**UE20CS352(OOAD with Java)**

**Self Learning Hands-on Assignment: MVC Framework**

|  |  |  |  |
| --- | --- | --- | --- |
| V Himadhith | PES1UG20CS478 | Section : H | Roll no : 45 |

**MVC Architecture Pattern:**

Model-View-Controller, or MVC, is a software architecture paradigm that divides an application into three interdependent parts. This design pattern is popular in web application development because it makes it easier to divide an application's concerns into discrete modules. The Controller functions as a mediator between the Model and View, handling user input and changing the Model and View appropriately. The Model represents the data and business logic of the application, the View represents the presentation layer.

**Advantages of MVC pattern:**

1. Separation of Concerns: Using the MVC pattern, an application is divided into various modules, each of which oversees a different concern. Developers can work separately on various components of the programme without interfering with each other's work because to this easier management of the codebase.
2. Testability: Writing unit tests for each application module is made simpler by the separation of concerns. This makes it possible for developers to find flaws early on and repair them.
3. Code Reusability: The MVC pattern encourages modular design, which makes it simpler to reuse code across various components of the programme. This shortens the development process and lessens the possibility that problems may be added because of redundant code.
4. Scalability: Because each module may be scaled individually, the MVC approach makes it simpler to scale an application. This enables the programme to accommodate growing user demand and traffic without compromising performance.

**Features of MVC Framework Chosen:**

1. Inversion of Control (IoC) Container: Spring provides an IoC container that manages the creation and lifecycle of objects in the application. This allows developers to focus on the business logic of the application rather than the mechanics of object creation and management.
2. Dependency Injection (DI): Spring supports DI, which allows objects to be loosely coupled and enables easy testing and maintenance of code. This means that objects can be easily replaced or modified without affecting other parts of the application.
3. Spring MVC: Spring includes its own implementation of the MVC architecture pattern, which provides a robust framework for building web applications. It includes features such as request mapping, view resolution, and form handling, as well as support for RESTful web services.
4. Aspect-Oriented Programming (AOP): Spring provides support for AOP, which enables cross-cutting concerns such as logging, security, and transaction management to be modularized and applied across multiple modules of the application.
5. Integration with other technologies: Spring provides integration with a wide range of other technologies, including Hibernate, JPA, Struts, and JSF. This allows developers to use the best tools for each part of the application, while still maintaining a consistent programming model.
6. Testing: Spring provides support for testing through its testing framework, which includes features such as JUnit integration, mock objects, and test context management.

**Problem definition with description of the chosen scenarios**

The Library Management System is a web application for managing a library built using Spring Boot, following the MVC (Model-View-Controller) design pattern, and uses mongoDB as its database. The application allows users to add and view books in the library's collection.

Users can view all books in the collection. They can also add new books to the collection.

The project aims to provide a user-friendly interface for managing a library collection, making it easier for librarians to keep track of books, and allowing patrons to search for and view books in the collection.

