

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 mengisi 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 dites

```
74 func GetMD5Hash(text string) string {
75     hash := md5.Sum([]byte(text))
76     return hex.EncodeToString(hash[:])
77 }
```

Kode untuk melakukan testing

```
run test | debug test
59 func TestMD5Hash(t *testing.T) {
60     tableTest := []struct {
61         label    string
62         text     string
63         expected string
64     }{
65         {label: "MD5 Origin", text: "supersecret", expected: "9a618248b64db62d15b300a07b00580b"},
66         {label: "MD5 with Small Origin Diff.", text: "Supersecret", expected: "6a582edbeb20c95978af4ebec2cedddc"},
67     }
68     for _, v := range tableTest {
69         t.Run(
70             v.label,
71             func(t *testing.T) {
72                 result := GetMD5Hash(v.text)
73                 if v.expected != result {
74                     t.Error("Expected:", v.expected, ",got:", result)
75                 }
76             },
77         )
78     }
79 }
80
```

2. Login API

Kode Function yang akan dites

```
12 //UserLogin untuk melakukan pengecekan pada password dan email yang dimasukan
13 func UserLogin(w http.ResponseWriter, r *http.Request) {
14     var response model.Response
15     email := r.URL.Query()["email"]
16     password := r.URL.Query()["password"]
17
18     if len(email) == 0 || len(password) == 0 {
19         controller.ResponseManager(&response, 400, "")
20         w.Header().Set("Content-Type", "application/json")
21         json.NewEncoder(w).Encode(response)
22         return
23     }
24
25     id, name, err := controller.UserLogin(email[0], password[0])
26
27     if len(name) > 0 {
28         GenerateToken(w, id, name, 0)
29         controller.ResponseManager(&response, 200, "Success Login")
30         w.Header().Set("Content-Type", "application/json")
31         json.NewEncoder(w).Encode(response)
32     } else if err != nil && err.Error() == "500" {
33         controller.ResponseManager(&response, 500, "")
34         w.Header().Set("Content-Type", "application/json")
35         json.NewEncoder(w).Encode(response)
36     } else {
37         controller.ResponseManager(&response, 404, "Login Failed")
38         w.Header().Set("Content-Type", "application/json")
39         json.NewEncoder(w).Encode(&response)
40     }
41 }
```

Untuk mencoba function ini maka diperlukan untuk membuat request dalam bentuk http dan kemudian membuat variabel penerima hasil dari proses 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

```
1 package handler
2
3 > import (...)
13 )
14
15 run test | debug test
16 func TestHandlerLoginRegularUser(t *testing.T) {
17     tableTest := []struct {
18         label    string
19         email     string
20         password  string
21         expected  int
22     }{
23         {label: "Login with Correct Credential", email: "albertusaaa@gmail.com", password: "12345678", expected: 200},
24         {label: "Login with Wrong Password", email: "dahdkj@gmail", password: "notpwd", expected: 404},
25         {label: "Login with Unknown User", email: "testingtest@gmail", password: "notexistuser", expected: 404},
26     }
27     for _, v := range tableTest {
28         t.Run(
29             v.label,
30             func(t *testing.T) {
31                 req, err := http.NewRequest("GET", "localhost:8081/login?email="+v.email+"&password="+v.password, nil)
32                 if err != nil {
33                     return
34                 }
35                 rec := httptest.NewRecorder()
36                 UserLogin(rec, req)
37
38                 res := rec.Result()
39                 defer res.Body.Close()
40
41                 b, err := ioutil.ReadAll(res.Body)
42                 if err != nil {
43                     return
44                 }
45                 if res.StatusCode != http.StatusOK {
46                     return
47                 }
48                 var bodyMapJson map[string]interface{}
49                 json.Unmarshal(b, &bodyMapJson)
50
51                 if v.expected != int(bodyMapJson["status"].(float64)) {
52                     t.Errorf("Expected: %d, v.expected: %d, got: %d, bodyMapJson[%q]", v.expected, bodyMapJson["status"])
53                 }
54             },
55         ),
56     }
57 }
```

