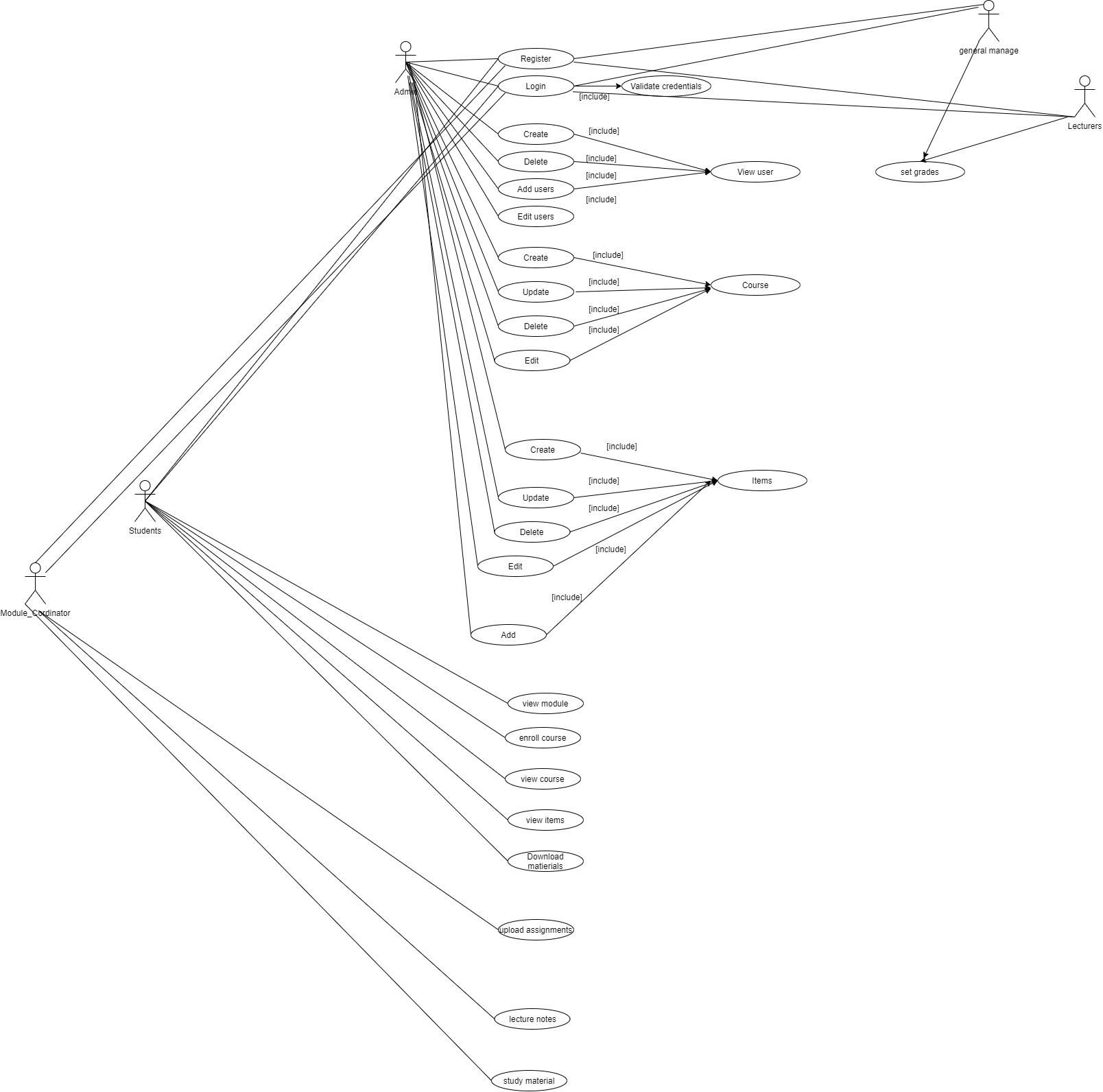
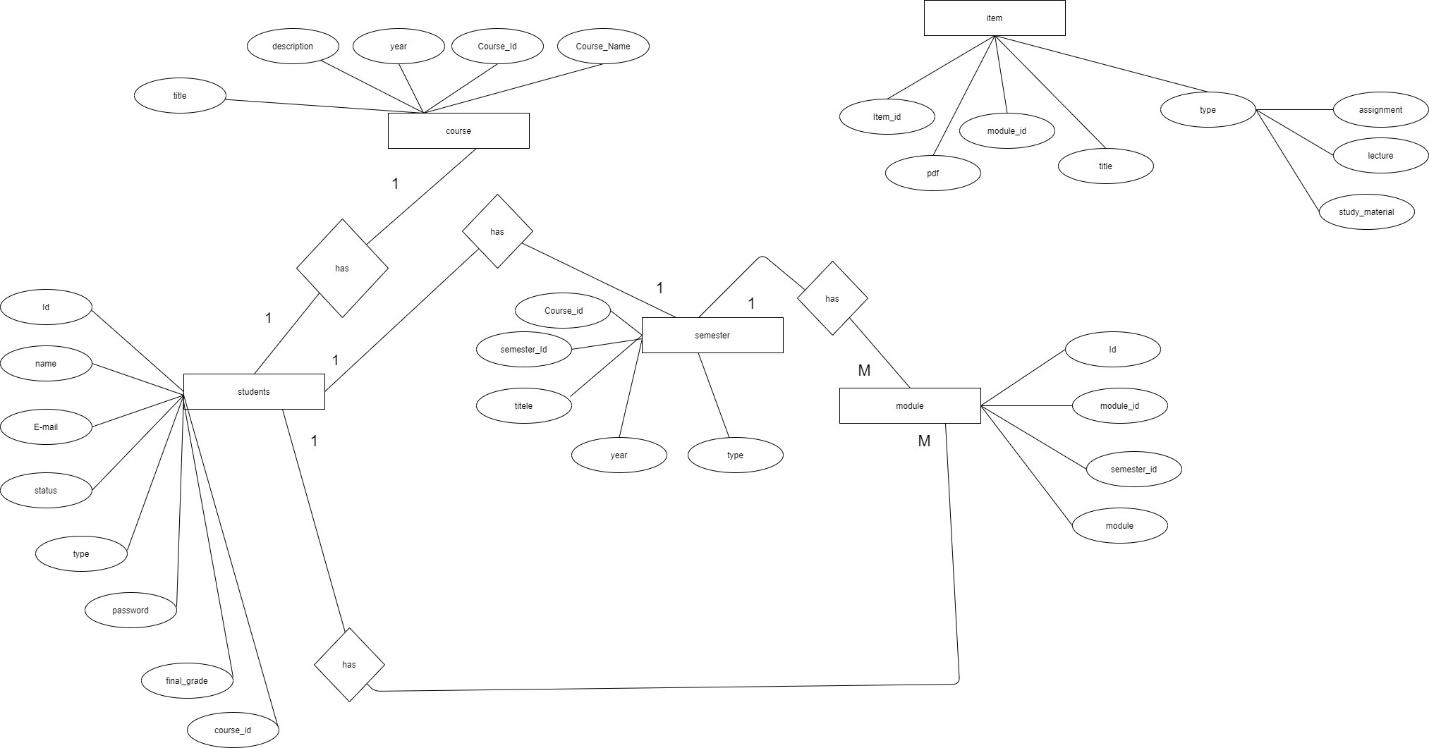
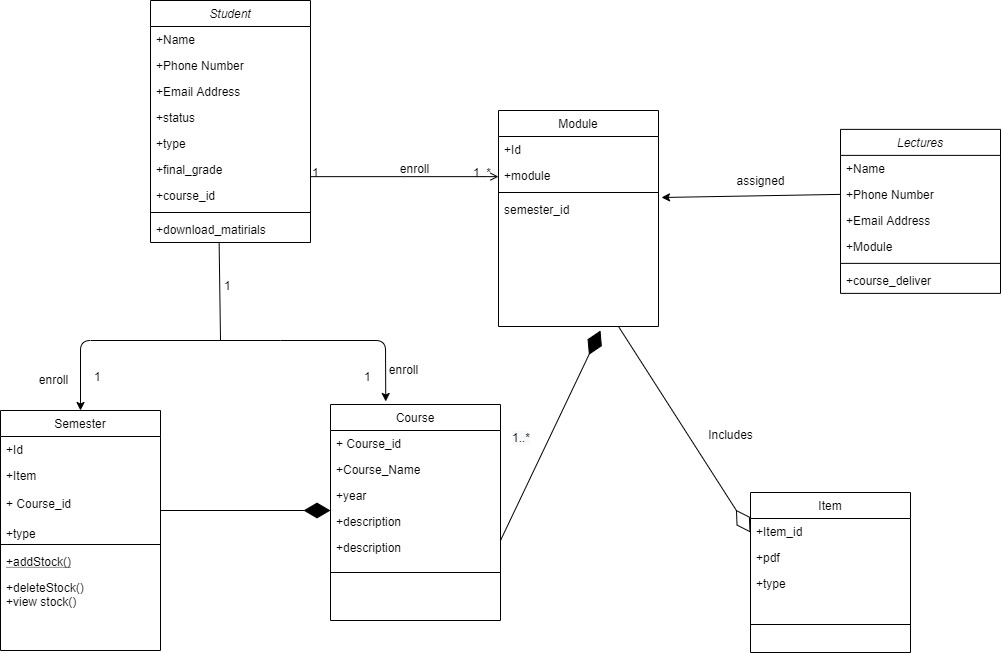
Use case diagram





ER diagrams

Class diagrams



Application Configuration and Run Manual

Comprehensive Application Installation Guide

Using of tools

* Visual Studio
* Bootstrap
* Microsoft Sql

Design pattern

Used singleton design pattern for the implementation.

