Anindito Izdihardian Wibisono
1506757466
APAP – C

Dalam tutorial kali ini, saya belajar mengenai implementasi model dan service dengan menggunakan Spring Boot. Model disini merupakan implementasi suatu object class yang dapat digunakan dalam MVC, sementara service pada Spring Boot diimplementasikan sebagai suatu Java interface. Di tutorial kali ini, service yang saya buat bersifat in-memory; artinya, ketika program dimatikan lalu dijalankan kembali, isinya akan kembali kosong tanpa ada sisa data yang tersimpan dari sesi sebelumnya.

LATIHAN 1: METHOD SELECTSTUDENT

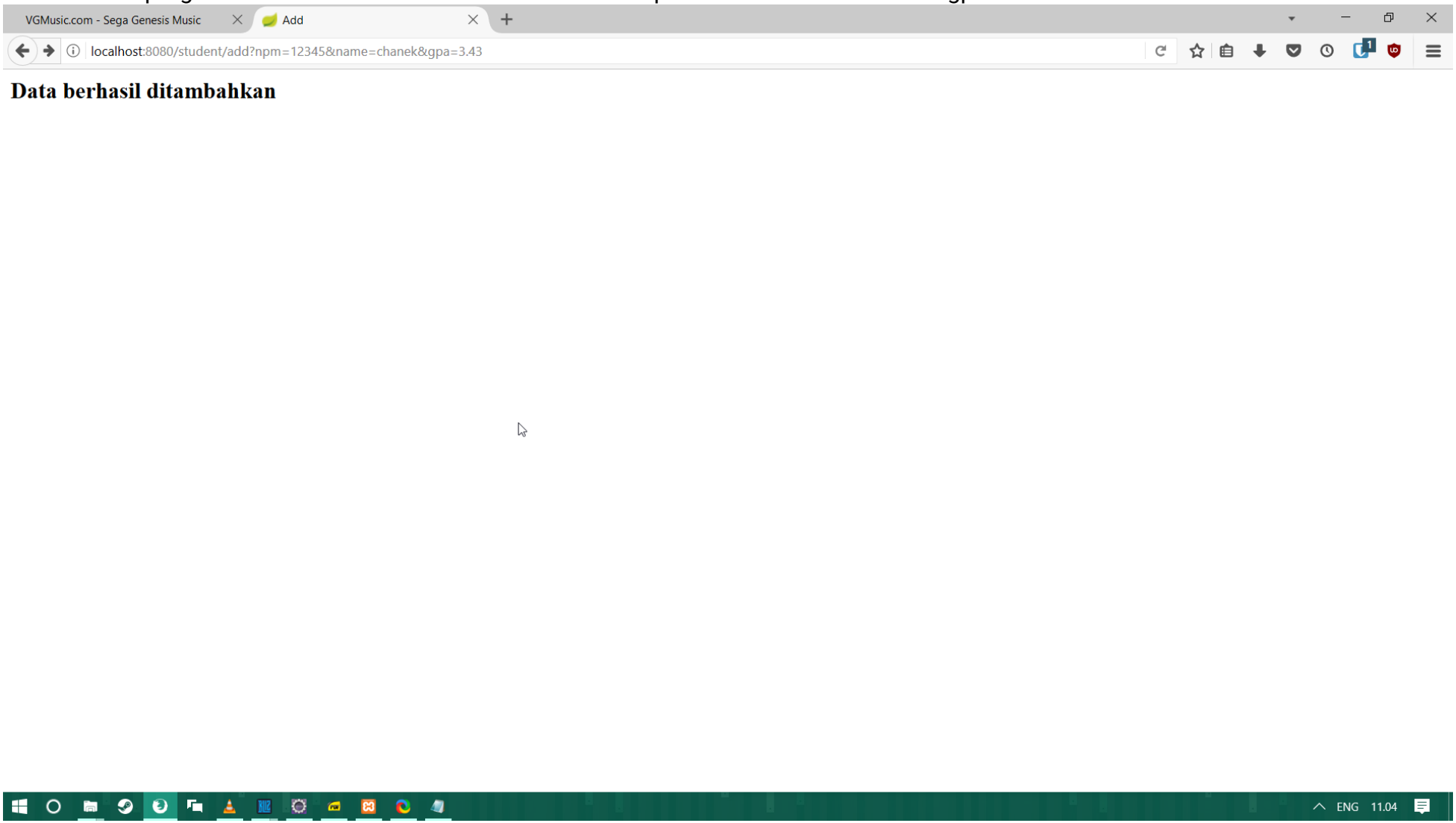
Berikut adalah potongan kode method selectStudent:

```
@Override
public StudentModel selectStudent(String selNpm) {
     for (StudentModel sm : studentList) {
          if (sm.getNpm().equalsIgnoreCase(selNpm)) {
               return sm;
          }
     }
     return null;
}
```

