Arpit Singh

19BCG10069

Appointy – Task 1

Technical Task

(Internship)

Steps:

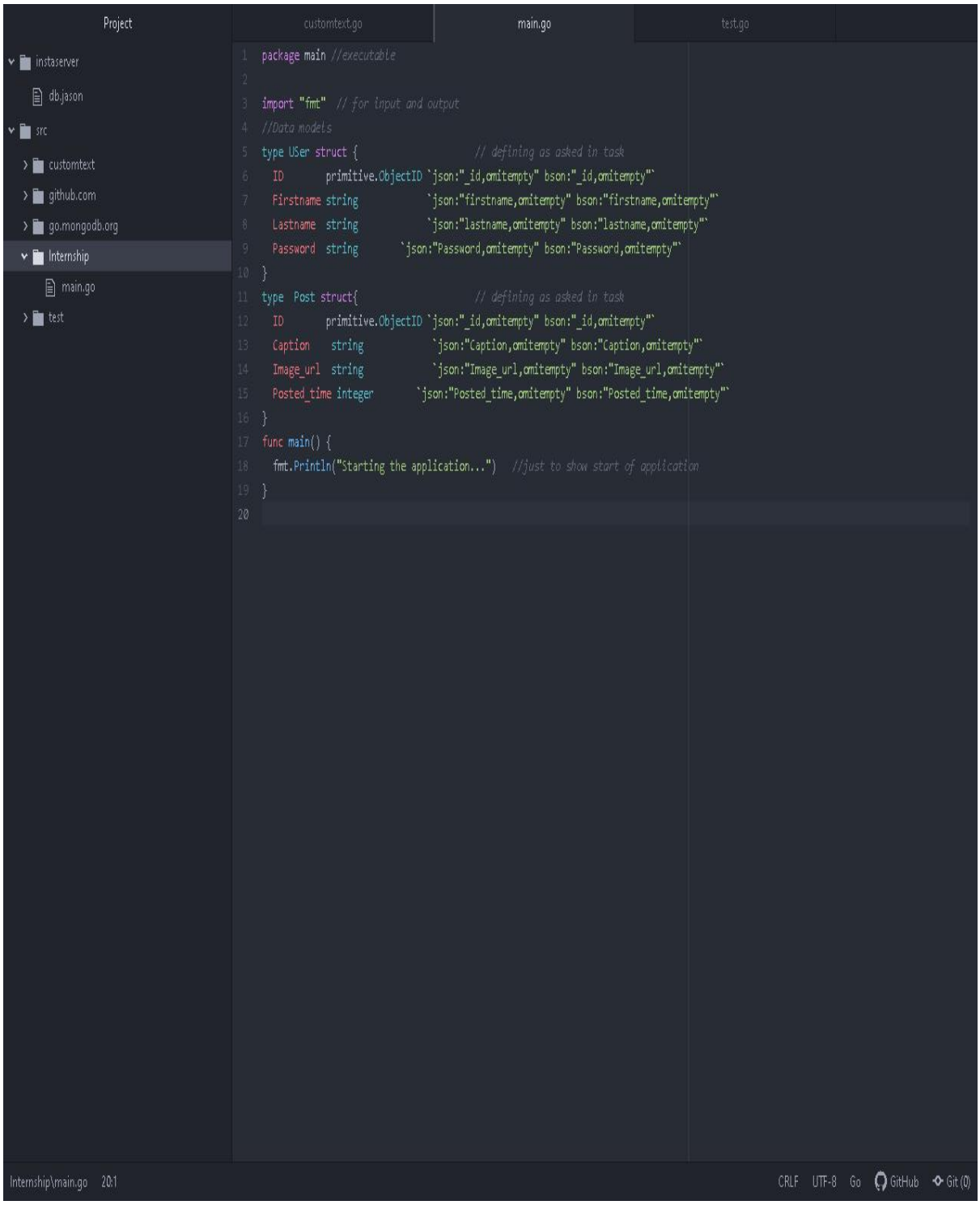
1. Installing Golang and Setting up the Environment
    a. Install(Setup for windows) golang from Golang.org
    b. Setup the System (Environment) variable as per the desired directory
        i. Use commands: go env; Check GoRoot and GoPath specifically.
    c. For Example: To set workstation as a folder in Desktop set env variable as: *C:\Users\ArpitSG\Desktop\Go-Workspace*.
2. Installing MongoDB and Setting up the Environment
    a. Install(Setup for windows) MongoDB from MongoDB.com(community server)
    b. Setup the System (Environment) variable as per the desired directory
    c. For Example: *C:\Program Files\MongoDB\Server\5.0\bin.*
3. Setup a Connection between MongoDB cluster(Server) and Golang(file)
    a. Import dependencies :
        i. go get github.com/gorilla/mux (For Managing htttp requests)
        ii. go get go.mongodb.org/mongo-driver/mongo(For Managing and establishing connection with MongoDB)

Coding

Create a file under GOPath Directory

(C:\Users\ArpitSG\Desktop\Go-Workspace\src\Internship\main.go)

Open a text Editor (Atom) and write the following code that just puts in basic data model and outline of our go file.



Json – for user understanding (Web client)

Bson – for MongoDB interpretation

Next lets create a mongo client and establish connection(within main function along with context time) each time a function of our application is implemented.

```
var client *mongo.Client      // a client would be setup for each time of implementation
func main() {
  fmt.Println("Starting the application...")   //just to show start of application
  ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)   // for establishing connection (context)
  clientOptions := options.Client().ApplyURI("mongodb://localhost:27017")   // port setup
  client, _ = mongo.Connect(ctx, clientOptions)  // mongo Connect
}
```

Next lets define the router to complete the connection process with MongoDB by passing in our router and port.

```
var client *mongo.Client      // a client would be setup for each time of implementation
func main() {
  var (
      client     *mongo.Client
      mongoURL = "mongodb://localhost:27017"
  )
  fmt.Println("Starting the application...")   //just to show start of application
  ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)   // for establishing connection (context)
  client, err := mongo.NewClient(options.Client().ApplyURI(mongoURL))  // mongo Connect
  err = client.Connect(ctx)   // to check if connection is established or not
  ctx, _ = context.WithTimeout(context.Background(), 10*time.Second)
  if err = client.Ping(ctx, readpref.Primary()); err != nil {   // ping mongo server
      fmt.Println("could not ping to mongo db service: %v\n", err)
      return
  }
```

Now lets Check everything is working properly.

