Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.
Digital Object Identifier 10.1109/ACCESS.2017.DOI

Agile Development of a Scalable Library Management System Using Scrum Methodology

ÖMER TEKIN YAVUZ[1], (Project Manager), ARIF EREN AYDIN[2], (Backend Developer), GÖKÇE KESKIN[3], (Database Developer), KADIR YILDIZ[4], (Backend Developer), MUHSIN AY[5], (Backend Developer), FEVZI GÜLER[6], (Frontend Developer), and YUNUS EMRE KARATAS[7], (FullStack Developer)
[1]Project Manager, GitHub: https://github.com/ÖmerYavuz

ABSTRACT The Library Site project aims to develop a comprehensive web-based system designed to streamline the operations of libraries, particularly those serving universities and public communities. The platform will offer functionalities such as catalog management, user account management, search and reservation features, and notification services. The goal is to deliver an intuitive, efficient, and scalable solution that enhances the overall library experience for both users and administrators.

INDEX TERMS Agile Development, Scrum, Library Management System, Scalability, Software Engineering

I. INTRODUCTION

LIBRARIES play a vital role in fostering education and community engagement. However, traditional library systems often face challenges related to scalability, user experience, and operational inefficiencies. To address these issues, the Library Site project proposes a modern, web-based library management system developed using agile methodologies, specifically Scrum.

The project aims to deliver a user-friendly platform that incorporates core functionalities such as catalog management, user account handling, search and reservation features, and notification services. By leveraging state-of-the-art technologies and an iterative development approach, the Library Site project ensures technical, operational, and financial feasibility while aligning with stakeholder requirements.

II. TECHNICAL FEASIBILITY

The technical feasibility of the Library Site project evaluates the tools, technologies, and infrastructure required for successful implementation.

A. FRONTEND TECHNOLOGY

The frontend will be developed using HTML, CSS, ASP.NET MVC, Razor, Bootstrap, and Tailwind CSS to ensure a responsive and interactive user interface. Additional tools, such as the GSAP (GreenSock) library, SASS/SCSS preprocessor, FontAwesome, and HeroIcons, will be integrated to enhance interactivity, animations, and design customization. These technologies will enable users to browse the catalog, search for books, and manage their accounts seamlessly.

B. BACKEND TECHNOLOGY

The backend will be powered by C#, utilizing the ASP.NET MVC framework to provide robust API endpoints for handling user requests, managing the catalog, and processing reservations. This framework ensures scalability, security, and seamless integration with the frontend.

C. DATABASE

MsSQL will be employed to manage the relational database, offering secure and organized storage of user accounts, book details, and reservation records.

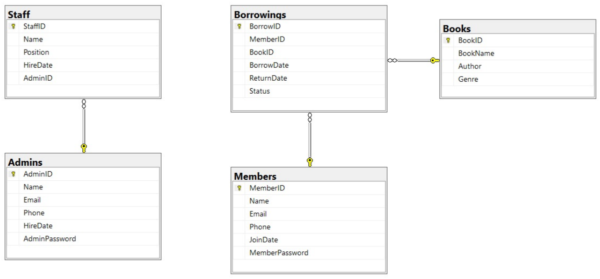
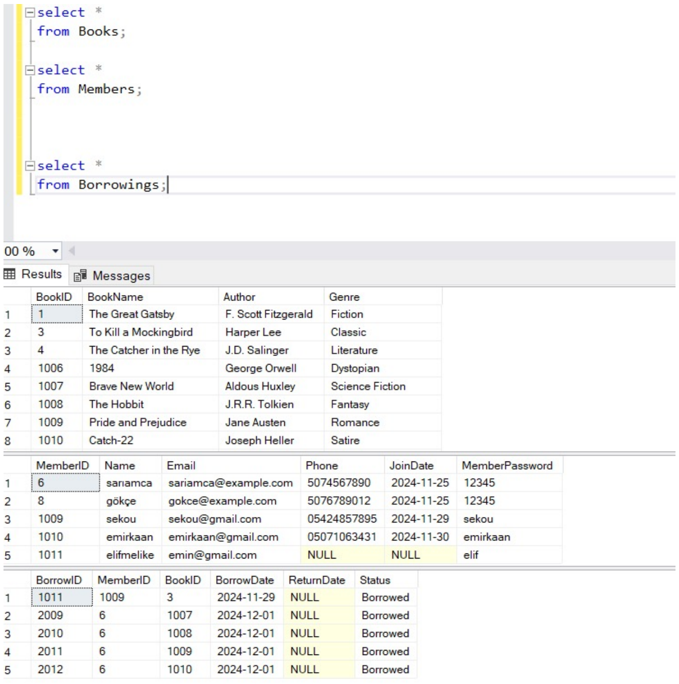


FIGURE 1. "Database schema showing the relational structure of the Library Site system, including tables for Staff, Admins, Members, Borrowings, and Books, with their respective attributes and relationships."



```
select *
from Books;

select *
from Members;


select *
from Borrowings;
```

	BookID	BookName	Author	Genre
1	1	The Great Gatsby	F. Scott Fitzgerald	Fiction
2	3	To Kill a Mockingbird	Harper Lee	Classic
3	4	The Catcher in the Rye	J.D. Salinger	Literature
4	1006	1984	George Orwell	Dystopian
5	1007	Brave New World	Aldous Huxley	Science Fiction
6	1008	The Hobbit	J.R.R. Tolkien	Fantasy
7	1009	Pride and Prejudice	Jane Austen	Romance
8	1010	Catch-22	Joseph Heller	Satire