Singleton pattern is one of the simplest design patterns. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

This pattern involves a single class which is responsible to create an object while making sure that only single object gets created. This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class. Singleton pattern is mostly used in multi-threaded and database applications. It is used in logging, caching, thread pools, configuration settings.

Example: In here singleton is used to set the application details by instantiating only one object.

Text

Description automatically generated

ASP.Net MVC template

Example:

* Controller Files:

Table

Description automatically generated

* Model Files:

Graphical user interface

Description automatically generated

* View Files:

Graphical user interface, text, application

Description automatically generated

Encapsulation:

The process of compartmentalizing the data, providing getters and setters for the attributes.

Example:

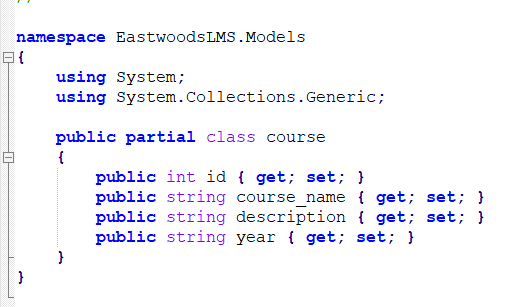
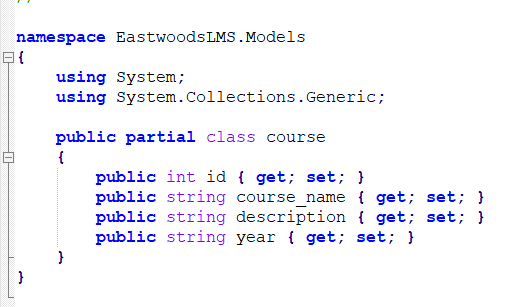


Figure shows the getters and setters for the class, to access the data very securely (secure meaning access modify level).

Abstraction:

The process of including the right attributes to decrease the complex level of the class.

Example: Class course can be simply written with the attributes (id, course name, description and year).



Inheritance:

The ability to implement the parent’s method by its child class is facilitated in Inheritance.

Example:

class User // base class (parent)

{

public int id // User id

public String name // Username

}

class Student: User // derived class (child)

{

public string courseId; // course id field which is unique for a student

}

Polymorphism

Ability to write 2 similar methods and use its function according to the calling style.

Example:

class Student

{

public String comment // Comment

public String commenters // commenting person’s name

// Writing a public comment

public void writeComment(String \_comment)

{

comment = \_comment;

}

// Writing a comment with a person’s name

public void writeComment(String comment, String commenters)

{

comment = \_comment;

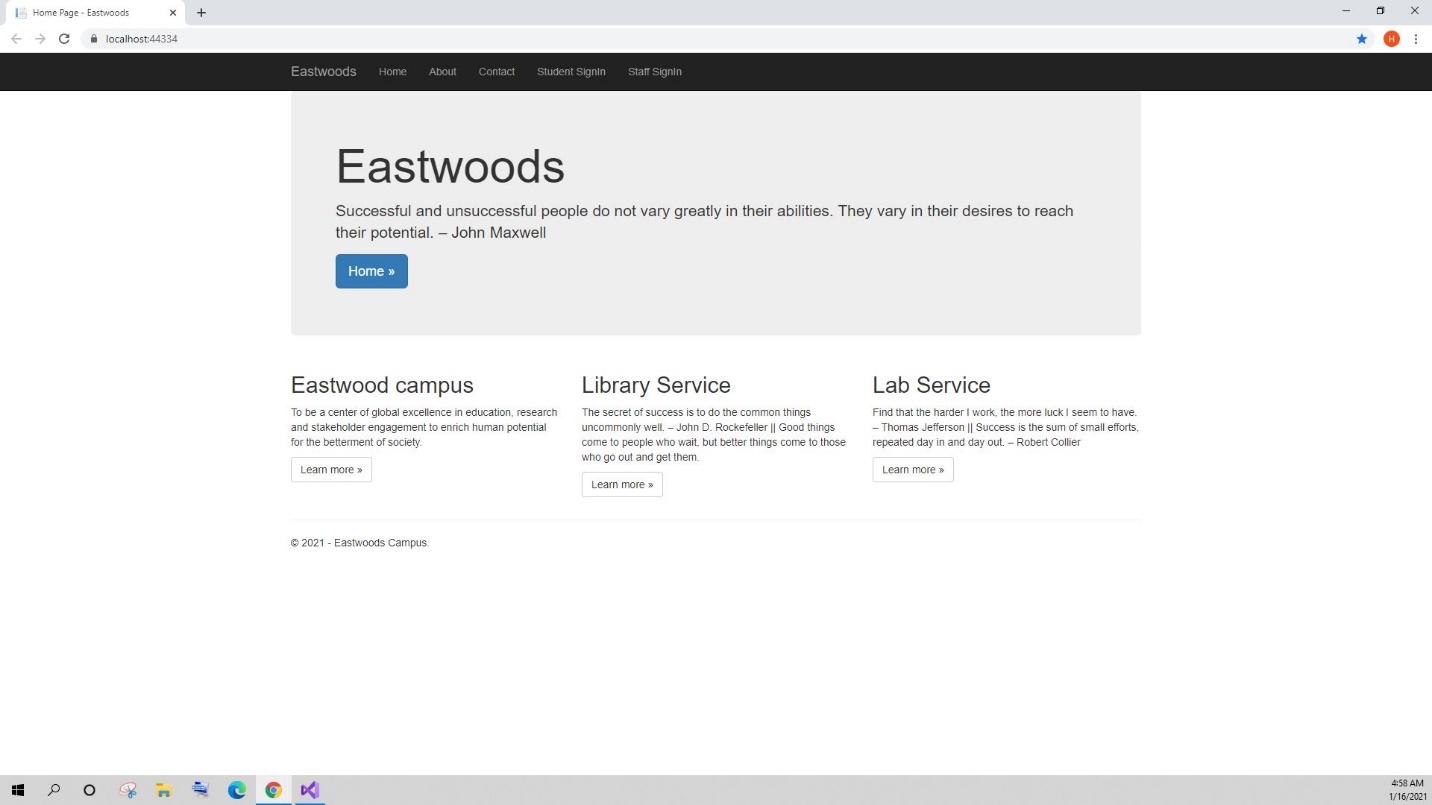
commenters += commenters; // example names: David, Richard

