REPORT 03

Full name: Pham Ngoc Tam
Teacher: Nguyen Thanh Hoa

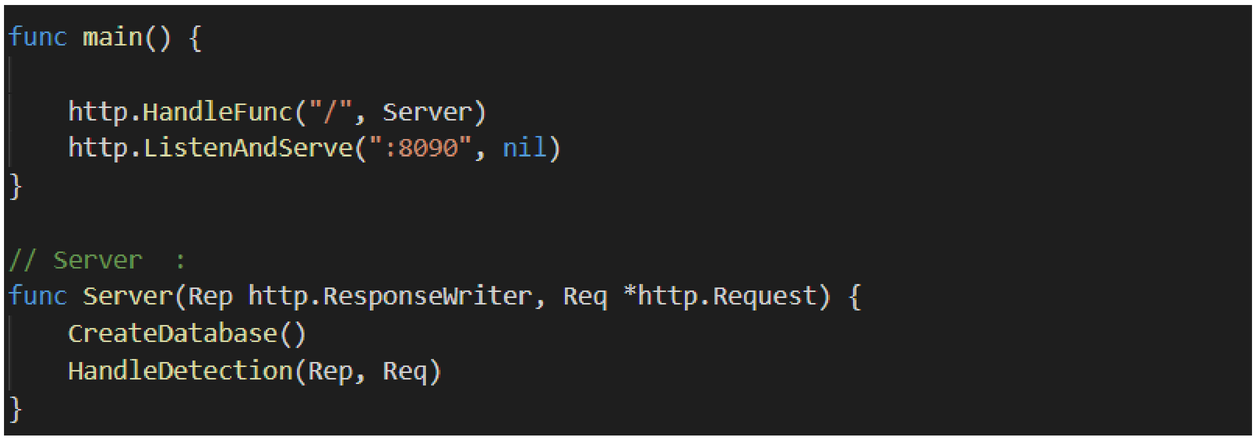
I.Report content

1.Project purpose

- Use ingredients of report 02 consist of Server.go, Client.go, but data will be saved in MySQL server, and when Client request data which existed in database, Server will reponse data from there, on the contrary, new data will be saved into database.

2.Ingredients of source code

- **Server:**



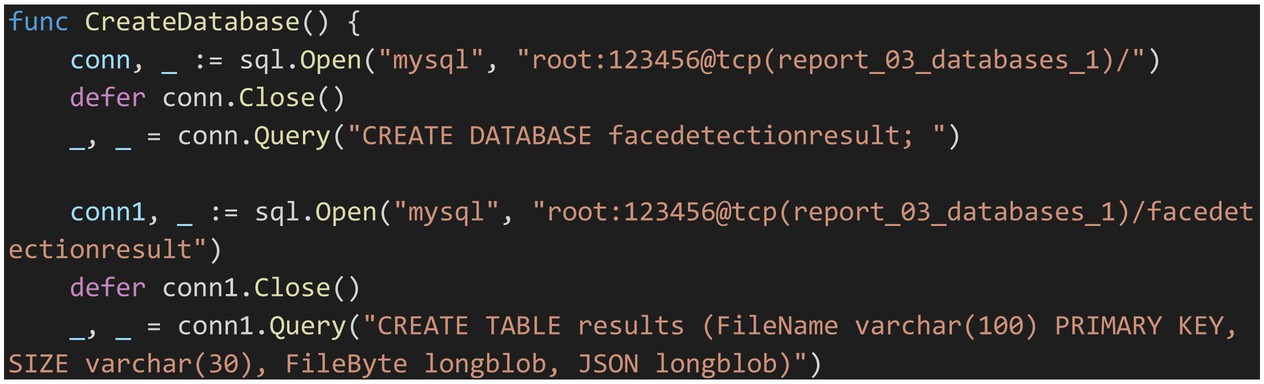
```
func main() {

    http.HandleFunc("/", Server)
    http.ListenAndServe(":8090", nil)
}

// Server  :
func Server(Rep http.ResponseWriter, Req *http.Request) {
    CreateDatabase()
    HandleDetection(Rep, Req)
}
```

➔At Server, First, Create a database which data of client will be saved in, and then implement main work **HandleDetection()**

+ **CreateDatabase**: create a database saving data client



```
func CreateDatabase() {
    conn, _ := sql.Open("mysql", "root:123456@tcp(report_03_databases_1)/")
    defer conn.Close()
    _, _ = conn.Query("CREATE DATABASE facedetectionresult; ")

    conn1, _ := sql.Open("mysql", "root:123456@tcp(report_03_databases_1)/facedet
ectionresult")
    defer conn1.Close()
    _, _ = conn1.Query("CREATE TABLE results (FileName varchar(100) PRIMARY KEY,
SIZE varchar(30), FileByte longblob, JSON longblob)")
```

