**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!
   1. Ubah fungsi untuk create user, sehingga password tersimpan dalam bentuk hash bcrypt

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)

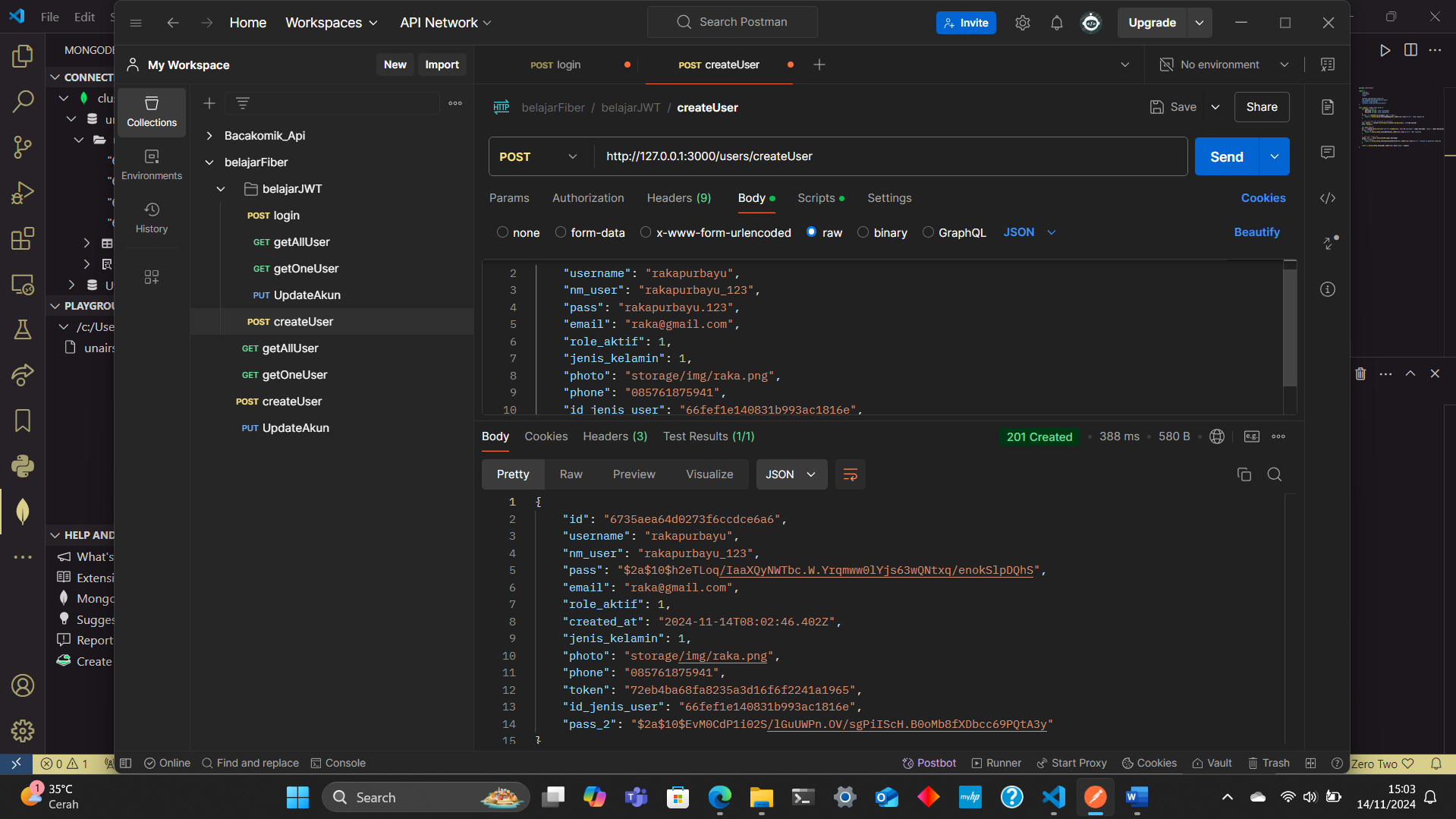
    if errIns != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": errIns.Error()})

