Project Name: Book Library Management System

Group Name: VMDI

Team Members:

Member 1: [Viktoriia, Role: Team Leader. Project Development, Project Design, Documentation.]

Member 2: [Diana, Role: Database, Development, Project Testing.]

Member 3: [Ingvild, Role: The member did not participate in the project.]

Member 4: [Maina, Role: Rating Development.]

GitHub Repository Link: <https://github.com/MainaPaina/EBookLibrarySystem>

Project Overview

* Objective: To develop an ASP.NET Core MVC-based Book Library Management System with SQL integration for storing and managing ebooks.
* Description: A web application that allows users to preview, download, add, edit, and delete ebooks. It also provides an option to rate books and manage them efficiently(Optional to add rating).
* Deadline: March 14, 2025, Friday

Project Requirements

* Languages and Frameworks: C#, ASP.NET Core MVC, HTML, CSS, JavaScript(optional)
* Database: SQL (EBooksLibrary)
* Tools: Visual Studio, SQL Server Management Studio
* NuGet Packages: [List of installed packages like Microsoft.EntityFrameworkCore, Microsoft.AspNetCore.Mvc, etc.]

Technology Stack

* Backend: C#, ASP.NET Core MVC
* Database: SQL Server or SqlLite
* Version Control: Git & GitHub
* Tools: Visual Studio 2022, SQL Server Management Studio

Database Design

* Database Name: EBooksLibrary
* Table Name: FreeEBooks
* Schema:

| Column Name | Data Type | Description |
| --- | --- | --- |
| BookID | UniqueIdentifier | GUID for security |
| BookTitle | varchar(255) | Title of the book |
| Genre | varchar(100) | Genre of the book |
| Image | varbinary(100) | Book cover image |
| Author | varchar(MAX) | Author of the book |
| Rating (Optional) | int | Book cover image |

Controllers, Models and Views:

What files and folders did you create and what is their purpose?

Path:

* Controllers -> BooksController.cs  
   Description: Handles operations related to books, such as adding, editing, and listing books.
* Controllers -> HomeController.cs  
   Description: Manages the home page and initial navigation of the application.
* Controllers -> RatingController.cs  
   Description: Handles the rating functionality, allowing users to rate books.
* Data -> ApplicationDBContext.cs  
   Description: The database context class, managing the connection to the database and the entities involved.
* Migrations -> 20250313140849\_test.Designer.cs  
   Description: Defines the schema for the migration process.
* Migrations -> 20250313140849\_test.cs  
   Description: Contains the migration logic for database schema changes.
* Migrations -> 20250313221639\_AddedRatingToBook.Designer.cs  
   Description: Defines the schema for adding a rating to the books.
* Migrations -> 20250313221639\_AddedRatingToBook.cs  
   Description: Contains the migration logic for adding the rating feature.
* Migrations -> 20250313232756\_AddedRatingTable.Designer.cs  
   Description: Defines the schema for the new rating table.
* Migrations -> 20250313232756\_AddedRatingTable.cs  
   Description: Contains the migration logic for the new rating table.
* Models -> Entities -> AddBookViewModel.cs  
   Description: Contains the model used to represent the data when adding a new book.
* Models -> Entities -> AddRatingModel.cs  
   Description: Represents the model used for adding ratings to books.
* Models -> Entities -> ErrorViewModel.cs  
   Description: Used to display error messages when something goes wrong during execution.
* Views -> Books -> Add.cshtml  
   Description: View for adding a new book to the library.
* Views -> Books -> Edit.cshtml  
   Description: View for editing book details.
* Views -> Books -> List.cshtml  
   Description: Displays a list of all books available in the library.
* Views -> Home -> Index.cshtml  
   Description: Home page of the application, typically shows initial navigation options.
* Views -> Home -> Privacy.cshtml  
   Description: Displays the privacy policy of the application.
* Views -> Rating -> AddRating.cshtml  
   Description: Allows users to rate books.
* Views -> Shared -> Error.cshtml  
   Description: Displays error messages.
* Views -> Shared -> \_Layout.cshtml  
   Description: Contains the layout and structure of the main application pages.
* Views -> Shared -> \_Layout.cshtml.css  
   Description: Contains CSS styles for the layout of the application.
* Views -> Shared -> \_ValidationScriptsPartial.cshtml  
   Description: Contains validation scripts that are used across different views.
* Views -> Shared -> \_ViewImports.cshtml  
   Description: Imports common namespaces and files for all views.
* Views -> Shared -> \_ViewStart.cshtml  
   Description: Defines the layout file and configurations that apply globally for views.
* wwwroot -> css  
   Description: Stores the CSS files that are used for styling the application.
* wwwroot -> images  
   Description: Contains images used in the application, such as logos, book covers, etc.
* wwwroot -> js  
   Description: Contains JavaScript files used for adding interactivity to the application.
* wwwroot -> lib  
   Description: Stores third-party libraries or frameworks used in the application (e.g., jQuery, Bootstrap).
* wwwroot -> favicon.ico  
   Description: The favicon icon used in the browser tab for the application.
* appsettings.Development.json  
   Description: Contains development environment settings for the application.
* appsettings.json  
   Description: Contains application-wide settings, such as database connection strings and configurations.

