Laporan Hasil Project Backend Praktikum

Nama : Muhamad Ghandi Nur Setiawan

Nim : 434221014

Kelas : C-1

Universitas Airlangga

Surabaya

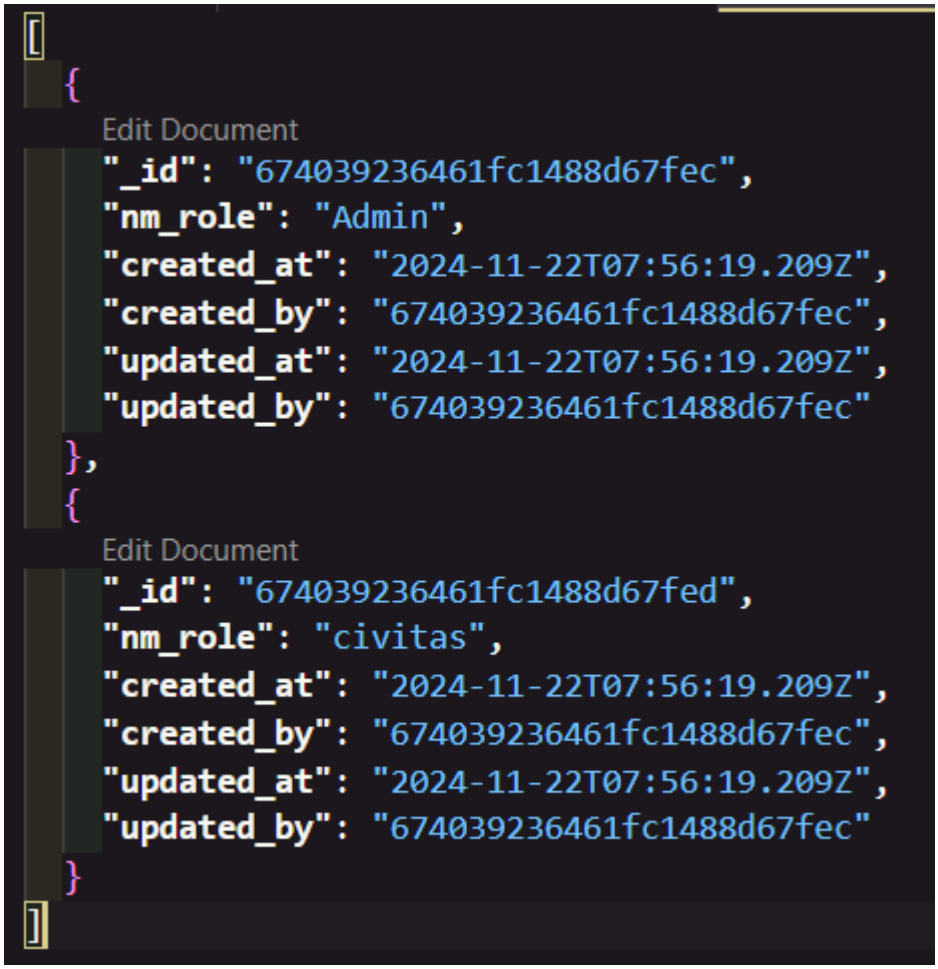
2024

Hasil Project Backend Praktikum

Tugas :

1. Silahkan anda buka soal UTS dan hasil disain MongoDb anda. Perhatikan hal berikut:
   - Tabel Role hanya memiliki 2 nilai, yaitu admin dan civitas.

```
// colection role
use('unairsatu_v2');
db.getCollection('role').updateMany(
    {},
    {
        $set: {
            created_by: ObjectId('674039236461fc1488d67fec'),
            updated_by: ObjectId('674039236461fc1488d67fec')
        }
    }
);
use('unairsatu_v2');
db.getCollection('role').insertMany([
    {
        "nm_role": "Admin",
        created_at: new Date(),
        created_by: 1,
        updated_at: new Date(),
        updated_by: 1
    },
    {
        "nm_role": "civitas",
        created_at: new Date(),
        created_by: 1,
        updated_at: new Date(),
        updated_by: 1
    }
]);
```



