

Assignment

ExpressJS Web Application

with MySQL and Mongo Databases

Contents

| | |
|---|----|
| Introduction | 3 |
| MySQL Database..... | 3 |
| Mongo Database | 4 |
| Marks | 4 |
| Submission of the Project | 5 |
| Overview of Web App | 5 |
| GET /(Home Page) | 5 |
| GET /students (Students Page) | 6 |
| GET /students/edit/:sid, POST /students/edit/:sid (Update Student Page) .7 | |
| GET /students/add, POST /students/add (Add Student Page) | 9 |
| GET /grades (Grades Page) | 12 |
| GET /lecturers (Lecturers (MongoDB) Page) | 13 |
| GET /lecturers/delete/:lid | 14 |

Introduction

Write an ExpressJS web application that queries and updates a MySQL database and a Mongo database.

The MySQL database is called [proj2024mysql](#) and can be downloaded from the *Project* section of Moodle for this module (*proj2024Mysql.sql*).

MySQL Database

The [proj2024mysql](#) database consists of 3 tables:

| student | |
|----------------------|--------------|
| Column Name | Details |
| sid | Student ID |
| name | Student Name |
| age | Student Age |

| module | |
|--------------------------|--|
| Column Name | Details |
| mid | Module ID |
| name | Module Name |
| credits | Module Credits |
| lecturer | The ID of the lecturer for this module |

| grade | |
|-----------------------|------------------------------------|
| Column Name | Details |
| sid | Student ID |
| mid | Module ID |
| grade | The Student's grade in this module |

Full details of the Primary and Foreign Keys as well as datatypes can be found in the database itself.

Import *proj2024Mysql.sql* into MySQL to setup the initial database.

Mongo Database

The Mongo database can be downloaded from the *Project* section of Moodle for this module (*proj2024MongoDB.json*).

The Mongo database name must be `proj2024MongoDB`, and the collection containing the documents must be called `lecturers`.

The file *proj2024MongoDB.json* contains:

```
{ "_id": "L001", "name": "Mark Collins", "did": "ART" }
{ "_id": "L002", "name": "Barbara O'Toole", "did": "ART" }
{ "_id": "L003", "name": "Paddy McDonagh", "did": "CIV" }
{ "_id": "L004", "name": "Josie Sullivan", "did": "COM" }
{ "_id": "L005", "name": "Tommy Hyde", "did": "COM" }
{ "_id": "L006", "name": "Anne Mulligan", "did": "ENG" }
{ "_id": "L007", "name": "Arthur McNamee", "did": "ENG" }
{ "_id": "L008", "name": "Fiona Murphy", "did": "MAT" }
{ "_id": "L009", "name": "John Armstrong", "did": "MAT" }
{ "_id": "L010", "name": "Cathal O'Donovan", "did": "MEC" }
{ "_id": "L011", "name": "Fergus McMathuna", "did": "SOC" }
```

Import *proj2024MongoDB.json* into MongoDB to setup the initial database as follows:

```
mongoimport.exe --db=proj2024MongoDB --collection=lecturers --
file="<path proj2024MongoDB.json>"
```

Marks

90% of the marks for this assignment will be for implementing the functionality in this document.

10% of the marks for this assignment will be for innovation, extra functionality, exceeding the requirements listed in this document.

Any innovation etc. must be clearly indicated in a file called *Innovation.pdf* stored in the root folder of your application.

NOTE: Students may be invited to an MS Teams meeting for a [viva](#) explanation of any or all parts of their submission.

Plagiarism will be dealt with in accordance with the university's [Student Code](#).

Submission of the Project

The zipped project (named GXXXXXXX.zip where GXXXXXXX is your student number) should be uploaded to the *Project* section of Moodle no later than **Friday, August 15th, 2025 at 9:00am.**

You must have a file entitled *GitLocation.pdf* in the root folder of your application which contains a link to the GIT repository you used when developing your project.

Overview of Web App

The web app must run on port 3004, and handle the following HTTP routes and methods:

GET /(*Home Page*)

The Home page consists of 3 links:

- One to the *Students* page
- One to the *Grades* page
- One to the *Lecturers* (MongoDB) page