	MemberID	Name	Email	Phone	JoinDate	MemberPassword
1	6	sarıamca	sariamca@example.com	5074567890	2024-11-25	12345
2	8	gökçe	gokce@example.com	5076789012	2024-11-25	12345
3	1009	sekou	sekou@gmail.com	05424857895	2024-11-29	sekou
4	1010	emirkaan	emirkaan@gmail.com	05071063431	2024-11-30	emirkaan
5	1011	elifmelike	emin@gmail.com	NULL	NULL	elif

	BorrowID	MemberID	BookID	BorrowDate	ReturnDate	Status
1	1011	1009	3	2024-11-29	NULL	Borrowed
2	2009	6	1007	2024-12-01	NULL	Borrowed
3	2010	6	1008	2024-12-01	NULL	Borrowed
4	2011	6	1009	2024-12-01	NULL	Borrowed
5	2012	6	1010	2024-12-01	NULL	Borrowed

FIGURE 2. "Snapshot of the working database with queries and results demonstrating the populated tables for Books, Members, and Borrowings, showcasing the real-time data integration and functionality."

D. DEPLOYMENT AND SCALABILITY

The system will be deployed on cloud platforms such as AWS or Azure to ensure scalability and availability. This infrastructure will allow the Library Site to handle an increasing number of users without compromising performance. With readily available expertise in these technologies and comprehensive documentation, the technical implementation is both feasible and manageable.

III. OPERATIONAL FEASIBILITY

Operational feasibility assesses the ease of integrating the Library Site into existing library workflows and its usability for end-users.

A. TARGET USERS

The system will cater to two primary user groups:

- **Patrons:** Users who will search the catalog, reserve books, and manage their accounts.
- **Administrators:** Staff who will manage inventory, track reservations, and generate reports.

B. USER INTERFACE DESIGN

"The platform prioritizes a user-friendly interface to ensure smooth navigation for users and admins of varying technical proficiency levels. Here is an example of the login process for the Library Site. This example demonstrates the completed integration of both backend and frontend components for the login functionality."

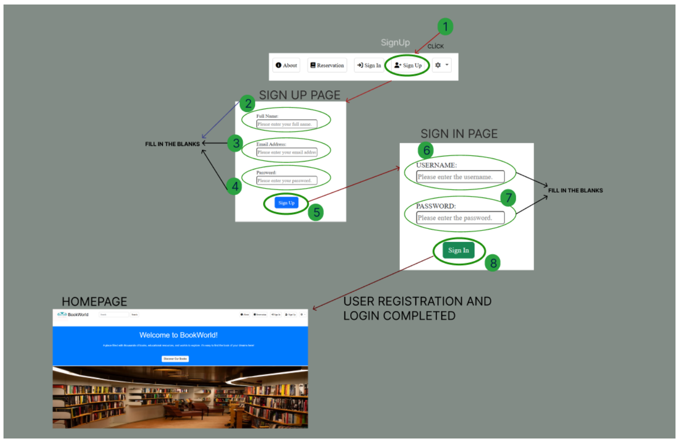
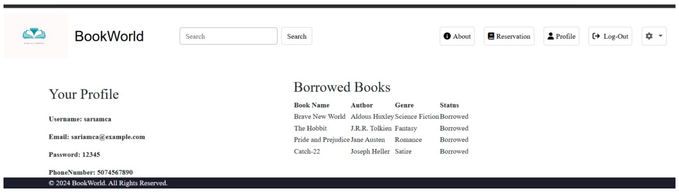


FIGURE 3. Step-by-Step Guide for Logging into the Library Site: A Frontend User Interface Walkthrough



Your Profile

Username: sarıamca

Email: sariamca@example.com

Password: 12345

PhoneNumber: 5074567890

Borrowed Books

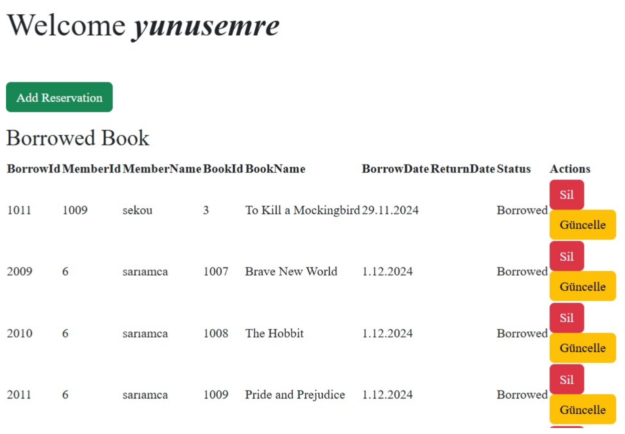
Book Name	Author	Genre	Status
Brave New World	Aldous Huxley	Science Fiction	Borrowed
The Hobbit	J.R.R. Tolkien	Fantasy	Borrowed
Pride and Prejudice	Jane Austen	Romance	Borrowed
Catch-22	Joseph Heller	Satire	Borrowed

© 2024 BookWorld. All Rights Reserved.

FIGURE 4. "User interface for "BookWorld" showcasing the profile page and a list of borrowed books, complete with book details, search functionality, and navigation options for seamless interaction."

C. LIBRARY STAFF ADAPTATION

Comprehensive documentation and training resources will be provided to library staff to facilitate seamless adoption of the system with minimal disruption to daily operations.



