Laporan Hasil Project Backend Praktikum

Nama : Muhamad Ghandi Nur Setiawan

Nim : 434221014

Kelas : C-1

Universitas Airlangga

Surabaya

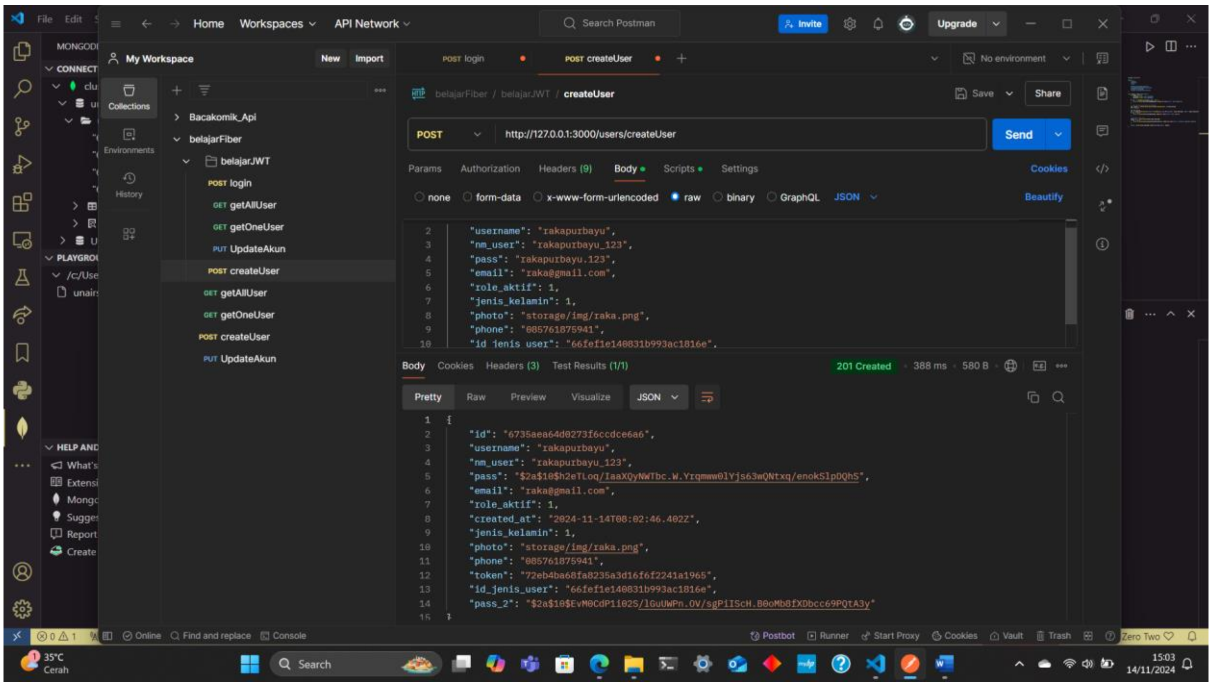
2024

Hasil Project Backend Praktikum

Tugas :

1. Implementasikan Bcrypt untuk menyimpan password anda!
   a. Ubah fungsi untuk create user, sehingga password tersimpan dalam bentuk hash bcrypt

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

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

    // Parse id_jenis_user dari string ke ObjectID
    idJenisUser, err :=
primitive.ObjectIDFromHex(user.Id_jenis_user.Hex())
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid id_jenis_user format"})
    }
    user.Id_jenis_user = idJenisUser

    // Hash the password
    hashedPassword, err :=
bcrypt.GenerateFromPassword([]byte(user.Pass), bcrypt.DefaultCost)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to hash password"})
    }

    // Generate token acak
    token, err := utils.GenerateRandomString(32)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to generate token"})
    }

    loc, err := time.LoadLocation("Asia/Jakarta")
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    user.Created_at = primitive.NewDateTimeFromTime(time.Now().In(loc))

    newUser := models.User{
        ID:            primitive.NewObjectID(),
        Username:      user.Username,
        Nm_user:       user.Nm_user,
        Pass:          string(hashedPassword), // Simpan password yang
sudah di-hash
        Email:         user.Email,
        Role_aktif:    user.Role_aktif,
        Created_at:    user.Created_at,
        Jenis_kelamin: user.Jenis_kelamin,
        Photo:         user.Photo,
        Phone:         user.Phone,
        Token:         token,
        Id_jenis_user: user.Id_jenis_user,
        Pass_2:        user.Pass_2,
    }

