



# Hello!

**We are**

M.Raghavi Reddy

Purnima Priyadarshini

Srishti Suman

Tejaswi Vykuntam





# Google Docs API



Let's start with.....

# What is an API?

A

Application


P

Programming

I

Interface

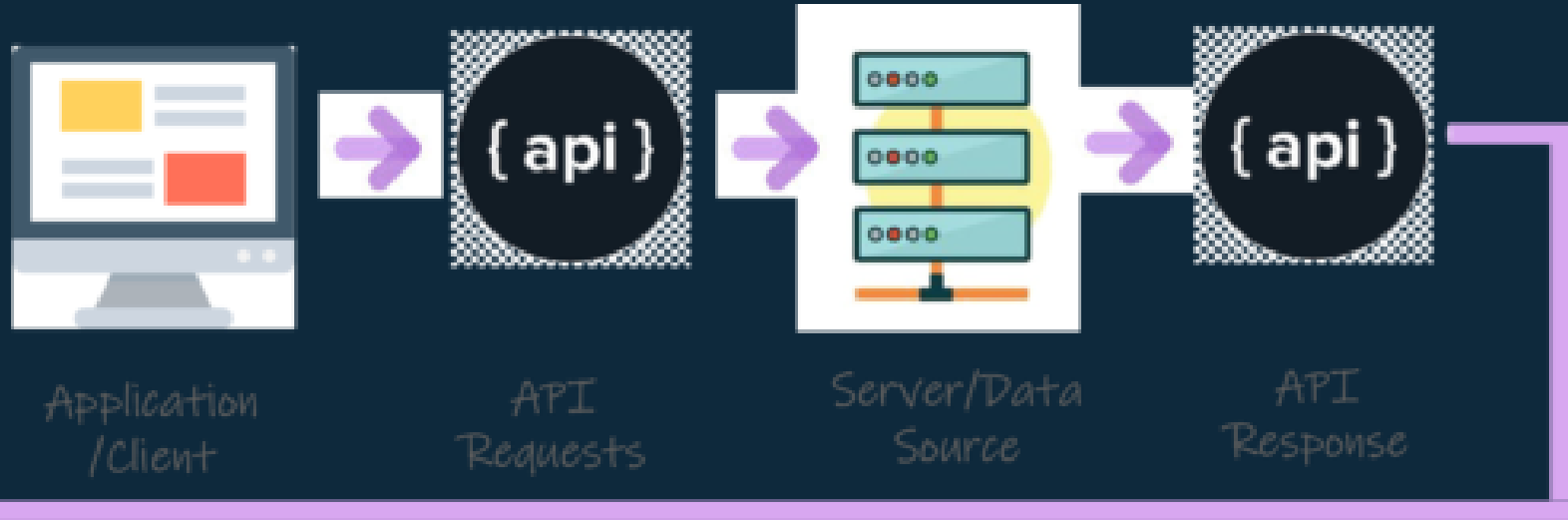
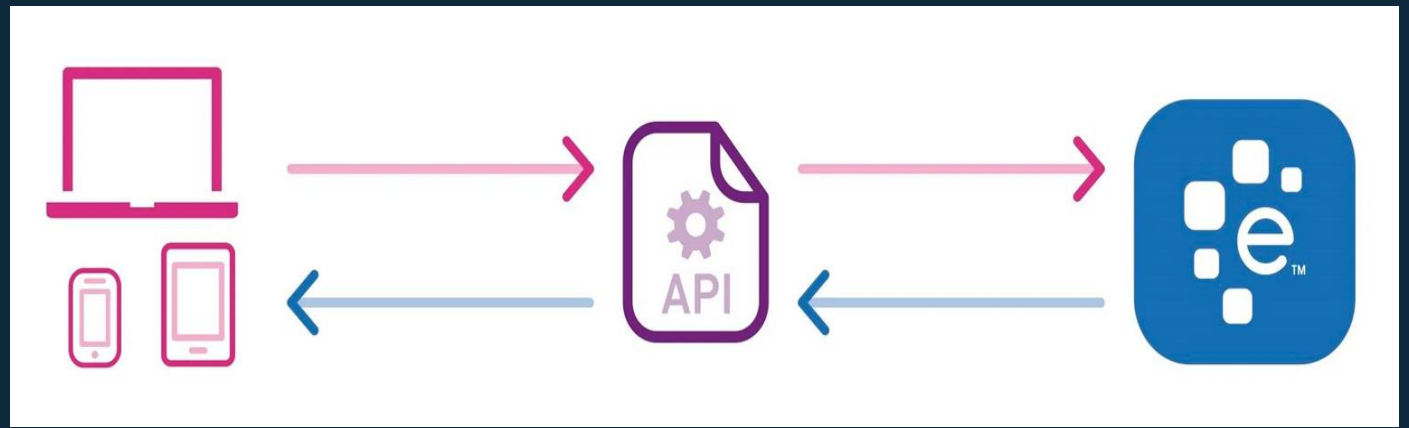