}

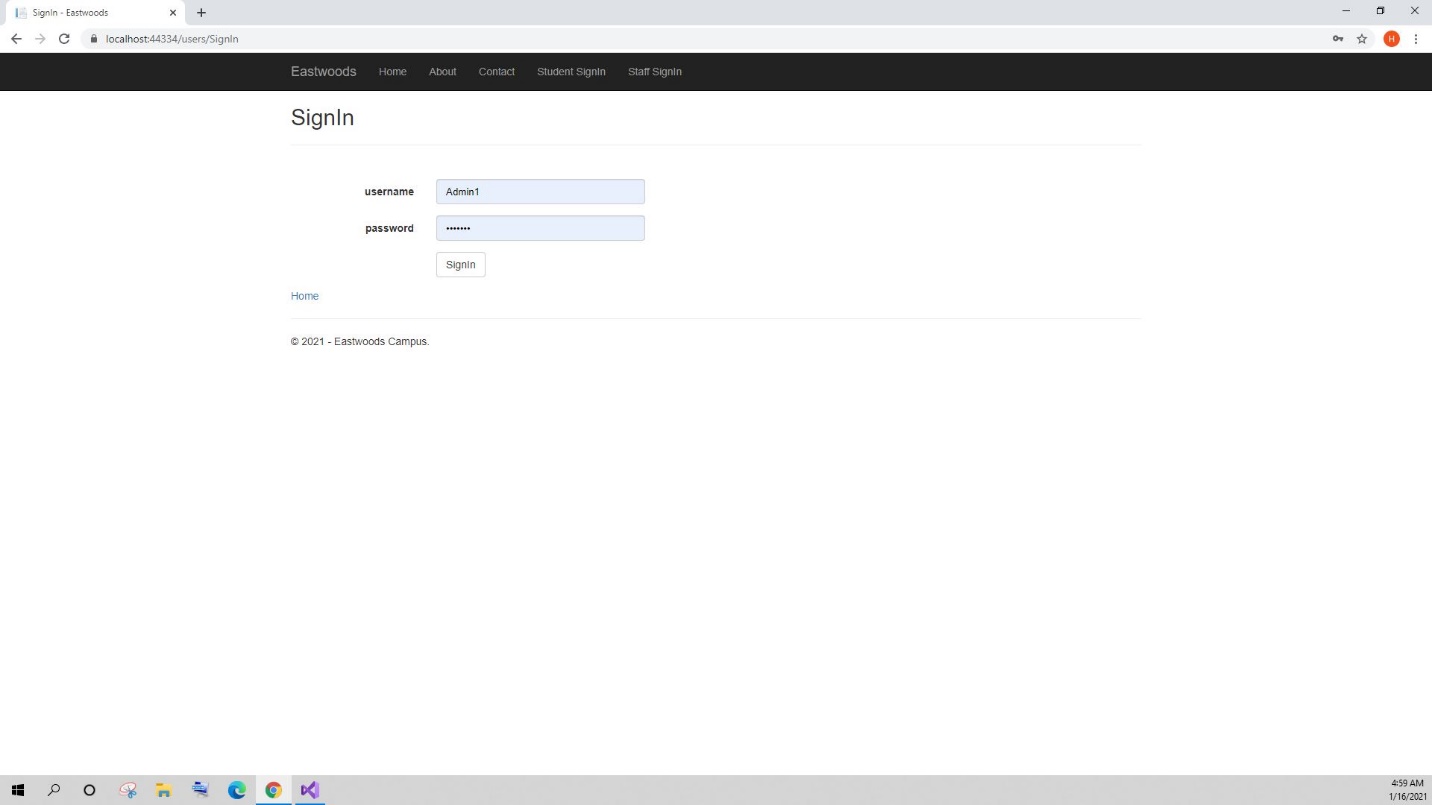
}

# Run manual

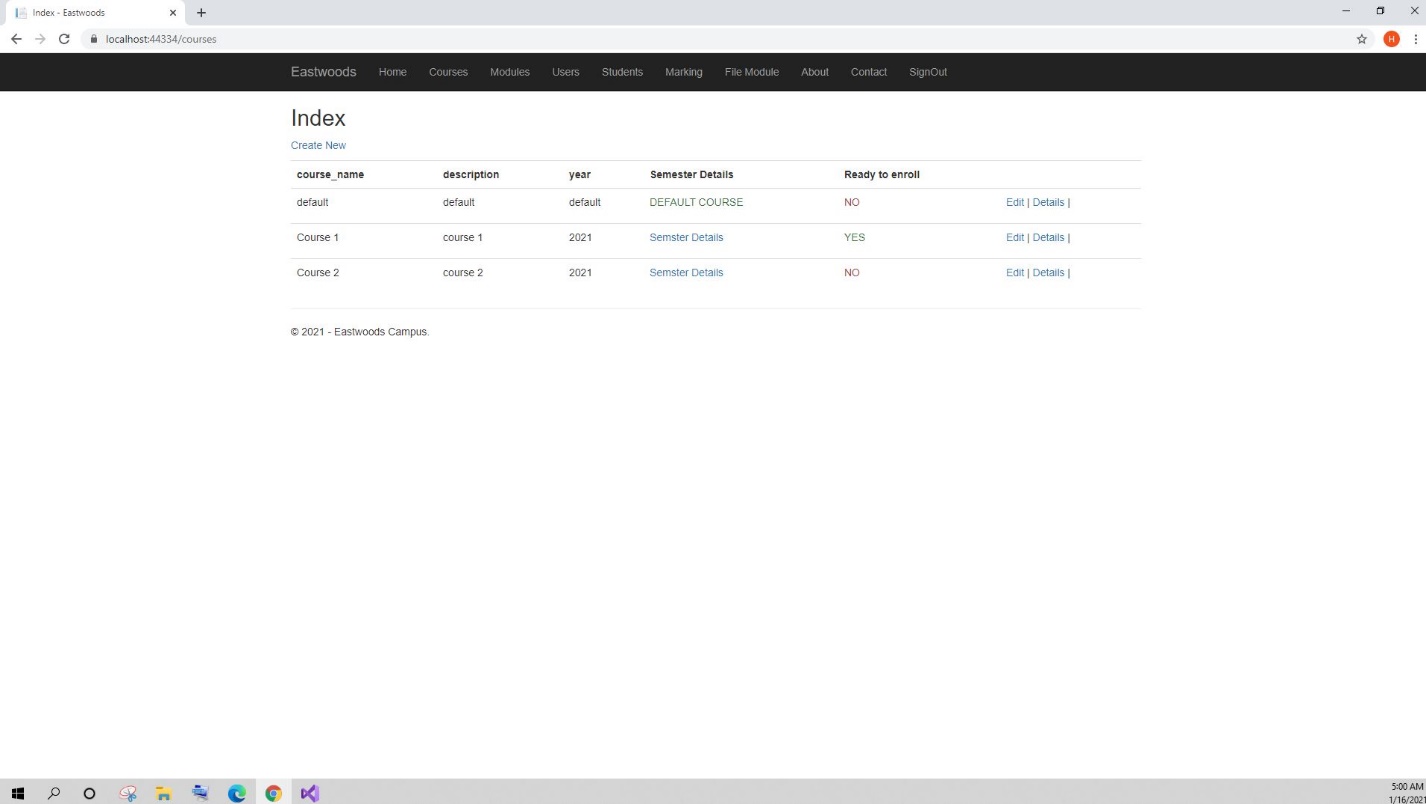
* Home page