    _, errIns := userCollection.InsertOne(ctx, newUser)
    if errIns != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
errIns.Error()})
    }

    return c.Status(http.StatusCreated).JSON(newUser)
}
```



POST http://127.0.0.1:3000/users/createUser

```
"username": "rakapurbayu",
"nm_user": "rakapurbayu_123",
"pass": "rakapurbayu.123",
"email": "raka@gmail.com",
"role_aktif": 1,
"jenis_kelamin": 1,
"photo": "storage/img/raka.png",
"phone": "085761875941",
"id_jenis_user": "66fef1e140831b993ac1816e",
```

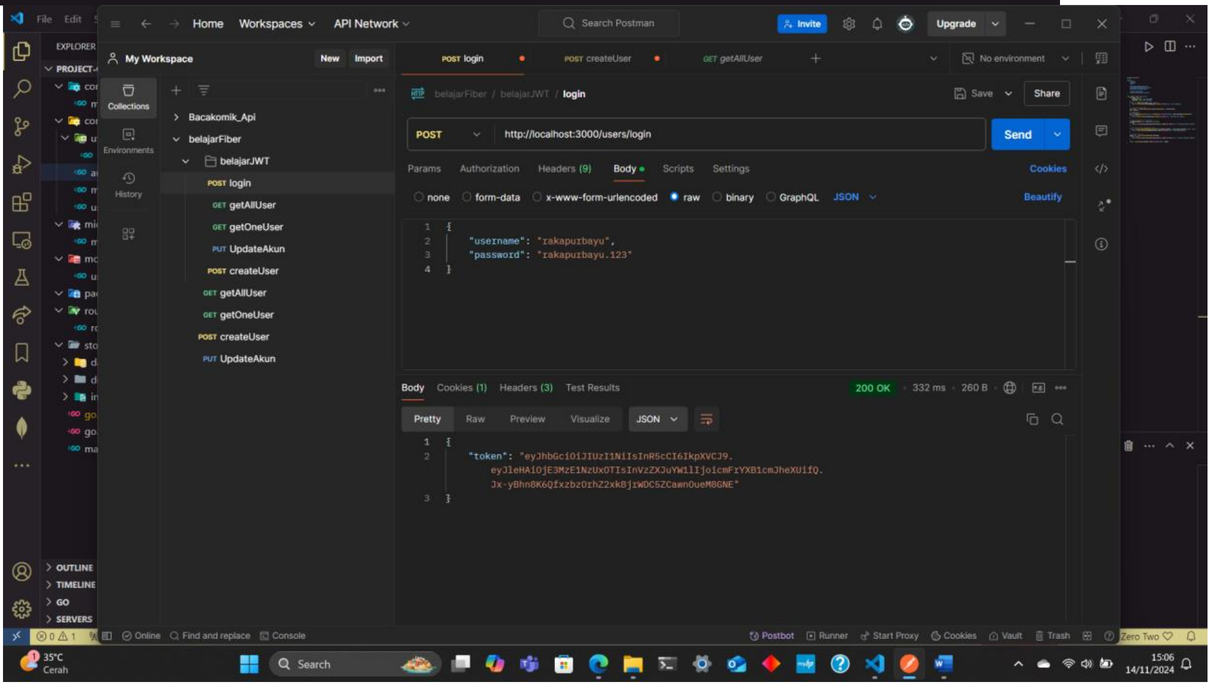
201 Created · 388 ms · 580 B

```
{
    "id": "6735aea64d0273f6ccdce6a6",
    "username": "rakapurbayu",
    "nm_user": "rakapurbayu_123",
    "pass": "$2a$10$h2eTLoq/IaaXQyNWTbc.W.Yrqmww0lYjs63wQNtxq/enokSlpDQhS",
    "email": "raka@gmail.com",
    "role_aktif": 1,
    "created_at": "2024-11-14T08:02:46.402Z",
    "jenis_kelamin": 1,
    "photo": "storage/img/raka.png",
    "phone": "085761875941",
    "token": "72eb4ba68fa8235a3d16f6f2241a1965",
    "id_jenis_user": "66fef1e140831b993ac1816e",
    "pass_2": "$2a$10$EvM0CdP1i02S/lGuUWPn.OV/sgPiISCH.B0oMb8fXDbcc69PQtA3y"
}
```

b. Ubah fungsi login anda, sehingga verifikasi password menggunakan bcrypt