To do so lets go to terminal (Cmd) and execute are main.go (file) with:

(go run main.go)

Note you should be in the GoPath file directory

(C:\Users\ArpitSG\Desktop\Go-Workspace\src\Internship\main.go)

```
# command-line-arguments
.\main.go:26:15: undefined: Time

C:\Users\ArpitSG\Desktop\Go-Workspace\src\Internship>go run main.go
Starting the application...
connected to nosql database: mongodb://localhost:27017
```

Now with this the connection to server is setup and we have how the basic data model of our application looks like.

Next lets perform the task as descirebed in the doc file:

So in the folder named TheMainAPI all the required coding tasks are done.

****NOTE: Proper comments are added in each file for better understanding.****

ThMainAPI folder has been divided as:

i. ***main.go***
ii. ***posts.go***
iii. ***go.sum***
iv. ***go.mod***
v. ***Database(Folder > db.go)***

***In* Main.go:**

The basic data model of our application has been implemented with proper http requests and json format.

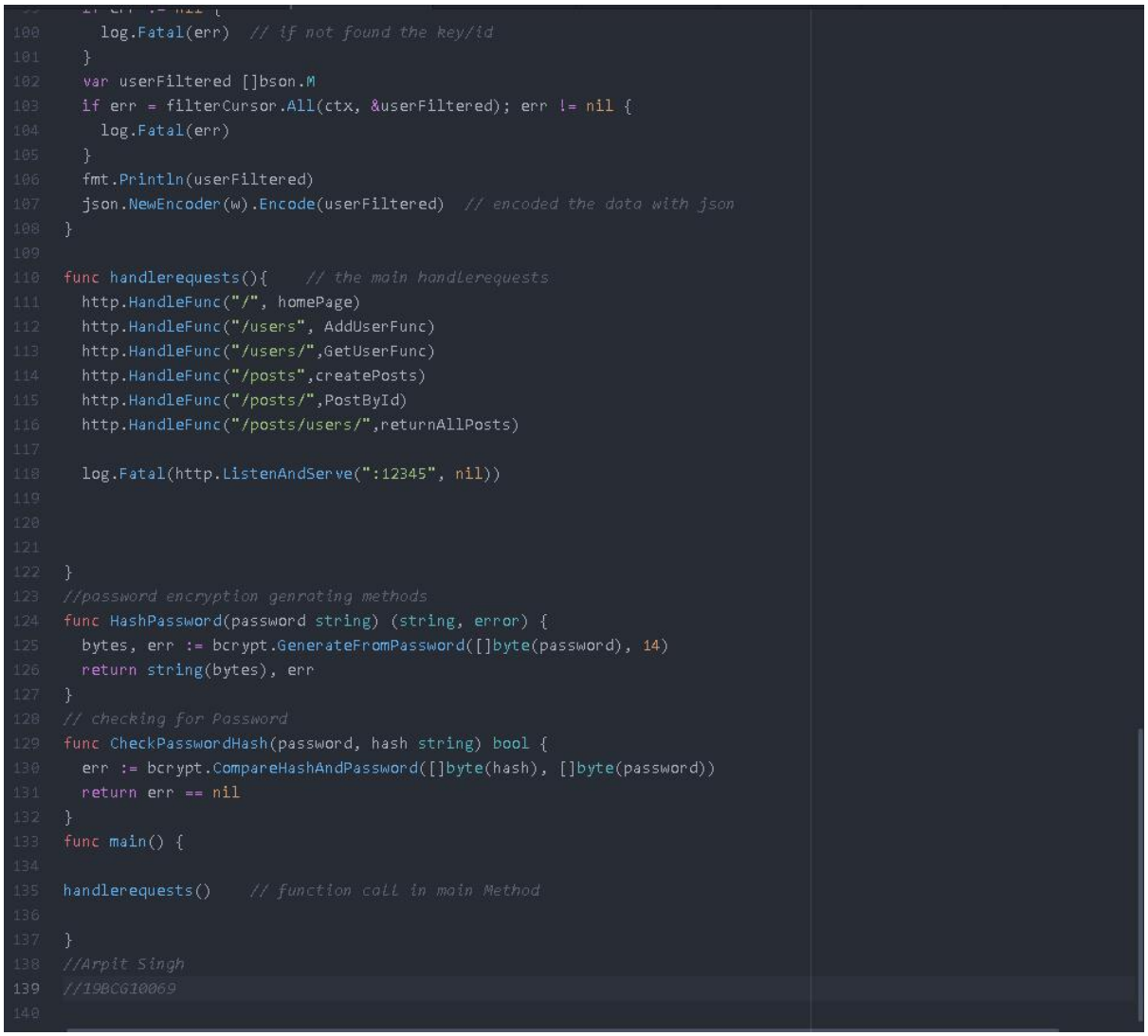
The User data model is implemented and data is stored in the collections accordingly.