Method tersebut meng-iterate arraylist studentList yang ada di memori, lalu mengecek apakah npm yang dimasukkan ada pada arraylist. Method mengembalikan object StudentModel jika npm yang dicari ada, atau null jika tidak ada.

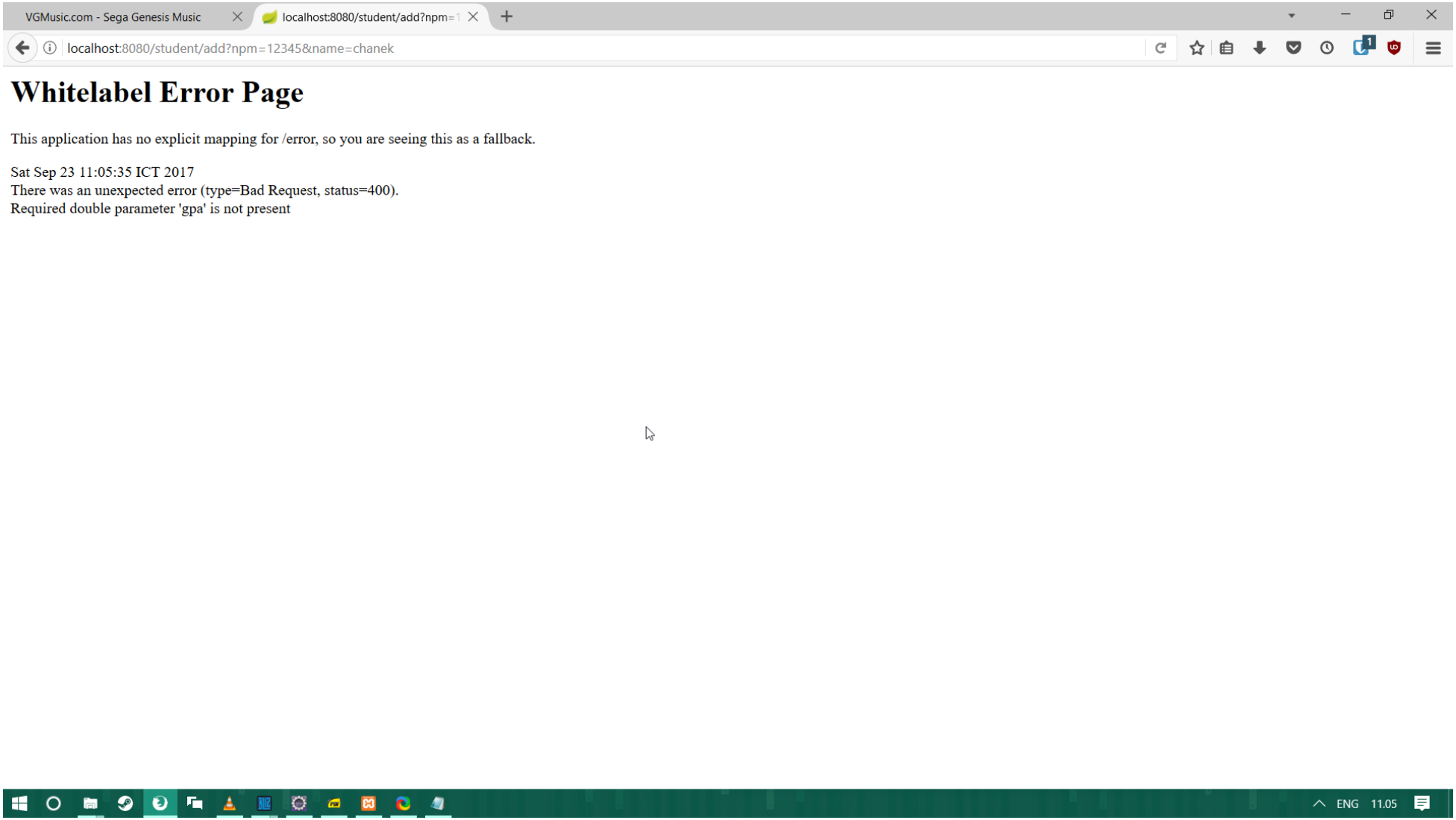
LATIHAN 2: FUNGSI ADD DAN CONTROLLER