```
func Login(c *fiber.Ctx) error {
    var input struct {
        Username string `json:"username"`
        Password string `json:"password"`
    }
    if err := c.BodyParser(&input); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Bad request"})
    }

    // Cek username di database
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

    var user bson.M
    err := config.GetCollection("users").FindOne(ctx,
bson.M{"username": input.Username}).Decode(&user)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"Username not found"})
    }

    // Ambil password hash dari database
    storedPasswordHash, ok := user["pass"].(string)
    if !ok {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Invalid password format"})
```

```
    }

    // Verifikasi password menggunakan bcrypt
    if err := bcrypt.CompareHashAndPassword([]byte(storedPasswordHash),
[]byte(input.Password)); err != nil {
        return
c.Status(http.StatusUnauthorized).JSON(fiber.Map{"error": "Invalid
password"})
    }

    // Generate token JWT
    token, err := utils.GenerateJWT(input.Username)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to generate token"})
    }

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



belajarFiber / belajarJWT / login

```
POST http://localhost:3000/users/login
```

```
{
    "username": "rakapurbayu",
    "password": "rakapurbayu.123"
}
```

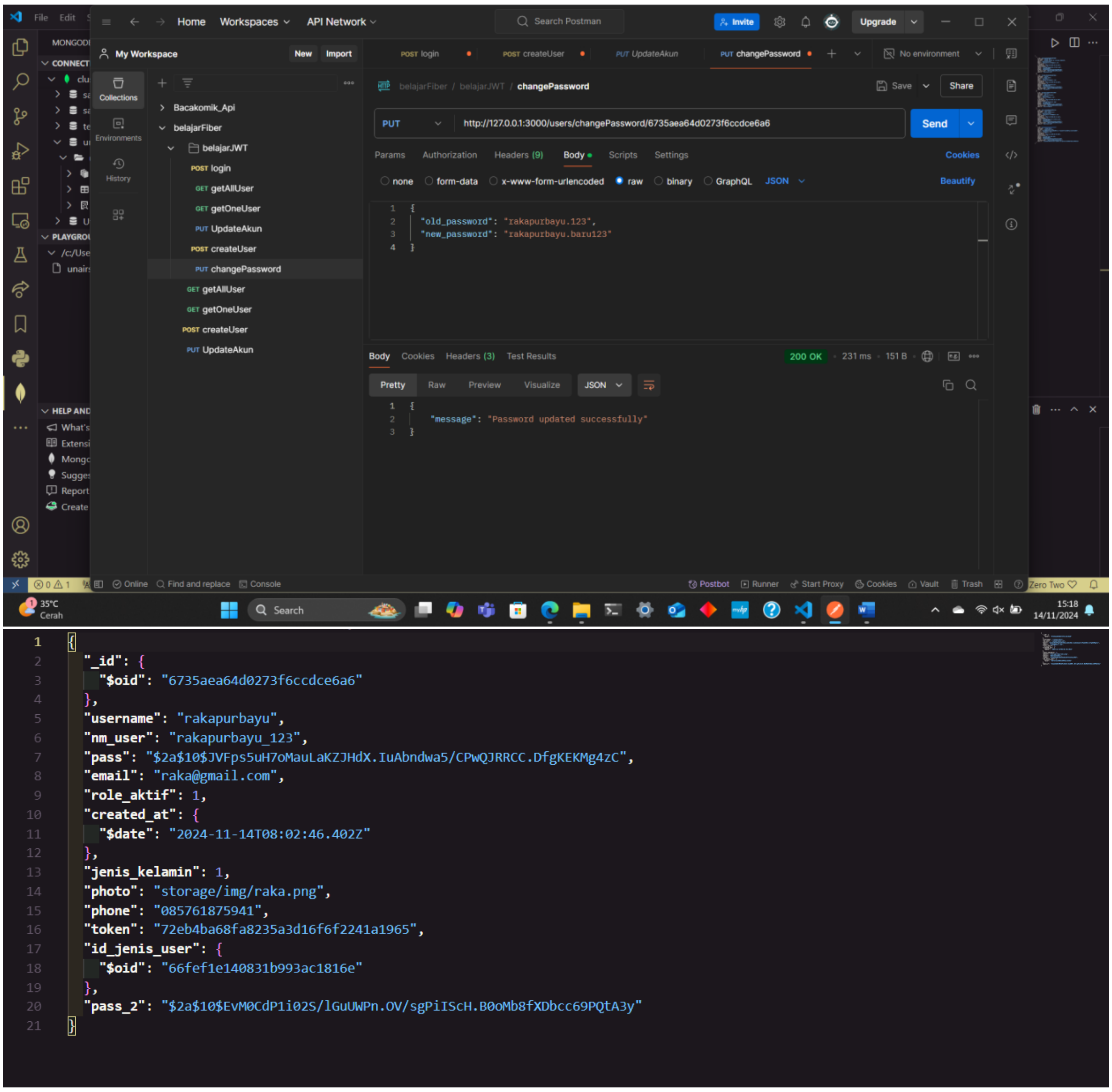
200 OK · 332 ms · 260 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJleHAiOjE3MzE1NzUxOTIsInVzZXJuYW1lIjoicmFrYXB1cmJheXUifQ.
        Jx-yBhn8K6QfxzbzOrhZ2xk8jrWDC5ZCawnOueM8GNE"
}
```

c. Buatlah fungsi untuk ubah password dan simpan password dalam bentuk hash bcrypt