    }

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

}



* 1. 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)

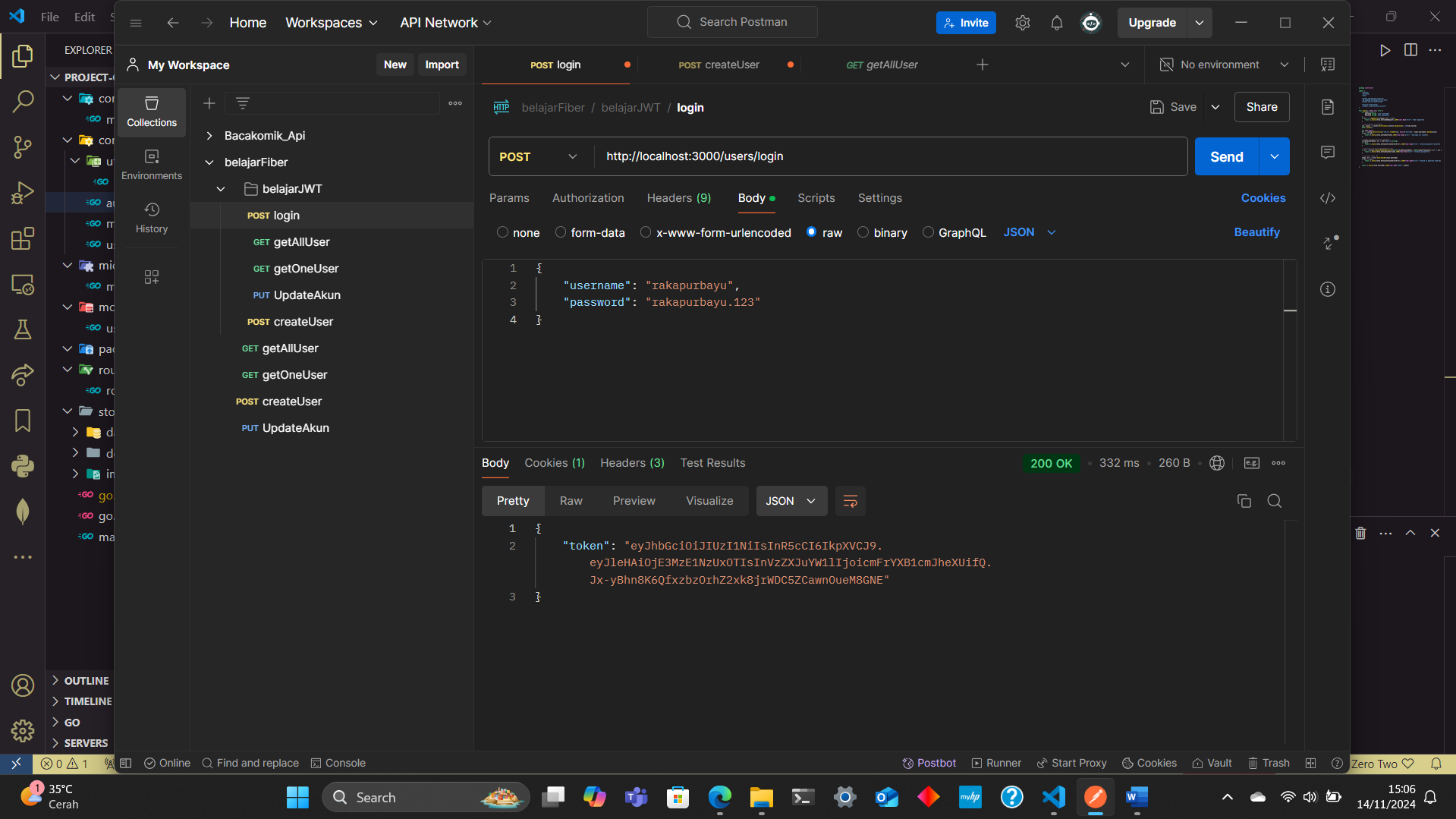
    if err != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": "Failed to generate token"})