All the handlers are added and stored in a function named handelfunction().

```go
package main

import (
  "TheMainAPI/DB"
  "context"
  "encoding/json"
  "fmt"
  "go.mongodb.org/mongo-driver/bson"
  "golang.org/x/crypto/bcrypt"
  "io/ioutil"
  "log"
  "net/http"
  "os"
  "reflect"
  "strconv"
  "time"
)
//Struct defined as per the task here instead of struct users_collectionare used
type UserDetails struct {
  Id  int `json: "Id"`     //unique int type
  Name  string    `json: "Name"`   // rest all strings
  Email string   `json: "Email"`
  Password string `json: "Password"`
}
var UserArr []UserDetails            // here all user are stored with the help of array
//handle functions
func homePage(w http.ResponseWriter, r *http.Request) {       // Function to display few greeting lines
  fmt.Fprintf(w, "This is a Instagram API(Task 1)")
  fmt.Println("Endpoint: homePage")    //endpoint indication
}

func AddUserFunc(w http.ResponseWriter, r *http.Request) {
  if r.Method == "POST" {                           // here Method post used to get the response from web users
    reqBody, _ := ioutil.ReadAll(r.Body)
    fmt.Fprintf(w, "%+v", string(reqBody))
    var users UserDetails     // variable to strore the data
    json.Unmarshal(reqBody, &users)
    // update our global userdetail array to include
    // our new user
    UserArr = append(UserArr, users)    // adding entered details with appendbackwards in the goLobal array

    json.NewEncoder(w).Encode(users)          // json used to encode the data from users var
```

```
41
42      json.NewEncoder(w).Encode(users)        // json used to encode the data frok users var
43
44      fmt.Println("Endpoint: users")        //endpoint indication
45      DatabaseName := "Instagram_DB"        // creating and moving the database to env
46      client, err := db.CreateDatabaseConnection(DatabaseName)
47      if err != nil {                             // checking connection to mongodb Server
48        fmt.Println("Failed to connect to Database")
49        panic(err)
50      }
51      defer client.Disconnect(context.TODO()) // this handels when user Disconnects
52      ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)
53      col := client.Database(DatabaseName).Collection("users_collection")
54      HashPassword(users.Password)            //encrypting password for secruing user password
55      fmt.Println(HashPassword(users.Password))   // showing user DEBUGed:Password
56      result, insertErr := col.InsertOne(ctx,users)   // result and err used in collection col
57      if insertErr != nil {
58        fmt.Println("InsertONE Error:", insertErr) // checking id user entered the data correctly or not
59        os.Exit(1)
60      } else {
61        fmt.Println("InsertOne() result type: ", reflect.TypeOf(result)) // inserting data in collection with reseult
62        fmt.Println("InsertOne() api result type: ", result)
63
64        newID := result.InsertedID   // assigning unique id each time at insertion
65        fmt.Println("InsertedOne(), newID", newID)     // showing the ID
66        fmt.Println("InsertedOne(), newID type:", reflect.TypeOf(newID))
67
68      }
69
70    }
71  }
72  func GetUserFunc(w http.ResponseWriter, r *http.Request) {    //function to perform the post and get tasks
73
74      keys, ok := r.URL.Query()["id"]
75
76      if !ok || len(keys[0]) < 1 {          // checking the parameters as passed
77        log.Println("Url Param 'key' is missing")
78        return
79      }
80
81      // Query()["key"] will return an array of items,
82      // we only want the single item
```

```
    // Query()["key"] will return an array of items,
    // we only want the single item.
    key := keys[0]

    log.Println("Url Param 'key' is: " + string(key))
    //var userDB UserDetails
  DatabaseName := "Instagram_DB"
  client, err := db.CreateDatabaseConnection(DatabaseName) // create a database if all criterias are fullfiled
  if err != nil {
    fmt.Println("Failed to connect to DB")
    panic(err)  // else print Error
  }
  defer client.Disconnect(context.TODO())  // for disconnecting Client
  ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)
  col := client.Database(DatabaseName).Collection("users_collection")
  id_key, err := strconv.Atoi(key)
  filterCursor, err := col.Find(ctx, bson.M{"id":id_key }) // this maps the user id as stroed previously in col Col
  fmt.Println(filterCursor)
  if err != nil {
    log.Fatal(err)  // if not found the key/id
  }
  var userFiltered []bson.M
  if err = filterCursor.All(ctx, &userFiltered); err != nil {
    log.Fatal(err)
  }
  fmt.Println(userFiltered)
  json.NewEncoder(w).Encode(userFiltered)  // encoded the data with json
}

func handlerequests(){    // the main handlerequests
  http.HandleFunc("/", homePage)
  http.HandleFunc("/users", AddUserFunc)
  http.HandleFunc("/users/",GetUserFunc)
  http.HandleFunc("/posts",createPosts)
  http.HandleFunc("/posts/",PostById)
  http.HandleFunc("/posts/users/",returnAllPosts)

  log.Fatal(http.ListenAndServe(":12345", nil))
```



```
    log.Fatal(err)  // if not found the key/id
  }
  var userFiltered []bson.M
  if err = filterCursor.All(ctx, &userFiltered); err != nil {
    log.Fatal(err)
  }
  fmt.Println(userFiltered)
  json.NewEncoder(w).Encode(userFiltered)  // encoded the data with json
}

func handlerequests(){    // the main handlerequests
  http.HandleFunc("/", homePage)
  http.HandleFunc("/users", AddUserFunc)
  http.HandleFunc("/users/",GetUserFunc)
  http.HandleFunc("/posts",createPosts)
  http.HandleFunc("/posts/",PostById)
  http.HandleFunc("/posts/users/",returnAllPosts)

  log.Fatal(http.ListenAndServe(":12345", nil))



}
//password encryption genrating methods
func HashPassword(password string) (string, error) {
  bytes, err := bcrypt.GenerateFromPassword([]byte(password), 14)
  return string(bytes), err
}
// checking for Password
func CheckPasswordHash(password, hash string) bool {
  err := bcrypt.CompareHashAndPassword([]byte(hash), []byte(password))
  return err == nil
}
func main() {

handlerequests()    // function call in main Method

}
//Arpit Singh
//19BCG10069
```

Next *Posts.go* similar to main.to :