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

    // Mendapatkan ID pengguna dari parameter
    id := c.Params("id")
    userID, err := primitive.ObjectIDFromHex(id)
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid ID"})
    }

    // Mendapatkan password lama dan baru dari request body
    var input struct {
        OldPassword string `json:"old_password"`
        NewPassword string `json:"new_password"`
    }
    if err := c.BodyParser(&input); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
err.Error()})
    }

    // Mengambil data pengguna berdasarkan ID
    var user models.User
    err = userCollection.FindOne(ctx, bson.M{"_id":
userID}).Decode(&user)
    if err != nil {
        return c.Status(http.StatusNotFound).JSON(fiber.Map{"error":
"User not found"})
    }

    // Verifikasi password lama dengan bcrypt
    if err := bcrypt.CompareHashAndPassword([]byte(user.Pass),
[]byte(input.OldPassword)); err != nil {
        return
c.Status(http.StatusUnauthorized).JSON(fiber.Map{"error": "Old password
is incorrect"})
    }

    // Hash password baru
    hashedNewPassword, err :=
bcrypt.GenerateFromPassword([]byte(input.NewPassword),
bcrypt.DefaultCost)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to hash new password"})
    }

    // Melakukan update password di database
    update := bson.M{"pass": string(hashedNewPassword)}
    _, err = userCollection.UpdateOne(ctx, bson.M{"_id": userID},
bson.M{"$set": update})
```

```go
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

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



PUT changePassword

belajarFiber / belajarJWT / changePassword

PUT http://127.0.0.1:3000/users/changePassword/6735aea64d0273f6ccdce6a6