2-1 Jalankan program dan buka localhost:8080/student/addnpm=12345&name=chanek&gpa=3.43



Objek Student dengan nama chanek, npm 12345, dan gpa 3.43 berhasil ditambahkan.

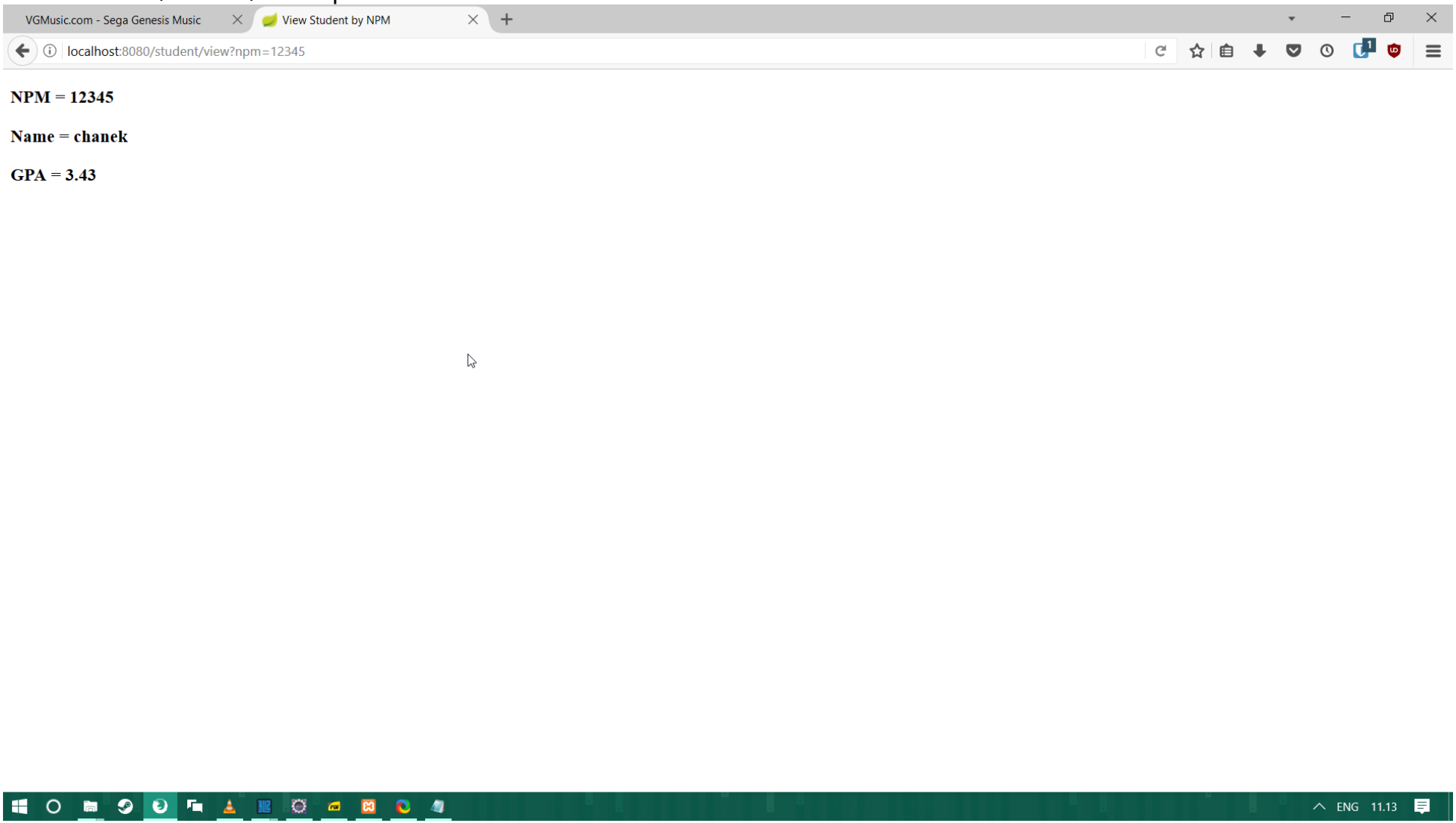
2-2 Buka localhost:8080/student/addnpm=12345&name=chanek