However here post data model is described

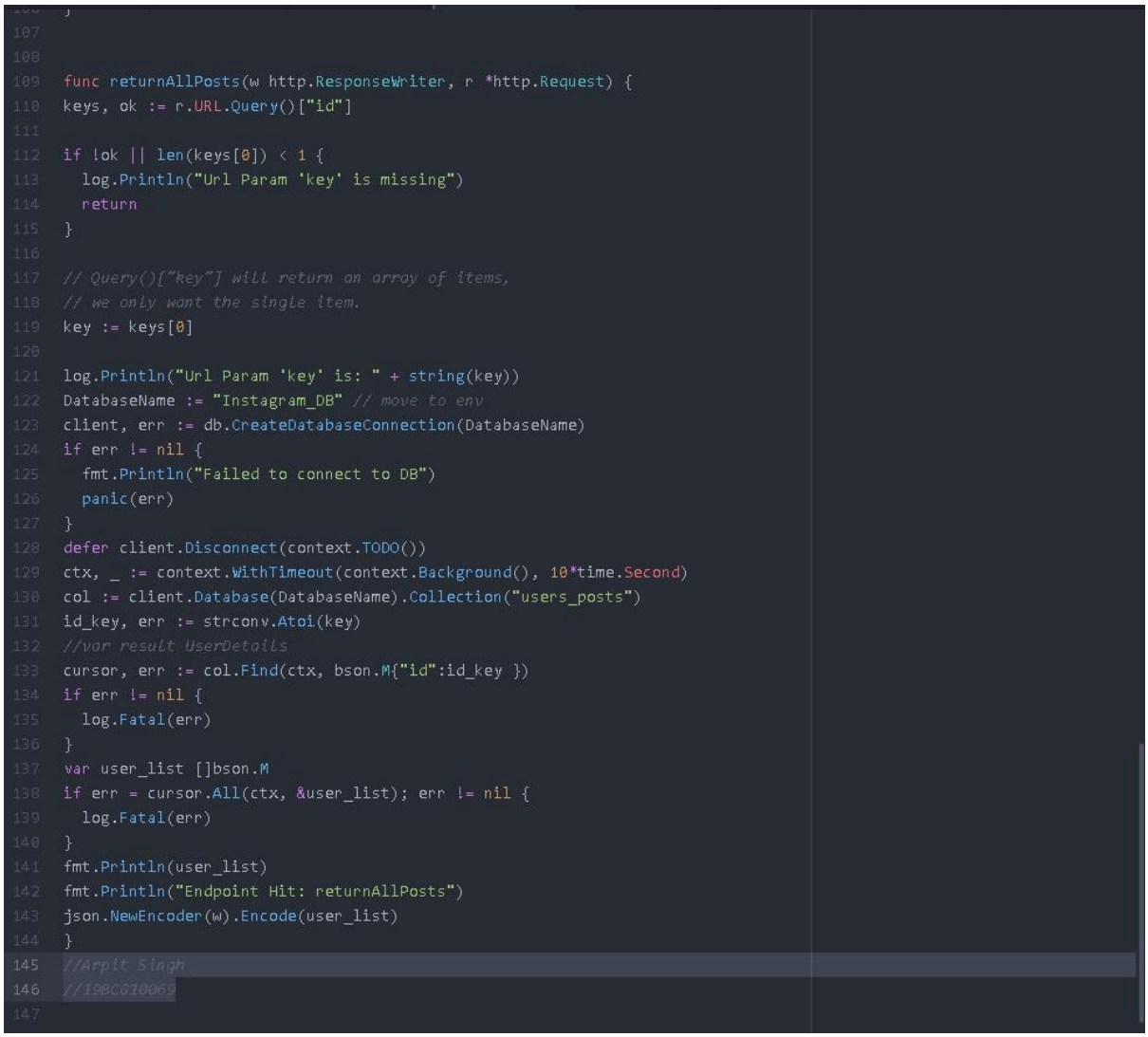
And search by postid/id logic has been implemented.

Main.go

```go
package main

import (
"TheMainAPI/db"
"context"
"encoding/json"
"fmt"
"go.mongodb.org/mongo-driver/bson"
"io/ioutil"
"log"
"net/http"
"os"
"reflect"
"strconv"
"time"
)

//Struct for post collection as described in the Task
type postsDetails struct {

Id  int `json: "Id"`      // id and post id given unique integer id for furture uses
PostId  int `json: "PostId"`
Caption  string    `json: "Caption"`
ImageURL string   `json: "ImageURL"`
PostedTS time.Time `json: "PostedTS"` // Time selector used for time datatype
}
var PostArr []postsDetails  // all post details stored in this array
func createPosts(w http.ResponseWriter, r *http.Request){
if r.Method == "POST" {

  reqBody, _ := ioutil.ReadAll(r.Body)  // r used for request var.
  fmt.Fprintf(w, "%+v", string(reqBody))
  var posts postsDetails  // var to add and include postsDetails
  json.Unmarshal(reqBody, &posts)
  // update our global post array to include
  // our new post with the help of appendbackwards.
  PostArr = append(PostArr, posts)

  json.NewEncoder(w).Encode(posts)  // encoded the data in json format

  fmt.Println("Endpoint : posts")
  DatabaseName := "Instagram_DB" // move to env
```

```go
    fmt.Println("Endpoint : posts")
    DatabaseName := "Instagram_DB" // move to env
    client, err := db.CreateDatabaseConnection(DatabaseName)
    if err != nil {   // here proccess is simimlar to main.go i.e. for the creation and connection to Server
      fmt.Println("Failed to connect to DB")
      panic(err)
    }
    defer client.Disconnect(context.TODO())

    ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)
    col := client.Database(DatabaseName).Collection("users_posts")
    result, insertErr := col.InsertOne(ctx,posts)   // insertion of posts proccess
    if insertErr != nil {
      fmt.Println("InsertONE Error:", insertErr)
      os.Exit(1)
    } else {
      fmt.Println("InsertOne() result type: ", reflect.TypeOf(result))
      fmt.Println("InsertOne() api result type: ", result)

      newID := result.InsertedID      // seprate insertion Id assigned each Time
      fmt.Println("InsertedOne(), newID", newID)
      fmt.Println("InsertedOne(), newID type:", reflect.TypeOf(newID))

    }

}

}

func PostById(w http.ResponseWriter, r *http.Request) {   // function/Logic to search the post with postid method

keys, ok := r.URL.Query()["postid"]

if !ok || len(keys[0]) < 1 {
  log.Println("Url Param 'key' is missing")
  return
}

// Query()["key"] will return an array of items,
// we only want the single item.
key := keys[0]