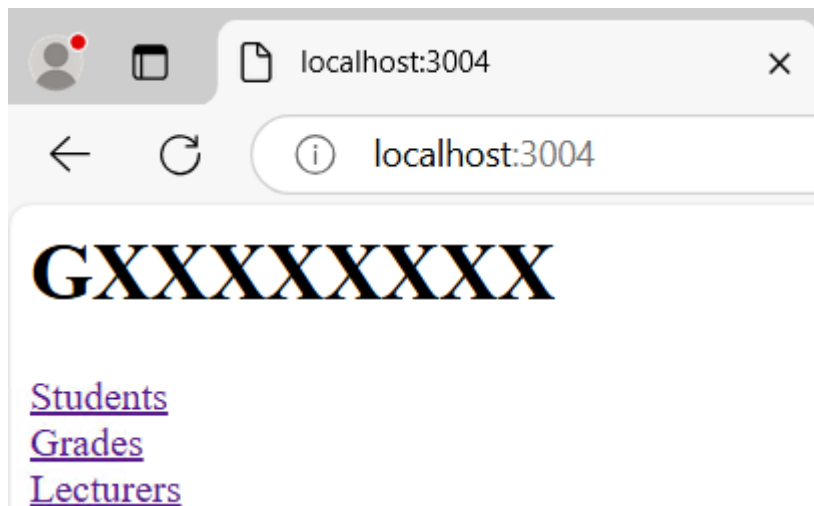


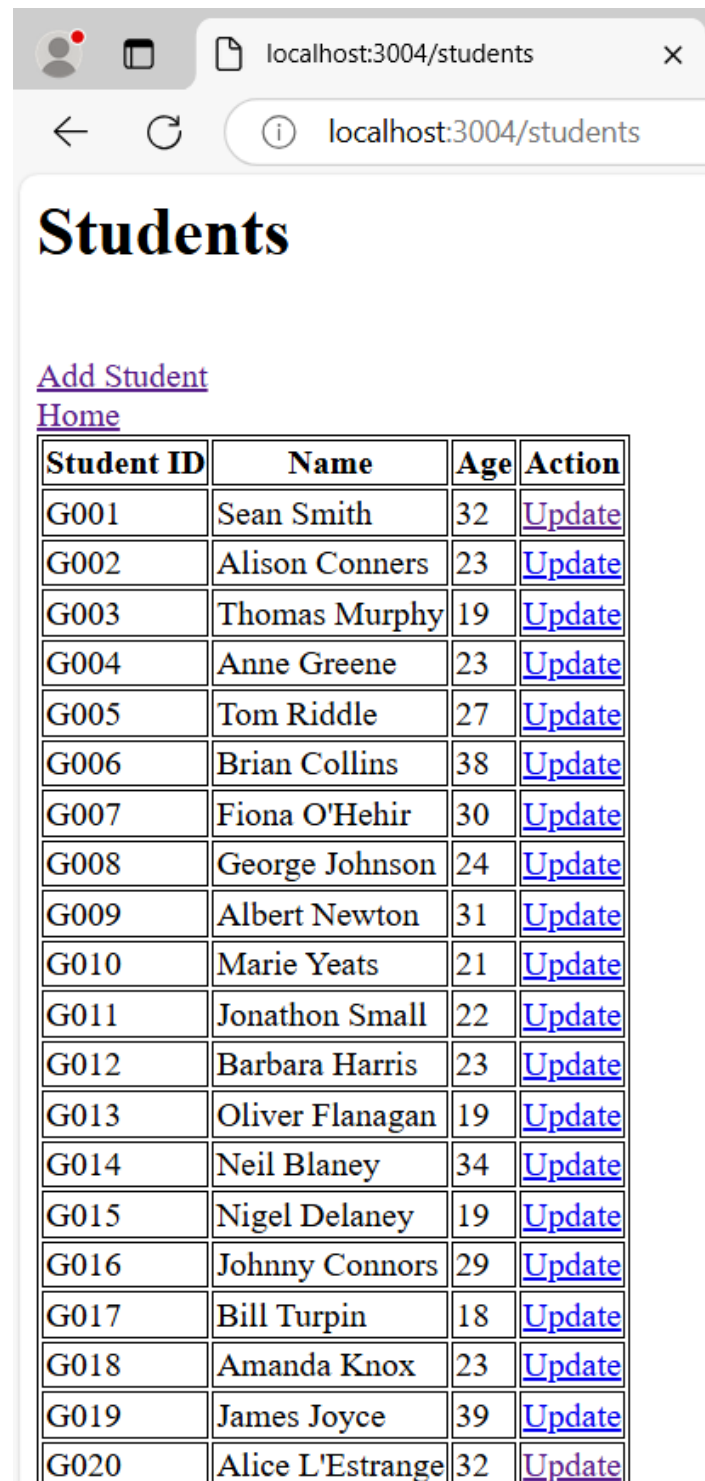
Figure 1 Main Page

GET /students (Students Page)

The *Students* page:

- Shows details of all Students.
- Has an *Update* link for each Student.
- Has an *Add Student* link.
- Has a link to the *Home* page.

The *Students Page* should show data in alphabetical order by student ID.