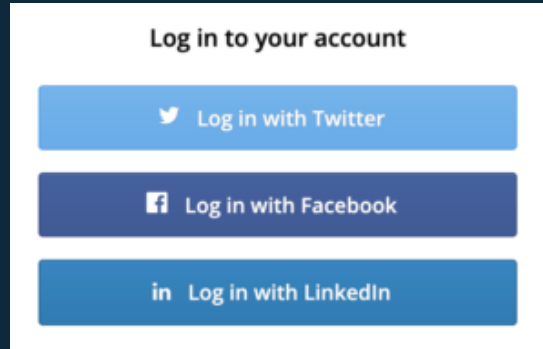
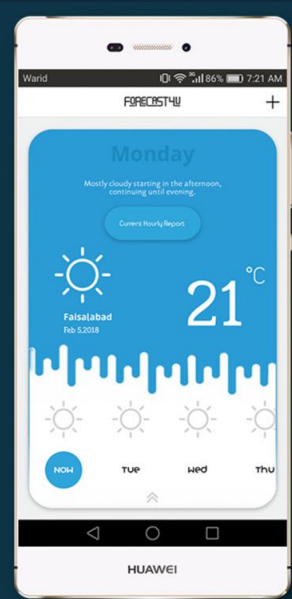
API is the acronym for **Application Programming Interface**, which is a software intermediary that allows two applications to talk to each other. Basically, a code that allows communication between two software programs. It delivers a request from the source to the destination and then brings back the response to the source.

- APIs are of great use because without writing the complete code themselves, they can add specifications to the application.



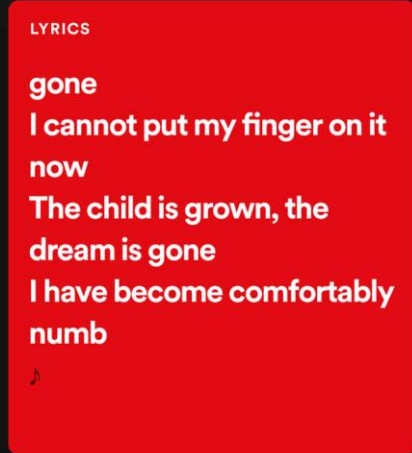


Spotify provides lyrics from Musixmatch in their app. This is done by calling Musixmatch API for the song that is currently being played and the lyrics are displayed in a small window.



Comfortably Numb  
Pink Floyd

4:20 6:22





Now...

What does Google Docs  
API offer?




Google Docs **API** lets you read and write documents programmatically so that you can integrate data from various sources leveraging the power of Google Docs.

**Apps can integrate with the Google Docs API to create polished documents from user-and -system provided data. The API allows you to do tasks such as :**

- **Automate processes (use of digital technology to perform processes in order to accomplish a workflow or function)**
- **Create documentation in bulk**
- **Generate invoices or contracts**







Docs API provides client libraries in many languages such as:

- ◇ Java
- ◇ Python
- ◇ PHP
- ◇ Ruby
- ◇ Node.js
- ◇ Go





The API uses three primary methods.

`documents.create`

To create new document

`documents.get`

To retrieve the entire contents of a specified document.

`documents.batchUpdate`

To atomically perform a set of updates on a specified document.





# Method : documents.create

Creates a blank document using the title given in the request. Other fields in the request, including any provided content, are ignored.

Returns the created document

## HTTPS request:

```
POST https://docs.googleapis.com/v1/documents
```





# API methods

- The *get* and *batchUpdate* methods require a document ID as a parameter to specify the target document. The *create* method returns an instance of the create document, from which you can read the ID.