```

```
// we only want the single item.
key := keys[0]

log.Println("Url Param 'key' is: " + string(key))
//var userDB UserDetails
DatabaseName := "Instagram_DB"
client, err := db.CreateDatabaseConnection(DatabaseName)
if err != nil {
  fmt.Println("Failed to connect to DB")
  panic(err)
}
defer client.Disconnect(context.TODO())
ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)
col := client.Database(DatabaseName).Collection("users_posts")
id_key, err := strconv.Atoi(key)  // similar to main.go just instead of key here postid is used
filterCursor, err := col.Find(ctx, bson.M{"postid":id_key })
fmt.Println(filterCursor)
if err != nil {
  log.Fatal(err)
}
var postFiltered []bson.M
if err = filterCursor.All(ctx, &postFiltered); err != nil {
  log.Fatal(err)
}
fmt.Println(postFiltered) // mapped with postid and added to the collection in json format.
json.NewEncoder(w).Encode(postFiltered)
}


func returnAllPosts(w http.ResponseWriter, r *http.Request) {
keys, ok := r.URL.Query()["id"]

if !ok || len(keys[0]) < 1 {
  log.Println("Url Param 'key' is missing")
  return
}

// Query()["key"] will return an array of items,
// we only want the single item.
key := keys[0]

log.Println("Url Param 'key' is: " + string(key))
```



```
}


func returnAllPosts(w http.ResponseWriter, r *http.Request) {
keys, ok := r.URL.Query()["id"]

if !ok || len(keys[0]) < 1 {
  log.Println("Url Param 'key' is missing")
  return
}

// Query()["key"] will return an array of items,
// we only want the single item.
key := keys[0]

log.Println("Url Param 'key' is: " + string(key))
DatabaseName := "Instagram_DB" // move to env
client, err := db.CreateDatabaseConnection(DatabaseName)
if err != nil {
  fmt.Println("Failed to connect to DB")
  panic(err)
}
defer client.Disconnect(context.TODO())
ctx, _ := context.WithTimeout(context.Background(), 10*time.Second)
col := client.Database(DatabaseName).Collection("users_posts")
id_key, err := strconv.Atoi(key)
//var result UserDetails
cursor, err := col.Find(ctx, bson.M{"id":id_key })
if err != nil {
  log.Fatal(err)
}
var user_list []bson.M
if err = cursor.All(ctx, &user_list); err != nil {
  log.Fatal(err)
}
fmt.Println(user_list)
fmt.Println("Endpoint Hit: returnAllPosts")
json.NewEncoder(w).Encode(user_list)
}
//Arpit Singh
//19BCG10065

```

Next we have *go.sum and go.mod:*

These 2 files are used to store and get all the required dependencies.

Mod – module.

Sum – checks the cryptographic checksums.