- Tabel Jenis_user memiliki nilai: Mahasiswa, Dosen, Tendik, KPS, Dekanat, Ketua_Unit dan Pimpinan_univ

```
// collection jenis users
use('unairsatu_v2');
db.getCollection('jenis_user').insertMany([
    {
        "nm_jenis_user": "Mahasiswa",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9985'),  //
aplikasi cyber campus
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9986'),  //
aplikasi pusba elpt
            },
            {
```

```
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998b'),  //
aplikasi helpdesk unair
            }
        ]
    },
    {
        "nm_jenis_user": "Dosen",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9985'),  //
aplikasi cyber campus
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9987'),  //
aplikasi dosen
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9988'),  //
aplikasi dosen v3
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998c'),  //
aplikasi dashboard
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
        ]
    },
    {
        "nm_jenis_user": "Tendik",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9985'),  //
aplikasi cyber campus
```

```
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9989'),  //
aplikasi e-office
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998c'),  //
aplikasi dashboard
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998b'),  //
aplikasi helpdesk unair
            }
        ]
    },
    {
        "nm_jenis_user": "KPS",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9985'),  //
aplikasi cyber campus
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9989'),  //
aplikasi e-office
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998c'),  //
aplikasi dashboard
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998b'),  //
aplikasi helpdesk unair
            }
        ]
    },
```

```
    {
        "nm_jenis_user": "Dekanat",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9989'),  //
aplikasi e-office
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998c'),  //
aplikasi dashboard
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998d'),  //
aplikasi simba
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998b'),  //
aplikasi helpdesk unair
            }
        ]
    },
    {
        "nm_jenis_user": "Ketua_Unit",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9989'),  //
aplikasi e-office
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998c'),  //
aplikasi dashboard
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998d'),  //
aplikasi simba
            },
```

```
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998b'),  //
aplikasi helpdesk unair
            }
        ]
    },
    {
        "nm_jenis_user": "Pimpinan_univ",
        "templates": [
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9984'),  //
aplikasi email unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9985'),  //
aplikasi cyber campus
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e9989'),  //
aplikasi e-office
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998c'),  //
aplikasi dashboard
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998d'),  //
aplikasi simba
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998a'),  //
aplikasi vpn unair
            },
            {
                "id_modul": ObjectId('67403a6e19038f97f16e998b'),  //
aplikasi helpdesk unair
            }
        ]
    }
]);
```

2. Sesuaikan struktur collection user anda dan modelnya, sehingga sesuai dengan disain MongoDB anda
   User.go

```go
package models

import (
    "go.mongodb.org/mongo-driver/bson/primitive"
)

type Moduls struct {
    ModulID  primitive.ObjectID `json:"modul_id" bson:"modul_id"`
    NmModul  string             `json:"nm_modul" bson:"nm_modul"`
    KetModul string             `json:"ket_modul" bson:"ket_modul"`
    Alamat   string             `json:"alamat" bson:"alamat"`
    GbrIcon  string             `json:"gbr_icon" bson:"gbr_icon"`
}

type User struct {
    ID           primitive.ObjectID `json:"id" bson:"_id,omitempty"`
    Username     string             `json:"username" bson:"username"`
    Nm_user      string             `json:"nm_user" bson:"nm_user"`
    Pass         string             `json:"pass" bson:"pass"`
    Email        string             `json:"email" bson:"email"`
    Role_aktif   primitive.ObjectID `json:"role_aktif"
bson:"role_aktif"`
```

```
    Created_at    primitive.DateTime `json:"created_at"
bson:"created_at"`
    Updated_at    primitive.DateTime `json:"updated_at"
bson:"updated_at"`
    Created_by    primitive.ObjectID `json:"created_by"
bson:"created_by"`
    Updated_by    primitive.ObjectID `json:"updated_by"
bson:"updated_by"`
    AuthKey       string             `json:"auth_key" bson:"auth_key"`
    Jenis_kelamin int                `json:"jenis_kelamin"
bson:"jenis_kelamin"`
    Photo         string             `json:"photo" bson:"photo"`
    Phone         string             `json:"phone" bson:"phone"`
    Token         string             `json:"token" bson:"token"`
    Id_jenis_user primitive.ObjectID `json:"id_jenis_user"
bson:"id_jenis_user"`
    Pass_2        string             `json:"pass_2" bson:"pass_2"`
    Moduls        []Moduls           `json:"moduls" bson:"moduls"`
}
```

