com、タスクがデザイン、詳細がホームページのデザイン、優先度が高で、メモには月曜日は作業不可と記載されています。開始日はSun Nov 10 2024、進捗率は0、ステータスは未着手、期日はFri Dec 20 2024、完成予定日はFri Nov 29 2024です。
Message #task_management

2.4. Slack で ChatGPT を利用してタスク確認

Slack の特定のチャンネルで ChatGPT ボットをメンションしてタスクに関する質問を投げかけることで、ボットがその質問に対する返答を行います。具体的な流れと処理の概要は以下の通りです：

1. Slack メッセージの受信

- handleSlackRequest 関数は、Slack から送信されたメッセージイベントを処理します。
- メッセージが message タイプである場合のみ処理を続けます。ボットからのメッセージや他のボット ID からのメッセージ（subtype === 'bot_message' や bot_id が含まれる場合）は無視されます。これにより、無駄なループや重複を防ぎます。

2. メッセージ内容の抽出とプロンプト生成

- slackMessage 変数に Slack から送られたメッセージ内容を格納します。
- メッセージがボットによるものではないことを確認した後、generateChatGPTPrompt 関数を呼び出して、Slack メッセージを元に ChatGPT へ送るためのプロンプトを作成します。このプロンプトは、ChatGPT に質問内容を明確に伝えるために整形されます。

3. ChatGPT に対する API 呼び出し

- callChatGPT 関数を使用して、生成されたプロンプトを ChatGPT API に送信します。ChatGPT は、タスクに関する質問や進捗状況の確認などに答えます。

4. 返答の Slack への送信

- ChatGPT からの返答が得られた後、それを returnSlackMessage 関数を使って、元の Slack チャンネルにメッセージとして送信します。
- returnSlackMessage 関数では、Slack の API (chat.postMessage）を呼び出して、指定されたチャンネル（postData.event.channel）にメッセージを送信します。メッセージ内容は ChatGPT から得た回答です。

5. Slack API を使用したメッセージ送信

- メッセージを送信するために、Slack API へのリクエストには認証トークン（token）が必要です。このトークンは、Slack の OAuth 認証によって取得します。
- メッセージの内容とターゲットチャンネルを指定して POST リクエストを Slack に送信します。

