LAPORAN TEST SCENARIO
TESTING DAN IMPLEMENTASI SISTEM

Anggota Kelompok :

1119002 - Albertus Septian Angkuw
1119019 - William Surjana
1119038 - Elangel Neilea Shaday

INSTITUT TEKNOLOGI HARAPAN BANGSA
DEPARTEMEN INFORMATIKA

TAHUN 2022

A. DESKRIPSI APLIKASI

E-Music adalah aplikasi mobile yang berisi playlist lagu, album lagu, serta penyanyi. Aplikasi ini dibuat dengan tujuan mempermudah user untuk mendengarkan lagu secara online serta mudah digunakan dan mudah diakses kapan saja dan dimana saja. Aplikasi ini dibuat menggunakan bahasa pemrograman kotlin untuk frontend dan bahasa pemrograman golang untuk backend.

Sumber kode yang sudah dibuat sebelumnya bisa diakses melalui repository https://github.com/albertusangkuw/TugasBesarAndroidEMusic

Pada E-Music terdapat beberapa page, antara lain:

1. Login Page

Pada halaman ini, user diminta untuk memasukkan email dan password untuk bisa masuk kedalam aplikasi E-Music. Selain lewat email, user bisa menggunakan facebook untuk masuk kedalam E-Music. Pada halaman ini juga terdapat pilihan untuk pindah ke page sign up/register apabila user belum memiliki akun pada E-Music. Tersedia juga pilihan untuk reset password jika user lupa password dan pilihan setting untuk mengatur aplikasi E-Music.

2. Register Page

Halaman ini biasanya dipakai apabila user belum memiliki akun. Halaman ini berfungsi untuk membuat akun baru untuk user. Pada halaman ini user akan diminta memasukkan email terlebih dahulu dan dilanjutkan dengan mengisikan password sehingga proses pembuatan akun bisa dilanjutkan.

3. Home Page

Pada halaman ini biasanya berisi daftar playlist yang sering diputar, top chart lagu, serta recommended playlist. Home page ini dibuat agar user lebih mudah dalam memilih lagu.

4. Search Page

Halaman ini dibuat untuk memudahkan user mencari musik atau lagu yang diinginkan. User bisa memasukkan nama lagu, artis, nama music, atau nama podcast/playlist ke dalam textbox. Selain itu ada beberapa pilihan genre music yang disesuaikan dengan riwayat music yang diputar pada device user. Selanjutnya, pada halaman ini juga user bisa mencari musik yang memiliki genre-genre tertentu serta peringkat musik top dunia.

5. Profile Artist Page

Halaman ini merupakan suatu tampilan untuk menunjukkan seluruh album yang dimiliki oleh penyanyi / artis / grup music tertentu. Selain menampilkan seluruh album, pada halaman ini user bisa melihat jumlah pendengar music tersebut, user juga bisa melakukan shuffle play pada halaman ini, serta bisa memfollow artis tersebut.

6. Profile Regular User Page

Halaman ini merupakan suatu tampilan untuk menunjukkan profile user. Pada halaman ini memuat jumlah playlist user, jumlah followers user, jumlah following user, serta menampilkan seluruh daftar playlist yang dimiliki oleh user.

7. On Going Music Page

Pada halamn ini memuat suatu music atau lagu yang sedang dimainkan. Selain itu halaman ini juga terdapat beberapa button yang dapat diakses / digunakan oleh user seperti play and pause, next song, repeat, shuffle, playlist, menu, serta available device.

8. Library Page

Pada halaman ini memuat playlist, artis, serta album yang disimpan oleh user. Pada playlist akan menampilkan seluruh daftar playlist, selain itu terdapat button create playlist yang berfungsi untuk membuat playlist baru; pada artis akan menampilkan seluruh daftar artis; dan pada album akan menampilkan seluruh daftar album. Halaman ini dibuat agar user dengan mudah mengetahui lagu-lagu yang disimpan tanpa perlu melakukan searching pada search page.

B. TEST SCENARIO PROGRAM

Untuk mengetahui aplikasi berjalan dengan baik dan data tersimpan dengan baik, disini kami melakukan skenario test pada aplikasi E-Music untuk bagian backend. Bahasa pemrograman yang digunakan yaitu Go. Dengan ini kami menggunakan package standard library “testing” dan “httptest” untuk melakukan unit test pada Go.