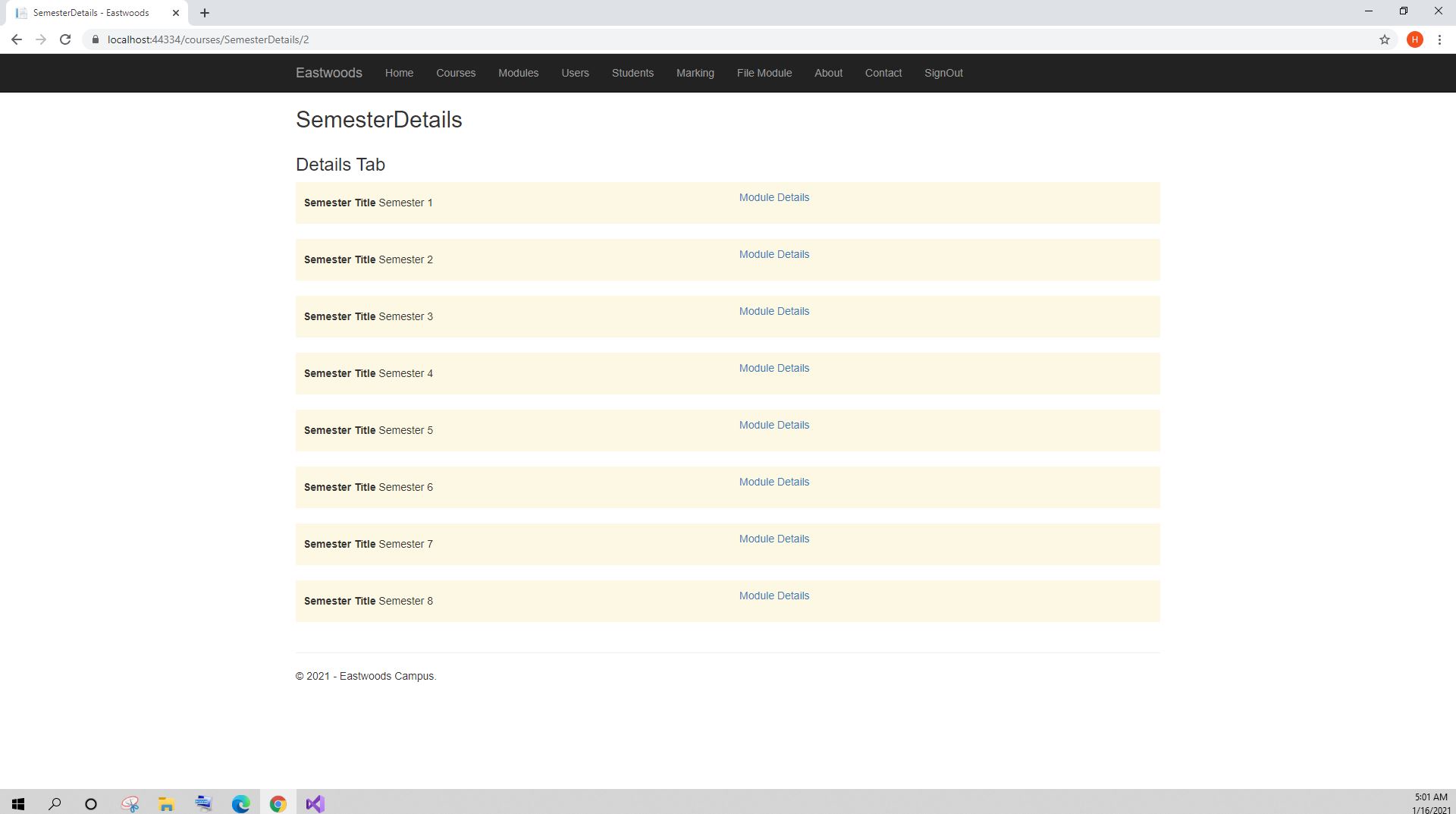
* Sign in



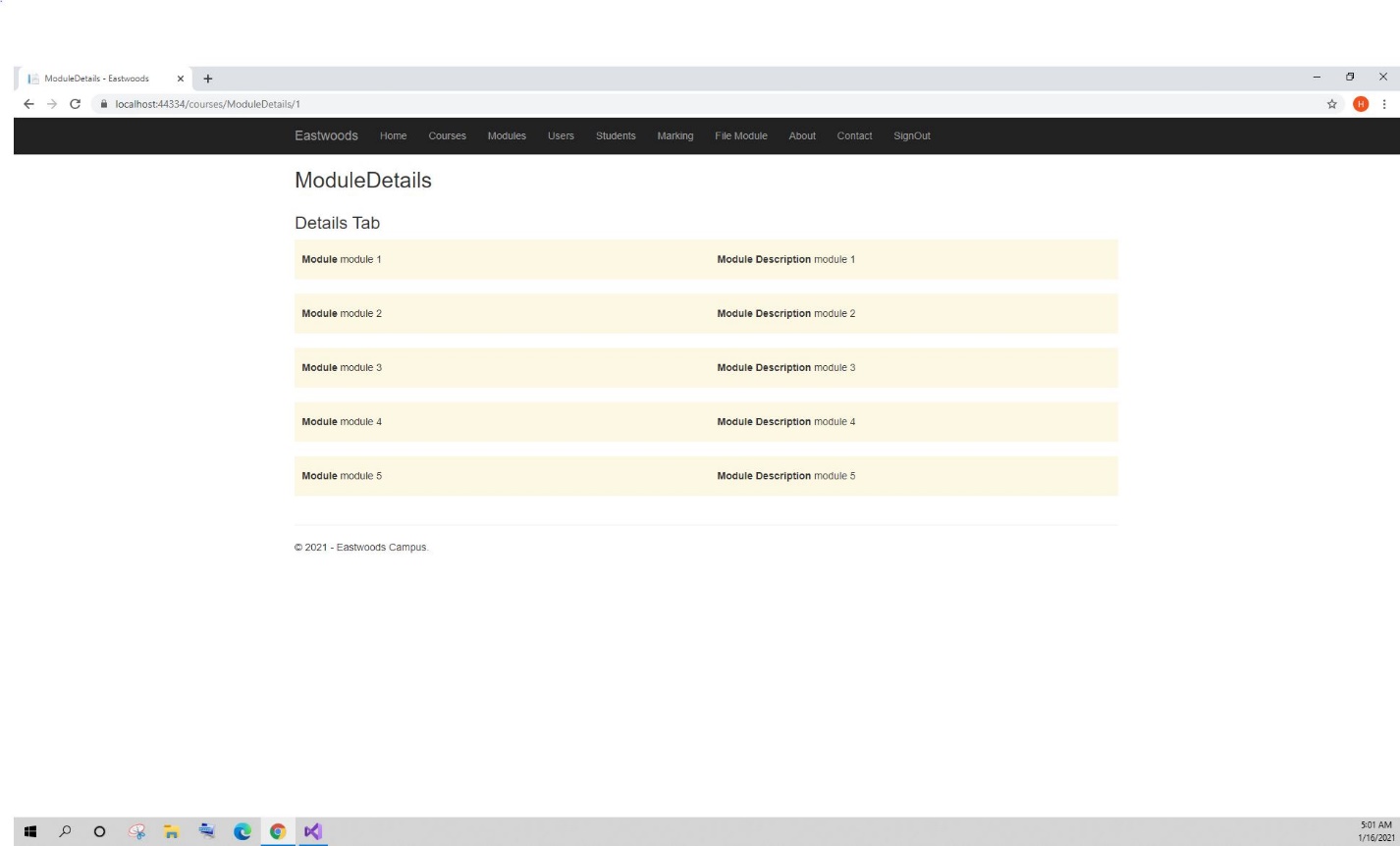
* Course List



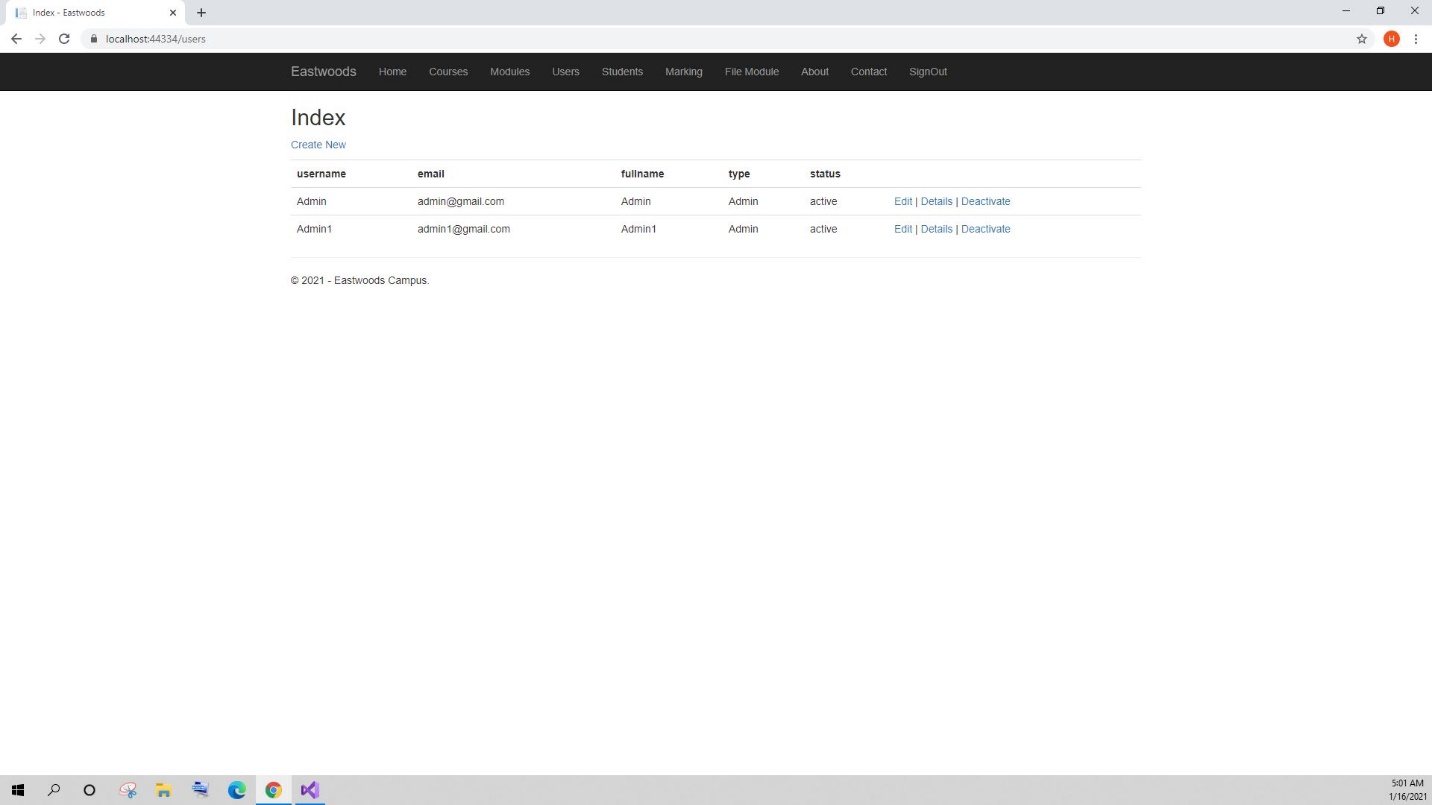