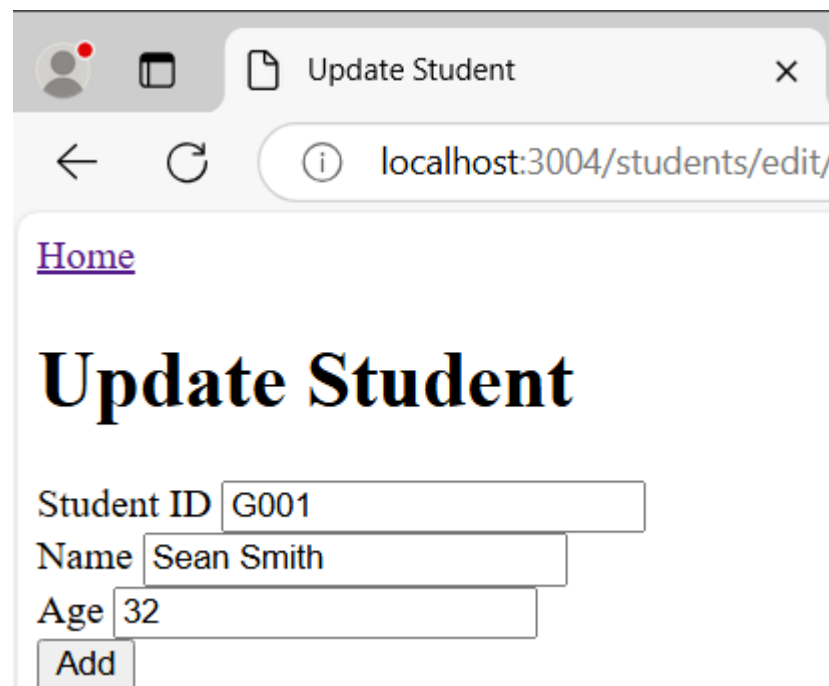


| Student ID | Name | Age | Action |
|------------|------------------|-----|------------------------|
| G001 | Sean Smith | 32 | Update |
| G002 | Alison Conners | 23 | Update |
| G003 | Thomas Murphy | 19 | Update |
| G004 | Anne Greene | 23 | Update |
| G005 | Tom Riddle | 27 | Update |
| G006 | Brian Collins | 38 | Update |
| G007 | Fiona O'Hehir | 30 | Update |
| G008 | George Johnson | 24 | Update |
| G009 | Albert Newton | 31 | Update |
| G010 | Marie Yeats | 21 | Update |
| G011 | Jonathon Small | 22 | Update |
| G012 | Barbara Harris | 23 | Update |
| G013 | Oliver Flanagan | 19 | Update |
| G014 | Neil Blaney | 34 | Update |
| G015 | Nigel Delaney | 19 | Update |
| G016 | Johnny Connors | 29 | Update |
| G017 | Bill Turpin | 18 | Update |
| G018 | Amanda Knox | 23 | Update |
| G019 | James Joyce | 39 | Update |
| G020 | Alice L'Estrange | 32 | Update |

Figure 2 *Students Page*

GET /students/edit/:sid, POST /students/edit/:sid (Update Student Page)

When the *Update* link is clicked beside a student, a GET request is sent to `/students/edit/:sid` and the student's details are shown.



Home

Update Student

Student ID

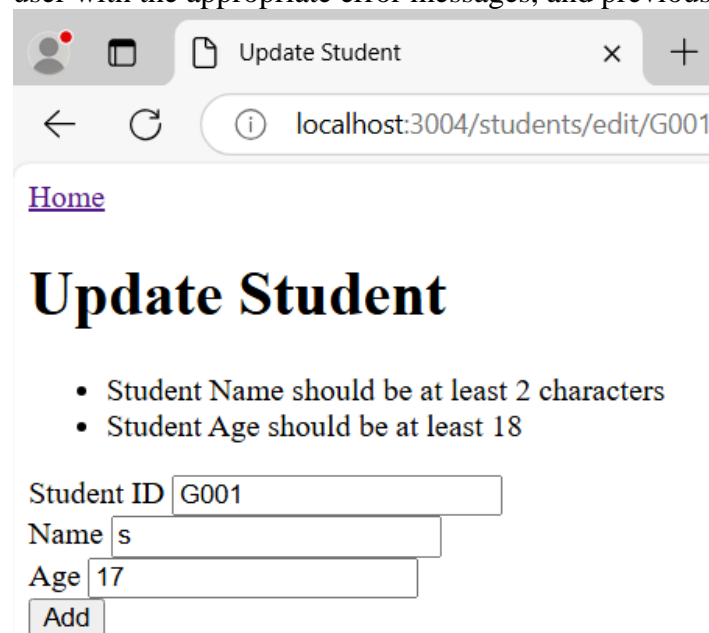
Name

Age

Figure 3 Update Student Page

- Student ID is not editable.
- Name should be a minimum of 2 characters.
- Age should be 18 or older.

If incorrect information is entered, the *Update Student* page should be returned to the user with the appropriate error messages, and previous data entered.



Home

Update Student

- Student Name should be at least 2 characters
- Student Age should be at least 18