➔Connect to MySQL server and create a database **facedetectionresult,** and then create a table result with 4 column: FileName, Size, FileByte, JSON. FileByte saves image under Base64 type and JSON saves JSON type about coordinate of face

+ HandleDetection()

```
func HandleDetection(Rep http.ResponseWriter, Req *http.Request) {
    if err := Req.ParseMultipartForm(1024 * 1024 * 10); err != nil {
        http.Error(Rep, err.Error(), http.StatusBadRequest)
        return
    }

    for key, value := range Req.MultipartForm.File {
        Option := Req.FormValue("Option")
        if key == "FileUpload" {
            for _, oneFileOfMultiFile := range value {
                check := IsAlreadyExistInDatabase(oneFileOfMultiFile)
                if check == true {
                    if Option == "1" {
                        ResponseImageToClient(oneFileOfMultiFile, Rep)
                    } else {
                        ResponseJSONtoClient(oneFileOfMultiFile, Rep)
                    }
                } else {
                    saveFileintoServer(oneFileOfMultiFile)
                    saveOriginalFileToDatabase(oneFileOfMultiFile)
                    Detection(Rep, Req)
                    SaveResultImageIntoDatabase(oneFileOfMultiFile.Filename)
                    saveResultJSONintoDatabse(oneFileOfMultiFile.Filename)
                    if Option == "1" {
                        ResponseImageToClient(oneFileOfMultiFile, Rep)
                    } else {
                        ResponseJSONtoClient(oneFileOfMultiFile, Rep)
                    }
                    os.Remove("ImageOut/" + oneFileOfMultiFile.Filename)
                    os.Remove("output.json")
                }
            }
        }
    }
}
```

➔Firstly, check condition, if IsAlreadyExistInDatabase is True, mean image existed in database so get information and response to Client immediately. On the contrary, Implement saving data into database with necessary information and then response to Client.

For more detail about functions, refer to Souce code attach with Report.

- **Client:**

```
-   func main() {
-       Client()
-       http.ListenAndServe(":8093", nil)
-   }
-
-   // Client :
-   func Client() {
-       PathIn := "Image/"
-       Option := "1"
-       uploadImage(PathIn, Option)
-   }
```

➔Because I am going to deploy in Docker container automatically, so I expose Option is 1. Constantly, Server will be response Image.

+ uploadImage()

```
func uploadImage(Path string, Option string) {

    files, _ := ioutil.ReadDir(Path)
    for _, file := range files {
        if filepath.Ext(file.Name()) != ".jpg" {
            continue
        }
        var bodyRequest bytes.Buffer
        MultiWriter := multipart.NewWriter(&bodyRequest)

        FileImageUp, _ := os.Open(Path + "/" + file.Name())
        defer FileImageUp.Close()
        WriterFile, _ := MultiWriter.CreateFormFile("FileUpload", file.Name())
        _, err := io.Copy(WriterFile, FileImageUp)
        if err != nil {
            log.Fatal(err)
        }

        MultiWriter.WriteField("Option", Option)
        MultiWriter.Close()
        MakeRequest(MultiWriter, bodyRequest, Option)
    }
}
```

➔We send image need to detect to Server, and get response image saved in folder Result.

****Detail of functions, refer to functions of report 02, it same each other*.**

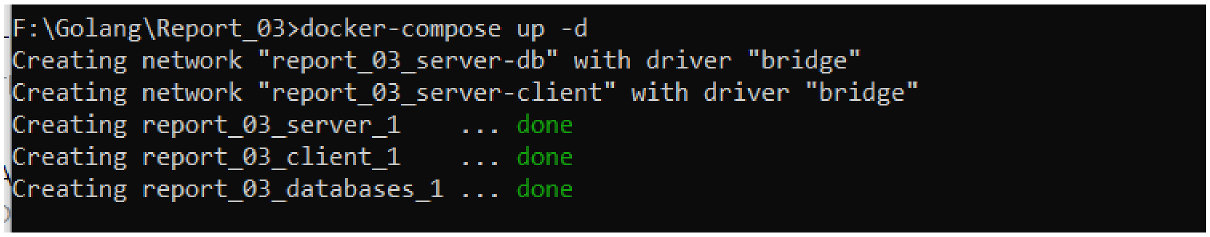
- **Docker-compose file:**

```
-   version: '3.7'
-   services:
-       databases:
-           image: mysql:latest
-           command: --default-authentication-plugin=mysql_native_password
-           networks:
-               - server-db
-           ports:
-           - "3306:3306"
-           environment:
-           - MYSQL_ROOT_PASSWORD=123456
-       server:
-           image: server-image
-           networks:
-               - server-db
-               - server-client
-           ports:
-           - "8090:8090"
-           volumes:
-               - type: volume
-                 source: server-data
-                 target: /usr/src/app/FaceD
-       client:
-           image: client-image
-           networks:
-               - server-client
-           ports:
-           - "8093:8093"
-           restart: always
-           volumes:
-               - type: volume
-                 source: client-data
-                 target: /usr/src/app/FaceD
-   networks:
-       server-db:
-           driver: bridge
-       server-client:
-           driver: bridge
-   volumes:
-       client-data:
-       server-data:
```