Welcome ***yunusemre***

Borrowed Book

BorrowId	MemberId	MemberName	BookId	BookName	BorrowDate	ReturnDate	Status	Actions
1011	1009	sekou	3	To Kill a Mockingbird	29.11.2024		Borrowed	Sil Güncelle
2009	6	sarıamca	1007	Brave New World	1.12.2024		Borrowed	Sil Güncelle
2010	6	sarıamca	1008	The Hobbit	1.12.2024		Borrowed	Sil Güncelle
2011	6	sarıamca	1009	Pride and Prejudice	1.12.2024		Borrowed	Sil Güncelle

FIGURE 5. "Admin Panel View: This interface demonstrates the administrative functionalities for managing borrowed books. Administrators can add reservations, update book details, and delete records efficiently, ensuring seamless library operations.

D. WORKFLOW INTEGRATION

The system will integrate with existing library systems for book inventory and notifications, enabling a seamless transition and operational consistency.

IV. FINANCIAL FEASIBILITY

Financial feasibility evaluates the project's cost implications, ensuring alignment with budgetary constraints.

A. DEVELOPMENT COSTS

The project team will include frontend and backend developers, UI/UX designers, and database administrators. Estimated costs for salaries, licenses, and tools are projected to remain within a reasonable range.

B. MAINTENANCE AND UPDATES

Post-deployment, routine maintenance and periodic updates will be required. These activities have been planned and budgeted to ensure long-term functionality and reliability.

C. COST JUSTIFICATION

The overall cost is justified by the long-term benefits of an efficient, modern library system. Funding options such as grants or sponsorships may further support the project.

V. CONCLUSION

The Library Site project is technically, operationally, and financially feasible. Its core functionalities, including catalog management and reservations, align with the project's scope. The use of modern technologies like ASP.NET MVC, C#, and MsSQL ensures technical viability, while user-centric design enhances usability. Financially, the project remains within budget, with predictable and manageable costs.

The next steps involve detailed requirements gathering, system design, and initiating development. Following an iterative approach with Scrum methodology will ensure timely delivery and continuous feedback incorporation. This project is poised to deliver significant value to libraries by enhancing user satisfaction and operational efficiency.

VI. SPRINT PLAN

The Library Site project followed an iterative Scrum-based approach with nine sprints, each focusing on specific milestones. Tables 1 and 2 summarize the duration, key activities, and deliverables of each sprint.

TABLE 1. Sprint Key Activities

Sprint	Duration	Key Activities
1	Week 1	Feasibility study and project planning
2	Week 2	Requirements gathering, initial system design
3	Week 3	Frontend development setup, basic UI components,migrated HTML/CSS to ASP.NET, transitioned to embedded processes,database schema creation,User Login,Admin Panel,Book Management Module development, including CRUD operations
4	Week 4	Backend setup,Functional catalog integration and admin panel backend features, improved version of fronted for admin panel
5	Week 5	Improved frontend-backend integration for completed tasks and search function
6	Week 6	Reservation system implementation
7	Week 7	Notification system development
8	Week 8	User feedback collection and testing
9	Week 9	Bug fixes, optimization, and deployment

VII. SPRINT DETAILS

A. SPRINT 1: WEEK 1

During the first week of the project, the focus was on conducting a feasibility study and creating a detailed project plan.

- **Activities:**
  - Conducted a comprehensive **feasibility study** to evaluate the technical, operational, and financial aspects of the Library Site project.
  - Identified key risks, including potential challenges in scaling the system and ensuring user adoption.
  - Collaborated with stakeholders to define high-level objectives for the system.
  - Developed a project timeline outlining all major milestones and deliverables.
- **Deliverables:**
  - A detailed **feasibility report** documenting the study's findings, risks, and recommendations.
  - A comprehensive **project timeline**, highlighting sprint-level goals and deadlines.
  - A **study plan** defining the team's learning objectives and initial research areas.

B. SPRINT 2: WEEK 2

In the second week, the focus shifted to gathering requirements and initiating the system's design.

TABLE 2. Sprint Deliverables

Sprint	Duration	Deliverables
1	Week 1	Feasibility report, project timeline, and study plan
2	Week 2	Requirements document, initial UML diagrams
3	Week 3	Developed the homepage prototype, including design and layout. Migrated HTML and CSS files into the ASP.NET environment for compatibility and backend integration. Implemented user login functionality, including sign-in, sign-up, and session management. Finalized the database schema, ensuring stable connections and addressing LAN/IP-related issues. Transitioned to an embedded process approach, replacing API architecture with direct backend integration. Conducted tests for routing systems, form submissions, and CRUD operations to validate frontend-backend interactions. Established admin panel functionalities, including user and content management workflows
4	Week 4	Functional catalog integration and further improvements on admin panel backend features.
5	Week 5	Functional search integration and making reservation development
6	Week 6	Reservation feature prototype
7	Week 7	In-app notification features
8	Week 8	User feedback report, bug list
9	Week 9	Final deployed system, project documentation

- **Activities:**
  - Conducted **stakeholder interviews** with library patrons and administrators to understand their needs and challenges.
  - Organized **brainstorming workshops** to prioritize the features and functionalities required for the system.
  - Created initial **UML diagrams** to visualize system components and workflows.
  - Documented both functional and non-functional requirements to guide the design and implementation phases.
- **Deliverables:**
  - A **requirements document** containing all gathered and prioritized system requirements.
  - Initial **UML diagrams**, including use case and system context diagrams, to provide a conceptual understanding of the system.

C. SPRINT 3: WEEK 3

In the third week, the focus was on integrating the frontend and backend systems using embedded processes. Significant progress was made in ensuring seamless functionality, implementing routing and controller logic, completing user login, developing the book management module, and establishing the database connection.