* Semester details

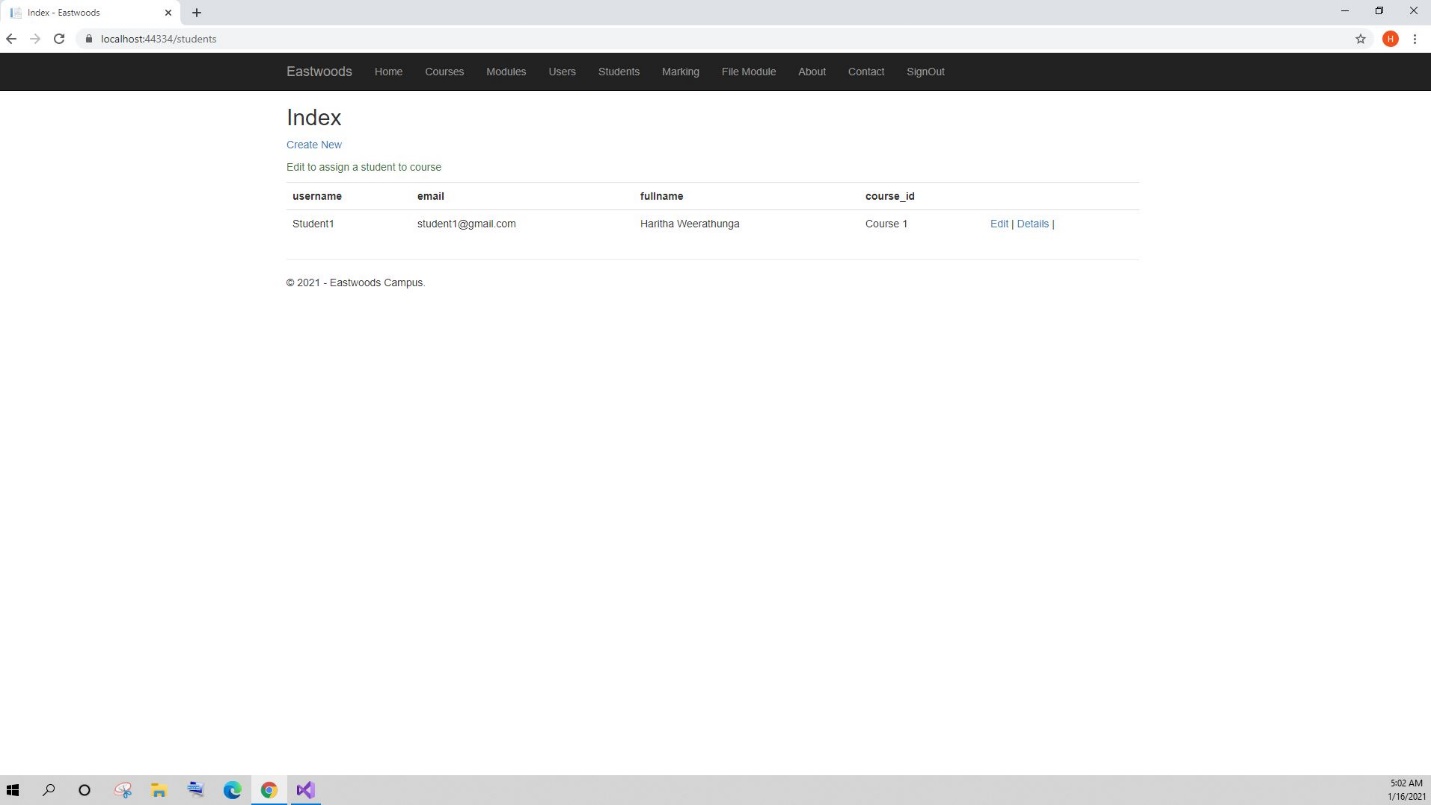


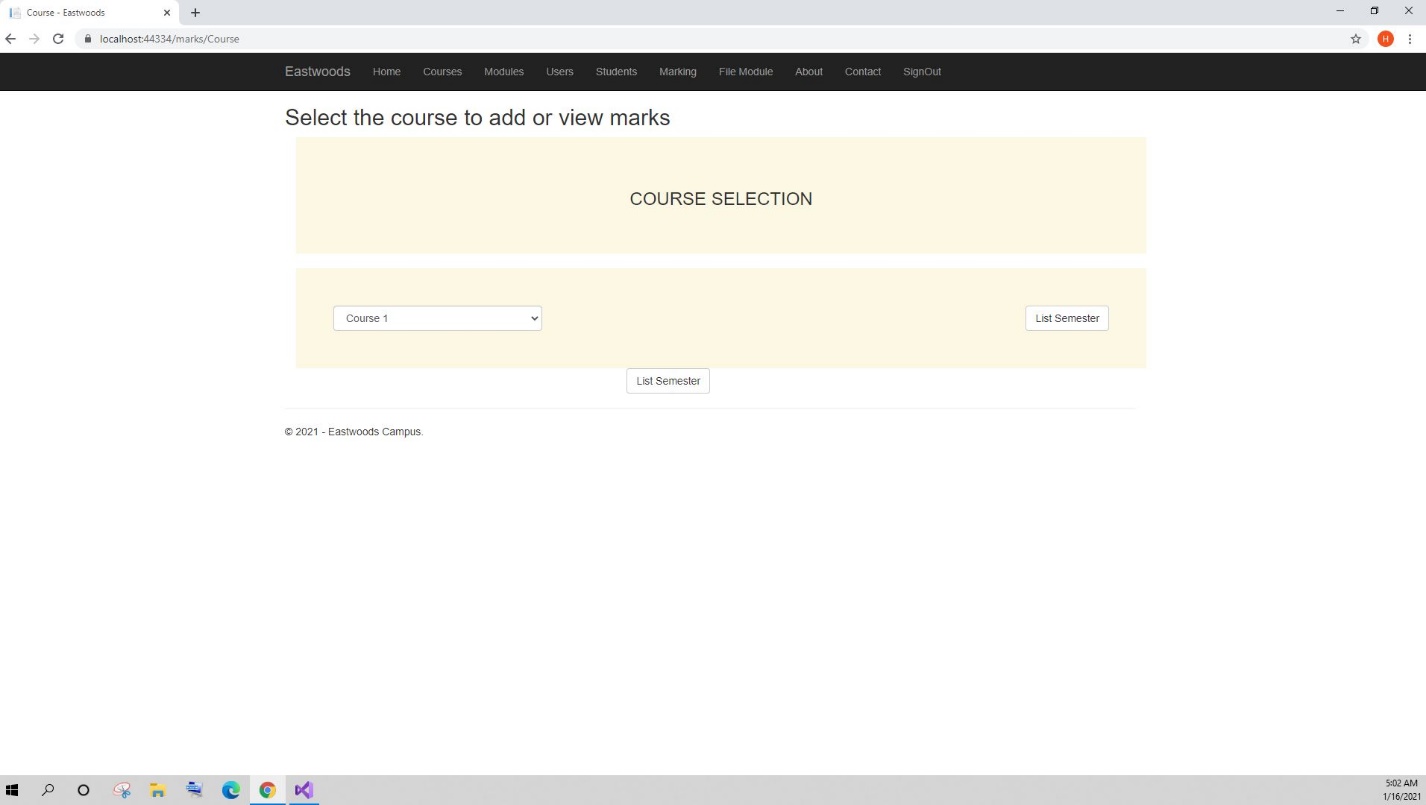
* Module details



