# **MEDIUM BLOG:** [**https://medium.com/@f200114/unlock-the-power-of-web-scraping-with-colly-in-golang-588da159e540**](https://medium.com/@f200114/unlock-the-power-of-web-scraping-with-colly-in-golang-588da159e540)

**Link 1 code:**

package main

import (

   "encoding/csv"

   "log"

   "os"

   "time"

   "github.com/gocolly/colly"

)

type scrapStruct struct {

   url   string

   image string

   title string

   text  string

}

func scrapeAndWriteCSV() []scrapStruct {

   var scrapData []scrapStruct

   // <-- gpt modification

   c := colly.NewCollector(

      colly.UserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"),

   )

   // Set a delay between requests to avoid being blocked

   c.SetRequestTimeout(time.Second \* 10)

   //-->

   c.OnHTML(".o-listease\_\_item", func(e \*colly.HTMLElement) {

      linkData := scrapStruct{}

      linkData.url = e.ChildAttr("a", "href")

      linkData.image = e.ChildAttr("img", "src")

      linkData.title = e.ChildAttr("a", "title")

      linkData.text = e.ChildText(".m-statement\_\_quote")

      scrapData = append(scrapData, linkData)

   })

   c.OnHTML(".m-teaser", func(e \*colly.HTMLElement) {

      linkData := scrapStruct{}

      linkData.url = e.ChildAttr("a", "href")

      linkData.image = e.ChildAttr("img", "src")

      linkData.title = e.ChildAttr("a", "title")

      scrapData = append(scrapData, linkData)

   })

   c.OnError(func(r \*colly.Response, err error) {

      log.Printf("Request URL: %s failed with response: %v\n", r.Request.URL, err)

   })

   c.Visit("https://www.politifact.com")

   // Wait for the collector to finish

   c.Wait()

   file, err := os.Create("link1.csv")

   if err != nil {

      log.Fatalln("Failed to create output CSV file", err)

   }

   defer file.Close()

   // Initializing a file writer

   writer := csv.NewWriter(file)

   headers := []string{

      "url",

      "image",

      "title",

      "text",

   }

   writer.Write(headers)

   for \_, dataArray := range scrapData {

      // Converting a data to an array of strings

      record := []string{

         dataArray.url,

         dataArray.image,

         dataArray.title,

         dataArray.text,

      }

      writer.Write(record)

   }

   writer.Flush()

   // Check for any errors in writing the CSV file

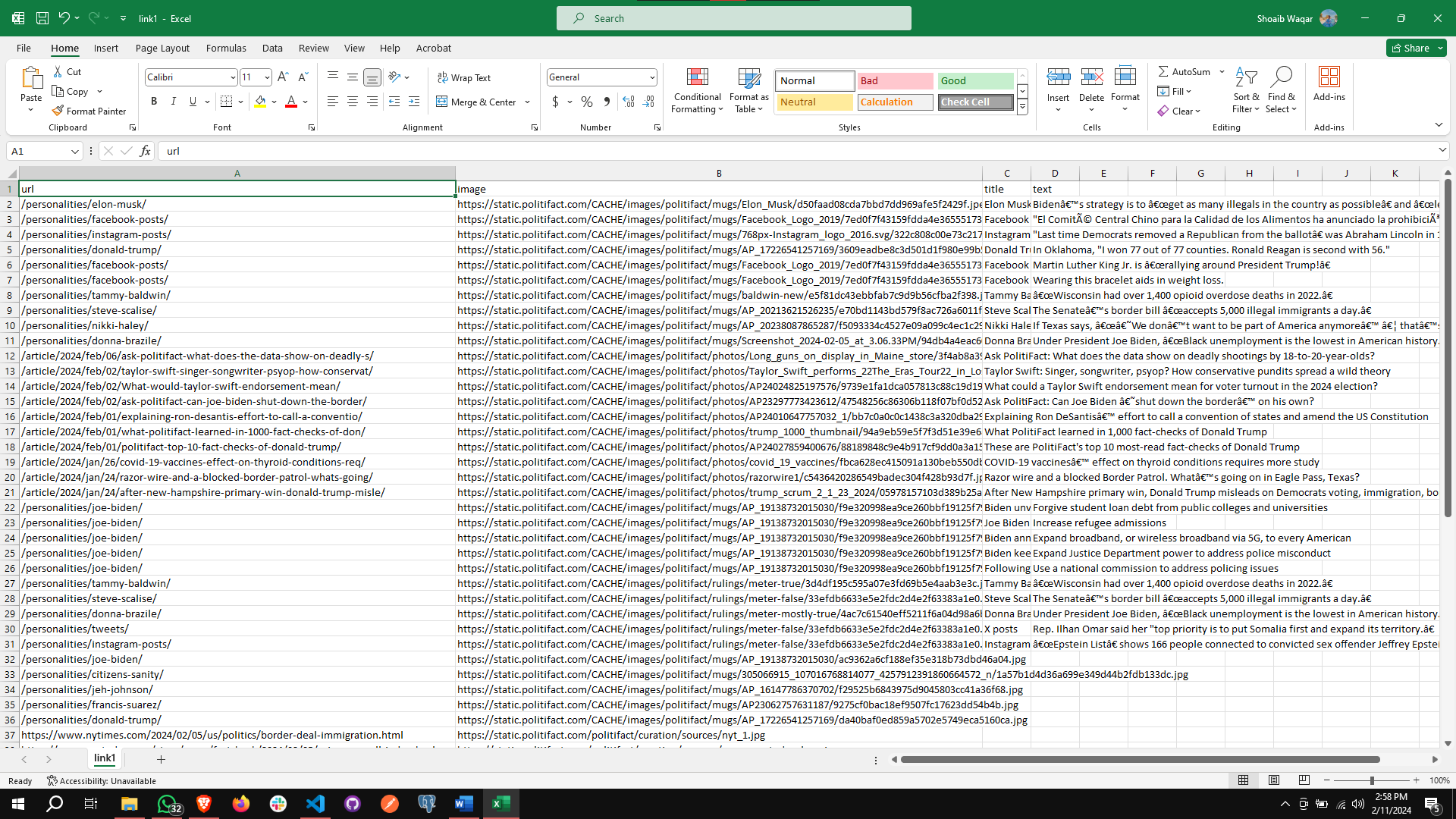
   if err := writer.Error(); err != nil {