- **Activities:**
  - Migrated **HTML and CSS files** into the **ASP.NET environment**, ensuring compatibility with the project structure and preparing for backend integration.
  - **Layout and Master Page Integration:** Created the `Layout.cshtml` file as a reusable master page template. Linked CSS and JS files for a consistent frontend design across all pages.
  - Developed **routing systems** to enable smooth page navigation.
  - Transitioned to an **embedded process approach**, integrating backend logic directly within the project rather than using APIs.
  - Implemented and tested **embedded controllers** such as the `BooksController`, responsible for CRUD operations, book management, and user functionalities.
  - **Routing and Controller Logic:** Finalized routing configurations and implemented controllers for handling form data and backend processing.
  - Integrated **user login functionality**, including sign-in, sign-up, and session management. Session handling and JWT authentication were implemented where applicable.
  - **Backend and Frontend Integration:** Completed model binding for form data, ensuring smooth interaction between the frontend and backend.
  - Developed initial **Book Management Module**, including functions like `GetBooks`, and `DeleteBook`.
  - Worked on **Book Categories and CRUD Functions:** Initiated development for book categories and CRUD operations. Created the `BooksController` with functions such as `GetBooks`, `GetBook`, `DeleteBook`, `PutBook`, `PostBook`, `GetBooks ByCategory`, and `GetAllCategories`. These functions were tested locally using Swagger, and no errors were found after resolving initial `Code:500` issues caused by missing database objects and misconfigured `appsettings.json`.
  - **Database Connectivity:** Established stable connections to the SQL Server database, addressing LAN and IP-related issues. The database schema was refined and integrated with backend controllers.
  - **Admin Panel Development:** Worked on backend functionalities for the admin panel, including content creation, editing, and deletion. Verified user and admin workflows across pages.
  - **GitHub Repository Management:** Regularly updated and managed the repository, ensuring all team members had access to the latest codebase and documentation. Repository adjustments were made

for better collaboration.
    - Ensured **responsive design** and optimized the frontend for mobile compatibility by adjusting layouts and testing across devices.
    - Conducted **functional tests**, validating all features, including routing, form submissions, session management, and book management operations.
- **Challenges:**
    - Encountered **database connectivity issues** related to IP restrictions and LAN configurations. These were resolved through debugging and system adjustments.
    - Resolved **Code:500 errors** in Swagger during testing, caused by missing database objects in `program.cs` and misconfigured default connection strings in `appsettings.json`.
    - Adapting to the decision to shift from **API architecture** to embedded processes required reworking some components but ultimately streamlined development.
    - Responsive design posed **alignment and layout challenges**, which were addressed through iterative testing and adjustments.
    - **Model binding errors** during form processing occasionally interrupted workflow, but these were resolved by enhancing data validation and testing procedures.
- **Key Deliverables:**
    - Completed homepage prototype with integrated routing and master page functionality.
    - Developed user login functionality, session management, and basic CRUD operations for book management.
    - Established backend functionality for the admin panel and integrated with the frontend.
    - Finalized responsive frontend design and ensured compatibility across devices.
    - Updated GitHub repository with clean code and proper documentation.
- **Outcomes:**
    - Seamless integration of frontend and backend components using embedded processes, resulting in a robust system architecture.
    - Functional user login system with session management that enhances security and user experience.
    - A fully operational book management module with CRUD capabilities, setting a strong foundation for catalog and inventory management.
    - Stable database connectivity, ensuring reliable data flow between components.
    - Responsive and mobile-friendly UI/UX design that meets accessibility standards.
    - Well-documented repository and workflow, facilitating team collaboration and future enhancements.
- **Next Steps:**
    - Refine the database schema and backend logic based on team feedback.
    - Continue with catalog search integration and advanced admin panel functionalities.
    - Prepare for the next sprint by consolidating feedback and addressing minor bugs identified during testing.

VIII. STRUCTURE OF THE SCRUM BOARD

The Scrum board is organized into distinct columns to facilitate task management and progress tracking:

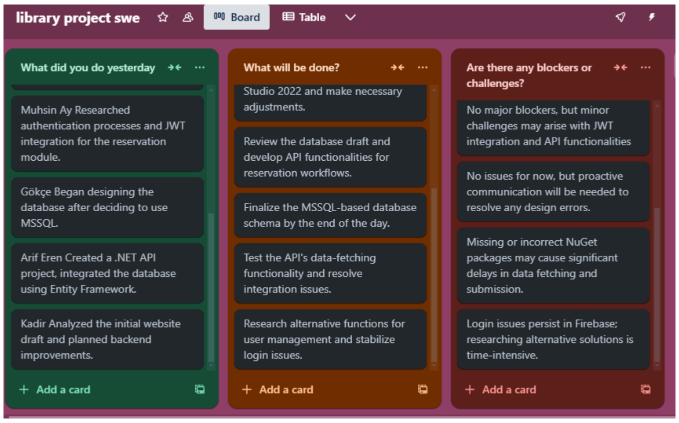
- **What Did You Do Yesterday?** This column records tasks that team members have completed during the previous work cycle.
- **What Will Be Done?** This column lists tasks scheduled for the current or upcoming work cycles, emphasizing ongoing priorities.
- **Are There Any Blockers or Challenges?** This column highlights issues or blockers that could impede the progress of specific tasks or the sprint overall.

Each task on the board is represented as a card containing:

- Task descriptions that outline the specific work to be done.
- Assigned team members to indicate responsibility for the task.
- Labels or tags for categorization, such as backend, frontend, or database.
- Task status and comments for real-time updates and collaboration.