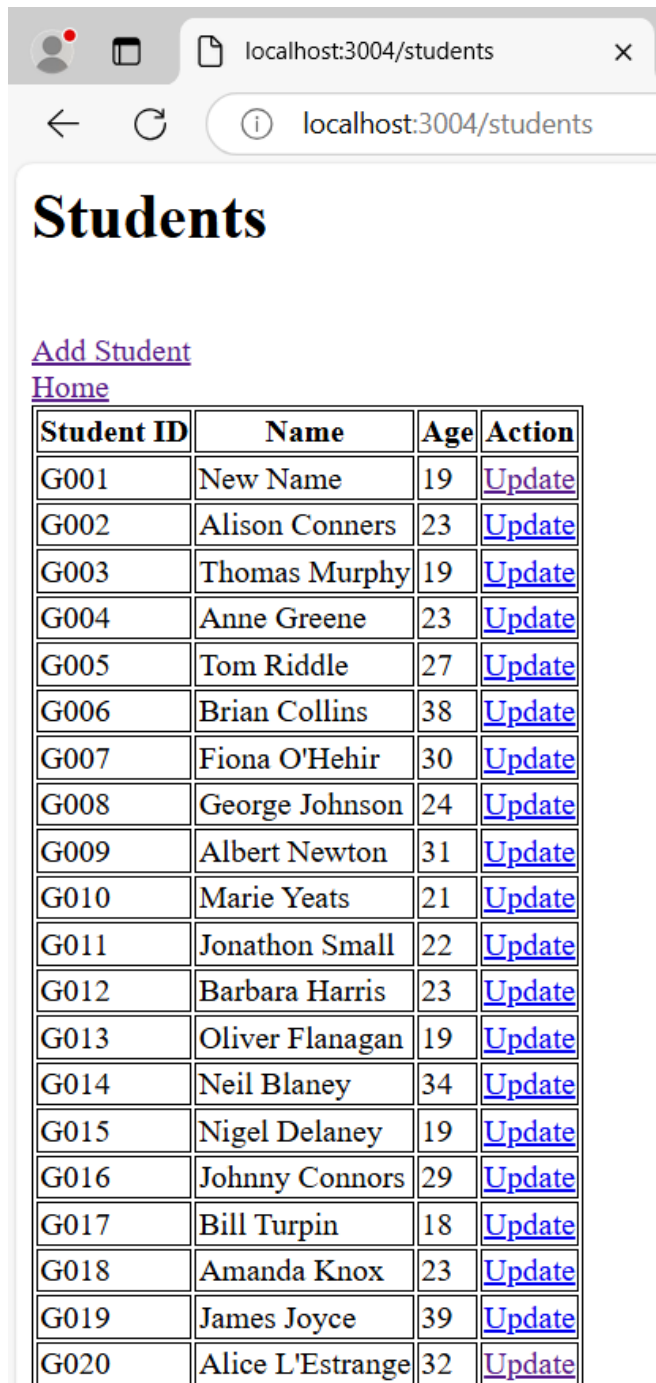
Student ID

Name

Age

Figure 4 Update Student Page with Errors

When a Student has been successfully updated in the MySQL database, the user is returned to the *Students Page*.

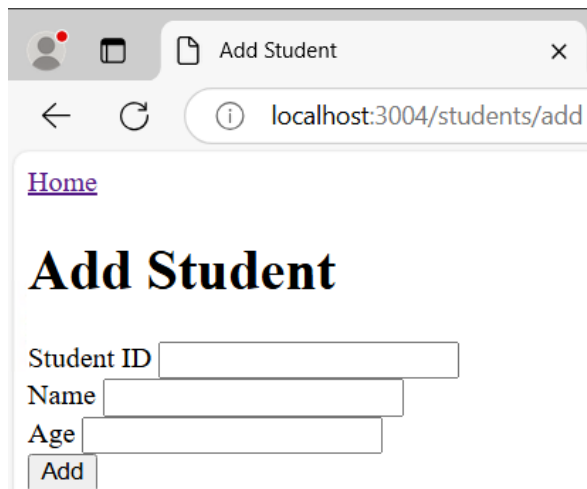


| Student ID | Name | Age | Action |
|------------|------------------|-----|------------------------|
| G001 | New Name | 19 | Update |
| G002 | Alison Connors | 23 | Update |
| G003 | Thomas Murphy | 19 | Update |
| G004 | Anne Greene | 23 | Update |
| G005 | Tom Riddle | 27 | Update |
| G006 | Brian Collins | 38 | Update |
| G007 | Fiona O'Hehir | 30 | Update |
| G008 | George Johnson | 24 | Update |
| G009 | Albert Newton | 31 | Update |
| G010 | Marie Yeats | 21 | Update |
| G011 | Jonathon Small | 22 | Update |
| G012 | Barbara Harris | 23 | Update |
| G013 | Oliver Flanagan | 19 | Update |
| G014 | Neil Blaney | 34 | Update |
| G015 | Nigel Delaney | 19 | Update |
| G016 | Johnny Connors | 29 | Update |
| G017 | Bill Turpin | 18 | Update |
| G018 | Amanda Knox | 23 | Update |
| G019 | James Joyce | 39 | Update |
| G020 | Alice L'Estrange | 32 | Update |

Figure 5 Student successfully updated.

GET /students/add, POST /students/add (Add Student Page)

When the *Add Student* link is clicked on the *Student Page*, a GET request is sent to `/students/add` the following form returned:



Home

Add Student

Student ID

Name

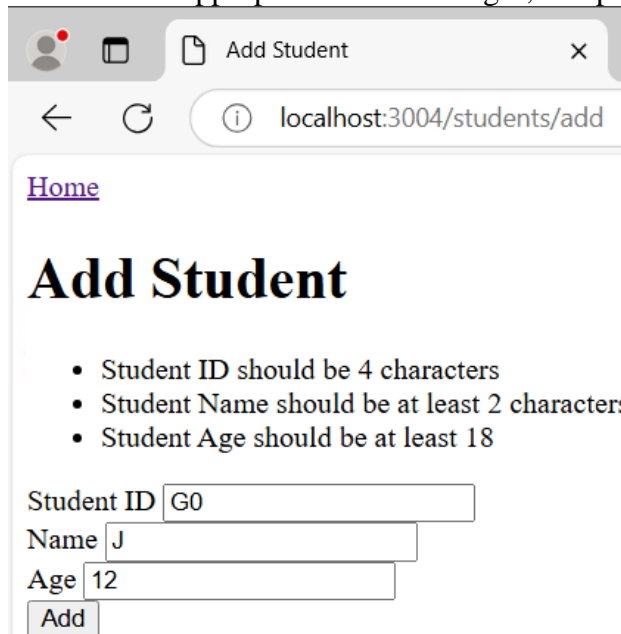
Age

Add

Figure 6 Add Student Page

- Student ID is 4 characters.
- Name should be a minimum of 2 characters.
- Age should be 18 or older.