Bagian yang akan kami uji pada aplikasi E-Music antara lain:

1. MD5 Hash from String
   Kode Function yang akan ditest

```go
func GetMD5Hash(text string) string {
    hash := md5.Sum([]byte(text))
    return hex.EncodeToString(hash[:])
}
```

   Kode untuk melakukan testing

```go
run test | debug test
func TestMD5Hash(t *testing.T) {
    tableTest := []struct {
        label    string
        text     string
        expected string
    }{
        {label: "MD5 Origin", text: "supersecret", expected: "9a618248b64db62d15b300a07b00580b"},
        {label: "MD5 with Small Origin Diff.", text: "Supersecret", expected: "6a582edbeb20c95978af4ebec2cedddc"},
    }
    for _, v := range tableTest {
        t.Run(
            v.label,
            func(t *testing.T) {
                result := GetMD5Hash(v.text)
                if v.expected ≠ result {
                    t.Error("Expected:", v.expected, ",got:", result)
                }
            },
        )
    }
}

```

2. Login API
   Kode Function yang akan ditest

```go
//UserLogin untuk melakukan pengecekan pada password dan email yang dimasukan
func UserLogin(w http.ResponseWriter, r *http.Request) {
    var response model.Response
    email := r.URL.Query()["email"]
    password := r.URL.Query()["password"]

    if len(email) == 0 || len(password) == 0 {
        controller.ResponseManager(&response, 400, "")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
        return
    }

    id, name, err := controller.UserLogin(email[0], password[0])

    if len(name) > 0 {
        GenerateToken(w, id, name, 0)
        controller.ResponseManager(&response, 200, "Success Login")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
    } else if err ≠ nil && err.Error() == "500" {
        controller.ResponseManager(&response, 500, "")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
    } else {
        controller.ResponseManager(&response, 404, "Login Failed")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(&response)
    }
}
```

Untuk mencoba function ini maka diperlukan untuk membuat request dalam bentuk http dan kemudian membuat variabel penerima hasil dari prosess http. Dari hasil/rec bisa diambil body dan diubah menjadi format JSON untuk dibaca dan dilihat hasil proses dari function.

Kode untuk melakukan testing pada Handler Login Regular User

```go
package handler

import (···
)

run test | debug test
func TestHandlerLoginRegularUser(t *testing.T) {
    tableTest := []struct {
        label    string
        email    string
        password string
        expected int
    }{
        {label: "Login with Correct Credential", email: "albertusaaa@gmail.com", password: "12345678", expected: 200},
        {label: "Login with Wrong Password", email: "dahdkj@gmail", password: "notpwd", expected: 404},
        {label: "Login with Unknown User", email: "testingtest@gmail", password: "notexistuser", expected: 404},
    }
    for _, v := range tableTest {
        t.Run(
            v.label,
            func(t *testing.T) {
                req, err := http.NewRequest("GET", "localhost:8081/login?email="+v.email+"&password="+v.password, nil)
                if err != nil {···
                }
                rec := httptest.NewRecorder()
                UserLogin(rec, req)

                res := rec.Result()
                defer res.Body.Close()

                b, err := ioutil.ReadAll(res.Body)
                if err != nil {···
                }
                if res.StatusCode != http.StatusOK {···
                }
                var bodyMapJson map[string]interface{}
                json.Unmarshal(b, &bodyMapJson)

                if v.expected != int(bodyMapJson["status"].(float64)) {
                    t.Error("Expected:", v.expected, ",got:", bodyMapJson["status"])
                }
            },
        )
    }
}
```

3. Login Regular User

Kode Function yang akan ditest

```go
//UserLogin untuk melakukan pengecekan pada password dan email yang dimasukan
func UserLogin(email string, password string) (string, string, error) {
    db := connect()
    defer db.Close()

    query := "SELECT iduser,username FROM user WHERE email=? AND password=? "

    sha := sha256.New()
    sha.Write([]byte(password))
    sha_password := hex.EncodeToString(sha.Sum(nil))

    rows, err := db.Query(query, email, sha_password)

    if err != nil {
        return "", "", errors.New("500")
    }

    //Check user login by name
    id := ""
    name := ""

    for rows.Next() {
        if err := rows.Scan(&id, &name); err != nil {
            log.Print(err.Error())
        }
    }
    if len(name) > 0 {
        return id, name, nil
    }
    return "", "", errors.New("404")
}
```