    }

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

}



* 1. 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})

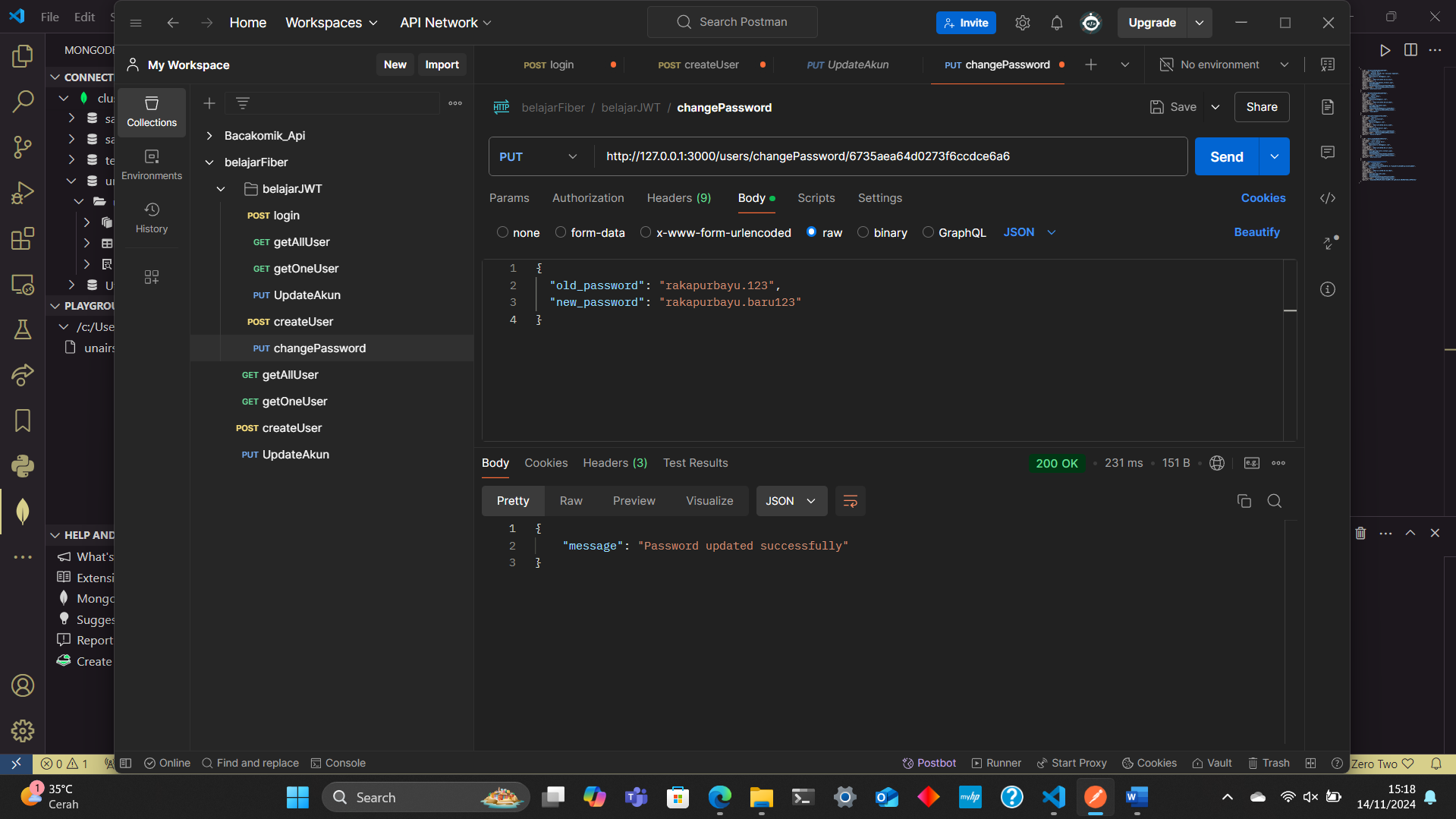
    if err != nil {

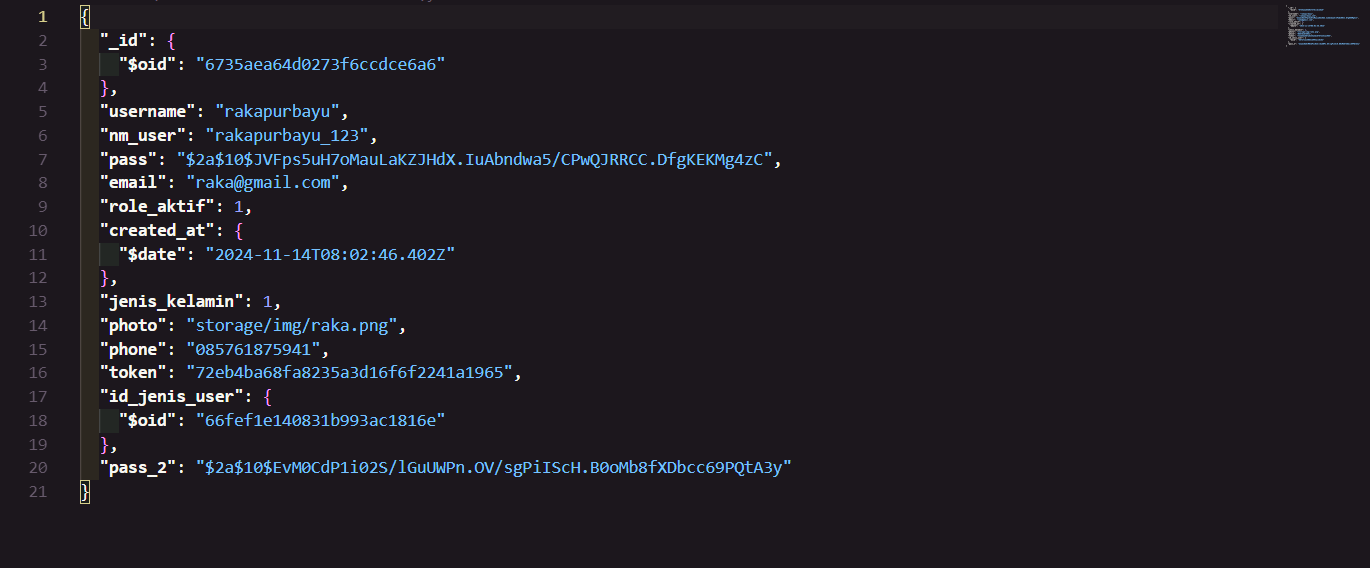
        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": err.Error()})

    }

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

}





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

*// 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)