Document ID :

Documents are referenced by their IDs. The ID of a document can be derived from the URL:

[https://docs.google.com/document/d/\*\*documentId\*\*/edit](https://docs.google.com/document/d/documentId/edit)

The document ID is a string containing letters, numbers and some special characters. The following regular expression can be used to extract the document ID from Google Docs URL:

```
/document/d/([a-zA-Z0-9- _]+)
```





Method :  
documents.get

Get the latest version of the specified document.

HTTPS request:

```
GET https://docs.googleapis.com/v1/documents/{documentId}
```





# Method : documents.batchUpdate

Applies one or more updates to the document.

- Each *request* is validated before being applied. If any request is not valid, then the entire request will fail and nothing will be applied.
- Some requests have *replies* to give you some information about how they are applied. Other requests do not need to return information; these each return an empty reply.
- You should use batchUpdate to group many requests into a single call where possible. This saves quota and improves performance.

HTTPS request :

```
POST https://docs.googleapis.com/v1/documents/{documentId}:batchUpdate
```





## For Example :

Suppose you call `batchUpdate` with four updates, and only the third one returns information. The response would have two empty replies, the reply to the third request and another empty reply in that order.





## REASON:

Because other users may be editing the document, the document might not exactly reflect your changes: your changes may be altered with respect to collaborator changes. If there are no collaborators, the document should reflect your changes. In any case, the updates in your request are guaranteed to be applied together atomically.