Jenis_user.go

```
package models

import (
    "go.mongodb.org/mongo-driver/bson/primitive"
)

// Template represents a single module template for a user type.
type Template struct {
    IDModul primitive.ObjectID `bson:"id_modul" json:"id_modul"`
}

// JenisUser represents a type of user with associated templates.
type JenisUser struct {
    ID          primitive.ObjectID  `bson:"_id" json:"id"`
    NmJenisUser string              `bson:"nm_jenis_user"
json:"nm_jenis_user"`
    Templates   []Template          `bson:"templates" json:"templates"`
}
```

Kategori.go

```
package models

import (
    "go.mongodb.org/mongo-driver/bson/primitive"
)
```

```go
// Kategori represents the structure of the 'kategori' collection in
MongoDB
type Kategori struct {
    ID        primitive.ObjectID `bson:"_id,omitempty" json:"id"`
    NmKategori string            `bson:"nm_kategori"
json:"nm_kategori"`
}
```

Modul.go

```go
package models

import (
    "go.mongodb.org/mongo-driver/bson/primitive"
    "time"
)

// Modul represents the structure of a module document in MongoDB
type Modul struct {
    ID           primitive.ObjectID `bson:"_id" json:"id"`
    IDKategori   primitive.ObjectID `bson:"id_kategori"
json:"id_kategori"`
    NmModul      string             `bson:"nm_modul" json:"nm_modul"`
    KetModul     string             `bson:"ket_modul" json:"ket_modul"`
    IsAktif      string             `bson:"is_aktif" json:"is_aktif"`
    Alamat       string             `bson:"alamat" json:"alamat"`
    Urutan       int                `bson:"urutan" json:"urutan"`
    GbrIcon      string             `bson:"gbr_icon" json:"gbr_icon"`
    CreatedAt    time.Time          `bson:"created_at"
json:"created_at"`
    CreatedBy    primitive.ObjectID `bson:"created_by"
json:"created_by"`
    UpdatedAt    time.Time          `bson:"updated_at"
json:"updated_at"`
    UpdatedBy    primitive.ObjectID `bson:"updated_by"
json:"updated_by"`
    Icon         string             `bson:"icon" json:"icon"`
}
```

Role.go

```go
package models

import (
    "go.mongodb.org/mongo-driver/bson/primitive"
    "time"
)

// Role represents the structure of the 'role' collection in MongoDB
type Role struct {
```

```
    ID        primitive.ObjectID `bson:"_id,omitempty" json:"id"`
    NmRole    string             `bson:"nm_role" json:"nm_role"`
    CreatedAt time.Time          `bson:"created_at" json:"created_at"`
    CreatedBy primitive.ObjectID `bson:"created_by" json:"created_by"`
    UpdatedAt time.Time          `bson:"updated_at" json:"updated_at"`
    UpdatedBy primitive.ObjectID `bson:"updated_by" json:"updated_by"`
}
```

3. Buatlah 2 middleware yang memiliki parameter:
    a. checkRole(role string) → memeriksa, apakah role yang digunakan sesuai dengan nilai role pada parameter

```
// Middleware to check role
func CheckRole(role string) func(http.Handler) http.Handler {
    return func(next http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r
*http.Request) {
            // Get Authorization header
            authHeader := r.Header.Get("Authorization")
            if authHeader == "" {
                http.Error(w, "Authorization header missing",
http.StatusUnauthorized)
                return
            }

            // Decode token (assuming it's base64-encoded JSON)
            parts := strings.Split(authHeader, " ")
            if len(parts) != 2 || parts[0] != "Bearer" {
                http.Error(w, "Invalid Authorization header format",
http.StatusUnauthorized)
                return
            }
            token := parts[1]
            data, err := base64.StdEncoding.DecodeString(token)
            if err != nil {
                http.Error(w, "Failed to decode token",
http.StatusUnauthorized)
                return
            }

            // Parse token data
            var payload map[string]interface{}
            err = json.Unmarshal(data, &payload)
            if err != nil {
                http.Error(w, "Failed to parse token",
http.StatusUnauthorized)
                return
            }

            // Check role
```

```
            if payload["role"] != role {
                http.Error(w, "Unauthorized role",
http.StatusForbidden)
                return
            }

            next.ServeHTTP(w, r)
        })
    }
}
```