      log.Fatalln("Error writing CSV:", err)

   }

   return scrapData

}



**Link 2 code:**

package main

import (

   "encoding/csv"

   "log"

   "os"

   "time"

   "github.com/gocolly/colly"

)

type link2 struct {

   url   string

   image string

   video string

   text  string

}

func scrapLink2() []link2 {

   var scrapData []link2

   // <-- gpt modification

   c := colly.NewCollector(

      colly.UserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"),

   )

   // Set a delay between requests to avoid being blocked

   c.SetRequestTimeout(time.Second \* 10)

   //-->

   c.OnHTML(".entry-header", func(e \*colly.HTMLElement) {

      linkData := link2{}

      linkData.url = e.ChildAttr("a", "href")

      linkData.image = e.ChildAttr("img", "src")

      linkData.text = e.ChildText("h4")

      scrapData = append(scrapData, linkData)

   })

   c.OnHTML(".embed-youtube", func(e \*colly.HTMLElement) {

      linkData := link2{}

      linkData.video = e.ChildAttr("iframe", "src")

      scrapData = append(scrapData, linkData)

   })

   c.OnError(func(r \*colly.Response, err error) {

      log.Printf("Request URL: %s failed with response: %v\n", r.Request.URL, err)

   })

   c.Visit("https://www.altnews.in")

   // Wait for the collector to finish

   c.Wait()

   file, err := os.Create("link2.csv")

   if err != nil {

      log.Fatalln("Failed to create output CSV file", err)

   }

   defer file.Close()

   // Initializing a file writer

   writer := csv.NewWriter(file)

   headers := []string{

      "url",

      "image",

      "video",

      "h4",

   }

   writer.Write(headers)

   for \_, dataArray := range scrapData {

      // Converting a data to an array of strings

      record := []string{

         dataArray.url,

         dataArray.image,

         dataArray.video,

         dataArray.text,

      }

      writer.Write(record)

   }

   writer.Flush()