```json
{
  "old_password": "rakapurbayu.123",
  "new_password": "rakapurbayu.baru123"
}
```

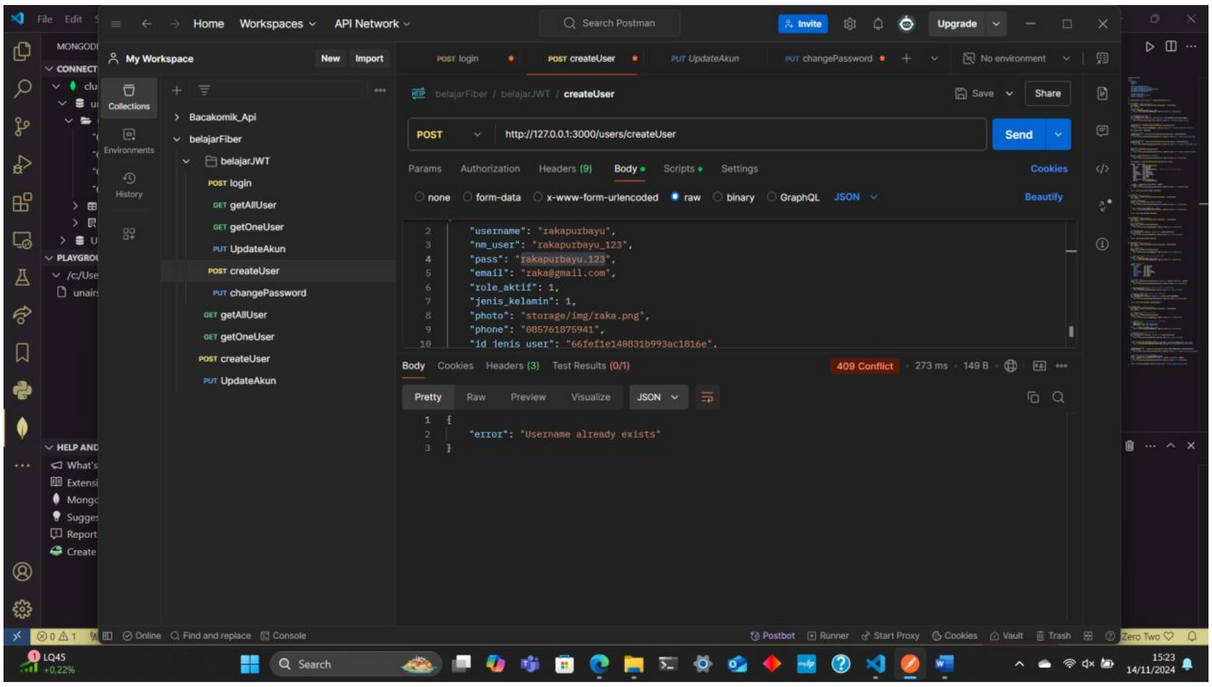
200 OK · 231 ms · 151 B

```json
{
    "message": "Password updated successfully"
}
```

```json
{
  "_id": {
    "$oid": "6735aea64d0273f6ccdce6a6"
  },
  "username": "rakapurbayu",
  "nm_user": "rakapurbayu_123",
  "pass": "$2a$10$JVFps5uH7oMauLaKZJHdX.IuAbndwa5/CPwQJRRCC.DfgKEKMg4zC",
  "email": "raka@gmail.com",
  "role_aktif": 1,
  "created_at": {
    "$date": "2024-11-14T08:02:46.402Z"
  },
  "jenis_kelamin": 1,
  "photo": "storage/img/raka.png",
  "phone": "085761875941",
  "token": "72eb4ba68fa8235a3d16f6f2241a1965",
  "id_jenis_user": {
    "$oid": "66fef1e140831b993ac1816e"
  },
  "pass_2": "$2a$10$EvM0CdP1i02S/lGuUWPn.OV/sgPiIScH.B0oMb8fXDbcc69PQtA3y"
}
```

2. Ubah fungsi create user, sehingga fungsi anda saat ini dapat memastikan bahwa username pada collection adalah unique

```go
// Create User
func CreateUser(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()
    var user models.User
    if err := c.BodyParser(&user); err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
err.Error()})
    }

    // Cek apakah username sudah ada
    var existingUser models.User
    err := userCollection.FindOne(ctx, bson.M{"username":
user.Username}).Decode(&existingUser)
    if err == nil {
        return c.Status(http.StatusConflict).JSON(fiber.Map{"error":
"Username already exists"})
    }

    // Parse id_jenis_user dari string ke ObjectID
    idJenisUser, err :=
primitive.ObjectIDFromHex(user.Id_jenis_user.Hex())
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Invalid id_jenis_user format"})
    }
    user.Id_jenis_user = idJenisUser

    // Hash the password
    hashedPassword, err :=
bcrypt.GenerateFromPassword([]byte(user.Pass), bcrypt.DefaultCost)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to hash password"})
    }

    // Hash the password 2
    hashedPassword2, err :=
bcrypt.GenerateFromPassword([]byte(user.Pass_2), bcrypt.DefaultCost)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to hash password 2"})
    }