Challenges Faced

* Handling Errors: Error handling was complex as we had to manage different exceptions for database operations and user input.

Bug Fixes:

Issue: Model data was not being passed correctly between the controller and the view, leading to incorrect data rendering.  
 Solution: I rectified this by ensuring that the correct model was passed from the controller to the view and used the @model directive correctly in the view files.

| Date | Hours Worked | Team Member | Task Description |
| --- | --- | --- | --- |
| Mar 5, 2025 | 0 hrs | Group discussion | The project discussion time cannot be considered final, as the appointed team leader was not handling their task properly and added a new member to the group, which was unnecessary. |
| Mar 5, 2025 | 8 hr | Diana | Database creation, migration, testing. |
| Mar 6, 2025 | 2 hrs | Maina | Creation of rating (optional). I want to acknowledge the excellent work! |
| Date | 8 hrs | Viktoriia | Development, creation, design. |
| Date | 0 hrs | Ingvild | Did not participate in the project. |

Learning Outcomes

* Backend Development: Improved C# and SQL skills, explored the possibility of connecting SQLLite for this project or an API. It was not successful, but it will be developed in the future.
* Frontend Development: Gained a better understanding of Bootstrap classes and had fun experimenting with them.
* Project Management: Improved collaboration and task management skills.
* Database Management: Gained hands-on experience in creating, managing, and migrating databases using SQL Server.
* Error Handling: Learned how to handle errors effectively in both the frontend and backend of the application.
* Version Control: Improved proficiency in using Git and GitHub for version control, ensuring the project's code was well-managed and updated.
* UI/UX Design: Gained practical experience in designing user-friendly interfaces and optimizing the user experience.
* Teamwork: Enhanced communication and collaboration skills while working with team members on different tasks.

Future Enhancements

* Advanced Search Filters: By date, author, rating.
* Admin Panel: For managing books and users.
* User Authentication: Login system for users.

Conclusion

The project was a valuable learning experience, allowing the team to develop both technical and collaborative skills.

What worked well:

* The development process was smooth, especially in terms of backend and database management. The team was able to implement core features like book addition, editing, and deleting, as well as the optional rating system.
* Collaboration and communication within the team were effective, which helped in completing tasks on time.
* The project was successfully structured, with clear roles assigned to each member, leading to efficient task management.

Areas for improvement:

* There were challenges with integrating the rating system and some issues with database migrations that caused delays in the early stages of development. These could be avoided in future projects by refining the initial setup and planning better.
* The user interface could have been more polished. Although it functions, additional design improvements could make it more user-friendly.
* More thorough testing could have been done during development to catch bugs earlier and ensure smoother functionality.

Overall, the project was a success, but there are clear opportunities to enhance both the technical and design aspects in future iterations.