Kode untuk melakukan testing pada Login Regular User

```
run test | debug test
func TestLoginRegularUser(t *testing.T) {
    tableTest := []struct {
        label    string
        email    string
        password string
        expected struct {
            id   string
            name string
            err  error
        }
    }{
        {
            label:    "Login with Known Correct Credentials",
            email:    "albertusaaa@gmail.com",
            password: "12345678",
            expected: struct {
                id   string
                name string
                err  error
            }{
                id:   "7c20f5ef71c38abb1d0c1f1b6eb4459f",
                name: "albertusaa222",
                err:  nil,
            },
        },
        {…
        },
    }
    for _, v := range tableTest {
        t.Run(
            v.label,
            func(t *testing.T) {
                id, name, err := UserLogin(v.email, v.password)
                if v.expected.id ≠ id {
                    t.Error("Expected:", v.expected.id, ",got:", id)
                }
                if v.expected.name ≠ name {
                    t.Error("Expected:", v.expected.name, ",got:", name)
                }
                if err ≠ nil && v.expected.err ≠ nil && errors.Is(v.expected.err, err) {
                    t.Error("Expected:", v.expected.err, ",got:", err)
                }
            },
        )
    }
}
```

4. Reset password

   Kode Function yang akan ditest

```
func ResetPassword(email string) (string, string, error) {
    db := connect()
    defer db.Close()

    query := "SELECT iduser,username FROM user WHERE email=? "

    if len(email) == 0 {
        return "", "", errors.New("404")
    }

    rows, err := db.Query(query, email)

    if err ≠ nil {
        return "", "", errors.New("500")
    }

    //Check user login by name
    id := ""
    name := ""

    for rows.Next() {
        if err := rows.Scan(&id, &name); err ≠ nil {
            log.Print(err.Error())
        }
    }
    if len(name) > 0 {
        return id, name, nil
    }
    return "", "", errors.New("404")
}
```

Kode untuk melakukan testing

```
run test | debug test
func TestResetPassword(t *testing.T) {
    tableTest := []struct {
        label    string
        email    string
        expected struct {
            id   string
            name string
            err  error
        }
    }{
        {
            label: "Reset Password with Known Email",
            email: "neilea@gmail.com",
            expected: struct {
                id   string
                name string
                err  error
            }{
                id:   "9f44c64241f6095571dea1b73144a5fb",
                name: "neilea",
                err:  nil,
            },
        },
        {...
        },
    }
    for _, v := range tableTest {
        t.Run(
            v.label,
            func(t *testing.T) {
                id, name, err := ResetPassword(v.email)
                if v.expected.id ≠ id {
                    t.Error("Expected:", v.expected.id, ",got:", id)
                }
                if v.expected.name ≠ name {
                    t.Error("Expected:", v.expected.name, ",got:", name)
                }
                if err ≠ nil && v.expected.err ≠ nil && errors.Is(v.expected.err, err) {
                    t.Error("Expected:", v.expected.err, ",got:", err)
                }
            },
        )
    }
}
```