Muncul Whitelabel Error (400 Bad Request), karena field gpa yang seharusnya wajib diisi tidak ada pada masukan.

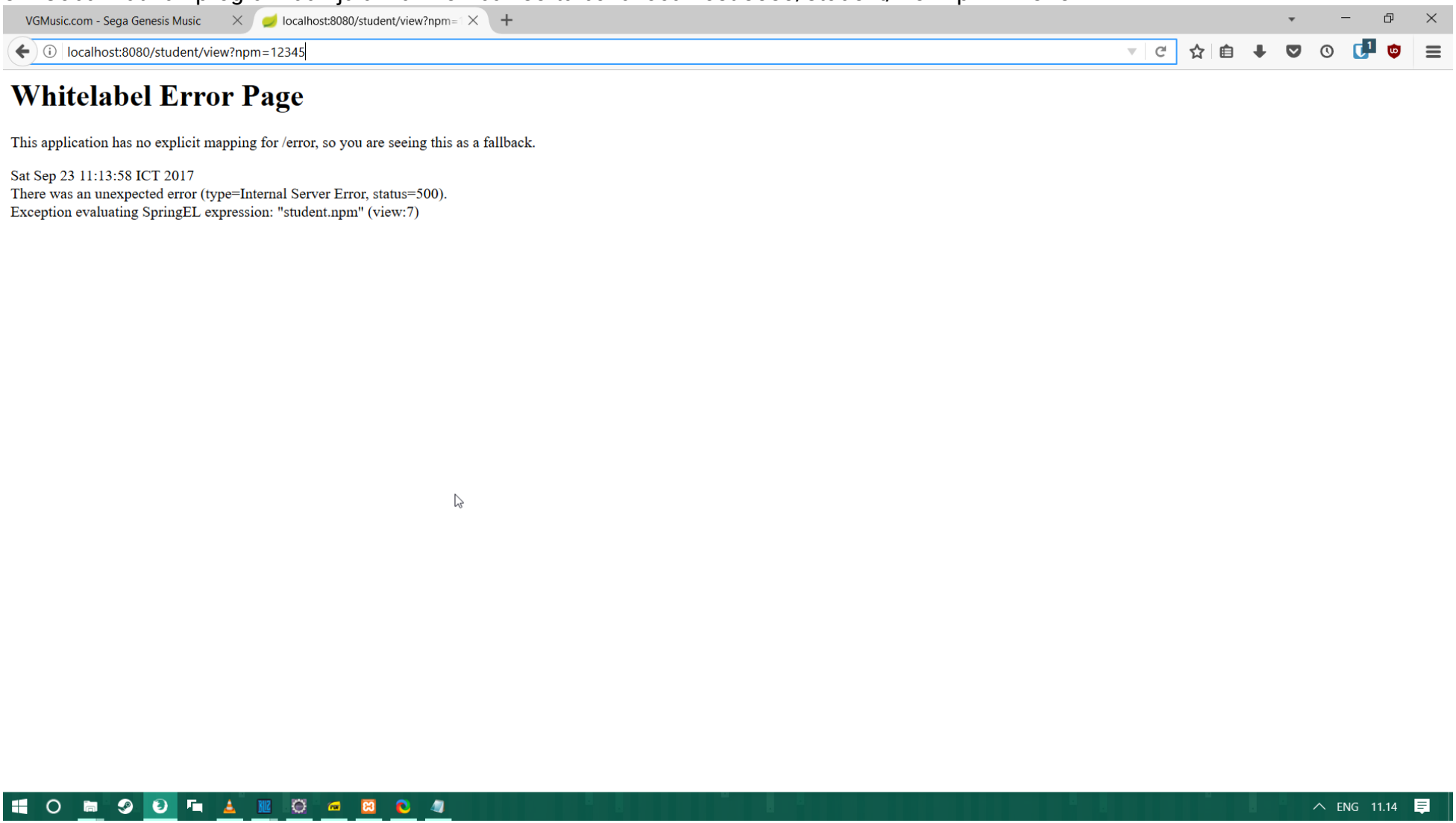
LATIHAN 3: METHOD VIEW BY NPM

3-1 Jalankan program dan buka localhost:8080/student/addnpm