    // Generate token acak
    token, err := utils.GenerateRandomString(32)
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to generate token"})
    }

    loc, err := time.LoadLocation("Asia/Jakarta")
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
err.Error()})
    }

    user.Created_at = primitive.NewDateTimeFromTime(time.Now().In(loc))

    newUser := models.User{
        ID:             primitive.NewObjectID(),
        Username:       user.Username,
        Nm_user:        user.Nm_user,
        Pass:           string(hashedPassword), // Simpan password yang
sudah di-hash
        Email:          user.Email,
        Role_aktif:     user.Role_aktif,
        Created_at:     user.Created_at,
        Jenis_kelamin: user.Jenis_kelamin,
        Photo:          user.Photo,
        Phone:          user.Phone,
        Token:          token,
        Id_jenis_user: user.Id_jenis_user,
        Pass_2:         string(hashedPassword2),
    }

    _, errIns := userCollection.InsertOne(ctx, newUser)
    if errIns != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
errIns.Error()})
    }

    return c.Status(http.StatusCreated).JSON(newUser)
}
```


POST http://127.0.0.1:3000/users/createUser

```
    "username": "rakapurbayu",
    "nm_user": "rakapurbayu_123",
    "pass": "rakapurbayu.123",
    "email": "raka@gmail.com",
    "role_aktif": 1,
    "jenis_kelamin": 1,
    "photo": "storage/img/raka.png",
    "phone": "085761875941",
    "id_jenis_user": "66fef1e140831b993ac1816e",
```

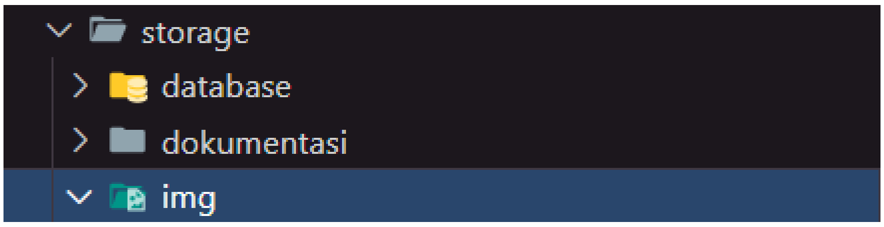
409 Conflict · 273 ms · 149 B

```
{
    "error": "Username already exists"
}
```

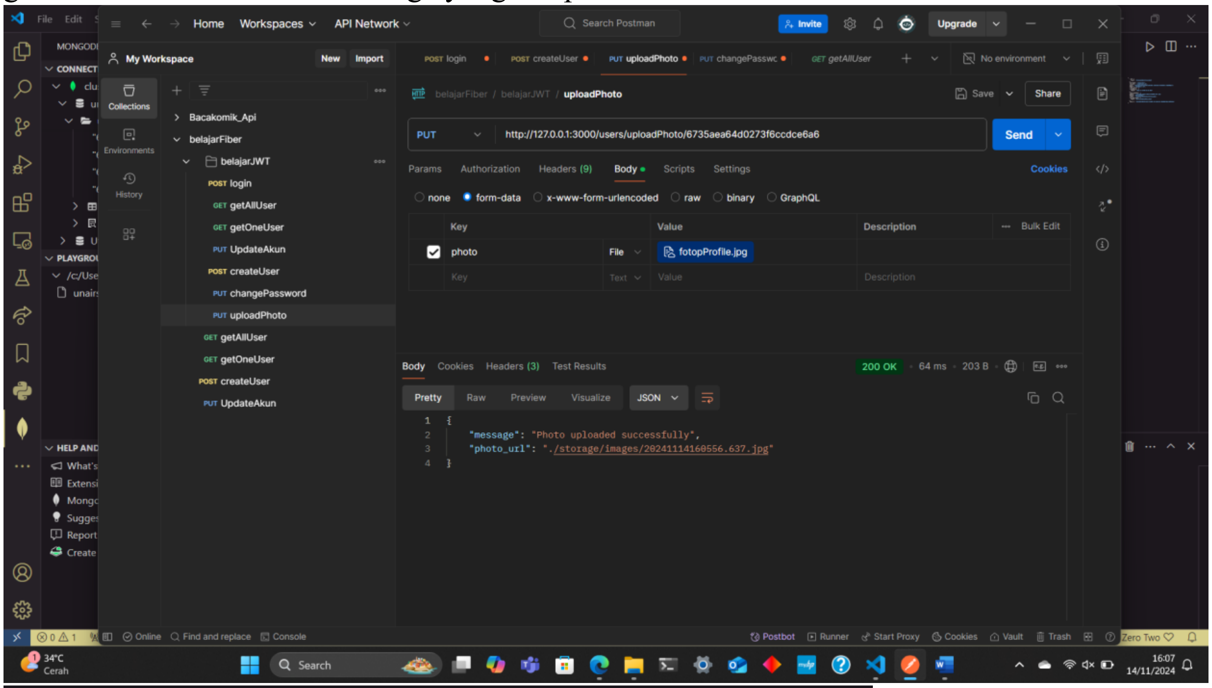
3. Buatlah fungsi upload image yang akan mengupdate field photo pada document users
    a. Dependencies baru yang mungkin akan anda perlukan
        i. Filepath
        ii. Os
        iii. Ftm