➔Build images from Dockerfile as Report 02, but this times deploy in docker-compose file, use images already existed.

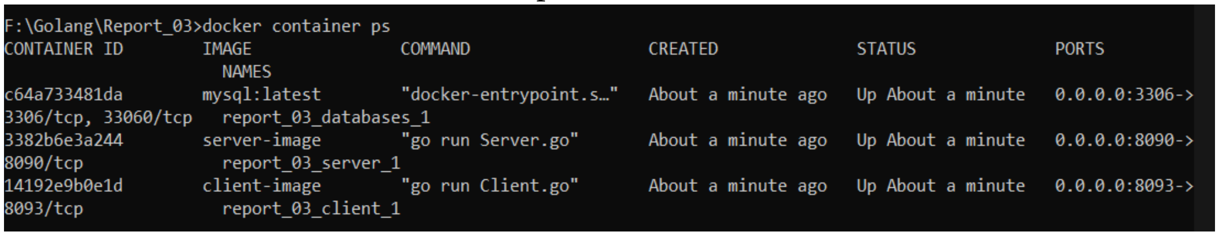
3.Test case and Result

Run command line: docker-compose up -d at directory of folder report



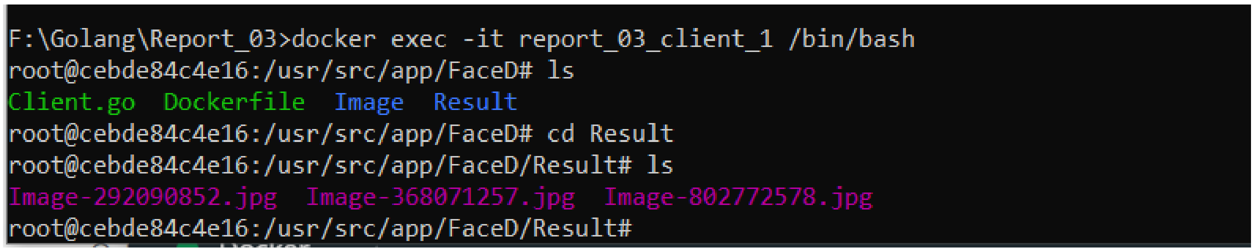
```
F:\Golang\Report_03>docker-compose up -d
Creating network "report_03_server-db" with driver "bridge"
Creating network "report_03_server-client" with driver "bridge"
Creating report_03_server_1    ... done
Creating report_03_client_1    ... done
Creating report_03_databases_1 ... done
```

Check container : docker container ps



```
F:\Golang\Report_03>docker container ps
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS              PORTS
                      NAMES
c64a733481da        mysql:latest        "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->
3306/tcp, 33060/tcp   report_03_databases_1
3382b6e3a244        server-image        "go run Server.go"       About a minute ago   Up About a minute   0.0.0.0:8090->
8090/tcp              report_03_server_1
14192e9b0e1d        client-image        "go run Client.go"       About a minute ago   Up About a minute   0.0.0.0:8093->
8093/tcp              report_03_client_1
```

Now, show terminal of Client and see Result



```
F:\Golang\Report_03>docker exec -it report_03_client_1 /bin/bash
root@cebde84c4e16:/usr/src/app/FaceD# ls
Client.go  Dockerfile  Image  Result
root@cebde84c4e16:/usr/src/app/FaceD# cd Result
root@cebde84c4e16:/usr/src/app/FaceD/Result# ls
Image-292090852.jpg  Image-368071257.jpg  Image-802772578.jpg
root@cebde84c4e16:/usr/src/app/FaceD/Result#
```

----END----