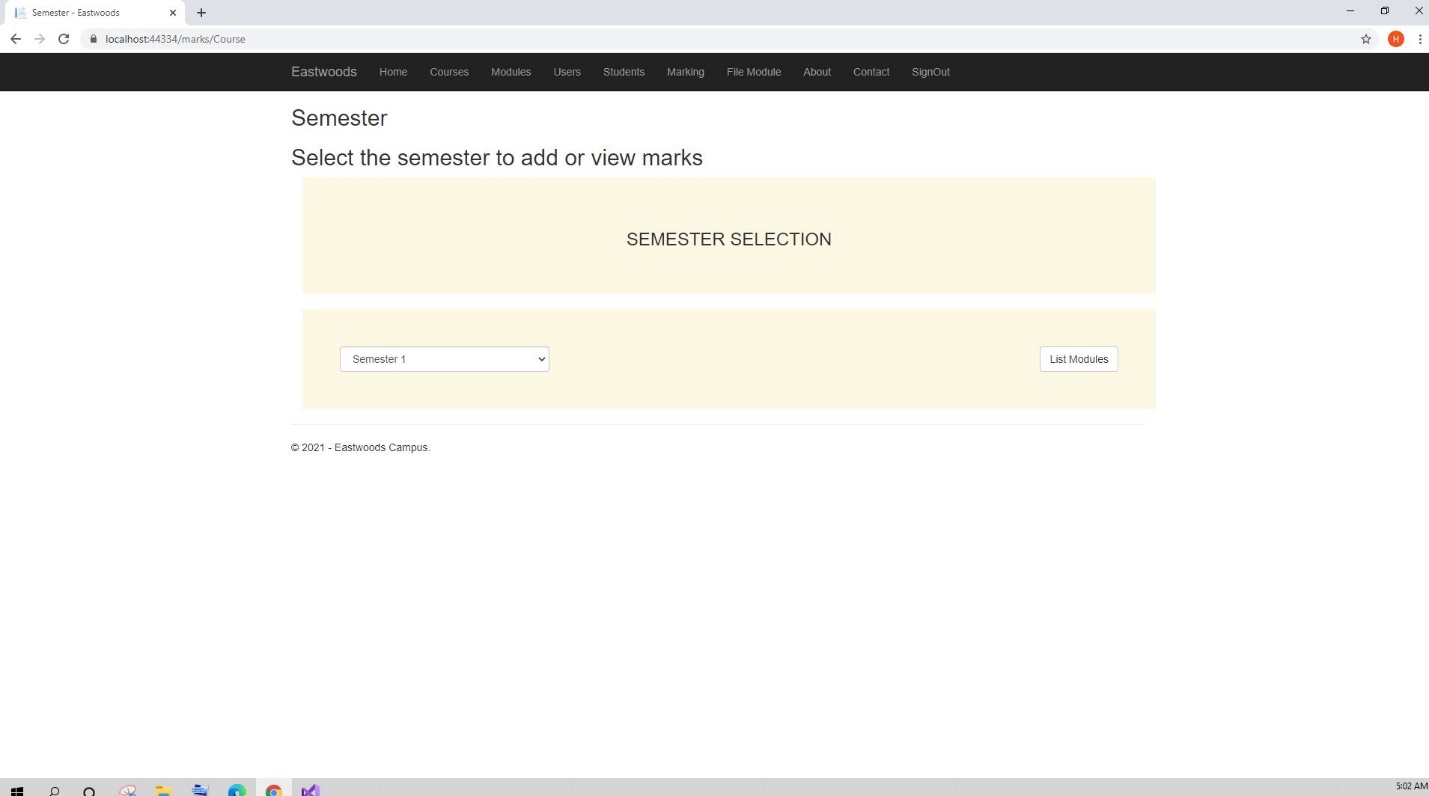
* User details



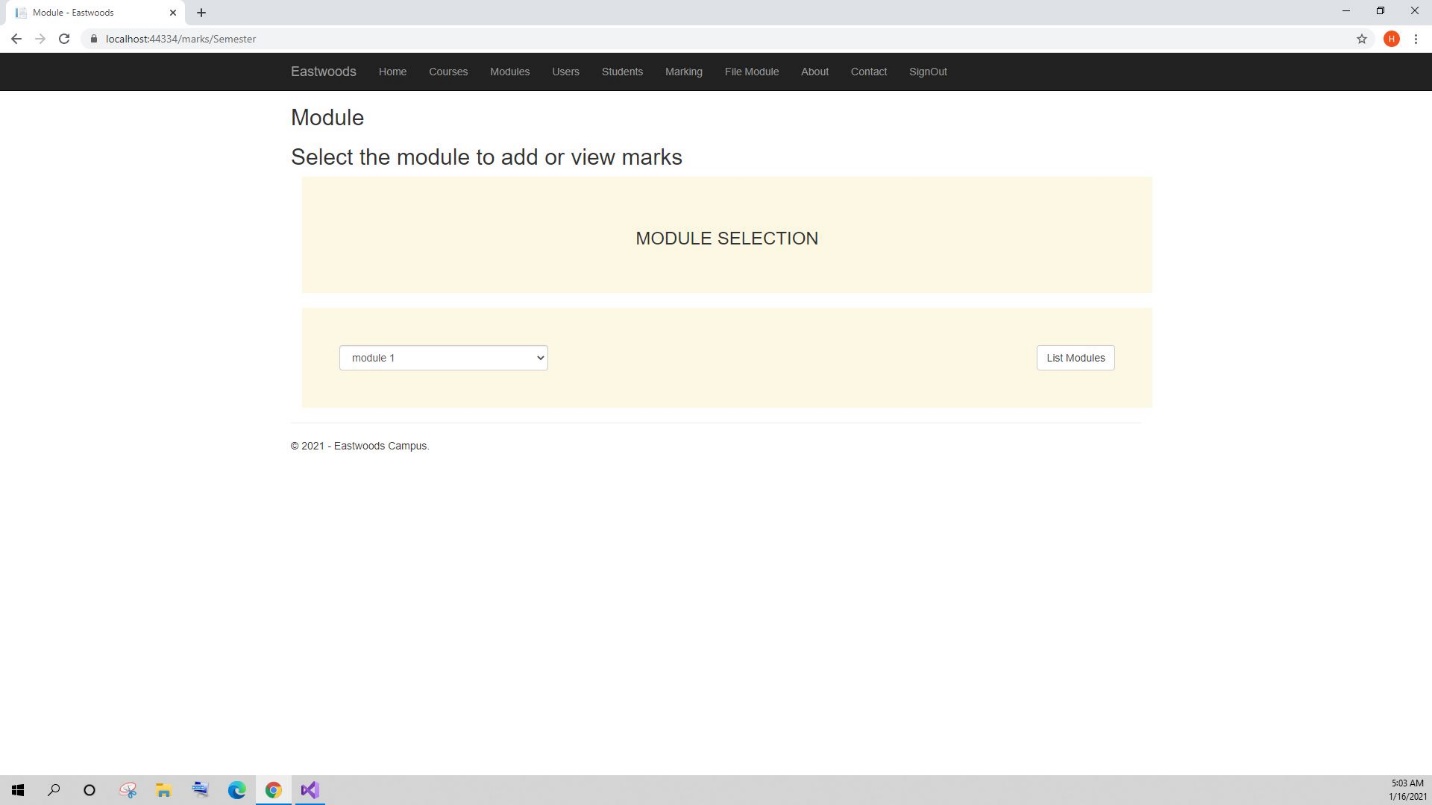
* Student details
* Add or edit mark view 1