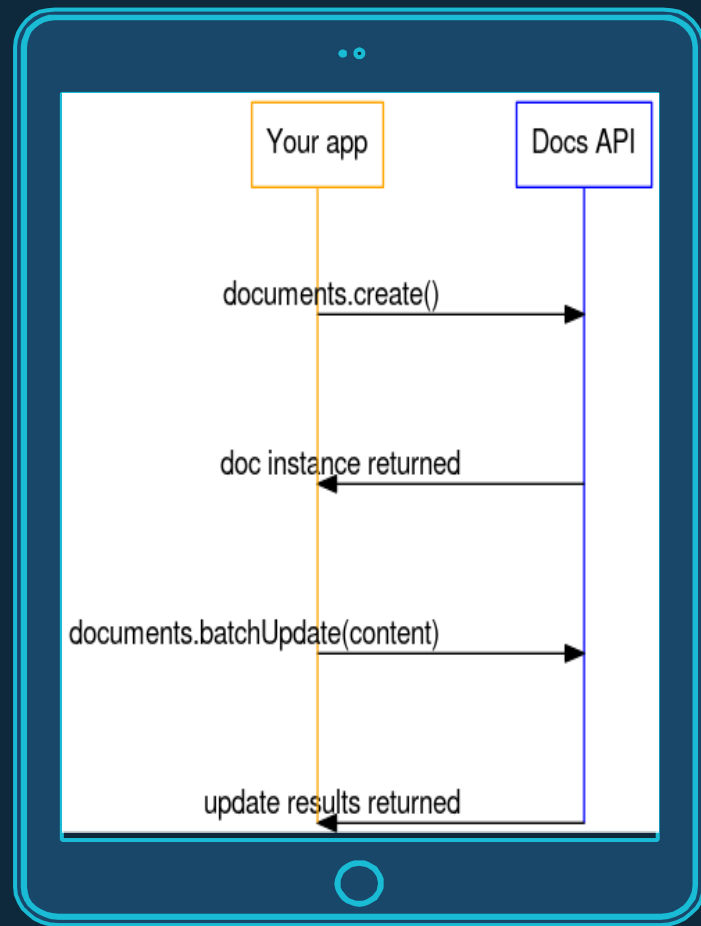
# Document update overflow



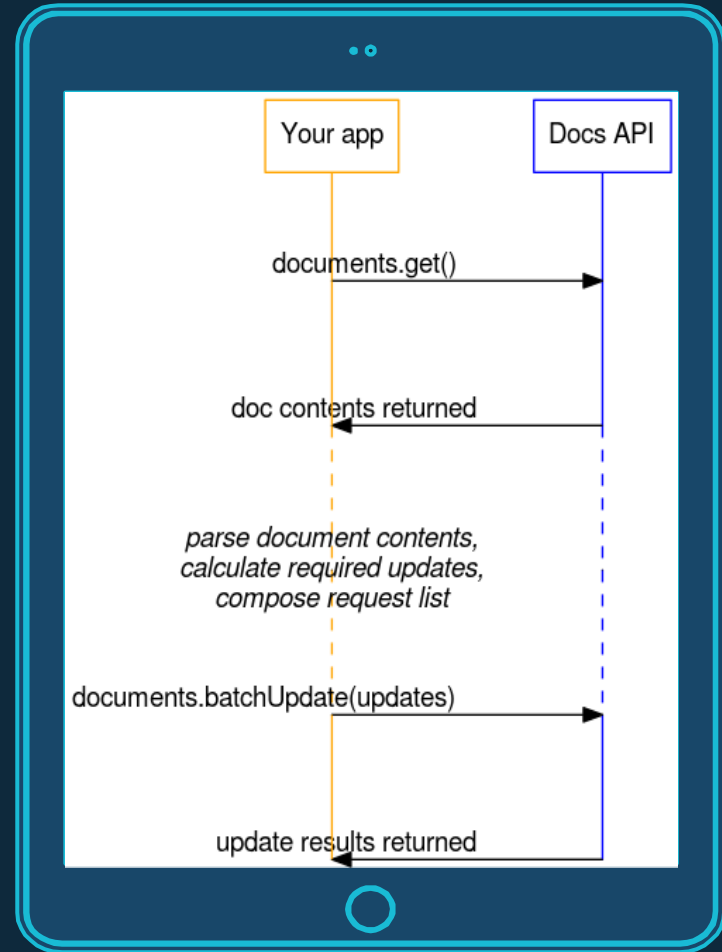
Security may be better than in paper files.



Creating and populating a new document is straightforward, since there is no existing content to worry about and there are no collaborators who can alter the document state.



Updating an existing file is more complex. Before you can make meaningful calls to update a document, you need to know the current state of the document: what elements make it up, what content is in those elements, and where all these things are located with the document.





# Best practices for best results



There are a number of principles that you should follow when using the Google Docs API.

- Edit backwards for efficiency
- Plan for collaboration
- Use WriteControl to ensure state consistency





## Edit backwards for efficiency

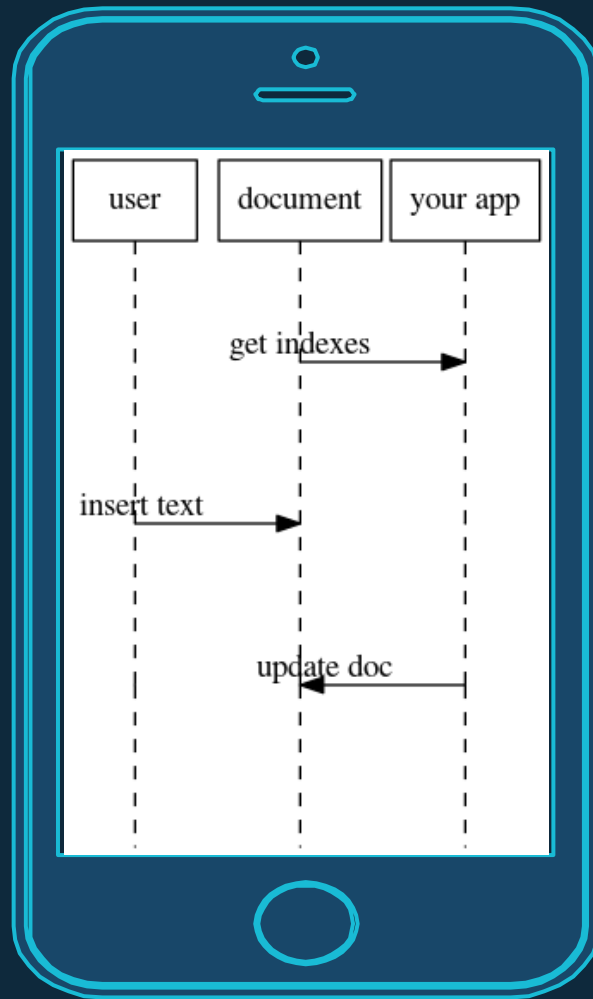
Within a single call to the `batchUpdate` method, order your requests in descending order of the index location. This eliminates the need to calculate the index changes due to insertions and deletions.





## Plan for collaboration

Expect the document state to change.  
Between one method call and another,  
the document might have been updated  
by other collaborators.