b. checkJenis_user(ju string) → memeriksa, apakah jenis user yang digunakan sesuai dengan jenis_user pada parameter

```
// Middleware to check jenis_user
func CheckJenisUser(ju string) func(http.Handler) http.Handler {
    return func(next http.Handler) http.Handler {
        return http.HandlerFunc(func(w http.ResponseWriter, r
*http.Request) {
            // Get Authorization header
            authHeader := r.Header.Get("Authorization")
            if authHeader == "" {
                http.Error(w, "Authorization header missing",
http.StatusUnauthorized)
                return
            }

            // Decode token (assuming it's base64-encoded JSON)
            parts := strings.Split(authHeader, " ")
            if len(parts) != 2 || parts[0] != "Bearer" {
                http.Error(w, "Invalid Authorization header format",
http.StatusUnauthorized)
                return
            }
            token := parts[1]
            data, err := base64.StdEncoding.DecodeString(token)
            if err != nil {
                http.Error(w, "Failed to decode token",
http.StatusUnauthorized)
                return
            }

            // Parse token data
            var payload map[string]interface{}
            err = json.Unmarshal(data, &payload)
            if err != nil {
                http.Error(w, "Failed to parse token",
http.StatusUnauthorized)
                return
            }
```

```
            // Check jenis_user
            if payload["jenis_user"] != ju {
                http.Error(w, "Unauthorized jenis_user",
http.StatusForbidden)
                return
            }

            next.ServeHTTP(w, r)
        })
    }
}
```

4. Buatlah group route untuk Admin dan gunakan middleware pada point 3.a.
    a. Pindahkan semua route terkait dengan CRUD pada users ke group ini

```
    admin := app.Group("/admin")
    admin.Use(middleware.AuthMiddleware) // Middleware diterapkan di
seluruh grup Admin

    // Routes CRUD Users
    admin.Post("/createUser", controllers.CreateUser)
    admin.Get("/getAllUser", controllers.GetUsers)
    admin.Get("/getUser/:id", controllers.GetUserOne)
    admin.Put("/updateUser/:id", controllers.UpdateUser)
    admin.Put("/changePassword/:id", controllers.ChangePassword)
    admin.Put("/uploadPhoto/:id", controllers.UploadPhoto)
    admin.Delete("/deleteUser/:id", controllers.DeleteUser)
```