GO.SUM:

```
// This file consists of all the dependencies used.




github.com/BurntSushi/toml v0.3.1/go.mod h1:xHWCNGjB5oqiDr8zfno3MHue2Ht5sIBksp03qcyfWMU=
github.com/davecgh/go-spew v1.1.0/go.mod h1:J7Y8YcW2NihsgmVo/mv3lAwl/skON4iLHjSsI+c5H38=
github.com/davecgh/go-spew v1.1.1/go.mod h1:J7Y8YcW2NihsgmVo/mv3lAwl/skON4iLHjSsI+c5H38=
github.com/go-stack/stack v1.8.0 h1:5SgMzNM5HxrEjV0ww2lTmX6E2Izsfxas4+YHWRs3Lsk=
github.com/go-stack/stack v1.8.0/go.mod h1:v0f6uXyyMGvRgIKkXu+yp6POWl0qKG85gN/melR3HDY=
github.com/gobuffalo/attrs v0.0.0-20190224210810-a9411de4debd/go.mod h1:4duuawTqi2wkkpB4ePgWMaai6/Kc6WEz83bhFwpHzj8
github.com/gobuffalo/depgen v0.0.0-20190329151759-d478694a28d3/go.mod h1:3STtPUQYuzV0gBVOY3vy6CfMm/ljR4pABfrTeHNLHUY=
github.com/gobuffalo/depgen v0.1.0/go.mod h1:+ifsuy7fhi15RWncXQQKjWS9JPkdah5sZvtHc2RXGlg=
github.com/gobuffalo/envy v1.6.15/go.mod h1:n7DRkBerg/aorDM8kbduw5dN3oXGswK5liaSCx4T5NI=
github.com/gobuffalo/envy v1.7.0/go.mod h1:n7DRkBerg/aorDM8kbduw5dN3oXGswK5liaSCx4T5NI=
github.com/gobuffalo/flect v0.1.0/go.mod h1:d2ehjJqGOH/Kjqcoz+F7jHTBbmDb38yXA598Hb50EGs=
github.com/gobuffalo/flect v0.1.1/go.mod h1:8JCgGVbRjJhVgD6399mQr4fx5rRfGKVzFjbj6RE/9UI=
github.com/gobuffalo/flect v0.1.3/go.mod h1:8JCgGVbRjJhVgD6399mQr4fx5rRfGKVzFjbj6RE/9UI=
github.com/gobuffalo/genny v0.0.0-20190329151137-27723ad26ef9/go.mod h1:rWs4Z12d1Zbf19rlsn0nurr75KqhYp52EAGGxTbBhNk
github.com/gobuffalo/genny v0.0.0-20190403191548-3ca520ef0d9e/go.mod h1:80lIj3kVJWwOrXWWMRzzdhW3DsrdjILVil/SFKBzF28
github.com/gobuffalo/genny v0.1.0/go.mod h1:XidbUqzak3lHdS//TPu2OgiFB+51Ur5f7CSnXZ/JDvo=
github.com/gobuffalo/genny v0.1.1/go.mod h1:5TExbEyY48pfunL4QSXxlDOmdsD44RRq4mVZ0Ex28Xk=
github.com/gobuffalo/gitgen v0.0.0-20190315122116-cc086187d211/go.mod h1:vEHJk/E9DmhejeLeNt7UVvlSGv3ziL+djtTr3yyzcC
github.com/gobuffalo/gogen v0.0.0-20190315121717-8f38393713f5/go.mod h1:V9QVDIxsgKNZs6L2IYiGR8datgMhB577vzTDqypH360
github.com/gobuffalo/gogen v0.1.0/go.mod h1:8NTelM5qd8RZ15VjQTFkAW6qOMx5wBbW4dSCS3BY8gg=
github.com/gobuffalo/gogen v0.1.1/go.mod h1:y8iBtmHmGc4qa3urIyo1shvOD8JftTtfcKi+71xfDNE=
github.com/gobuffalo/logger v0.0.0-20190315122211-86e12af44bc2/go.mod h1:QdxcLw541hSGtBnhUc4gaNIXRjiDppFGaDqzbrBd3v
github.com/gobuffalo/mapi v1.0.1/go.mod h1:4VAGh89y6rVOvm5A8fKFxYG+wIW6LO1FMTG9hnKStFc=
github.com/gobuffalo/mapi v1.0.2/go.mod h1:4VAGh89y6rVOvm5A8fKFxYG+wIW6LO1FMTG9hnKStFc=
github.com/gobuffalo/packd v0.0.0-20190315124812-a385830c7fc0/go.mod h1:M2Juc+hhDXf/PnmBANFCqx4DM3wRbgDvnVWeG2RIxq4
github.com/gobuffalo/packd v0.1.0/go.mod h1:M2Juc+hhDXf/PnmBANFCqx4DM3wRbgDvnVWeG2RIxq4=
github.com/gobuffalo/packr/v2 v2.0.9/go.mod h1:emmyGweYTm6Kdper+iywB6YK5YzuKchGtJQZ0Odn4pQ=
github.com/gobuffalo/packr/v2 v2.2.0/go.mod h1:CaAwI0GPIAv+5wKLtv8Afwl+Cm78K/I/VCm/3ptBN+0=
github.com/gobuffalo/syncx v0.0.0-20190224160051-33c29581e754/go.mod h1:HhnNqWY95UYwwW3uSASeV7vtgYkT2t16hJgV3AEPUpw
github.com/golang/snappy v0.0.1 h1:Qgr9rKW7uDUkrbSmQeiDsGa8SjGyCOGtuasMWwvp2P4=
github.com/golang/snappy v0.0.1/go.mod h1:/XxbfmMg8lxefKM7IXC3fBNl/7bRcc72aCRzEWrmP2Q=
github.com/google/go-cmp v0.5.2/go.mod h1:v8dTdLbMG2kIc/vJvl+f65V22dbkXbowE6jgT/gNBxE=
github.com/gorilla/mux v1.8.0 h1:i40aqfkR1h2SlN9hojwV5ZA91wcXFOvkdNIeFDP5koI=
github.com/gorilla/mux v1.8.0/go.mod h1:DVbg23sWSpFRCP0SfiEN6jmj59UnW/n46BH5rLB71So=
github.com/inconshreveable/mousetrap v1.0.0/go.mod h1:PxqpIevigyE2G7u3NXJIT2ANytuPF1OarO4DADm73n8=
```

```
github.com/joho/godotenv v1.3.0/go.mod h1:7hK45KPybAkOC6peb+G5yklZfMxEjkZhHbwpqxOKXbg=
github.com/karrick/godirwalk v1.8.0/go.mod h1:H5KPZjojv4lE+QYImBI8xVtrBRgYrIVsaRPx4tDPEn4=
github.com/karrick/godirwalk v1.10.3/go.mod h1:RoGL9dQei4vP9ilrpETWE8CLOZ1kiN0LhBygSwrAsHA=
github.com/klauspost/compress v1.13.6 h1:P76CopJELS0TiO2mebmnzgWaajssP/EszplttgQxcgc=
github.com/klauspost/compress v1.13.6/go.mod h1:/3/Vjq9QcHkK5uEr5lBEmyoZ1iFhe47etQ6QUkpK6sk=
github.com/konsorten/go-windows-terminal-sequences v1.0.1/go.mod h1:T0+1ngSBFLxvqU3pZ+m/2kptfBszLMUkC4ZK/EgS/cQ=
github.com/konsorten/go-windows-terminal-sequences v1.0.2/go.mod h1:T0+1ngSBFLxvqU3pZ+m/2kptfBszLMUkC4ZK/EgS/cQ=
github.com/kr/pretty v0.1.0/go.mod h1:dAy3ld7l9f0ibDNOQOHHMYYIbhfbHSm3C4ZsoJORNo=
github.com/kr/pty v1.1.1/go.mod h1:pFQYn66WHrOpPYNljwOMqo10TkYh1fy3cYio2l3bCsQ=
github.com/kr/text v0.1.0/go.mod h1:4Jbv+DJW3UT/LiOwJeYQe1efqtUx/iVham/4vfdArNI=
github.com/markbates/oncer v0.0.0-20181203154359-bf2de49a0be2/go.mod h1:Ld9puTsIW75CHf65OeIOkyKbteujpZVXDpWK6YGZbxE
github.com/markbates/safe v1.0.1/go.mod h1:nAqgmRi7cY2nqMc92/bSEeQA+R4OheNU2T1kNSCBdG0=
github.com/montanaflynn/stats v0.0.0-20171201202039-1bf9dbcd8cbe/go.mod h1:wL8QJuTMNUDYhXwkmfOly8iTdp5TEcJFWZD2D7SI
github.com/pelletier/go-toml v1.7.0/go.mod h1:vwGMzjaWMwyfHwgIBhI2YUM4fB6nL6lVAvS1LBMMhTE=
github.com/pkg/errors v0.8.0/go.mod h1:bwawxfHBFNV+L2hUp1rHADufV3IMtnDRdf1r5NINEl0=
github.com/pkg/errors v0.8.1/go.mod h1:bwawxfHBFNV+L2hUp1rHADufV3IMtnDRdf1r5NINEl0=
github.com/pkg/errors v0.9.1 h1:FEBLx1zS214owpjy7qsBeixbURkuhQAwrK5UwLGTwt4=
github.com/pkg/errors v0.9.1/go.mod h1:bwawxfHBFNV+L2hUp1rHADufV3IMtnDRdf1r5NINEl0=
github.com/pmezard/go-difflib v1.0.0/go.mod h1:iKH77koFhYxTK1pcRnkKkqfTogsbg7gZNVY4sRDYZ/4=
github.com/rogpeppe/go-internal v1.1.0/go.mod h1:M8bDsm7K2OlrFYOpmOWEs/qY81heoFRclV5y23lUDJ4=
github.com/rogpeppe/go-internal v1.2.2/go.mod h1:M8bDsm7K2OlrFYOpmOWEs/qY81heoFRclV5y23lUDJ4=
github.com/rogpeppe/go-internal v1.3.0/go.mod h1:M8bDsm7K2OlrFYOpmOWEs/qY81heoFRclV5y23lUDJ4=
github.com/sirupsen/logrus v1.4.0/go.mod h1:LxeOpSwHxABJmUn/MG1IvRgCAasNZTLOkJPxbbu5VWo=
