WEEK1_TASK

Bu uygulama, öğrenci veritabanını yönetmek için basit bir RESTful API sağlar. API, öğrenci ekleme, öğrenci bilgilerini alma ve tüm öğrenci bilgilerini listeleme gibi temel işlevleri destekler.

Kurulum ve Kullanım

1. Bağımlılıkların Yüklenmesi: Projenin kök dizininde terminali açın ve aşağıdaki komutu çalıştırarak gerekli bağımlılıkları yükleyin:

```
npm install
```

2. Veritabanı Bağlantısı: Veritabanı bağlantısını yapılandırmak için Helper/mysql.js dosyasını açın ve MySQL bağlantı ayarlarınızı yapılandırın.
3. Sunucuyu Başlatma: Projenin kök dizininde terminali açın ve aşağıdaki komutu çalıştırarak sunucuyu başlatın:

```
node app.js
```

API Rotaları

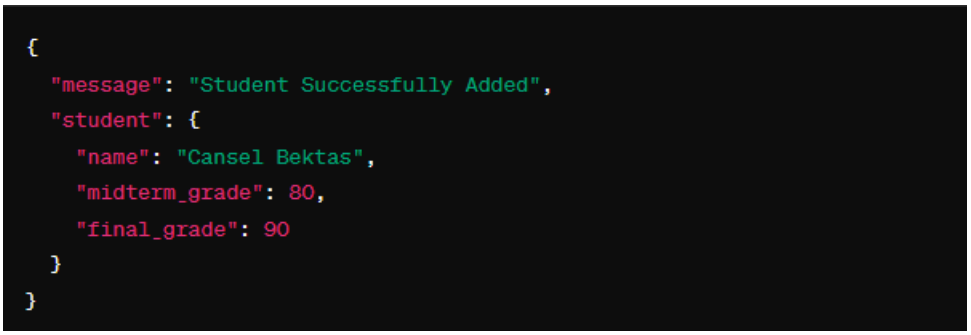
Postman veya benzeri bir araç kullanarak API rotalarını test edin.

Öğrenci Ekleme

- Metod: POST
- Rota: /students/add
- Açıklama: Yeni bir öğrenci ekler.
- Örnek İstek Gövdesi:

```
{
  "name": "Cansel Bektas",
  "midterm_grade": 80,
  "final_grade": 90
}
```

- Örnek Başarı Cevabı:



```
{
  "message": "Student Successfully Added",
  "student": {
    "name": "Cansel Bektas",
    "midterm_grade": 80,
    "final_grade": 90
  }
}
```

- Hata Durumları:

500 - Internal Server Error: Veritabanına öğrenci eklenirken bir hata oluştuğunda döndürülür.

Öğrenci Bilgisi Alma

- Metod: GET
- Rota: /students/getById/:id
- Açıklama: Belirtilen ID'ye sahip bir öğrencinin bilgilerini getirir.
- Örnek Başarı Cevabı:

```
{
  "message": "Student Successfully Found",
  "student": {
    "name": "Cansel Bektas",
    "midterm_grade": 80,
    "final_grade": 90,
    "average": 85
  }
}
```

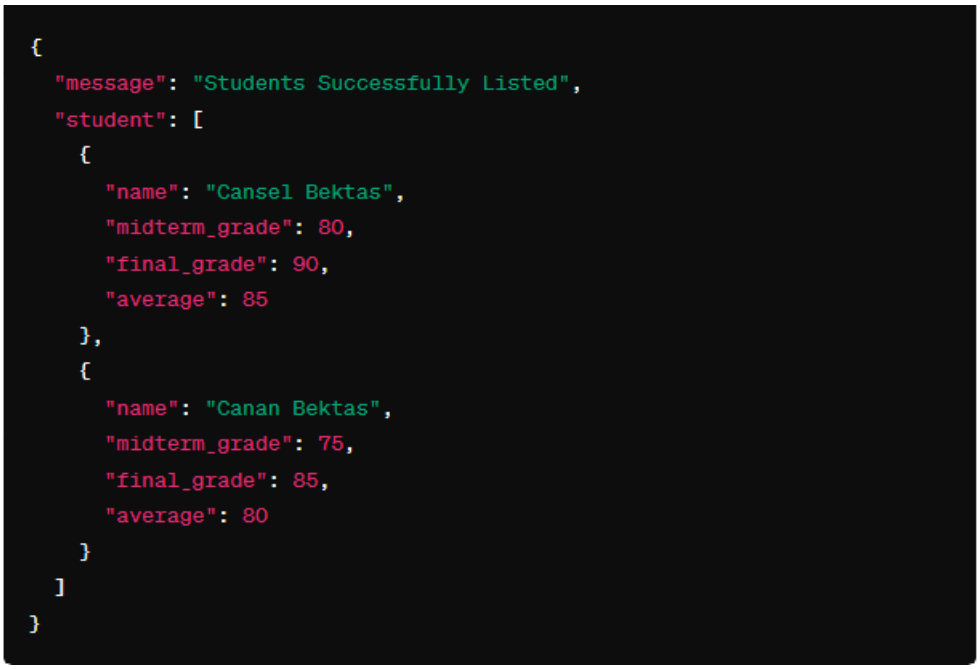
- Hata Durumları:

  404 - Not Found: Belirtilen ID'ye sahip bir öğrenci bulunamadığında döndürülür.