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/view?npm=12345



Data objek Student dengan npm 12345 berhasil ditampilkan.

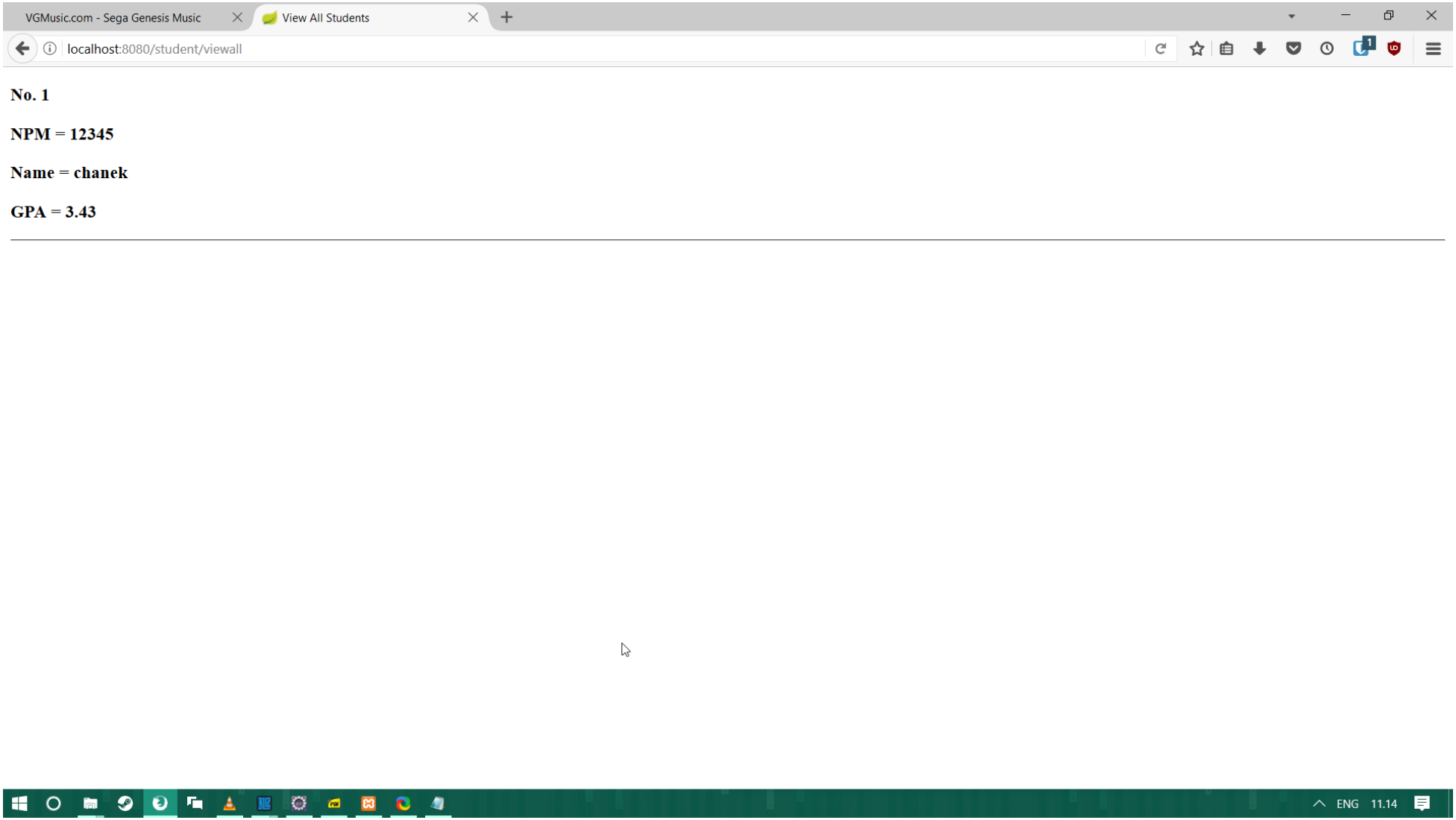
3-2 Coba matikan program dan jalankan kembali serta buka localhost:8080/student/viewnpm=12345