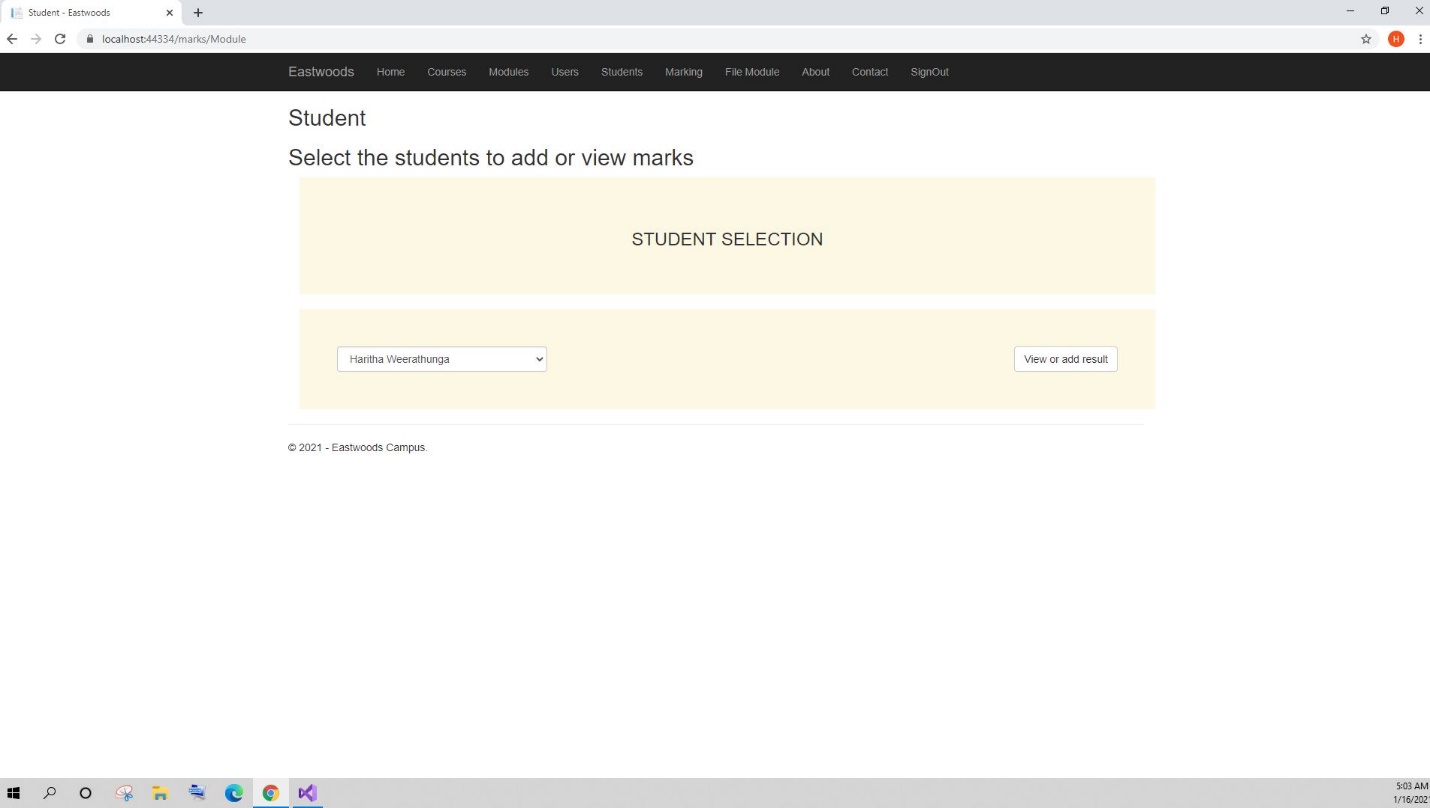
* Add or edit mark view 2



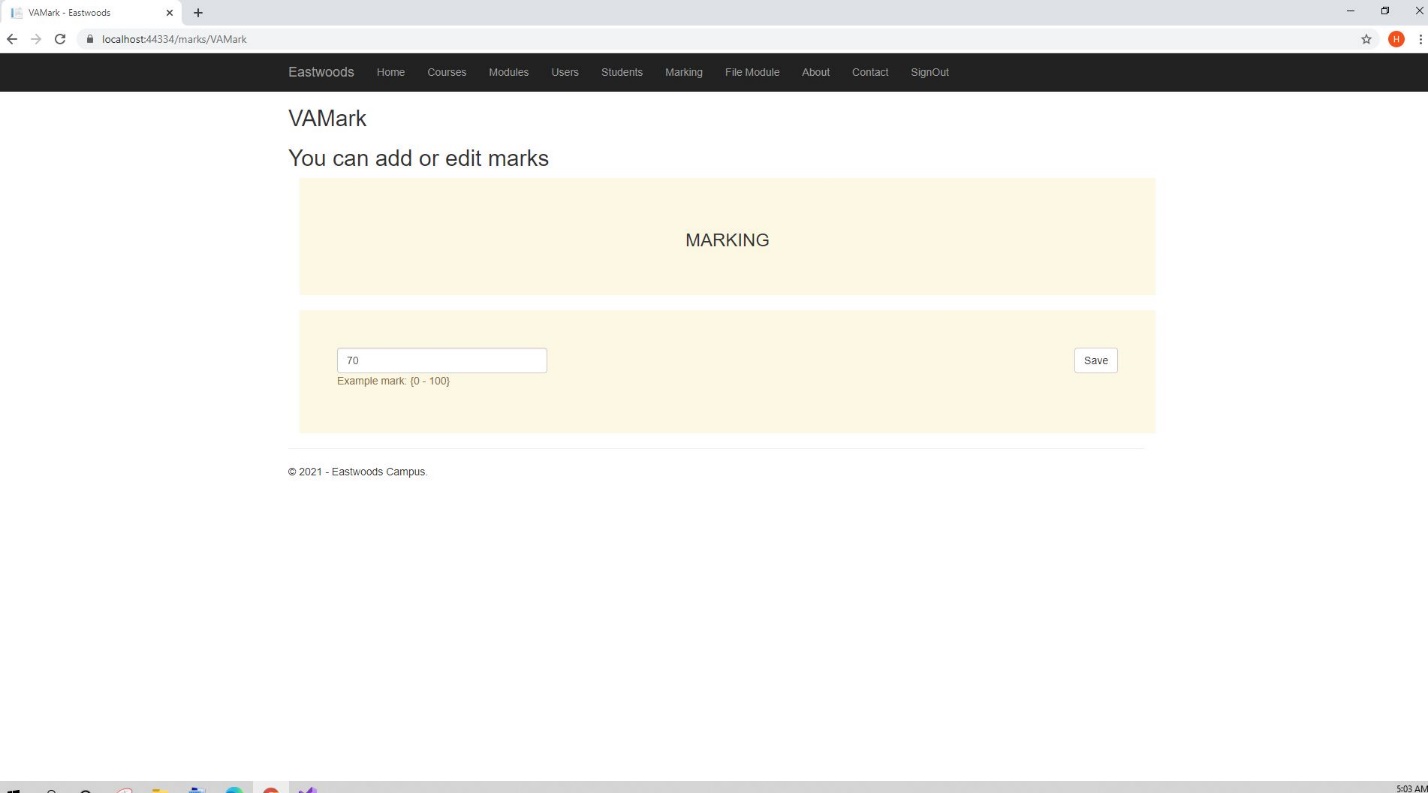
* Add or edit mark view 3



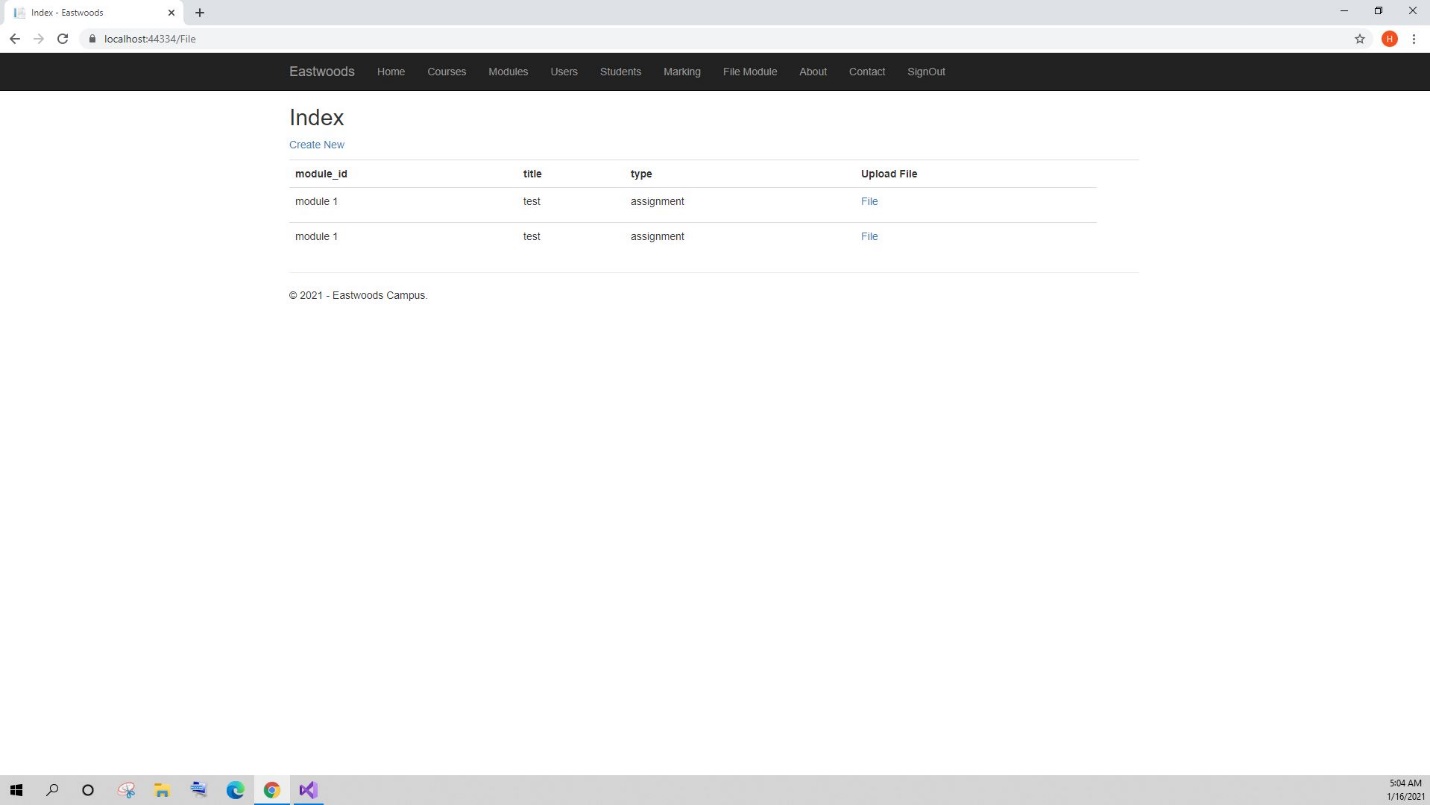
* Add or edit mark view 4



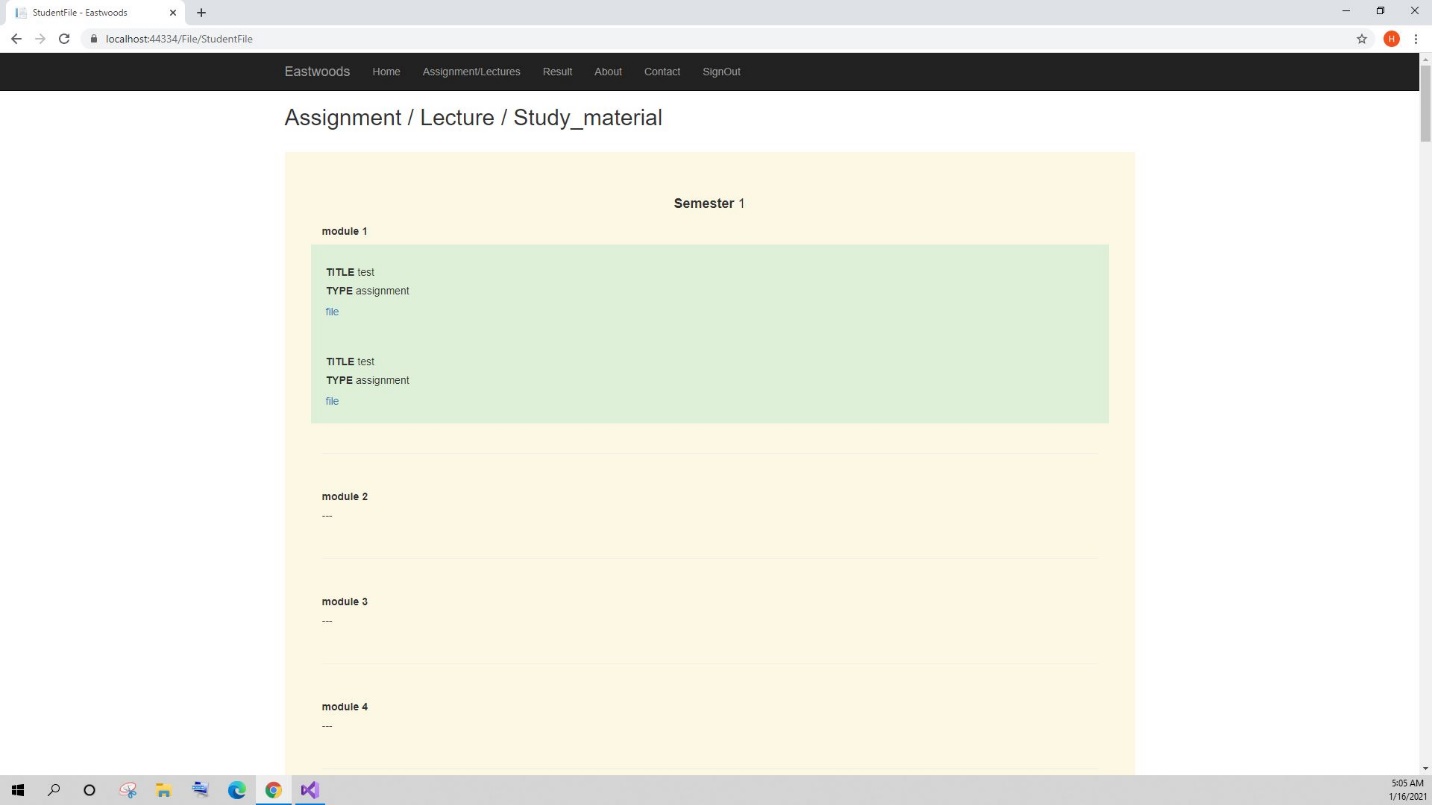