If incorrect information is entered, the *ADD Student* page should be returned to the user with the appropriate error messages, and previous data entered.



Home

Add Student

- Student ID should be 4 characters
- Student Name should be at least 2 characters
- Student Age should be at least 18

Student ID

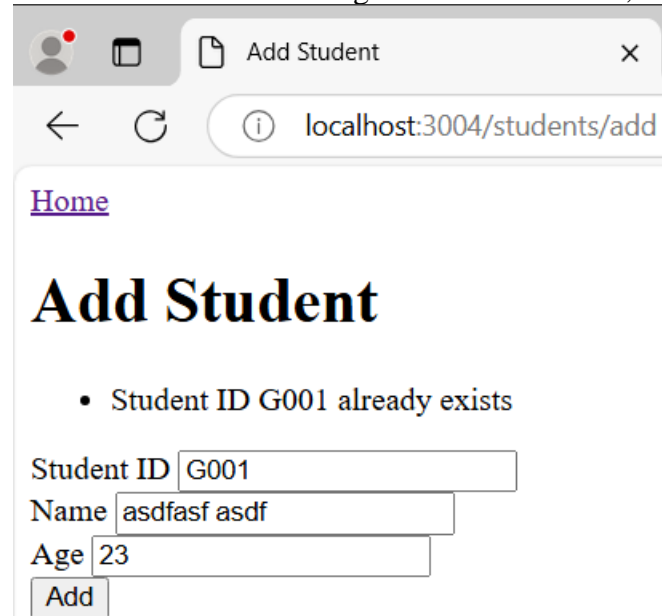
Name

Age

Add

Figure 7 Add Student Page with Errors

If a student with an existing student ID is added, an error message should be shown:



The screenshot shows a web browser window with a single tab titled 'Add Student'. The address bar displays 'localhost:3004/students/add'. The page content includes a link for 'Home', a main heading 'Add Student', and a bulleted error message: 'Student ID G001 already exists'. Below the message is a form with three input fields: 'Student ID' containing 'G001', 'Name' containing 'asdfasf asdf', and 'Age' containing '23'. An 'Add' button is positioned at the bottom of the form.

[Home](#)

Add Student

- Student ID G001 already exists

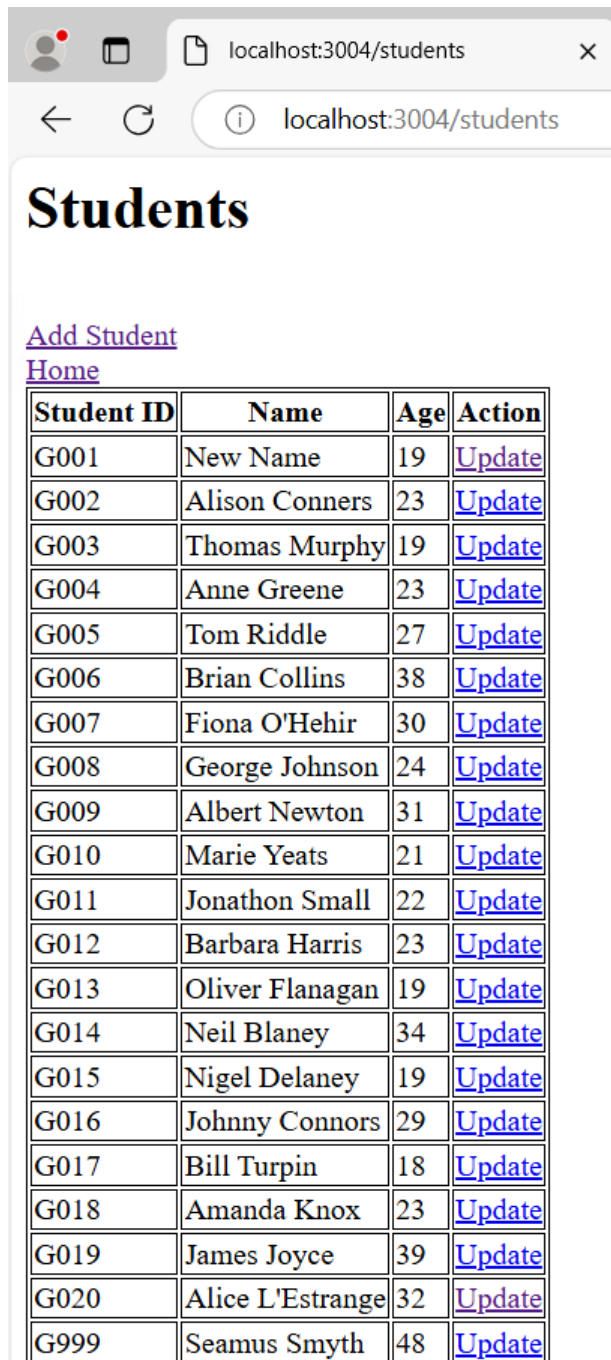
Student ID

Name

Age

Figure 8 Student with ID G001 already exists

When a Student has been successfully added to the MySQL database, the user is returned to the *Students Page*.