The board structure ensures a clear overview of progress, enabling efficient collaboration and tracking within the team. float

TRELLO WORKFLOW SCREENSHOTS



What did you do yesterday

Muhsin Ay Researched authentication processes and JWT integration for the reservation module.

Gökçe Began designing the database after deciding to use MSSQL.

Arif Eren Created a .NET API project, integrated the database using Entity Framework.

Kadir Analyzed the initial website draft and planned backend improvements.

What will be done?

Studio 2022 and make necessary adjustments.

Review the database draft and develop API functionalities for reservation workflows.

Finalize the MSSQL-based database schema by the end of the day.

Test the API's data-fetching functionality and resolve integration issues.

Research alternative functions for user management and stabilize login issues.

Are there any blockers or challenges?

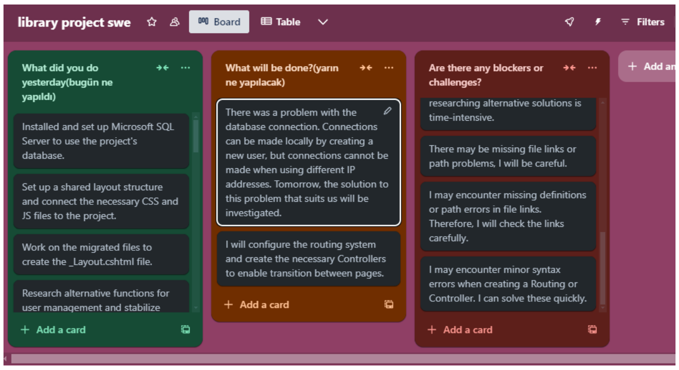
No major blockers, but minor challenges may arise with JWT integration and API functionalities

No issues for now, but proactive communication will be needed to resolve any design errors.

Missing or incorrect NuGet packages may cause significant delays in data fetching and submission.

Login issues persist in Firebase; researching alternative solutions is time-intensive.

FIGURE 6. Scrum Workflow Screenshot for 22.11 - Daily Scrum Progress and Updates.



library project swe

What did you do yesterday(bugün ne yapıldı)

Installed and set up Microsoft SQL Server to use the project's database.

Set up a shared layout structure and connect the necessary CSS and JS files to the project.

Work on the migrated files to create the _Layout.cshtml file.

Research alternative functions for user management and stabilize

What will be done?(yarın ne yapılacak)

There was a problem with the database connection. Connections can be made locally by creating a new user, but connections cannot be made when using different IP addresses. Tomorrow, the solution to this problem that suits us will be investigated.

I will configure the routing system and create the necessary Controllers to enable transition between pages.

Are there any blockers or challenges?

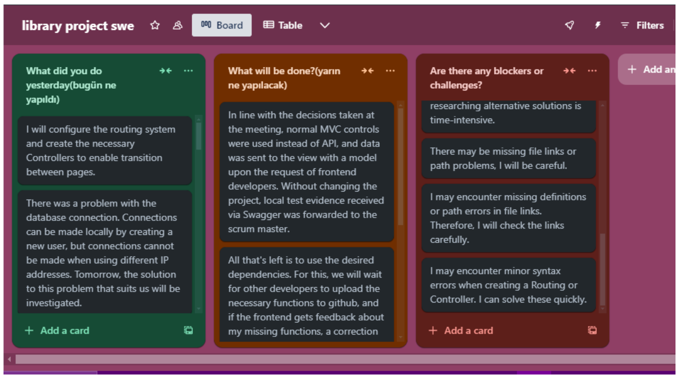
researching alternative solutions is time-intensive.

There may be missing file links or path problems, I will be careful.

I may encounter missing definitions or path errors in file links. Therefore, I will check the links carefully.

I may encounter minor syntax errors when creating a Routing or Controller. I can solve these quickly.

FIGURE 7. Scrum Workflow Screenshot for 23.11 - Daily Scrum Progress and Updates.



library project swe

What did you do yesterday(bugün ne yapıldı)

I will configure the routing system and create the necessary Controllers to enable transition between pages.

There was a problem with the database connection. Connections can be made locally by creating a new user, but connections cannot be made when using different IP addresses. Tomorrow, the solution to this problem that suits us will be investigated.

What will be done?(yarın ne yapılacak)

In line with the decisions taken at the meeting, normal MVC controls were used instead of API, and data was sent to the view with a model upon the request of frontend developers. Without changing the project, local test evidence received via Swagger was forwarded to the scrum master.

All that's left is to use the desired dependencies. For this, we will wait for other developers to upload the necessary functions to github, and if the frontend gets feedback about my missing functions, a correction

Are there any blockers or challenges?

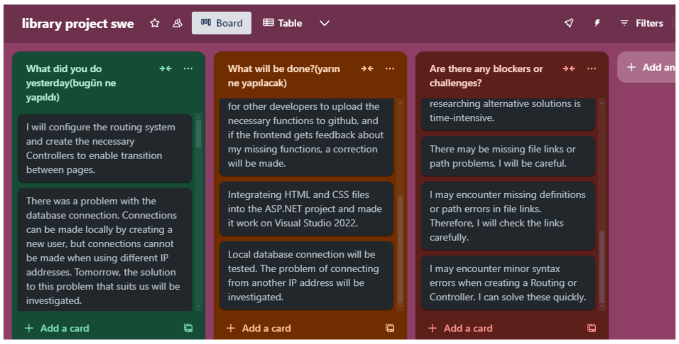
researching alternative solutions is time-intensive.