```
import (
    "context"
    "net/http"
    "time"
    "fmt"
    "os"
    "path/filepath"

    "github.com/gofiber/fiber/v2"
    "go.mongodb.org/mongo-driver/bson"
    "go.mongodb.org/mongo-driver/bson/primitive"
    "go.mongodb.org/mongo-driver/mongo"
    "golang.org/x/crypto/bcrypt"

    "project-crud/config"
    "project-crud/controllers/utils"
    "project-crud/models"
)
```

    b. Buatlah directory baru pada root directory anda:
        i. Root > storage > images
    Letakan image yang anda upload pada directory tersebut (./storage/images)



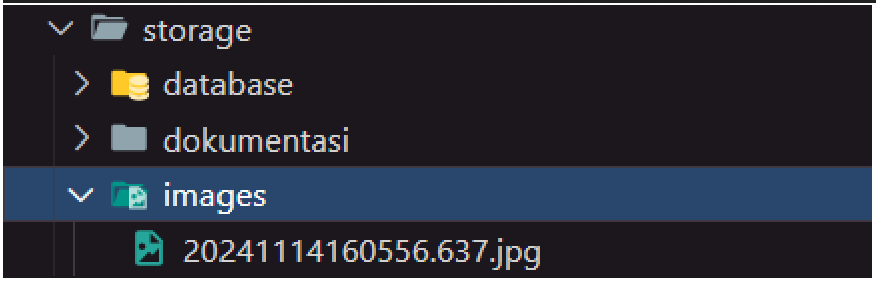
c. Ubah nama file yang diupload dengan format: YYYYMMDDHHmmSSsss.[file extension]
   Dimana:
   Y : tahun
   M : bulan
   D : tanggal
   H : jam
   m : menit
   S : detik
   s : mili detik
   gunakan file extension dari image yang di upload

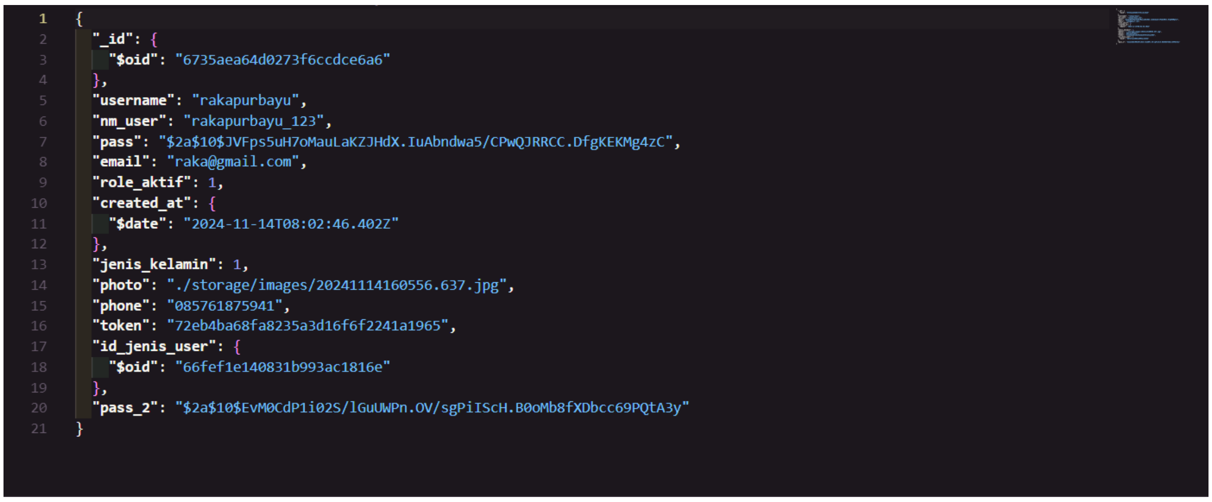

PUT http://127.0.0.1:3000/users/uploadPhoto/6735aea64d0273f6ccdce6a6

	Key		Value	Description
✓	photo	File	fotopProfile.jpg	

200 OK · 64 ms · 203 B