| Student ID | Name | Age | Action |
|------------|------------------|-----|------------------------|
| G001 | New Name | 19 | Update |
| G002 | Alison Conners | 23 | Update |
| G003 | Thomas Murphy | 19 | Update |
| G004 | Anne Greene | 23 | Update |
| G005 | Tom Riddle | 27 | Update |
| G006 | Brian Collins | 38 | Update |
| G007 | Fiona O'Hehir | 30 | Update |
| G008 | George Johnson | 24 | Update |
| G009 | Albert Newton | 31 | Update |
| G010 | Marie Yeats | 21 | Update |
| G011 | Jonathon Small | 22 | Update |
| G012 | Barbara Harris | 23 | Update |
| G013 | Oliver Flanagan | 19 | Update |
| G014 | Neil Blaney | 34 | Update |
| G015 | Nigel Delaney | 19 | Update |
| G016 | Johnny Connors | 29 | Update |
| G017 | Bill Turpin | 18 | Update |
| G018 | Amanda Knox | 23 | Update |
| G019 | James Joyce | 39 | Update |
| G020 | Alice L'Estrange | 32 | Update |
| G999 | Seamus Smyth | 48 | Update |

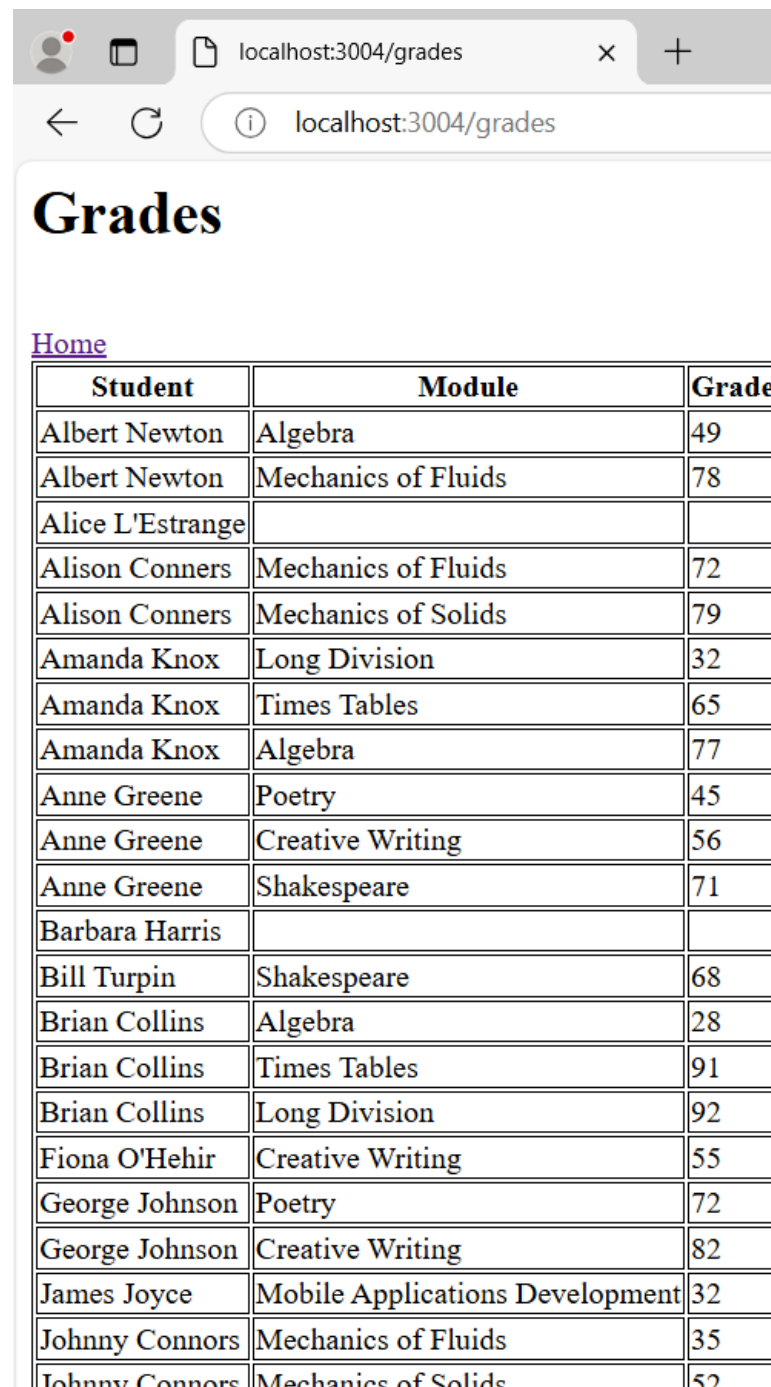
Figure 9 List of Students with newly added Student

GET /grades (Grades Page)

The *Grades* page:

- Shows each Student's *name*.
- Shows the name of each Module studied by a Student.
- Shows the grade received in each Module by a Student.
- If a student isn't studying any Module, then just the Student name should be shown.
- Has a link to the *Home* page.