This can lead to errors if your indexes are wrong. With multiple users editing a document using the UI, Google Docs takes care of this transparently, but as an API client your app must manage this. Even if you don't anticipate collaboration on the document, it is important to program defensively and make sure the document state remains consistent. See [Use WriteControl](#) for one way to ensure consistency.








## Use WriteControl

When you read and then update a document, you can ensure that the document has not been updated by using the `WriteControl` field in the `BatchUpdate` method. This lets you make edits to the doc only if the version has not changed.

When you call `BatchUpdate`, it will return an error and update nothing if the document revision has changed.





# How can we utilize the Google Docs API to boost efficiency in the workplace ?


## Automate Document Creation with the Google Docs API


```
Title = 'My Document'
Body = {
    'Title' : title
}

doc = service.documents() \
    .create(body=body) .execute()
print('Created document with title: {0}'.format{
doc.get('title'))
```

Got to create an invoice for every customer , and hopefully that's a lot. The creation of invoice will get better with the addition of the Docs API.

You're ready to move past manual creation, but luckily the Document Doc Create method can do that for you.





Now you can create all the individual invoices in one run of your invoice service.

### Productivity :

Instead of creating a new empty document every time, you simply make a copy of your invoice template.

Here is where the Drive API will help to speed up.





We are using python for the demo.  
The libraries are fairly easy to include and implement





To install:

```
Demo> pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
```

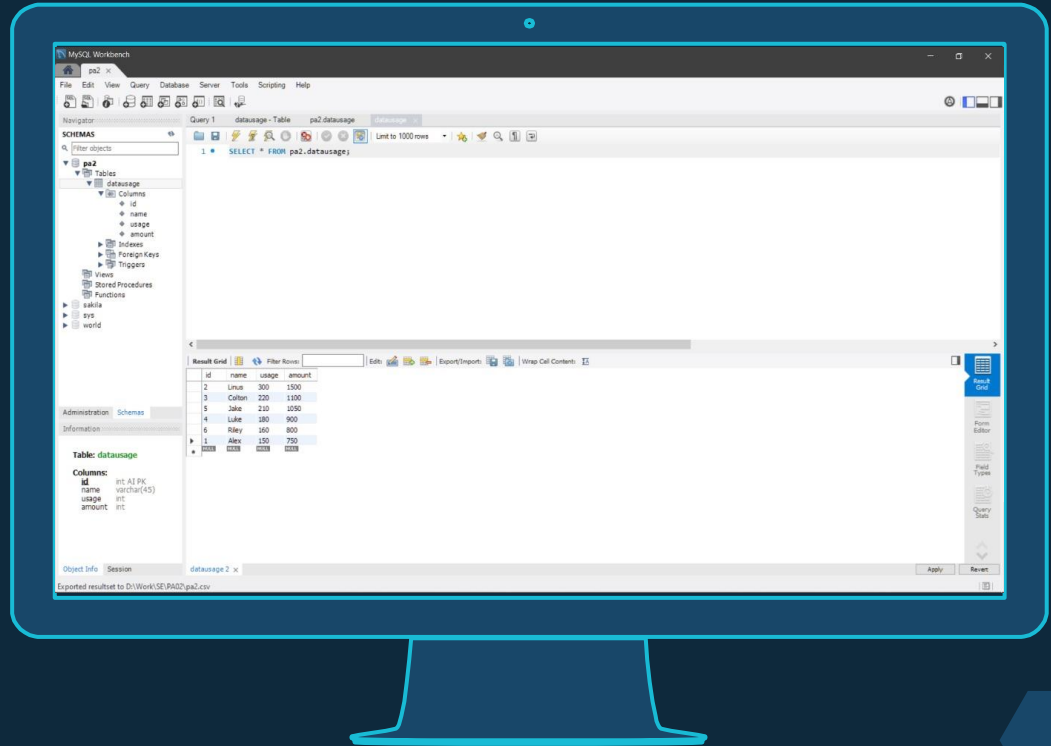
To include:

```
1 # For use with Docs API
2 from __future__ import print_function
3 import pickle
4 import os.path
5 from googleapiclient.discovery import build
6 from google_auth_oauthlib.flow import InstalledAppFlow
7 from google.auth.transport.requests import Request
```



## Our use case

Implementing Google Docs API to automate Invoice generation using a pre-existing template and populating it with data from an SQL server/ CSV file.





Thanks!