3. Login Regular User

Kode Function yang akan dites

```
18 //UserLogin untuk melakukan pengecekan pada password dan email yang dimasukan
19 func UserLogin(email string, password string) (string, string, error) {
20     db := connect()
21     defer db.Close()
22
23     query := "SELECT iduser,username FROM user WHERE email=? AND password=? "
24
25     sha := sha256.New()
26     sha.Write([]byte(password))
27     sha_password := hex.EncodeToString(sha.Sum(nil))
28
29     rows, err := db.Query(query, email, sha_password)
30
31     if err != nil {
32         return "", "", errors.New("500")
33     }
34
35     //Check user login by name
36     id := ""
37     name := ""
38
39     for rows.Next() {
40         if err := rows.Scan(&id, &name); err != nil {
41             log.Print(err.Error())
42         }
43     }
44     if len(name) > 0 {
45         return id, name, nil
46     }
47     return "", "", errors.New("404")
48 }
```

Kode untuk melakukan testing pada Login Regular User

```
run test | debug test
73 func TestLoginRegularUser(t *testing.T) {
74     tableTest := []struct {
75         label    string
76         email    string
77         password string
78         expected struct {
79             id    string
80             name string
81             err   error
82         }
83     }{
84         {
85             label:    "Login with Known Correct Credentials",
86             email:    "albertusaaa@gmail.com",
87             password: "12345678",
88             expected: struct {
89                 id    string
90                 name string
91                 err   error
92             }{
93                 id:    "7c20f5ef71c38abb1d0c1f1b6eb4459f",
94                 name: "albertusaa222",
95                 err:   nil,
96             },
97         },
98     }
99     for _, v := range tableTest {
100         t.Run(
101             v.label,
102             func(t *testing.T) {
103                 id, name, err := UserLogin(v.email, v.password)
104                 if v.expected.id != id {
105                     t.Error("Expected:", v.expected.id, ",got:", id)
106                 }
107                 if v.expected.name != name {
108                     t.Error("Expected:", v.expected.name, ",got:", name)
109                 }
110                 if err != nil && v.expected.err != nil && errors.Is(v.expected.err, err) {
111                     t.Error("Expected:", v.expected.err, ",got:", err)
112                 }
113             },
114         )
115     }
116 }
```

4. Reset password

Kode Function yang akan dites

```
50 func ResetPassword(email string) (string, string, error) {
51     db := connect()
52     defer db.Close()
53
54     query := "SELECT iduser,username FROM user WHERE email=? "
55
56     if len(email) == 0 {
57         return "", "", errors.New("404")
58     }
59
60     rows, err := db.Query(query, email)
61
62     if err != nil {
63         return "", "", errors.New("500")
64     }
65
66     //Check user login by name
67     id := ""
68     name := ""
69
70     for rows.Next() {
71         if err := rows.Scan(&id, &name); err != nil {
72             log.Print(err.Error())
73         }
74     }
75
76     if len(name) > 0 {
77         return id, name, nil
78     }
79     return "", "", errors.New("404")
80 }
```

Kode untuk melakukan testing

```
run test | debug test
8 func TestResetPassword(t *testing.T) {
9     tableTest := []struct {
10         label    string
11         email    string
12         expected struct {
13             id    string
14             name string
15             err  error
16         }
17     }{
18         {
19             label: "Reset Password with Known Email",
20             email: "neilea@gmail.com",
21             expected: struct {
22                 id    string
23                 name string
24                 err  error
25             }{
26                 id: "9f44c64241f6095571dea1b73144a5fb",
27                 name: "neilea",
28                 err: nil,
29             },
30         },
31     },
32 }
33
34 for _, v := range tableTest {
35     t.Run(
36         v.label,
37         func(t *testing.T) {
38             id, name, err := ResetPassword(v.email)
39             if v.expected.id != id {
40                 t.Error("Expected:", v.expected.id, ",got:", id)
41             }
42             if v.expected.name != name {
43                 t.Error("Expected:", v.expected.name, ",got:", name)
44             }
45             if err != nil && v.expected.err != nil && errors.Is(v.expected.err, err) {
46                 t.Error("Expected:", v.expected.err, ",got:", err)
47             }
48         },
49     )
50 }
51 }
```