**Code:**

**Model class**

**Text

Description automatically generatedText

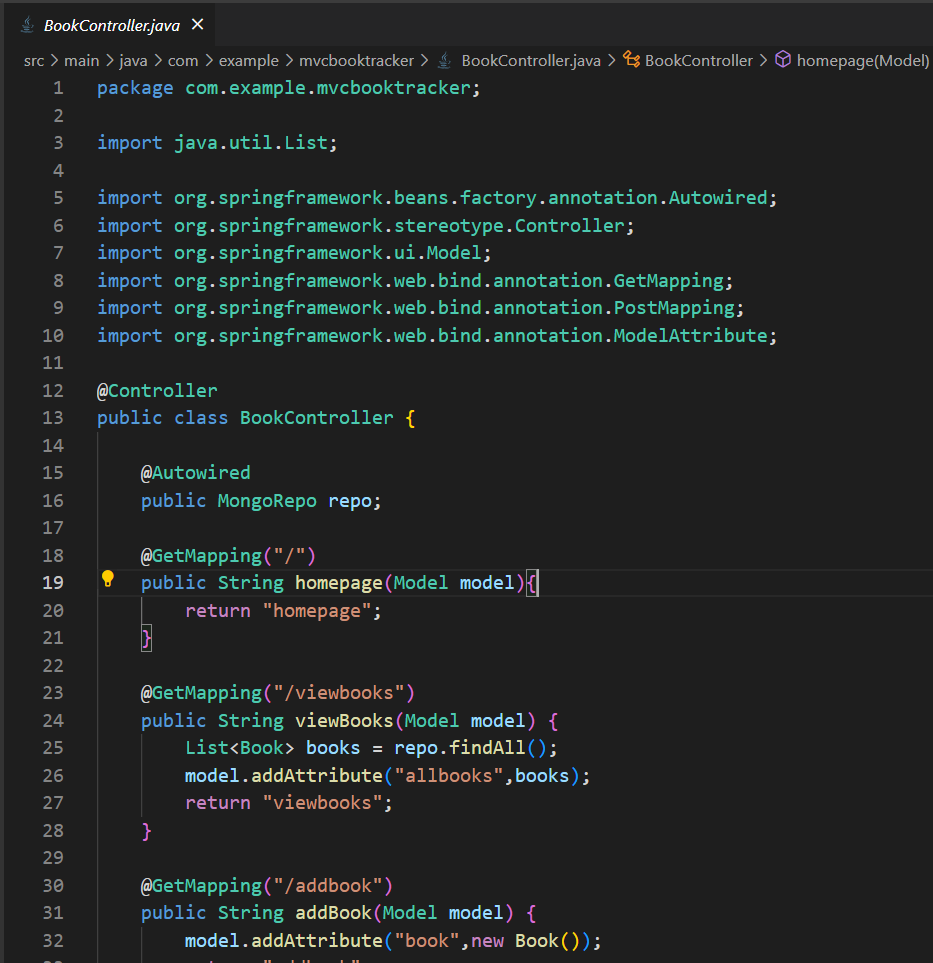
Description automatically generated**

**View class**

**Text

Description automatically generated**

**Controller class**

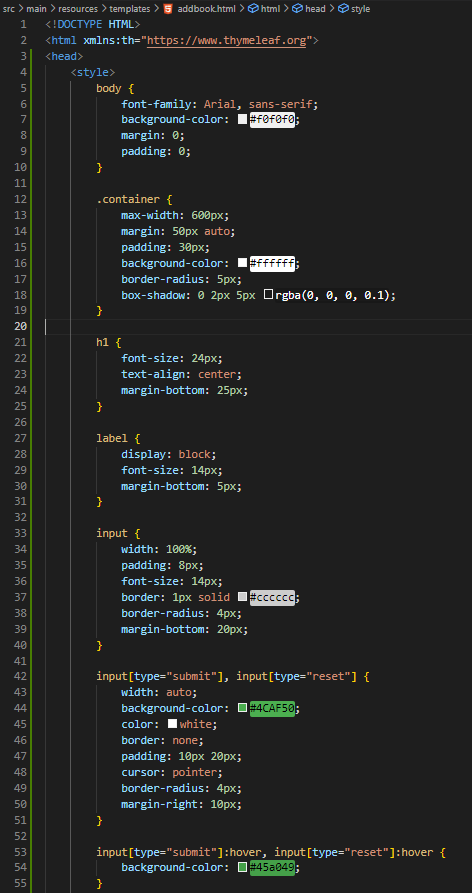
****

**Text

Description automatically generated**

**Text

Description automatically generatedAddpage.html**

****

**Text

Description automatically generated**