There may be missing file links or path problems, I will be careful.

I may encounter missing definitions or path errors in file links. Therefore, I will check the links carefully.

I may encounter minor syntax errors when creating a Routing or Controller. I can solve these quickly.

FIGURE 8. Scrum Workflow First Screenshot for 24.11 - Daily Scrum Progress and Updates.



library project swe

What did you do yesterday(bugün ne yapıldı)

I will configure the routing system and create the necessary Controllers to enable transition between pages.

There was a problem with the database connection. Connections can be made locally by creating a new user, but connections cannot be made when using different IP addresses. Tomorrow, the solution to this problem that suits us will be investigated.

What will be done?(yarın ne yapılacak)

for other developers to upload the necessary functions to github, and if the frontend gets feedback about my missing functions, a correction will be made.

Integrateing HTML and CSS files into the ASP.NET project and made it work on Visual Studio 2022.

Local database connection will be tested. The problem of connecting from another IP address will be investigated.

Are there any blockers or challenges?

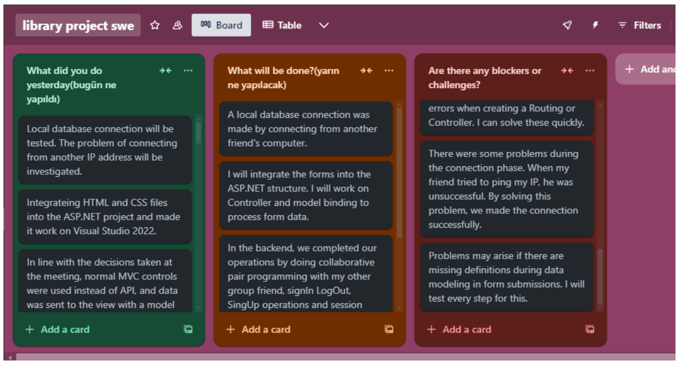
researching alternative solutions is time-intensive.

There may be missing file links or path problems, I will be careful.

I may encounter missing definitions or path errors in file links. Therefore, I will check the links carefully.

I may encounter minor syntax errors when creating a Routing or Controller. I can solve these quickly.

FIGURE 9. Scrum Workflow Second Screenshot for 24.11 - Daily Scrum Progress and Updates.



library project swe

What did you do yesterday(bugün ne yapıldı)

Local database connection will be tested. The problem of connecting from another IP address will be investigated.

Integrateing HTML and CSS files into the ASP.NET project and made it work on Visual Studio 2022.

In line with the decisions taken at the meeting, normal MVC controls were used instead of API, and data was sent to the view with a model

What will be done?(yarın ne yapılacak)

A local database connection was made by connecting from another friend's computer.

I will integrate the forms into the ASP.NET structure. I will work on Controller and model binding to process form data.

In the backend, we completed our operations by doing collaborative pair programming with my other group friend, signIn LogOut, SingUp operations and session

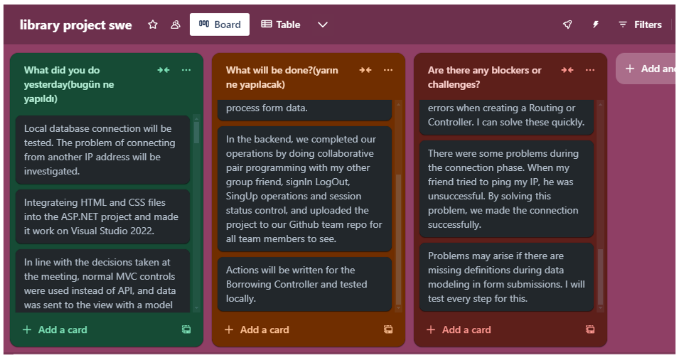
Are there any blockers or challenges?

errors when creating a Routing or Controller. I can solve these quickly.

There were some problems during the connection phase. When my friend tried to ping my IP, he was unsuccessful. By solving this problem, we made the connection successfully.

Problems may arise if there are missing definitions during data modeling in form submissions. I will test every step for this.

FIGURE 10. Scrum Workflow First Screenshot for 25.11 - Daily Scrum Progress and Updates.



library project swe

What did you do yesterday(bugün ne yapıldı)

Local database connection will be tested. The problem of connecting from another IP address will be investigated.

Integrateing HTML and CSS files into the ASP.NET project and made it work on Visual Studio 2022.

In line with the decisions taken at the meeting, normal MVC controls were used instead of API, and data was sent to the view with a model

What will be done?(yarın ne yapılacak)

process form data.

In the backend, we completed our operations by doing collaborative pair programming with my other group friend, signIn LogOut, SingUp operations and session status control, and uploaded the project to our Github team repo for all team members to see.

Actions will be written for the Borrowing Controller and tested locally.

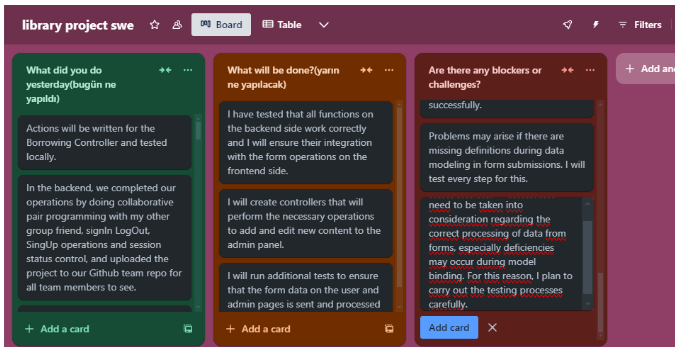
Are there any blockers or challenges?

errors when creating a Routing or Controller. I can solve these quickly.