The *Grades Page* should show data in alphabetical order by student name, and within that in grade order from lowest to highest.



| Student | Module | Grade |
|------------------|---------------------------------|-------|
| Albert Newton | Algebra | 49 |
| Albert Newton | Mechanics of Fluids | 78 |
| Alice L'Estrange | | |
| Alison Connors | Mechanics of Fluids | 72 |
| Alison Connors | Mechanics of Solids | 79 |
| Amanda Knox | Long Division | 32 |
| Amanda Knox | Times Tables | 65 |
| Amanda Knox | Algebra | 77 |
| Anne Greene | Poetry | 45 |
| Anne Greene | Creative Writing | 56 |
| Anne Greene | Shakespeare | 71 |
| Barbara Harris | | |
| Bill Turpin | Shakespeare | 68 |
| Brian Collins | Algebra | 28 |
| Brian Collins | Times Tables | 91 |
| Brian Collins | Long Division | 92 |
| Fiona O'Hehir | Creative Writing | 55 |
| George Johnson | Poetry | 72 |
| George Johnson | Creative Writing | 82 |
| James Joyce | Mobile Applications Development | 32 |
| Johnny Connors | Mechanics of Fluids | 35 |
| Johnny Connors | Mechanics of Solids | 57 |

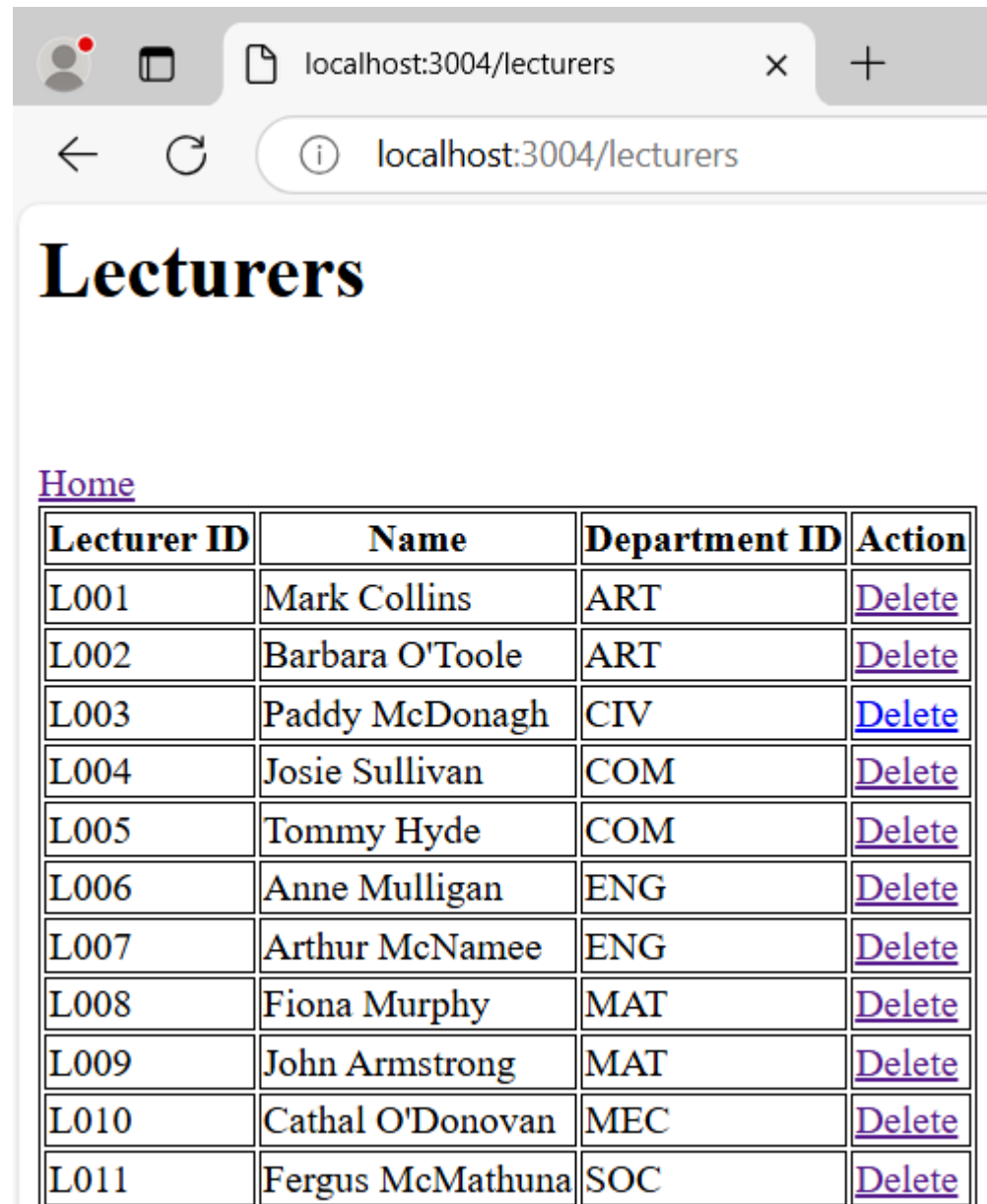
Figure 10 Grades Page

GET /lecturers (Lecturers (MongoDB) Page)

The *Lecturers* (MongoDB) page:

- Shows details of all Lecturers (from MongoDB).
- Has a *Delete* link for each Lecturer.
- Has a link back to the *Main* page.