```
{
    "message": "Photo uploaded successfully",
    "photo_url": "./storage/images/20241114160556.637.jpg"
}
```





```json
{
  "_id": {
    "$oid": "6735aea64d0273f6ccdce6a6"
  },
  "username": "rakapurbayu",
  "nm_user": "rakapurbayu_123",
  "pass": "$2a$10$JVFps5uH7oMauLaKZJHdX.IuAbndwa5/CPwQJRRCC.DfgKEKMg4zC",
  "email": "raka@gmail.com",
  "role_aktif": 1,
  "created_at": {
    "$date": "2024-11-14T08:02:46.402Z"
  },
  "jenis_kelamin": 1,
  "photo": "./storage/images/20241114160556.637.jpg",
  "phone": "085761875941",
  "token": "72eb4ba68fa8235a3d16f6f2241a1965",
  "id_jenis_user": {
    "$oid": "66fef1e140831b993ac1816e"
  },
  "pass_2": "$2a$10$EvM0CdP1i02S/lGuUWPn.OV/sgPiIScH.B0oMb8fXDbcc69PQtA3y"
}
```

```go
// Upload Photo
func UploadPhoto(c *fiber.Ctx) error {
    ctx, cancel := context.WithTimeout(context.Background(),
10*time.Second)
    defer cancel()

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

    // Retrieve uploaded file
    file, err := c.FormFile("photo")
    if err != nil {
        return c.Status(http.StatusBadRequest).JSON(fiber.Map{"error":
"Failed to retrieve file"})
    }

    // Create directory if it doesn't exist
    if _, err := os.Stat("./storage/images"); os.IsNotExist(err) {
        err := os.MkdirAll("./storage/images", os.ModePerm)
        if err != nil {
            return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to create directory"})
        }
    }

    // Generate new filename based on timestamp
    timestamp := time.Now().Format("20060102150405.000")
    extension := filepath.Ext(file.Filename)
    newFileName := fmt.Sprintf("%s%s", timestamp, extension)
    filePath := fmt.Sprintf("./storage/images/%s", newFileName)

    // Save the file to ./storage/images directory
    if err := c.SaveFile(file, filePath); err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to save file"})
    }

    // Update user document with the new file path in the `photo` field
    update := bson.M{"photo": filePath}
    _, err = userCollection.UpdateOne(ctx, bson.M{"_id": userID},
bson.M{"$set": update})
    if err != nil {
        return
c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error":
"Failed to update user photo"})
    }

    return c.Status(http.StatusOK).JSON(fiber.Map{"message": "Photo
uploaded successfully", "photo_url": filePath})
}
```

4. Dengan menyelesaikan tugas nomor 3, anda telah mempersiapkan fungsi-fungsi dasar yang akan diperlukan dalam proyek membangun unairsatu. Sekarang, buatlah list fungsi apa saja yang diperlukan pada unairsatu!
   Jawab :
   List fungsi – fungsi dasar yang di perlukan untuk membangun unairsatu yaitu sebagai berikut :
   - Autentikasi (Login), fungsi login untuk memungkinkan pengguna (mahasiswa, staf, dan dosen) mengakses aplikasi menggunakan kredensial mereka.
   - Registrasi (Pembuatan Akun), fungsi pembuatan akun untuk menambahkan akun baru pengguna baru.
   - Perubahan Kata Sandi (Change Password), fungsi untuk memungkinkan pengguna mengganti kata sandi.
   - Pengelolaan Aplikasi (Role Access), fungsi untuk mengelola akses pengguna ke aplikasi-aplikasi yang ada berdasarkan role.
   - Upload Foto, Fungsi untuk memungkinkan pengguna melakuakn upload foto profil.