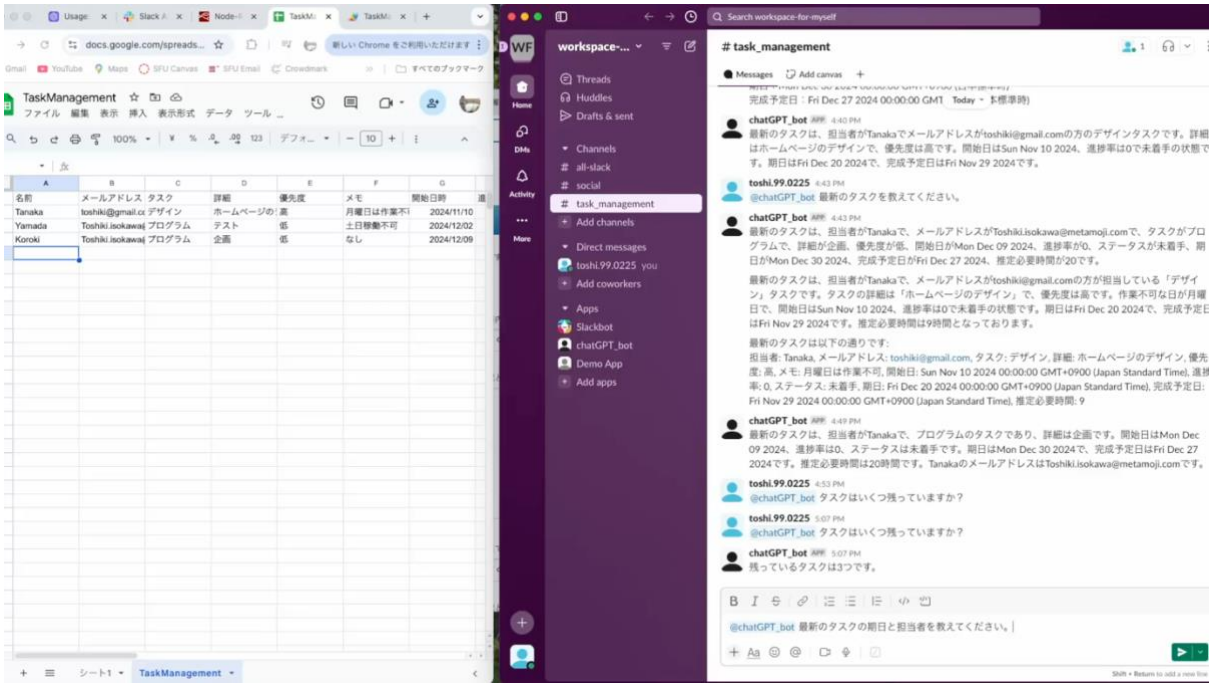
流れをまとめると、

→ Slack からメッセージを受け取る。

→ メッセージが bot からではないことを確認し、ChatGPT に送るプロンプトを作成。

→ 作成したプロンプトを使って ChatGPT に問い合わせ、返答を受け取る。

→ ChatGPT からの返答を、元の Slack チャンネルに通知。



2.5. ChatGPT を利用してスケジュール作成

ユーザーが eYACHO からスケジュールを立ててほしいタスクを指定し、特別なコメント（例：土日稼働不可能）を提供することによって、ChatGPT にスケジュールを自動的に作成させます。システムは、ユーザーから提供されたコメント、タスクのステータス項目（期日、推定時間、進捗率など）、およびその他の条件を基に、最適なスケジュールを生成します。

主要な処理の流れ

1. **タスクデータの取得**
   getTaskData 関数で、eYACHO システムから現在のタスク情報を取得します。このデータには、タスク名、優先度、進捗率、期日、推定時間などが含まれています。
2. **タスクデータのフォーマット化**
   generateTaskDataString 関数で、取得したタスク情報を ChatGPT が理解できる形式に整形します。これにより、各タスクの詳細情報（例：担当者、優先度、進捗率など）を ChatGPT に渡します。
3. **プロンプトの生成**
   generateSchedulePrompt 関数では、タスクデータとユーザーから提供されたコメントを元に、スケジュールを作成するためのプロンプトを構成します。プロンプト内では、以下の条件を考慮してスケジュールを作成するよう ChatGPT に指示します。
   - メモ、進捗率、優先度、推定時間、開始日時から期日までの期間
   - ユーザーのコメント（例：土日稼働不可能）
4. **ChatGPT へのリクエスト**
   上記のプロンプトを基に、ChatGPT にリクエストを送信し、スケジュールを生成させます。ChatGPT は、与えられた条件に基づいてタスクの詳細に対応するスケジュールを日本語で返答します。

プロンプトの構成

生成されるプロンプトは以下の形式で、ChatGPT にスケジュールを組むよう指示します。

- 「担当者名（タスク):」
  例：「田中 太郎（タスク名):」
    - 「◯月◯日(◯): ◯:00-◯:00, ◯時間作業」
    - 「◯月◯日(◯): ◯:00-◯:00, ◯時間作業」

これにより、タスクごとに必要な時間を割り当て、進捗や優先度に応じたスケジュールが作成されます。

2.6. ChatGPT が作ったスケジュールを Slack で通知

ChatGPT が作成したスケジュールを Slack に通知するものです。具体的には、sendSlackNotification 関数を利用して、生成されたスケジュール情報を Slack に送信します。この関数では、スケジュールデータを Slack メッセージフォーマットに組み込み、Slack の Incoming Webhook を使って指定したチャンネルに通知します。これにより、スケジュールが作成された際に、手動でメッセージを送ることなく、チームメンバーに自動的に通知が届きます。sendSlackNotification で通知を送る際、通知メッセージにはスケジュールの詳細（担当者、タスク名、作業時間など）が含まれ、リアルタイムで進捗を共有することができます。

2.7. ChatGPT が作ったスケジュールを Google カレンダーに追加

ChatGPT が生成したスケジュールを解析し、Google カレンダーに自動的にイベントとして追加する処理を実現します。

1. スケジュールレスポンスの解析

まず、ChatGPT が生成したスケジュールを parseScheduleResponse 関数で解析します。返答のフォーマットは、タスクごとに担当者名、タスク名、日付、作業時間が記載されたもので、これを正しく抽出して Google カレンダーに追加できる形式に整えます。

- parseScheduleResponse 関数では、ChatGPT から受け取ったレスポンスを処理します。スケジュールの各行を分割して、担当者、タスク名、日付、作業時間を取得します。
- 日付と時間は、Google カレンダーに追加するための Date オブジェクトに変換され、タスク情報は tasks 配列に格納されます。