github.com/sirupsen/logrus v1.4.1/go.mod h1:ni0Sbl8bgC9z8RoU9G6nDWqqs/fq4eDPysMBDgk/93Q=
github.com/sirupsen/logrus v1.4.2/go.mod h1:tLMulIdttU9McNUspp0xgXVQah82FyeX6MwdIuYE2rE=
github.com/spf13/cobra v0.0.3/go.mod h1:1l0Ry5zgKvJasoi3XT1TypsSe7PqH0Sj9dhYf7v3XqQ=
github.com/spf13/pflag v1.0.3/go.mod h1:DYY7MBk1bdzusC3SYhjObp+wFpr4gzcvqqNjLnInEg4=
github.com/stretchr/objx v0.1.0/go.mod h1:HFkY916IF+rwdDfMAkV7OtwuqBVzrE8GR6GFx+wExME=
github.com/stretchr/objx v0.1.1/go.mod h1:HFkY916IF+rwdDfMAkV7OtwuqBVzrE8GR6GFx+wExME=
github.com/stretchr/testify v1.2.2/go.mod h1:a8OnRcib4nhh0OaRAV+Yts87kKdq0PP7pXfy6kDkUVs=
github.com/stretchr/testify v1.3.0/go.mod h1:M5WIy9Dh21IEIfnGCwXGc5bZfKNJtfHm1UVUgZn+9EI=
github.com/stretchr/testify v1.6.1/go.mod h1:6Fq8oRcR53rry900zMqJjRRixrwX3KX962/h/Wwjteg=
github.com/tidwall/pretty v1.0.0/go.mod h1:XNkn88O1ChpSDQmQeStsy+sBenx6DDtFZJxhVysOjyk=
github.com/xdg-go/pbkdf2 v1.0.0 h1:Su7DPu48wXMwC3bs7MCNG+z4FhcyEuz5dlvchbq0B0c=
github.com/xdg-go/pbkdf2 v1.0.0/go.mod h1:jrpuAogTd400dnrH08LKmI/xc1MbPOebTwRqcT5RDeI=
github.com/xdg-go/scram v1.0.2 h1:akYIkZ28e6A96dkWNJQu3nmCzH3YfwMPQExUYDaRv7w=
github.com/xdg-go/scram v1.0.2/go.mod h1:1WAq6h33pAW+iRreB34OORO2Nf7qel3VV3fjBj+hCSs=
github.com/xdg-go/stringprep v1.0.2 h1:6iq84/ryjjeRmMJwxutI51F2GIPlP5BfTvXHeYjyhBc=
github.com/xdg-go/stringprep v1.0.2/go.mod h1:8F9zXuvzgwmyT5DUm4GUfZGDdT3W+LCvS6+da4O5kxM=
github.com/youmark/pkcs8 v0.0.0-20181117223130-1be2e3e5546d h1:splanxYIlg+5LfHAM6xpdFEAYOk8iySO56hMFq6uLyA=
github.com/youmark/pkcs8 v0.0.0-20181117223130-1be2e3e5546d/go.mod h1:rHwXgn7JulP+udvsHwJoVG1YGAP6VLg4y9I5dyZdqmA=
```

```
github.com/xdg-go/scram v1.0.2/go.mod h1:1WAq6h33pAW+iRreB34OORO2Nf7qel3VV3fjBj+hCSs=
github.com/xdg-go/stringprep v1.0.2 h1:6iq84/ryjjeRmMJwxutI51F2GIPlP5BfTvXHeYjyhBc=
github.com/xdg-go/stringprep v1.0.2/go.mod h1:8F9zXuvzgwmyT5DUm4GUfZGDdT3W+LCvS6+da4O5kxM=
github.com/youmark/pkcs8 v0.0.0-20181117223130-1be2e3e5546d h1:splanxYIlg+5LfHAM6xpdFEAYOk8iySO56hMFq6uLyA=
github.com/youmark/pkcs8 v0.0.0-20181117223130-1be2e3e5546d/go.mod h1:rHwXgn7JulP+udvsHwJoVG1YGAP6VLg4y9I5dyZdqmA=
go.mongodb.org/mongo-driver v1.7.3 h1:G4l/eYY9VrQAK/AUgkV0koQKzQnyddnWxrd/Etf0jIs=
go.mongodb.org/mongo-driver v1.7.3/go.mod h1:NqaYOwnXWr5Pm7AOpO5QFxKJ503nbMse/R79oO62zWg=
golang.org/x/crypto v0.0.0-20180904163835-0709b304e793/go.mod h1:6SG95UA2DQfeDnfUPMdvaQW0Q7yPrPDi9nlGo2tz2b4=
golang.org/x/crypto v0.0.0-20190308221718-c2843e01d9a2/go.mod h1:djNgcEr1/C05ACkg1iLfiJU5Ep61QUkGW8qpdssI0+w=
golang.org/x/crypto v0.0.0-20190422162423-af44ce270edf/go.mod h1:WFFai1msRO1wXaEeE5yQxYXgSfI8pQAWXbQop6sCtWE=
golang.org/x/crypto v0.0.0-20200302210943-78000ba7a073 h1:xMPOj6Pz6UipU1wXLkrtqpHbR0AVFnyPEQq/wRWz9lM=
golang.org/x/crypto v0.0.0-20200302210943-78000ba7a073/go.mod h1:LzIPMQfyMNhhGPhUkYOs5KpL4U8rLKemX1yGLhDgUto=
golang.org/x/net v0.0.0-20190311183353-d8887717615a/go.mod h1:t9HGtf8HONx5eT2rtn7q6eTqICYqUVnKs3thJo3Qplg=
golang.org/x/net v0.0.0-20190404232315-eb5bcb51f2a3/go.mod h1:t9HGtf8HONx5eT2rtn7q6eTqICYqUVnKs3thJo3Qplg=
golang.org/x/sync v0.0.0-20190227155943-e225da77a7e6/go.mod h1:RxMgew5VJxzue5/jJTE5uejpjVlOe/izrB70Jof72aM=
golang.org/x/sync v0.0.0-20190412183630-56d357773e84/go.mod h1:RxMgew5VJxzue5/jJTE5uejpjVlOe/izrB70Jof72aM=
golang.org/x/sync v0.0.0-20190423024810-112230192c58/go.mod h1:RxMgew5VJxzue5/jJTE5uejpjVlOe/izrB70Jof72aM=
golang.org/x/sync v0.0.0-20190911185100-cd5d95a43a6e h1:vcxGaoTs7kV8m5Np9uUNQin4BrLOthgV7252N8V+FwY=
golang.org/x/sync v0.0.0-20190911185100-cd5d95a43a6e/go.mod h1:RxMgew5VJxzue5/jJTE5uejpjVlOe/izrB70Jof72aM=
golang.org/x/sys v0.0.0-20180905080454-ebe1bf3edb33/go.mod h1:STP8DvDyc/dI5b8T5hshtkjS+E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190215142949-d0b11bdaac8a/go.mod h1:STP8DvDyc/dI5b8T5hshtkjS+E42TnysNCUPdjciGhY=
golang.org/x/sys v0.0.0-20190403152447-81d4e9dc473e/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20190412213103-97732733099d/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20190419153524-e8e3143a4f4a/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20190422165155-953cdadca894/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/sys v0.0.0-20190531175056-4c3a928424d2/go.mod h1:h1NjWce9XRLGQEsW7wpKNCjG9DtNlClVuFLEZdDNbEs=
golang.org/x/text v0.3.0/go.mod h1:NqM8EUOU14njkJ3fqMW+pc6Ldnwhi/IjpwHt7yyuwOQ=
golang.org/x/text v0.3.5 h1:i6eZZ+zk0SOf0xgBpEpPD18qWcJda6q1sxt3S0kzyUQ=
golang.org/x/text v0.3.5/go.mod h1:5Zoc/QRtKVWzQhOtBMvqHzDpF6irO9z98xDceosuGiQ=
golang.org/x/tools v0.0.0-20180917221912-90fa682c2a6e/go.mod h1:n7NCudcB/nEzxVGmLbDWY5pfWTLqBcC2KZ6jyYvM4mQ=
golang.org/x/tools v0.0.0-20190329151228-23e29df326fe/go.mod h1:LCzVGOaR6xXOjkQ3onu1FJEFr0SW1gC7cKk1uF8kGRs=
golang.org/x/tools v0.0.0-20190416151739-9c9e1878f421/go.mod h1:LCzVGOaR6xXOjkQ3onu1FJEFr0SW1gC7cKk1uF8kGRs=
golang.org/x/tools v0.0.0-20190420181800-aa740d480789/go.mod h1:LCzVGOaR6xXOjkQ3onu1FJEFr0SW1gC7cKk1uF8kGRs=
golang.org/x/tools v0.0.0-20190531172133-b3315ee88b7d/go.mod h1:/rFqwRUd4F7ZHNgwSSTFct+R/Kf4OFW1sUzUTQQTgfc=
golang.org/x/xerrors v0.0.0-20191204190536-9bdfabe68543/go.mod h1:I/5z698sn9Ka8TeJc9MKroUUfqBBauWjQqLJ2OPfmY0=
gopkg.in/check.v1 v0.0.0-20161208181325-20d25e280405/go.mod h1:Co6ibVJAznAaIkqp8huTwlJQCZ016jof/cbN4VW5Yz0=
gopkg.in/check.v1 v1.0.0-20180628173108-788fd7840127/go.mod h1:Co6ibVJAznAaIkqp8huTwlJQCZ016jof/cbN4VW5Yz0=
gopkg.in/errgo.v2 v2.1.0/go.mod h1:hNsd1EY+bozCKY1Ytp96fpM3vjJbqLJn88ws8XvfDNI=
gopkg.in/yaml.v2 v2.2.8/go.mod h1:hI93XBmqTisBFMUTm0b8Fm+jr3Dg1NNxqwp+5A1VGuI=
gopkg.in/yaml.v3 v3.0.0-20200313102051-9f266ea9e77c/go.mod h1:K4uyk7z7BCEPqu6E+C64Yfv1cQ7kz7rIZviUmN+EgEM=
```