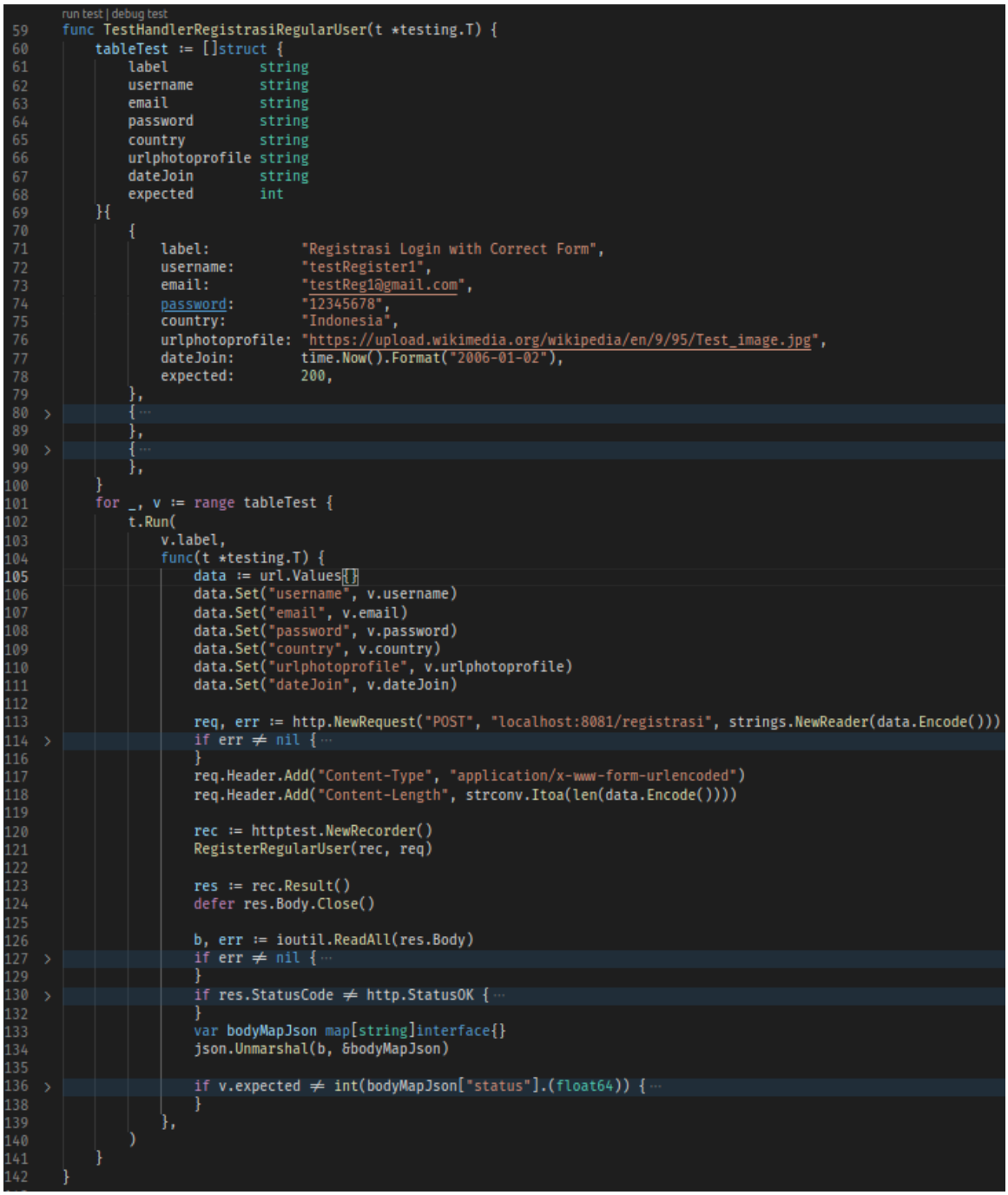
5. Handler Registrasi Regular User

Kode Function yang akan ditest

```
//Register Regular User to database
func RegisterRegularUser(w http.ResponseWriter, r *http.Request) {
    var response model.UserResponse
    err := r.ParseForm()
    if err ≠ nil {
        controller.ResponseManager(&response.Response, 400, "Insert Failed ")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
        return
    }
    username := r.Form.Get("username")
    email := r.Form.Get("email")
    password := r.Form.Get("password")
    country := r.Form.Get("country")
    urlphotoprofile := r.Form.Get("urlphotoprofile")
    dateJoin := r.Form.Get("dateJoin")
    categories := 2

    errQuery := controller.RegisterRegularUser(username, email, password, country, urlphotoprofile, dateJoin, categories)
    if errQuery == nil {
        controller.ResponseManager(&response.Response, 200, "")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
    } else if errQuery.Error() == "400" {
        controller.ResponseManager(&response.Response, 400, "Insert Failed ")
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
    } else {
        controller.ResponseManager(&response.Response, 500, errQuery.Error())
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(response)
    }
}
```