    b. Buat fungsi untuk CRUD collection modul. Modul adalah collection yang berisi semua opsi aplikasi pada halaman unairsatu

```
package controllers

import (
    "context"
    "net/http"
    "project-crud_baru/models"
    "time"

    "github.com/gofiber/fiber/v2"
    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/bson/primitive"
    "go.mongodb.org/mongo-driver/mongo"
)

// MongoDB collection reference
var modulCollection *mongo.Collection

// Initialize the MongoDB collection
func InitModulCollection(db *mongo.Database) {
```

```
    modulCollection = db.Collection("modul")
}

// Create a new module
func CreateModul(c *fiber.Ctx) error {
    var modul models.Modul

    // Parse the request body
    if err := c.BodyParser(&modul); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid request body"})
    }

    // Set additional fields
    modul.ID = primitive.NewObjectID()
    modul.CreatedAt = primitive.NewDateTimeFromTime(time.Now())
    modul.UpdatedAt = primitive.NewDateTimeFromTime(time.Now())

    // Insert the module into the database
    _, err := modulCollection.InsertOne(context.Background(), modul)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to create modul"})
    }

    return c.Status(http.StatusCreated).JSON(modul)
}

// Get all modules
func GetAllModul(c *fiber.Ctx) error {
    var modules []models.Modul

    // Retrieve all documents from the modul collection
    cursor, err := modulCollection.Find(context.Background(), bson.M{})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to retrieve modules"})
    }
    defer cursor.Close(context.Background())

    // Iterate through the cursor and decode each document
    for cursor.Next(context.Background()) {
        var modul models.Modul
        if err := cursor.Decode(&modul); err != nil {
```

```
            return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to decode modul"})
        }
        modules = append(modules, modul)
    }

    return c.JSON(modules)
}

// Get a single module by ID
func GetModulByID(c *fiber.Ctx) error {
    id := c.Params("id")

    // Convert the string ID to ObjectID
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    var modul models.Modul
    // Find the modul by ID
    err = modulCollection.FindOne(context.Background(), bson.M{"_id":
objID}).Decode(&modul)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"Modul not found"})
    }

    return c.JSON(modul)
}

// Update a module by ID
func UpdateModul(c *fiber.Ctx) error {
    id := c.Params("id")

    // Convert the string ID to ObjectID
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    var modul models.Modul
    // Parse the request body
    if err := c.BodyParser(&modul); err != nil {
```

```
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid request body"})
    }

    // Update the UpdatedAt field
    modul.UpdatedAt = primitive.NewDateTimeFromTime(time.Now())

    // Create an update document
    update := bson.M{
        "$set": bson.M{
            "id_kategori": modul.IDKategori,
            "nm_modul":    modul.NmModul,
            "ket_modul":   modul.KetModul,
            "is_aktif":    modul.IsAktif,
            "alamat":      modul.Alamat,
            "urutan":      modul.Urutan,
            "gbr_icon":    modul.GbrIcon,
            "updated_at":  modul.UpdatedAt,
            "updated_by":  modul.UpdatedBy,
            "icon":        modul.Icon,
        },
    }

    // Update the modul in the database
    _, err = modulCollection.UpdateOne(context.Background(),
bson.M{"_id": objID}, update)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to update modul"})
    }

    return c.JSON(fiber.Map{"message": "Modul updated successfully"})
}

// Delete a module by ID
func DeleteModul(c *fiber.Ctx) error {
    id := c.Params("id")

    // Convert the string ID to ObjectID
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    // Delete the modul from the database
```

```
    _, err = modulCollection.DeleteOne(context.Background(),
bson.M{"_id": objID})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to delete modul"})
    }

    return c.JSON(fiber.Map{"message": "Modul deleted successfully"})
}
```

c. Buat fungsi untuk CRUD aplikasi dari collection modul ke collection template_modul. Template_modul ini berisi list aplikasi template pada setiap

```
package controllers

import (
    "context"
    "net/http"

    "github.com/gofiber/fiber/v2"
    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/bson/primitive"
    "go.mongodb.org/mongo-driver/mongo"
    "project-crud_baru/config"
    "project-crud_baru/models"
)

var jenisUserCollection *mongo.Collection =
config.GetCollection("jenis_user")

// CreateJenisUser handles creating a new JenisUser
func CreateJenisUser(c *fiber.Ctx) error {
    var jenisUser models.JenisUser
    if err := c.BodyParser(&jenisUser); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
err.Error()})
    }

    // Set ID as an ObjectID
    jenisUser.ID = primitive.NewObjectID()

    // Insert into the database
    _, err := jenisUserCollection.InsertOne(context.Background(),
jenisUser)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }
```

```
    return c.Status(http.StatusCreated).JSON(fiber.Map{"message":
"JenisUser created successfully", "id": jenisUser.ID})
}

// GetJenisUsers handles retrieving all JenisUser
func GetJenisUsers(c *fiber.Ctx) error {
    var jenisUsers []models.JenisUser
    cursor, err := jenisUserCollection.Find(context.Background(),
bson.M{})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    if err := cursor.All(context.Background(), &jenisUsers); err != nil
{
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"data": jenisUsers})
}

// GetJenisUser handles retrieving a single JenisUser by ID
func GetJenisUser(c *fiber.Ctx) error {
    id := c.Params("id")
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    var jenisUser models.JenisUser
    err = jenisUserCollection.FindOne(context.Background(),
bson.M{"_id": objID}).Decode(&jenisUser)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"JenisUser not found"})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"data": jenisUser})
}

// UpdateJenisUser handles updating a JenisUser by ID
func UpdateJenisUser(c *fiber.Ctx) error {
```

```
    id := c.Params("id")
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    var jenisUser models.JenisUser
    if err := c.BodyParser(&jenisUser); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
err.Error()})
    }

    // Update the document in the collection
    update := bson.M{
        "$set": bson.M{
            "nm_jenis_user": jenisUser.NmJenisUser,
            "templates":     jenisUser.Templates,
        },
    }

    _, err = jenisUserCollection.UpdateOne(context.Background(),
bson.M{"_id": objID}, update)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message": "JenisUser
updated successfully"})
}

// DeleteJenisUser handles deleting a JenisUser by ID
func DeleteJenisUser(c *fiber.Ctx) error {
    id := c.Params("id")
    objID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    _, err = jenisUserCollection.DeleteOne(context.Background(),
bson.M{"_id": objID})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
```

```
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message": "JenisUser
deleted successfully"})
}
```