**homepage.html**

**adding css for all the files will make the file pretty big so I am skipping the css part.**

**Text

Description automatically generated**

**Viewbooks.html**

**Text

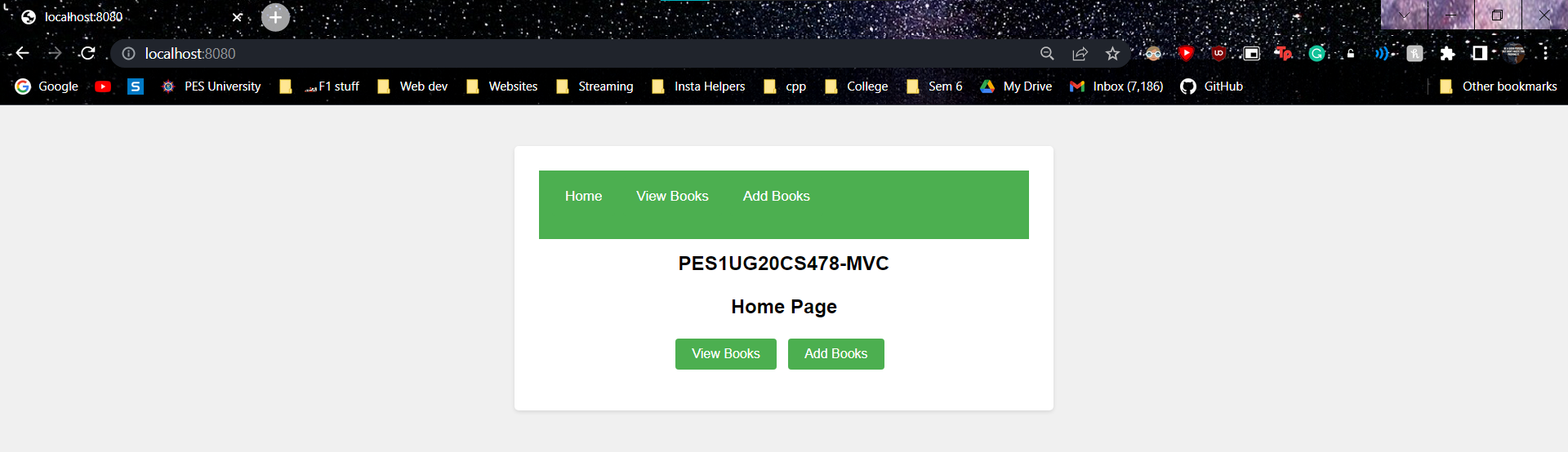
Description automatically generated**

**Console with application running:  
Graphical user interface

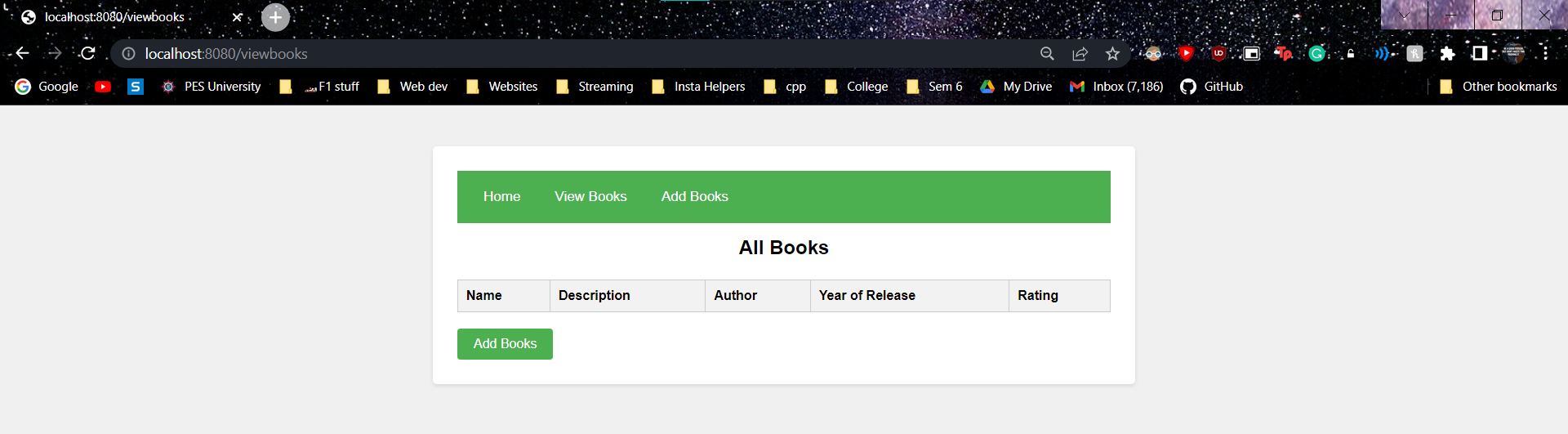
Description automatically generated with medium confidence**