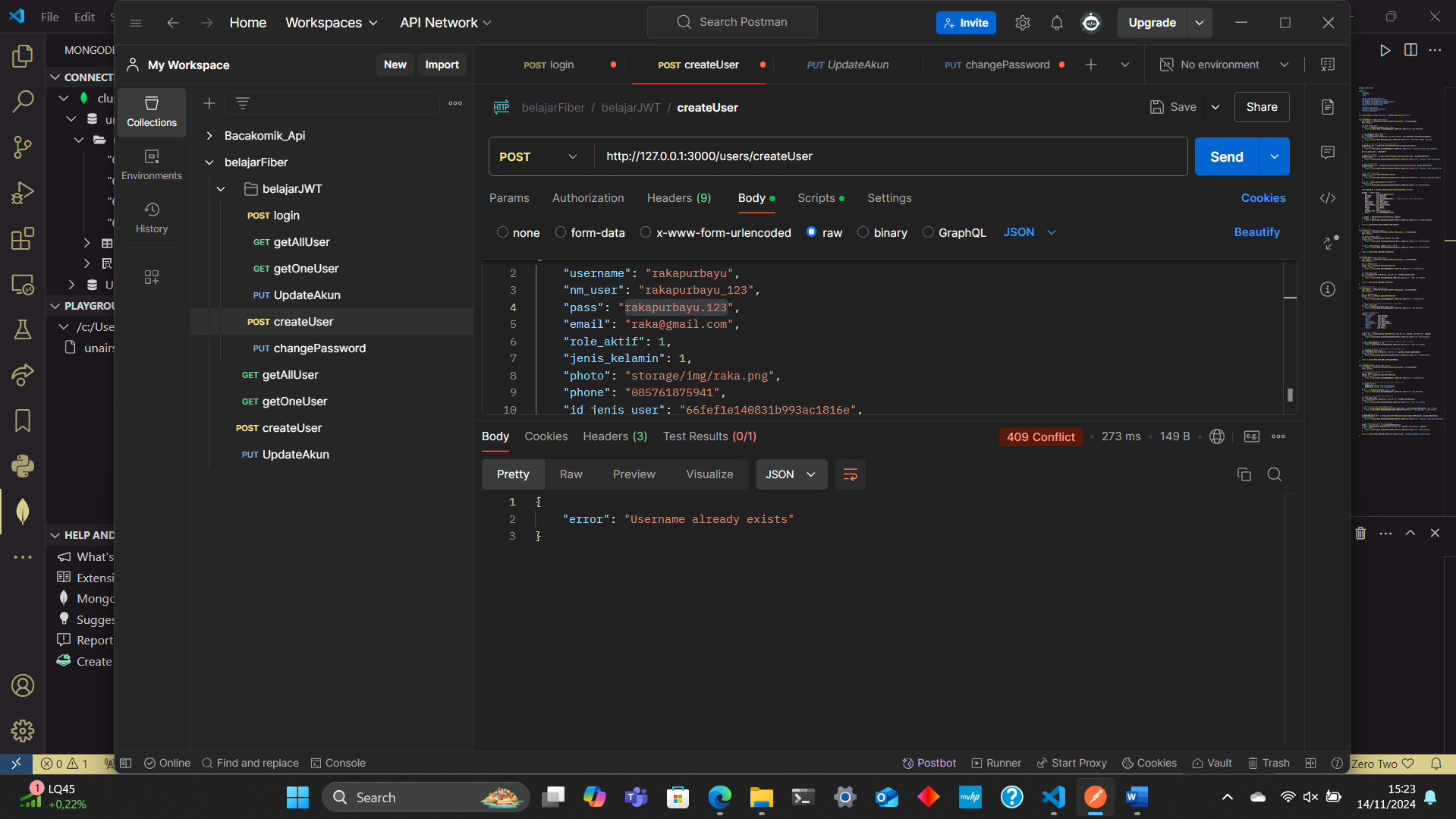
    if errIns != nil {

        return c.Status(http.StatusInternalServerError).JSON(fiber.Map{"error": errIns.Error()})

    }

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

}



1. Buatlah fungsi upload image yang akan mengupdate field photo pada document users
   1. Dependencies baru yang mungkin akan anda perlukan
      1. Filepath
      2. Os
      3. 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"

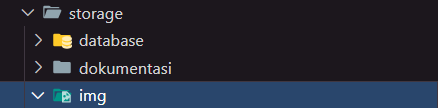
    "project-crud/controllers/utils"

    "project-crud/models"

)

* 1. Buatlah directory baru pada root directory anda:
     1. Root > storage > images

Letakan image yang anda upload pada directory tersebut (./storage/images)



* 1. Ubah nama file yang diupload dengan format: YYYYMMDDHHmmSSsss.[file extension]

Dimana:

Y : tahun

M : bulan

D : tanggal

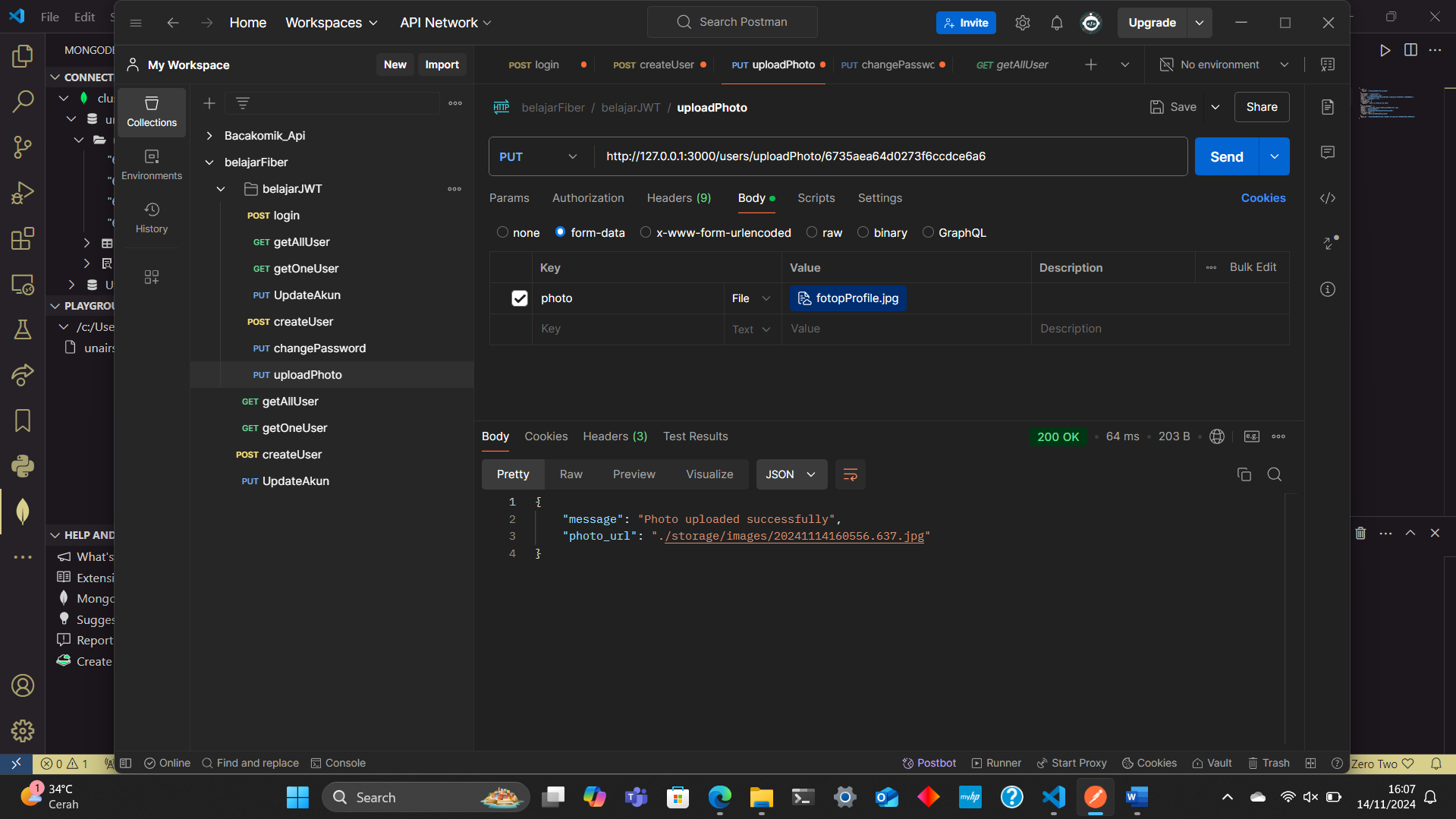
H : jam

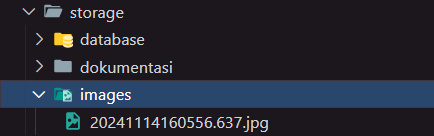
m : menit

S : detik

s : mili detik

gunakan file extension dari image yang di upload







*// 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})

}

1. 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.