The *Lecturers Page* should show data in alphabetical order by lecturer ID.



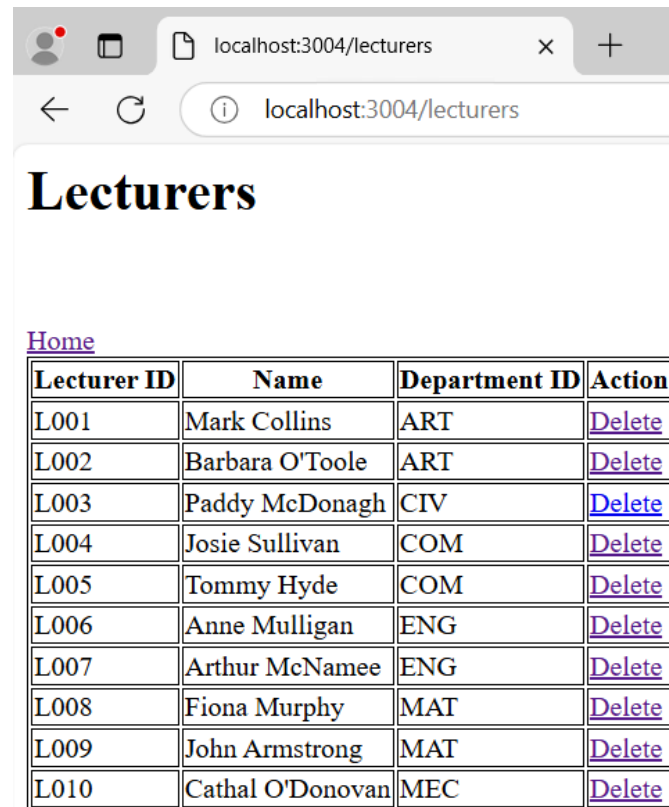
| Lecturer ID | Name | Department ID | Action |
|-------------|------------------|---------------|------------------------|
| L001 | Mark Collins | ART | Delete |
| L002 | Barbara O'Toole | ART | Delete |
| L003 | Paddy McDonagh | CIV | Delete |
| L004 | Josie Sullivan | COM | Delete |
| L005 | Tommy Hyde | COM | Delete |
| L006 | Anne Mulligan | ENG | Delete |
| L007 | Arthur McNamee | ENG | Delete |
| L008 | Fiona Murphy | MAT | Delete |
| L009 | John Armstrong | MAT | Delete |
| L010 | Cathal O'Donovan | MEC | Delete |
| L011 | Fergus McMathuna | SOC | Delete |

Figure 11 *Lecturers (MongoDB)*

GET /lecturers/delete/:lid

When the *Delete* link is clicked beside a lecturer, a GET request is sent to `/lecturers/delete/:lid`.

If the lecturer does not teach any modules, then he/she can be deleted from MongoDB, and the user returned to *Lecturers Page*.



| Lecturer ID | Name | Department ID | Action |
|-------------|------------------|---------------|------------------------|
| L001 | Mark Collins | ART | Delete |
| L002 | Barbara O'Toole | ART | Delete |
| L003 | Paddy McDonagh | CIV | Delete |
| L004 | Josie Sullivan | COM | Delete |
| L005 | Tommy Hyde | COM | Delete |
| L006 | Anne Mulligan | ENG | Delete |
| L007 | Arthur McNamee | ENG | Delete |
| L008 | Fiona Murphy | MAT | Delete |
| L009 | John Armstrong | MAT | Delete |
| L010 | Cathal O'Donovan | MEC | Delete |

Figure 12 List of Lecturers after L011 has successfully been deleted

If the lecturer does teach a module, then an error message should be returned to the user, and the lecturer should not be deleted from MongoDB.

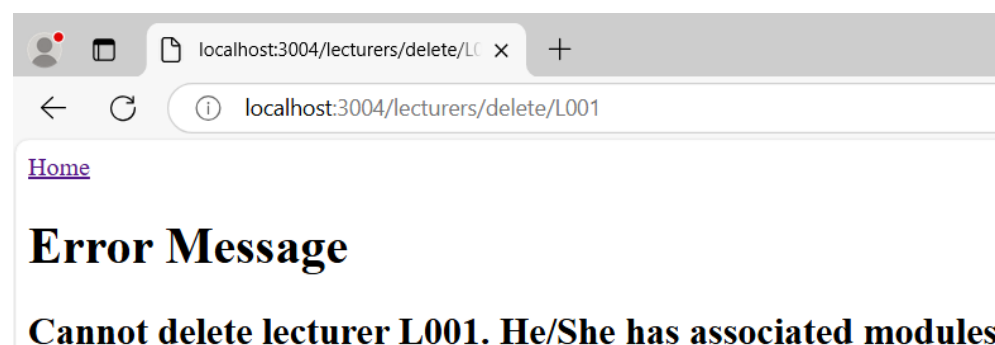


Figure 13 Cannot delete Lecturer L001 from MongoDB as he/she has associated modules.