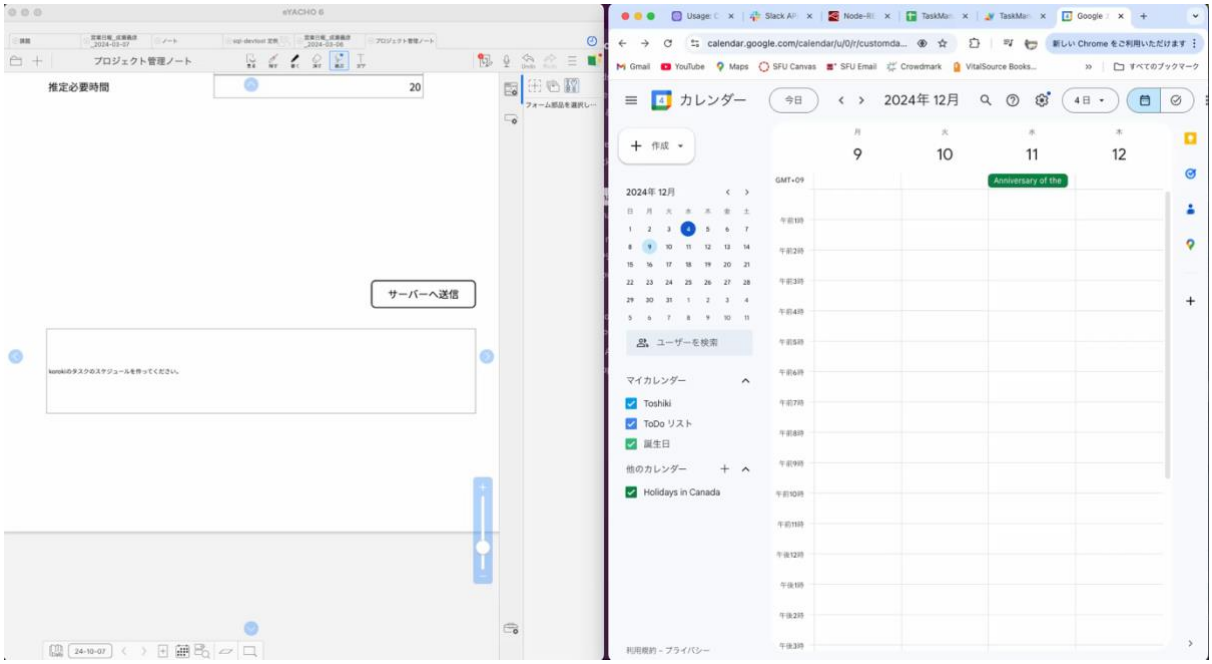
2. Google カレンダーへのイベント追加

addTasksToCalendar 関数では、tasks 配列に格納されたタスク情報を元に、Google カレンダーにイベントを追加します。具体的には、以下の処理が行われます：

- 担当者、タスク名、開始日時、終了日時をもとに、Google カレンダーにイベントを作成します。
- CalendarApp.getDefaultCalendar()を使用してデフォルトのカレンダーを取得し、createEvent メソッドでイベントを追加します。
- イベントには、タスク名や担当者名、説明として「タスク」「担当者」などの情報が記載されます。

機能の流れまとめ

- **スケジュール生成のためのプロンプト作成**
  generateSchedulePrompt 関数で、ユーザーコメントとタスクデータを基にプロンプトを作成。
- **ChatGPT へのプロンプト送信**
  作成したプロンプトを callChatGPT 関数で送信し、ChatGPT からスケジュールのレスポンスを取得。
- **スケジュールレスポンスの解析**
  取得したレスポンスを parseScheduleResponse 関数で解析し、タスク情報を整形して tasks 配列に格納。
- **Google カレンダーへのイベント追加**
  解析したタスク情報を元に、addTasksToCalendar 関数で Google カレンダーにイベントを追加。


プロジェクト管理ノート
推定必要時間
20
サーバーへ送信
kurokiのタスクのスケジュールを作ってください。
カレンダー
今日
2024年 12月
作成
9
10
11
12
Anniversary of the
マイカレンダー
Toshiki
ToDo リスト
誕生日
他のカレンダー
Holidays in Canada

3.改良できる点や取り組んでいる点

1. **eYACHO 側のデザイン改良**
    - GAS を用いた実装をメインに進めてきたため、eYACHO 側の UI/UX が簡素な状態です。
    - UI をリッチにすることで、利用者がより直感的に操作できるよう改良が必要です。
2. **より良いスケジュール作成**
    - ChatGPT によるスケジュール生成の質を高めるため、プロンプトを改善する必要があります。
    - 現在のプロンプトが複雑であるため、よりシンプルかつ的確なスケジュールを生成できる設計に取り組んでいます。
3. **全タスクの表示**
    - eYACHO 上で Google スプレッドシートに保存されている全タスクを表示できるようにすることに取り組んでいます。
    - タスクの進捗や管理を容易にし、ユーザーが eYACHO だけで状況を把握できるようにすることが目標です。
4. **リスケジュール機能の追加**
    - ChatGPT が作成したスケジュール通りに進まない場合、スケジュールを再編成（リスケジュール）できる機能を開発中です。
    - 変更されたスケジュールが Google カレンダーに自動で反映される仕組みを構築する予定です。

Github リポジトリ
(https://github.com/Toshiki-Isokawa/taskManagement)