There were some problems during the connection phase. When my friend tried to ping my IP, he was unsuccessful. By solving this problem, we made the connection successfully.

Problems may arise if there are missing definitions during data modeling in form submissions. I will test every step for this.

FIGURE 11. Scrum Workflow Second Screenshot for 25.11 - Daily Scrum Progress and Updates.



library project swe

What did you do yesterday(bugün ne yapıldı)

Actions will be written for the Borrowing Controller and tested locally.

In the backend, we completed our operations by doing collaborative pair programming with my other group friend, signIn LogOut, SingUp operations and session status control, and uploaded the project to our Github team repo for all team members to see.

What will be done?(yarın ne yapılacak)

I have tested that all functions on the backend side work correctly and I will ensure their integration with the form operations on the frontend side.

I will create controllers that will perform the necessary operations to add and edit new content to the admin panel.

I will run additional tests to ensure that the form data on the user and admin pages is sent and processed

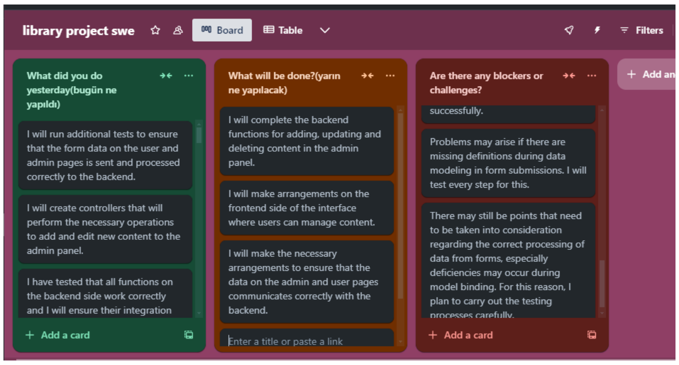
Are there any blockers or challenges?

successfully.

Problems may arise if there are missing definitions during data modeling in form submissions. I will test every step for this.

need to be taken into consideration regarding the correct processing of data from forms, especially deficiencies may occur during model binding. For this reason, I plan to carry out the testing processes carefully.

FIGURE 12. Scrum Workflow Screenshot for 26.11 - Daily Scrum Progress and Updates.



What did you do yesterday(bugün ne yapıldı)

I will run additional tests to ensure that the form data on the user and admin pages is sent and processed correctly to the backend.

I will create controllers that will perform the necessary operations to add and edit new content to the admin panel.

I have tested that all functions on the backend side work correctly and I will ensure their integration

What will be done?(yarın ne yapılacak)

I will complete the backend functions for adding, updating and deleting content in the admin panel.

I will make arrangements on the frontend side of the interface where users can manage content.

I will make the necessary arrangements to ensure that the data on the admin and user pages communicates correctly with the backend.

Enter a title or paste a link

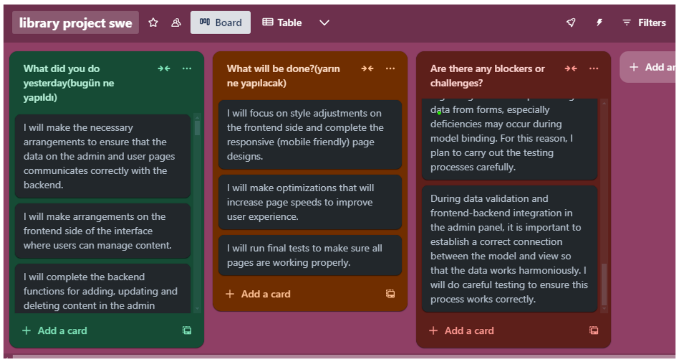
Are there any blockers or challenges?

successfully.

Problems may arise if there are missing definitions during data modeling in form submissions. I will test every step for this.

There may still be points that need to be taken into consideration regarding the correct processing of data from forms, especially deficiencies may occur during model binding. For this reason, I plan to carry out the testing processes carefully.

FIGURE 13. Scrum Workflow Screenshot for 27.11 - Daily Scrum Progress and Updates.



What did you do yesterday(bugün ne yapıldı)

I will make the necessary arrangements to ensure that the data on the admin and user pages communicates correctly with the backend.

I will make arrangements on the frontend side of the interface where users can manage content.

I will complete the backend functions for adding, updating and deleting content in the admin

What will be done?(yarın ne yapılacak)

I will focus on style adjustments on the frontend side and complete the responsive (mobile friendly) page designs.

I will make optimizations that will increase page speeds to improve user experience.

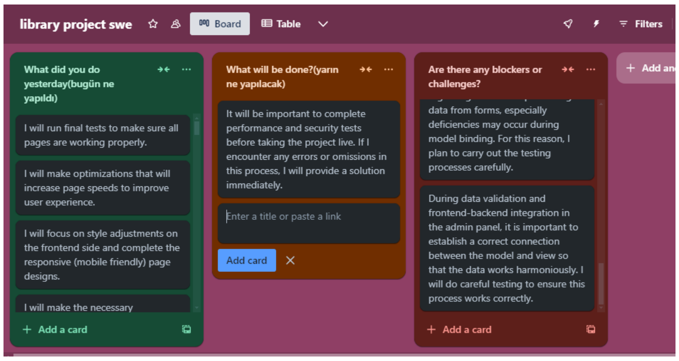
I will run final tests to make sure all pages are working properly.