GO.MOD:

```
// this file consists of all github repos used along with there versions
module TheMainAPI

go 1.17

require (
  github.com/go-stack/stack v1.8.0 // indirect
  github.com/golang/snappy v0.0.1 // indirect
  github.com/gorilla/mux v1.8.0 // indirect
  github.com/klauspost/compress v1.13.6 // indirect
  github.com/pkg/errors v0.9.1 // indirect
  github.com/xdg-go/pbkdf2 v1.0.0 // indirect
  github.com/xdg-go/scram v1.0.2 // indirect
  github.com/xdg-go/stringprep v1.0.2 // indirect
  github.com/youmark/pkcs8 v0.0.0-20181117223130-1be2e3e5546d // indirect
  go.mongodb.org/mongo-driver v1.7.3 // indirect
  golang.org/x/crypto v0.0.0-20200302210943-78000ba7a073 // indirect
  golang.org/x/sync v0.0.0-20190911185100-cd5d95a43a6e // indirect
  golang.org/x/text v0.3.5 // indirect
)

```

Lastly we have a Folder Database(Database>db.go):

Here we setup a connection with MongoDB server and Check it as well *(as done at start of document).

Db.go

```go
// as described in the word document here we establish a connection to mongoDB server
package db

import (
  "context"
  "fmt"
  "go.mongodb.org/mongo-driver/mongo"
  "go.mongodb.org/mongo-driver/mongo/options"
  "log"
)

func CreateDatabaseConnection(dbName string) (*mongo.Client, error) { // function to create a connectionwith mongodb
  clientOptions := options.Client().ApplyURI("mongodb://localhost:27017")

  // Connect to MongoDB
  client, err := mongo.Connect(context.TODO(), clientOptions)

  if err != nil {
    log.Fatal(err)      // if unable print Error
  }

  // Check the connection
  err = client.Ping(context.TODO(), nil)

  if err != nil {
    log.Fatal(err)
  }

  fmt.Println("Connected to MongoDB!")
  return client, err
}

```

The Folder .Xmlfiles has been added to give browser feasibility.

All the files can be accessed at:

https://github.com/TSM-ArpitSG/InstagramAPI