Kode untuk melakukan testing, pada kasus ini kita bisa menggunakan function”Run” untuk menjalankan kode testing secara terpisah berdasarkan setiap kasus. Dengan menggunakan data array yang terstruktur maka test bisa dilakukan secara rapi dan mudah dalam menangani banyak kasus.



```
run test | debug test
func TestHandlerRegistrasiRegularUser(t *testing.T) {
    tableTest := []struct {
        label           string
        username        string
        email           string
        password        string
        country         string
        urlphotoprofile string
        dateJoin        string
        expected        int
    }{
        {
            label:           "Registrasi Login with Correct Form",
            username:        "testRegister1",
            email:           "testReg1@gmail.com",
            password:        "12345678",
            country:         "Indonesia",
            urlphotoprofile: "https://upload.wikimedia.org/wikipedia/en/9/95/Test_image.jpg",
            dateJoin:        time.Now().Format("2006-01-02"),
            expected:        200,
        },
        {···
        },
        {···
        },
    }
    for _, v := range tableTest {
        t.Run(
            v.label,
            func(t *testing.T) {
                data := url.Values{}
                data.Set("username", v.username)
                data.Set("email", v.email)
                data.Set("password", v.password)
                data.Set("country", v.country)
                data.Set("urlphotoprofile", v.urlphotoprofile)
                data.Set("dateJoin", v.dateJoin)

                req, err := http.NewRequest("POST", "localhost:8081/registrasi", strings.NewReader(data.Encode()))
                if err != nil {···
                }
                req.Header.Add("Content-Type", "application/x-www-form-urlencoded")
                req.Header.Add("Content-Length", strconv.Itoa(len(data.Encode())))

                rec := httptest.NewRecorder()
                RegisterRegularUser(rec, req)

                res := rec.Result()
                defer res.Body.Close()

                b, err := ioutil.ReadAll(res.Body)
                if err != nil {···
                }
                if res.StatusCode != http.StatusOK {···
                }
                var bodyMapJson map[string]interface{}
                json.Unmarshal(b, &bodyMapJson)

                if v.expected != int(bodyMapJson["status"].(float64)) {···
                }
            },
        )
    }
}
```

6. Update Playlist

Kode Function yang akan ditest

```go
//UpdateUser is update data user like name, age, and address by id user
func UpdatePlaylist(updatedList string, valuesList []interface{}) error {
    db := connect()
    defer db.Close()

    if updatedList != "" {
        res, errQuery := db.Exec("UPDATE playlist SET "+updatedList+" WHERE idplaylist=?",
            valuesList... ,
        )
        if errQuery != nil {
            fmt.Println(errQuery.Error())
            fmt.Println(valuesList[0])
            return errors.New("500")
        } else {
            nums, _ := res.RowsAffected()
            if nums > 0 {
                return nil
            } else {
                return errors.New("400")
            }
        }
    }
    return errors.New("400")
}
```

Kode untuk melakukan testing

```go
run test | debug test
func TestUpdatePlaylist(t *testing.T) {
    updatedList := "NamePlaylist=?,UrlImageCover=?"
    var valuesList []interface{}
    valuesList = append(valuesList, "Fun Playlist")
    valuesList = append(valuesList, "https://image.shutterstock.com/723500997.jpg")
    valuesList = append(valuesList, 1)
    result := UpdatePlaylist(updatedList, valuesList)
    if result != nil {
        t.Error("Update Playlist Gagal : " + result.Error())
    }
}
```

7. Delete User

Kode Function yang akan ditest

```go
//DeleteUser is delete user by id user
func DeleteUser(userID string) error {
    db := connect()
    defer db.Close()

    // Asumsi Data hanya ada di table users dan regular_user
    res, errQuery := db.Exec("DELETE FROM regular_user WHERE iduser=?",
        userID,
    )
    if errQuery != nil {
        return errors.New("500")
    } else {
        resUser, errQuery2 := db.Exec("DELETE FROM user WHERE iduser=?",
            userID,
        )
        if errQuery2 != nil {
            return errors.New("500")
        }
        nums, _ := res.RowsAffected()
        nums2, _ := resUser.RowsAffected()
        if nums > 0 && nums2 > 0 {
            return nil
        } else {
            return errors.New("400")
        }
    }
}
```

Kode untuk melakukan testing

```
run test | debug test
func TestDeleteUser(t *testing.T) {
    userID := "a39df23e422a97f01a3b0ba58d7c9d8e"
    result := DeleteUser(userID)
    if result != nil {
        t.Error("Test Delete User Gagal : " + result.Error())
    }
}
```

8. Like Song

Kode Function yang akan ditest

```
func LikeSong(IDmusic string, anotherID string) error {
    db := connect()
    defer db.Close()

    if len(IDmusic) > 0 && len(anotherID) > 0 {
        _, errQuery := db.Exec("INSERT INTO song_like(iduser,idsong) VALUES(?,?)",
            anotherID, IDmusic,
        )
        if errQuery == nil {
            return nil
        } else {
            return errors.New("500")
        }
    }
    return errors.New("400")
}
```

Kode untuk melakukan testing

```
run test | debug test
func TestLikeSong(t *testing.T) {
    idMusic := "1"
    idUser := "7c20f5ef71c38abb1d0c1f1b6eb4459f"
    result := LikeSong(idMusic, idUser)
    if result != nil {
        t.Error("Like Song Gagal : " + result.Error())
    }
}
```