**UIs related to the 2 scenarios:**

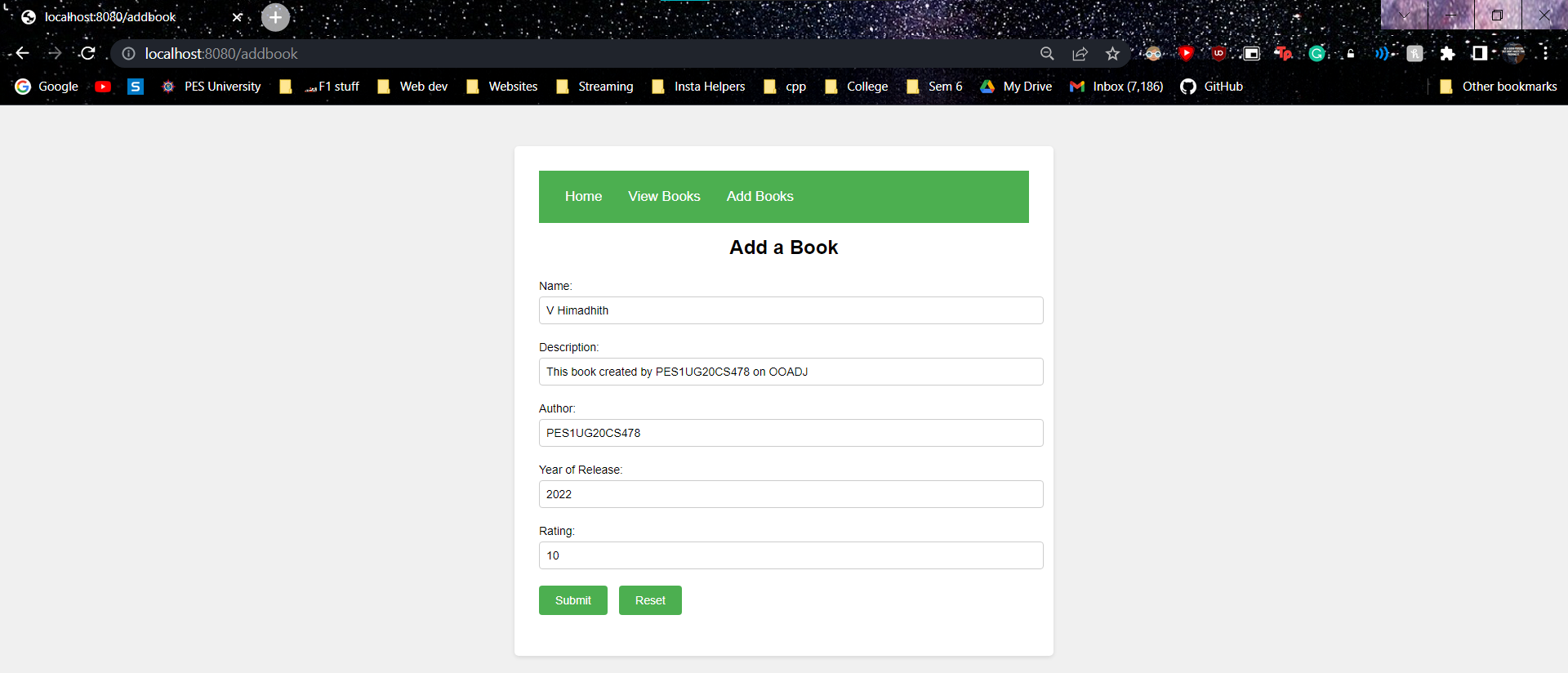
Homepage

****

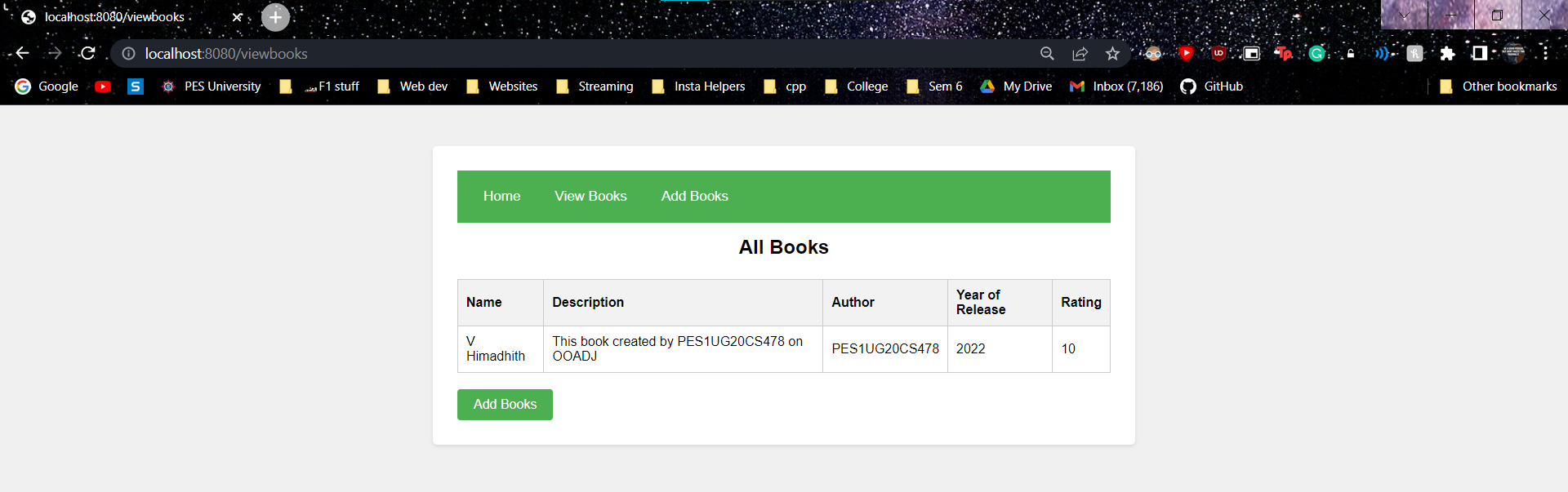
Before adding any books

****

The form to add books

****

Results after adding books

****

**Graphical user interface, application

Description automatically generated**

**MongoDB:**

**Graphical user interface, text, application, email

Description automatically generated**