5. Handler Registrasi Regular User

Kode Function yang akan dites

```
131 //Register Regular User to database
132 func RegisterRegularUser(w http.ResponseWriter, r *http.Request) {
133     var response model.UserResponse
134     err := r.ParseForm()
135     if err != nil {
136         controller.ResponseManager(&response.Response, 400, "Insert Failed ")
137         w.Header().Set("Content-Type", "application/json")
138         json.NewEncoder(w).Encode(response)
139         return
140     }
141     username := r.Form.Get("username")
142     email := r.Form.Get("email")
143     password := r.Form.Get("password")
144     country := r.Form.Get("country")
145     urlphotoprofile := r.Form.Get("urlphotoprofile")
146     dateJoin := r.Form.Get("dateJoin")
147     categories := 2
148
149     errQuery := controller.RegisterRegularUser(username, email, password, country, urlphotoprofile, dateJoin, categories)
150     if errQuery == nil {
151         controller.ResponseManager(&response.Response, 200, "")
152         w.Header().Set("Content-Type", "application/json")
153         json.NewEncoder(w).Encode(response)
154     } else if errQuery.Error() == "400" {
155         controller.ResponseManager(&response.Response, 400, "Insert Failed ")
156         w.Header().Set("Content-Type", "application/json")
157         json.NewEncoder(w).Encode(response)
158     } else {
159         controller.ResponseManager(&response.Response, 500, errQuery.Error())
160         w.Header().Set("Content-Type", "application/json")
161         json.NewEncoder(w).Encode(response)
162     }
163 }
```

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
59 func TestHandlerRegistrasiRegularUser(t *testing.T) {
60     tableTest := []struct {
61         label      string
62         username    string
63         email       string
64         password    string
65         country     string
66         urlphotoprofile string
67         dateJoin    string
68         expected    int
69     }{
70         {
71             label:      "Registrasi Login with Correct Form",
72             username:   "testRegister1",
73             email:      "testReg1@gmail.com",
74             password:   "12345678",
75             country:    "Indonesia",
76             urlphotoprofile: "https://upload.wikimedia.org/wikipedia/en/9/95/Test_image.jpg",
77             dateJoin:    time.Now().Format("2006-01-02"),
78             expected:    200,
79         },
80         { ...
81     },
82     { ...
83     },
84     { ...
85     },
86     { ...
87     },
88     { ...
89     },
90     { ...
91     },
92     { ...
93     },
94     { ...
95     },
96     { ...
97     },
98     { ...
99     },
100    }
101    for _, v := range tableTest {
102        t.Run(
103            v.label,
104            func(t *testing.T) {
105                data := url.Values{}
106                data.Set("username", v.username)
107                data.Set("email", v.email)
108                data.Set("password", v.password)
109                data.Set("country", v.country)
110                data.Set("urlphotoprofile", v.urlphotoprofile)
111                data.Set("dateJoin", v.dateJoin)
112
113                req, err := http.NewRequest("POST", "localhost:8081/registrasi", strings.NewReader(data.Encode()))
114                if err != nil { ...
115                }
116                req.Header.Add("Content-Type", "application/x-www-form-urlencoded")
117                req.Header.Add("Content-Length", strconv.Itoa(len(data.Encode())))
118
119                rec := httptest.NewRecorder()
120                RegisterRegularUser(rec, req)
121
122                res := rec.Result()
123                defer res.Body.Close()
124
125                b, err := ioutil.ReadAll(res.Body)
126                if err != nil { ...
127                }
128                if res.StatusCode != http.StatusOK { ...
129                }
130                var bodyMapJson map[string]interface{}
131                json.Unmarshal(b, &bodyMapJson)
132
133                if v.expected != int(bodyMapJson["status"].(float64)) { ...
134                }
135            },
136        )
137    }
138 }
139
140
141
142 }
```


6. Update Playlist

Kode Function yang akan dites

```
194 //UpdateUser is update data user like name, age, and address by id user
195 func UpdatePlaylist(updatedList string, valuesList []interface{}) error {
196     db := connect()
197     defer db.Close()
198
199     if updatedList != "" {
200         res, errQuery := db.Exec("UPDATE playlist SET "+updatedList+" WHERE idplaylist=?",
201             valuesList...)
202         if errQuery != nil {
203             fmt.Println(errQuery.Error())
204             fmt.Println(valuesList[0])
205             return errors.New("500")
206         } else {
207             nums, _ := res.RowsAffected()
208             if nums > 0 {
209                 return nil
210             } else {
211                 return errors.New("400")
212             }
213         }
214     }
215     return errors.New("400")
216 }
217 }
```

Kode untuk melakukan testing

```
run test | debug test
23 func TestUpdatePlaylist(t *testing.T) {
24     updatedList := "NamePlaylist=?,UrlImageCover=?"
25     var valuesList []interface{}
26     valuesList = append(valuesList, "Fun Playlist")
27     valuesList = append(valuesList, "https://image.shutterstock.com/723500997.jpg")
28     valuesList = append(valuesList, 1)
29     result := UpdatePlaylist(updatedList, valuesList)
30     if result != nil {
31         t.Error("Update Playlist Gagal : " + result.Error())
32     }
33 }
34 }
```

7. Delete User

Kode Function yang akan dites