* Add or edit mark



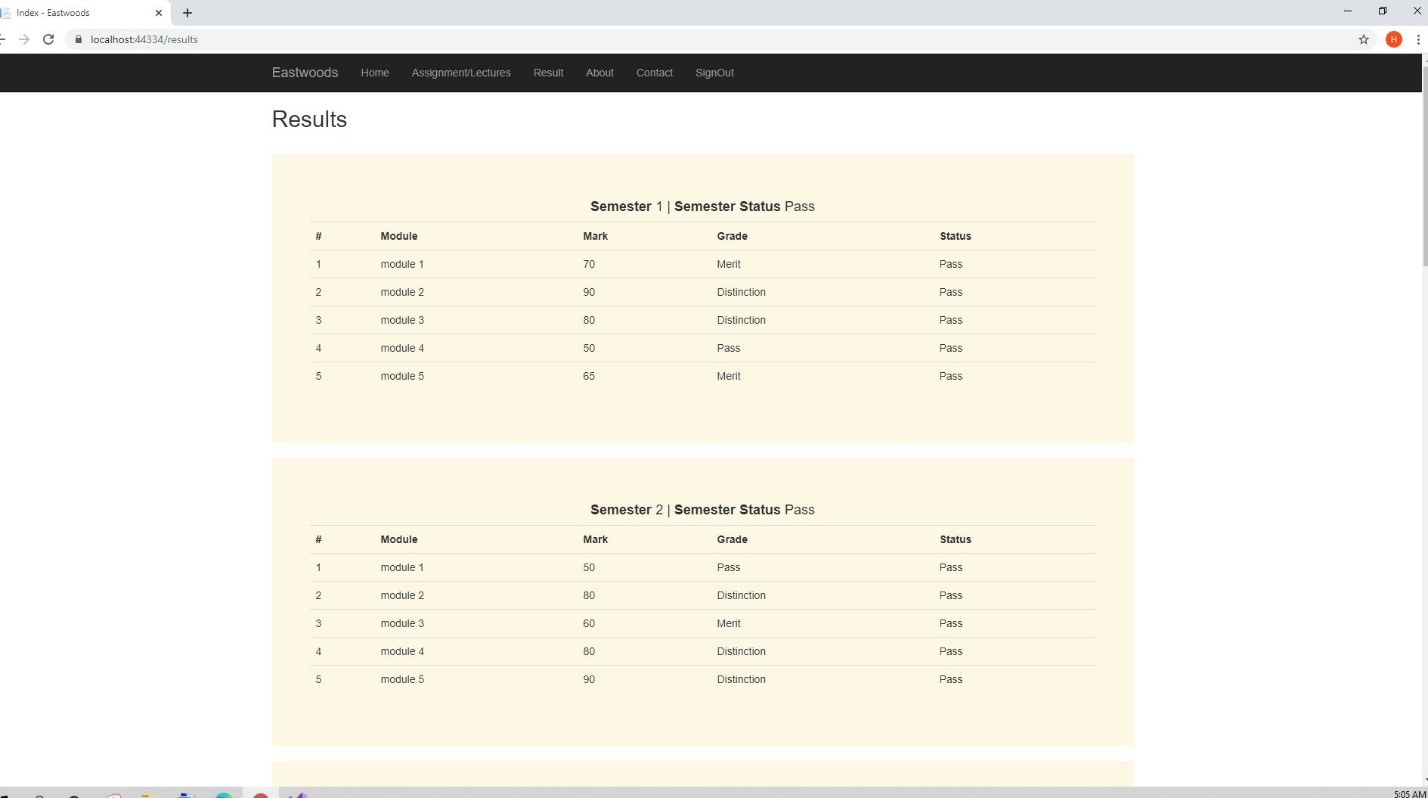
* Module list



* File list



* Grade list



# Installation Guide

* Prerequisite:
  + Visual studio 2019 installed.
* Unzip the folder.
* Open folder.
* Open .sln file and run the project.