Muncul Whitelabel Error (500 Internal Server Error), karena arraylist yang dicek tidak mengandung npm yang dicari. Hal ini karena arraylist yang digunakan sifatnya in-memory, sehingga isinya akan terhapus apabila program dimatikan.

LATIHAN 4: METHOD VIEW ALL

4-1 Jalankan program dan buka localhost:8080/student/addnpm=12345&name=chanek&gpa=3.43 lalu buka localhost:8080/student/viewall

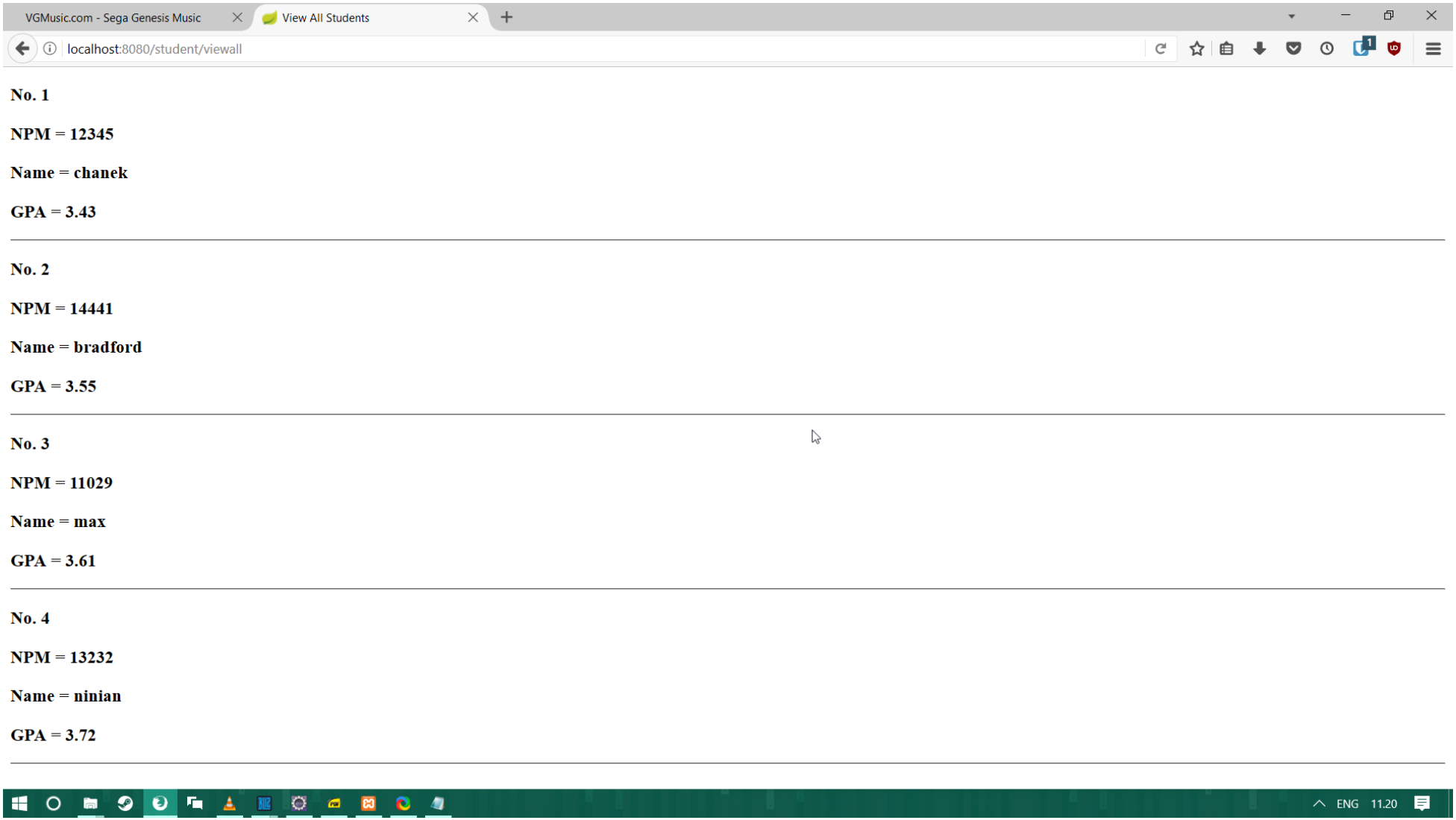


Data seluruh objek Student yang ada berhasil ditampilkan.

4-2 Coba tambahkan data Student lainnya dengan NPM yang berbeda, lalu buka localhost:8080/student/viewall

Berikut adalah perintah menambahkan Student lain:

```
localhost:8080/student/addnpm=14441&name=bradford&gpa=3.55
localhost:8080/student/addnpm=11029&name=max&gpa=3.61
localhost:8080/student/addnpm=13232&name=ninian&gpa=3.72
```



Hasil viewall setelah adanya tambahan objek Student. Student baru dengan nama “bradford”, “max”, dan “ninian” ikut muncul bersama data Student bernama “chanek”.

LATIHAN 5: FITUR TAMBAHAN VIEW BY PATH DAN DELETE

Berikut adalah potongan kode method viewByPath:

```
@RequestMapping("/student/view/{npm}")
    public String viewByPath (@PathVariable(value = "npm", required = true) String npm, Model model)
{
     if (npm.isEmpty()) {
          return "viewError";
     } else {
          StudentModel student = studentService.selectStudent(npm);
          if (student == null) {
               return "viewError";
          }
          model.addAttribute("student", student);
          return "view";
     }
    }
```