  500 - Internal Server Error: Veritabanından öğrenci bilgileri alınırken bir hata oluştuğunda döndürülür.

Tüm Öğrenci Bilgilerini Listeleme

- Metod: GET
- Rota: /students/getAll
- Açıklama: Tüm öğrenci bilgilerini getirir.
- Örnek Başarı Cevabı:



```
{
  "message": "Students Successfully Listed",
  "student": [
    {
      "name": "Cansel Bektas",
      "midterm_grade": 80,
      "final_grade": 90,
      "average": 85
    },
    {
      "name": "Canan Bektas",
      "midterm_grade": 75,
      "final_grade": 85,
      "average": 80
    }
  ]
}
```

- Hata Durumları:

  404 - Not Found: Veritabanında öğrenci bulunamadığında döndürülür.

  500 - Internal Server Error: Veritabanından öğrenci bilgileri alınırken bir hata oluştuğunda döndürülür.

Kodlar Açıklamaları:

Post

```
// Öğrenci Ekleme Endpoint'i
app.post('/students/add', (req, res)=>{

    // İstek gövdesinden gelen verileri alır.
    const{name,midterm_grade,final_grade}=req.body;

    // MySQL komutunu hazırlar.
    const mysqlCommand=`INSERT INTO students (name, midterm_grade, final_grade) VALUES (?, ?, ?)`;

    // Veritabanına ekleme işlemini gerçekleştirir.

    dbConnection.query(mysqlCommand,[name,midterm_grade,final_grade],(err,result)=>{
            if(err){

                 // Eğer bir hata oluşursa, hata mesajını konsola yazdırır
                 // istemciye 500 (Sunucu Hatası) koduyla bir hata mesajı gönderir.
                console.log("Database query error: ", err);
                res.status(500).send("Database's Error"+err.stack);

            } else {

                // Başarıyla eklendiğinde, eklenen öğrencinin bilgilerini içeren bir nesne oluşturur.
                const addedStudent={
                    name:name,
                    midterm_grade:midterm_grade,
                    final_grade:final_grade,
                };

                // İstemciye 200 (Başarılı) koduyla bir JSON yanıtı gönderir.
                res.status(200).json({
                    message:"Student  Succesfully Added",
                    student:addedStudent
                });

            }
    });
});
```

Get

```
app.get('/students/getById/:id', (req, res) => {

    //// Veritabanından öğrenciyi belirli bir ID'ye göre sorgular.
    dbConnection.query(
        "SELECT * FROM students WHERE id=?",
        [req.params.id],
        (err, results, fields) => {
            if (err) {
                // // Eğer bir sorgulama hatası oluşursa, hatayı konsola yazdırır
                //istemciye 500 (Sunucu Hatası) koduyla bir hata mesajı gönderir.
                console.log("Database query error: ", err);
                res.status(500).send("Database's Error" + err.stack);
            }

            //  // Eğer belirtilen ID'ye sahip bir öğrenci bulunamazsa, istemciye 404 (Bulunamadı) koduyla bir hata mesajı gönderir.
            if (results.length === 0) {
                    res.status(404).json({
                    message: 'Student Not Found',
                });
            }

            //// Bulunan öğrencinin bilgilerini alır ve ortalama notunu hesaplar.
            const student = results[0]
            const foundedStudent = {
                name: student.name,
                midterm_grade: student.midterm_grade,
                final_grade: student.final_grade,
                average: (student.final_grade + student.midterm_grade) / 2
            };

            // // İstemciye 200 (Başarılı) koduyla bulunan öğrencinin bilgilerini içeren bir yanıt gönderir.
                 res.status(200).json({
                message: "Student Successfully Found",
                student: foundedStudent
            });
        }
```

```
app.get('/students/getAll', (req, res) => {

     // Tüm öğrencileri seçmek için bir SQL sorgusu yapılır.
    dbConnection.query(
        "SELECT * FROM students ",
        [req.params.id],
        (err, results, fields) => {
            if (err) {
                // Eğer bir sorgulama hatası oluşursa, hatayı konsola yazdırır
                //istemciye 500 (Sunucu Hatası) koduyla bir hata mesajı gönderir.
                console.log("Database query error: ", err);
                 res.status(500).send("Database's Error" + err.stack);
            }

            //// Eğer veritabanında öğrenci bulunamazsa, istemciye 404 (Bulunamadı) koduyla bir hata mesajı gönderir.
            if (results.length === 0) {
                 res.status(404).json({
                    message: 'Students Not Exist',
                });
            }

            // Bulunan öğrencilerin bilgilerini içeren bir dizi oluşturulur.
            const students = [];

            // Her öğrenci için bilgiler alınır ve ortalama notu hesaplanır, sonra diziye eklenir.
            results.forEach(studentData => {
                const student = studentData;
                const foundedStudent = {
                    name: student.name,
                    midterm_grade: student.midterm_grade,
                    final_grade: student.final_grade,
                    average: (student.final_grade + student.midterm_grade) / 2
                };
                students.push(foundedStudent);
            });


            // İstemciye 200 (Başarılı) koduyla tüm öğrencilerin bilgilerini içeren bir yanıt gönderilir.
              res.status(200).json({
                message: "Students Successfully Listed",
                student: students
            });
        }
    );
```