d. jenis_user. Silahkan tentukan sendiri, apakah anda akan melakukan secara bulk atau satuan

e. Buat fungsi untuk menampilkan semua modul yang dimiliki oleh seorang user

```
func GetUserModules(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    // Retrieve user ID from URL parameters
    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    // Find user by ID
    var user models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&user)
    if err != nil {
        // If user not found, return error message
        if err == mongo.ErrNoDocuments {
            return
c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "User not
found"})
        }
        // Other errors (e.g., database issues)
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    // Return the user's modules if user found
    if len(user.Moduls) == 0 {
        return c.Status(http.StatusOK).JSON(fiber.Map{"message": "User
has no modules"})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"moduls":
user.Moduls})
}
```

f. Buat fungsi untuk pindah jenis_user. Hal yang dilakukan ketika pindah user adalah sbb:
   i. Tentukan user mana akan dipindahkan atau diberi jenis_user yang mana
   ii. Hapus semua modul yang dimiliki oleh user tersebut
   iii. Tambahkan modul baru ke user tersebut, sesuai dengan modul yang ada pada template_modul pada jenis_user tersebut

```
func ChangeJenisUser(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    // Retrieve user ID and new jenis_user ID from URL parameters
    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID format"})
    }

    idJenisUser := c.Params("id_jenis_user")
    jenisUserID, err := primitive.ObjectIDFromHex(idJenisUser)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid jenis_user ID format"})
    }

    // Find user by ID
    var user models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&user)
    if err != nil {
        // If user not found, return error message
        if err == mongo.ErrNoDocuments {
            return
c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "User not
found"})
        }
        // Other errors (e.g., database issues)
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    // Find jenis_user by ID
    jenisUserCollection := config.GetCollection("jenis_user")
    var jenisUser models.JenisUser
    err = jenisUserCollection.FindOne(ctx, bson.M{"_id":
jenisUserID}).Decode(&jenisUser)
```

```
    if err != nil {
        // If jenis_user not found, return error message
        if err == mongo.ErrNoDocuments {
            return
c.Status(http.StatusNotFound).JSON(fiber.Map{"error": "Jenis_user not
found"})
        }
        // Other errors (e.g., database issues)
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    // Remove all existing modules from the user
    update := bson.M{"$set": bson.M{"moduls": []models.Moduls{}}}
    _, err = userCollection.UpdateOne(ctx, bson.M{"_id": userID},
update)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    // Add new modules to the user based on the jenis_user's
template_modul
    update = bson.M{"$set": bson.M{"moduls": jenisUser.Templates}}
    _, err = userCollection.UpdateOne(ctx, bson.M{"_id": userID },
update)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message":
"Jenis_user changed successfully"})
}
```

g. Buat fungsi untuk CUD pada modul yang dimiliki oleh user. Fungsi ini bertujuan untuk menambahkan satu modul ke user tersebut. Terkadang terdapat user yang memiliki privilege khusus untuk mengakses modul tertentu. Contoh, dosen biasa yang ditunjuk sebagai salah satu tim satu data, sehingga dosen tersebut memiliki akses ke satu data dimana sebenarnya satu data hanya diberikan kepada jenis_user kps, dekanat, ketua_unit dan pimpinan_univ