   // Check for any errors in writing the CSV file

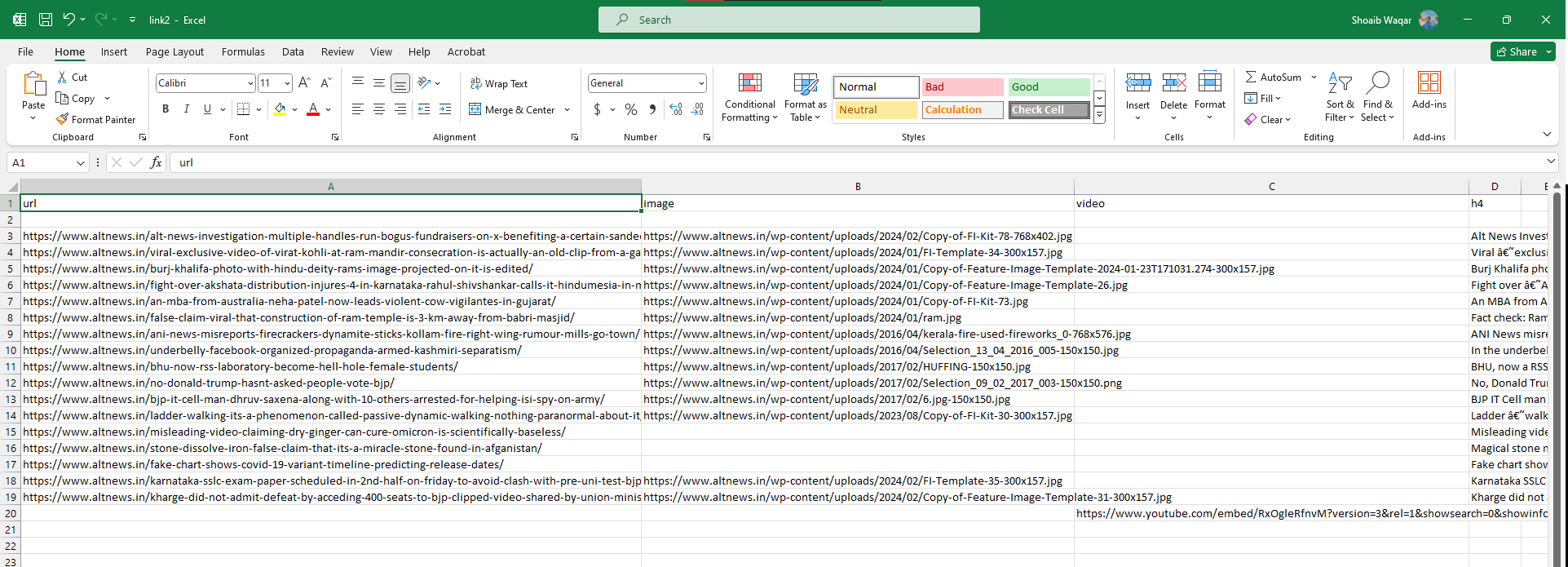
   if err := writer.Error(); err != nil {

      log.Fatalln("Error writing CSV:", err)

   }

   return scrapData

}



**Link 3 code:**

package main

import (

   "encoding/csv"

   "log"

   "os"

   "time"

   "github.com/gocolly/colly"

)

type link3 struct {

   url   string

   image string

   video string

   text  string

}

func scrapLink3() []link3 {

   var scrapData []link3

   // <-- gpt modification

   c := colly.NewCollector(

      colly.UserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3"),

   )

   // Set a delay between requests to avoid being blocked

   c.SetRequestTimeout(time.Second \* 10)

   //-->

   c.OnHTML(".status-card\_\_content", func(e \*colly.HTMLElement) {

      linkData := link3{}

      linkData.url = e.ChildAttr("a", "href")

      linkData.image = e.ChildAttr("img", "src")

      linkData.text = e.ChildText("span.status-card\_\_description")

      scrapData = append(scrapData, linkData)

   })

   c.OnError(func(r \*colly.Response, err error) {

      log.Printf("Request URL: %s failed with response: %v\n", r.Request.URL, err)

   })

   c.Visit("https://mastodon.social/explore")

   // Wait for the collector to finish

   c.Wait()

   file, err := os.Create("link3.csv")

   if err != nil {

      log.Fatalln("Failed to create output CSV file", err)

   }

   defer file.Close()

   // Initializing a file writer

   writer := csv.NewWriter(file)

   headers := []string{

      "url",

      "image",

      "video",

      "text",

   }

   writer.Write(headers)

   for \_, dataArray := range scrapData {

      // Converting a data to an array of strings

      record := []string{

         dataArray.url,

         dataArray.image,

         dataArray.video,

         dataArray.text,

      }

      writer.Write(record)

   }

   writer.Flush()

   // Check for any errors in writing the CSV file

   if err := writer.Error(); err != nil {

      log.Fatalln("Error writing CSV:", err)

   }

   return scrapData

}

**Main.go code:**

package main

import (

   "fmt"

)

func main() {

   scrapeAndWriteCSV()

   scrapLink2()

   scrapLink3()

   fmt.Println("Scraping and CSV writing completed successfully.")

}

//   go run main.go link1.go link2.go link3.go