```
340 //DeleteUser is delete user by id user
341 func DeleteUser(userID string) error {
342     db := connect()
343     defer db.Close()
344
345     // Asumsi Data hanya ada di table users dan regular_user
346     res, errQuery := db.Exec("DELETE FROM regular_user WHERE iduser=?",
347         userID,
348     )
349     if errQuery != nil {
350         return errors.New("500")
351     } else {
352         resUser, errQuery2 := db.Exec("DELETE FROM user WHERE iduser=?",
353             userID,
354         )
355         if errQuery2 != nil {
356             return errors.New("500")
357         }
358         nums, _ := res.RowsAffected()
359         nums2, _ := resUser.RowsAffected()
360         if nums > 0 && nums2 > 0 {
361             return nil
362         } else {
363             return errors.New("400")
364         }
365     }
366 }
```

Kode untuk melakukan testing

```
run test | debug test
64 func TestDeleteUser(t *testing.T) {
65     userID := "a39df23e422a97f01a3b0ba58d7c9d8e"
66     result := DeleteUser(userID)
67     if result != nil {
68         t.Error("Test Delete User Gagal : " + result.Error())
69     }
70 }
```

8. Like Song

Kode Function yang akan dites

```
116 func LikeSong(IDmusic string, anotherID string) error {
117     db := connect()
118     defer db.Close()
119
120     if len(IDmusic) > 0 && len(anotherID) > 0 {
121         _, errQuery := db.Exec("INSERT INTO song_like(iduser,idsong) VALUES(?,?)",
122             anotherID, IDmusic,
123         )
124         if errQuery == nil {
125             return nil
126         } else {
127             return errors.New("500")
128         }
129     }
130     return errors.New("400")
131 }
```

Kode untuk melakukan testing

```
run test | debug test
5 func TestLikeSong(t *testing.T) {
6     idMusic := "1"
7     idUser := "7c20f5ef71c38abb1d0c1f1b6eb4459f"
8     result := LikeSong(idMusic, idUser)
9     if result != nil {
10         t.Error("Like Song Gagal : " + result.Error())
11     }
12 }
```

9. Followed Playlist

Kode Function yang akan dites

```
219 func FollowedPlaylist(IDplaylist string, IDuser string) error {
220     db := connect()
221     defer db.Close()
222
223     if len(IDplaylist) > 0 && len(IDuser) > 0 {
224         _, errQuery := db.Exec("INSERT INTO playlist_following(idplaylist,iduser) VALUES(?,?)",
225             IDplaylist, IDuser,
226         )
227         if errQuery == nil {
228             return nil
229         } else {
230             return errors.New("500")
231         }
232     } else {
233         return errors.New("400")
234     }
235 }
```

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 dites

```

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 dites

```

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 dites

```
50 func ResponseManager(response *model.Response, status int16, message string) {
51     if message != "" {
52         message = " - " + message
53     }
54     switch {
55     case status == 200:
56         response.Status = 200
57         response.Message = "OK"
58     case status == 400:
59         response.Status = 400
60         response.Message = "Bad Request"
61     case status == 404:
62         response.Status = 404
63         response.Message = "Not Found"
64     case status == 500:
65         response.Status = 500
66         response.Message = "Internal Server Error"
67     default:
68         response.Status = 501
69         response.Message = "Not Implemented"
70     }
71     response.Message += message
72 }
```

Kode untuk melakukan testing

```
run test | debug test
9 func TestResponseManager(t *testing.T) {
10     tableTest := []struct {
11         label          string
12         responseManager model.Response
13         status          int16
14         message         string
15         expected        model.Response
16     }{
17         {
18             label:          "Response Ok",
19             responseManager: model.Response{},
20             status:          200,
21             message:         "Success",
22             expected:        model.Response{Status: 200, Message: "OK - Success"},
23         },
24         { ... },
25         { ... },
26         { ... },
27         {
28             label:          "Response Not Available",
29             responseManager: model.Response{},
30             status:          700,
31             message:         "Crash were found between Services",
32             expected:        model.Response{Status: 501, Message: "Not Implemented - Crash were found between Services"},
33         },
34     },
35     for _, v := range tableTest {
36         t.Run(
37             v.label,
38             func(t *testing.T) {
39                 ResponseManager(&v.responseManager, v.status, v.message)
40                 if v.expected != v.responseManager {
41                     t.Error("Expected:", v.expected, ",got:", v.responseManager)
42                 }
43             },
44         )
45     }
46 }
47
48
49
50
51
52
53
54
55
56
57 }
```