Berikut adalah potongan kode method deleteStudent serta method controllernya:

```
public boolean deleteStudent(String delNpm) {
          for (StudentModel sm : studentList) {
               if (sm.getNpm().equalsIgnoreCase(delNpm)) {
                    studentList.remove(studentList.indexOf(sm));
                    return true;
               }
          }
          return false;
     }
```

```
 @RequestMapping("/student/delete/{npm}")
    public String deleteByPath (@PathVariable(value = "npm", required = true) String npm, Model
model) {
     if (npm.isEmpty()) {
          return "deleteError";
     } else {
          if (studentService.deleteStudent(npm) == false) {
               return "deleteError";
          } else {
               return "delete";
          }
     }
    }
```

Tahapan yang dilakukan dalam mengimplementasikan kedua fitur tersebut adalah:

1. Siapkan logic dasar dari method
2. Implementasikan fungsi dalam kode di file yang bersesuaian (StudentController untuk viewByPath, InMemoryStudentService untuk deleteStudent)
3. Implementasikan halaman view yang diperlukan method di folder templates
4. Uji method yang telah dibuat

Cara kerja method deleteStudent mirip dengan method selectStudent. Method meng-iterate terhadap arraylist, lalu menghapus entri dengan npm yang dimasukkan jika benar ada di arraylist tersebut. Bedanya adalah di return type, yaitu boolean dalam kasus deleteStudent. Nilai boolean ini diperlukan method di controller untuk mennentukan view manakah (sukses/gagal) yang perlu ditampilkan browser.