9. Followed Playlist

Kode Function yang akan ditest

```
func FollowedPlaylist(IDplaylist string, IDuser string) error {
    db := connect()
    defer db.Close()

    if len(IDplaylist) > 0 && len(IDuser) > 0 {
        _, errQuery := db.Exec("INSERT INTO playlist_following(idplaylist,iduser) VALUES(?,?)",
            IDplaylist, IDuser,
        )
        if errQuery == nil {
            return nil
        } else {
            return errors.New("500")
        }
    } else {
        return errors.New("400")
    }
}
```

Kode untuk melakukan testing

```
run test | debug test
14  func TestFollowedPlaylist(t *testing.T) {
15      idPlaylist := "1"
16      idUser := "7c20f5ef71c38abb1d0c1f1b6eb4459f"
17      result := FollowedPlaylist(idPlaylist, idUser)
18      if result != nil {
19          t.Error("Followed Playlist Gagal : " + result.Error())
20      }
21  }
```

10. AddSongPlaylist

    Kode Function yang akan ditest

```
283 func AddSongPlaylist(IDplaylist string, IDmusic string) error {
284     db := connect()
285     defer db.Close()
286
287     _, errQuery := db.Exec("INSERT INTO playlist_song(idplaylist,idsong) VALUES(?,?)",
288         IDplaylist, IDmusic,
289     )
290     if errQuery == nil {
291         return nil
292     } else {
293         return errors.New("500")
294     }
295 }
```

    Kode untuk melakukan testing

```
run test | debug test
5   func TestAddSongPlaylist(t *testing.T) {
6       idPlaylist := "1"
7       idMusic := "1"
8       result := AddSongPlaylist(idPlaylist, idMusic)
9       if result != nil {
10          t.Error("Add Song to Playlist Gagal : " + result.Error())
11      }
12  }
```

11. Liked Album

    Kode Function yang akan ditest

```
175 func LikedAlbum(IDalbum string, IDuser string) error {
176     db := connect()
177     defer db.Close()
178
179     if len(IDalbum) > 0 && len(IDuser) > 0 {
180         _, errQuery := db.Exec("INSERT INTO album_following(idalbum,iduser) VALUES(?,?)",
181             IDalbum, IDuser,
182         )
183         if errQuery == nil {
184             return nil
185         } else {
186             return errors.New("500")
187         }
188     } else {
189         return errors.New("400")
190     }
191 }
```

    Kode untuk melakukan testing

```
run package tests | run file tests
1   package controller
2
3   import "testing"
4
    run test | debug test
5   func TestLikedAlbum(t *testing.T) {
6       result := LikedAlbum("1", "7c20f5ef71c38abb1d0c1f1b6eb4459f")
7       if result != nil {
8           t.Error("Like Album Gagal : " + result.Error())
9       }
10  }
```

12. Response Manager

Kode Function yang akan ditest

```go
func ResponseManager(response *model.Response, status int16, message string) {
    if message != "" {
        message = " - " + message
    }
    switch {
    case status == 200:
        response.Status = 200
        response.Message = "OK"
    case status == 400:
        response.Status = 400
        response.Message = "Bad Request"
    case status == 404:
        response.Status = 404
        response.Message = "Not Found"
    case status == 500:
        response.Status = 500
        response.Message = "Internal Server Error"
    default:
        response.Status = 501
        response.Message = "Not Implemented"
    }
    response.Message += message
}
```

Kode untuk melakukan testing

```go
run test | debug test
func TestResponseManager(t *testing.T) {
    tableTest := []struct {
        label           string
        responseManager model.Response
        status          int16
        message         string
        expected        model.Response
    }{
        {
            label:           "Response Ok",
            responseManager: model.Response{},
            status:          200,
            message:         "Success",
            expected:        model.Response{Status: 200, Message: "OK - Success"},
        },
        {…
        },
        {…
        },
        {
            label:           "Response  Not Available",
            responseManager: model.Response{},
            status:          700,
            message:         "Crash were found between Services",
            expected:        model.Response{Status: 501, Message: "Not Implemented - Crash were found between Services"},
        },
    }
    for _, v := range tableTest {
        t.Run(
            v.label,
            func(t *testing.T) {
                ResponseManager(&v.responseManager, v.status, v.message)
                if v.expected != v.responseManager {
                    t.Error("Expected:", v.expected, ",got:", v.responseManager)
                }
            },
        )
    }
}
```