Are there any blockers or challenges?

data from forms, especially deficiencies may occur during model binding. For this reason, I plan to carry out the testing processes carefully.

During data validation and frontend-backend integration in the admin panel, it is important to establish a correct connection between the model and view so that the data works harmoniously. I will do careful testing to ensure this process works correctly.

FIGURE 14. Scrum Workflow Screenshot for 28.11 - Daily Scrum Progress and Updates.



What did you do yesterday(bugün ne yapıldı)

I will run final tests to make sure all pages are working properly.

I will make optimizations that will increase page speeds to improve user experience.

I will focus on style adjustments on the frontend side and complete the responsive (mobile friendly) page designs.

I will make the necessary

What will be done?(yarın ne yapılacak)

It will be important to complete performance and security tests before taking the project live. If I encounter any errors or omissions in this process, I will provide a solution immediately.

Enter a title or paste a link

Are there any blockers or challenges?

data from forms, especially deficiencies may occur during model binding. For this reason, I plan to carry out the testing processes carefully.

During data validation and frontend-backend integration in the admin panel, it is important to establish a correct connection between the model and view so that the data works harmoniously. I will do careful testing to ensure this process works correctly.

FIGURE 15. Scrum Workflow Screenshot for 29.11 - Daily Scrum Progress and Updates.

BENEFITS OF THE SCRUM APPROACH

This Scrum-based approach allowed us to effectively track progress on a daily basis and ensured that the team stayed aligned with project goals. By utilizing the structured columns and task cards on the Scrum board:

- We were able to **monitor daily activities**, ensuring that each team member was aware of what had been completed and what was planned for the day.
- **Potential problems or blockers** were identified and addressed promptly, minimizing disruptions to the workflow.
- Team members could easily **review what others had accomplished**, fostering collaboration and ensuring no task was overlooked.
- **Planning for the next day** became straightforward, as tasks and challenges were clearly outlined, allowing for better preparation and continuity in workflow.

This method not only improved communication and accountability among team members but also facilitated swift resolution of issues, enabling the project to progress smoothly.

WEEK 1, WEEK 2 AND 3 SCRUM SUMMARY

WEEK 1: FEASIBILITY REPORT AND PLANNING

During the first week, our primary focus was on preparing the **feasibility report**, creating the **project timeline**, and drafting the **study plan**. Daily Scrum meetings ensured that tasks were assigned to team members, tracked consistently, and completed on time. The Scrum board facilitated clear communication and accountability, allowing the team to align on goals and address potential challenges early.

WEEK 2: REQUIREMENTS AND INITIAL DESIGN

In the second week, the team worked on developing the **requirements document** and creating **initial UML diagrams** to outline the system design. Tasks were broken down into manageable increments and assigned through the Scrum board, ensuring progress was tracked effectively. This iterative process allowed us to validate and refine requirements while laying a strong foundation for the project's architecture.

WEEK 3: WORKFLOW DEMONSTRATION

The screenshots provided in this report represent the Scrum workflows from **Week 3**. They illustrate the detailed task assignments, progress tracking, and problem resolution processes implemented to manage the project's third sprint. These images highlight our structured approach to handling tasks, addressing blockers, and ensuring the seamless transition of completed work to the final deliverables.
Overall, the Scrum methodology enabled us to:

- Monitor daily progress and ensure alignment with project milestones.
- Quickly identify and resolve blockers to maintain the project's momentum.
- Provide a clear and structured workflow for each team member.

This approach has proven instrumental in maintaining efficiency and collaboration throughout the project's development.

IX. VIII. REQUIREMENTS ENGINEERING PROCESS

The requirements for the Library Site system were developed following a structured requirements engineering process, including elicitation, analysis, validation, and management. This ensures the system aligns with project goals and user needs.

A. A. 1. REQUIREMENTS ELICITATION

Activities:

- **Stakeholder Interviews:** Engaged with administrators and end-users (library staff) to identify key system needs and challenges.
  - *Example Questions:*
    - For users: "How do you currently search for books? What features would simplify this process?"
    - For administrators: "What challenges do you face in managing the book catalog or reservations?"
- **Workshops:** Conducted brainstorming sessions to prioritize functionalities such as catalog management, user accounts, and admin controls.
  - *Example Outputs:*
    - Users requested a fast and intuitive book search feature with filtering options.
    - Administrators highlighted the need for efficient catalog and reservation management tools.
- **Observation:** Analyzed existing workflows in libraries to identify inefficiencies in catalog updates, book tracking, and reservation handling.
- **Document Analysis:** Reviewed user feedback and relevant library system policies to define initial requirements.

Deliverable: A list of raw requirements directly gathered from stakeholders, serving as the basis for analysis.

B. B. 2. REQUIREMENTS ANALYSIS

Refining and evaluating the raw requirements to ensure they are clear, consistent, and feasible.

Activities:

- **Prioritization:** Classified requirements using the MoSCoW method.
  - *Must-have:* Book search, catalog management, user account system, and reservation functionality.
  - *Should-have:* Notifications for book availability and reservation updates.
  - *Could-have:* Advanced reporting tools for administrators.
- **Conflict Resolution:** Resolved differences in how administrators and users expect catalog and reservation features to behave.
- **Feasibility Study:** Evaluated technical, operational, and financial feasibility of each requirement.
  - Example: Verified that the system could support simultaneous searches and catalog updates without performance degradation.
- **Requirement Modeling:** Created use case diagrams and user stories to represent system functionality.
  - *Example Use Case:* An administrator updates the book catalog with new entries, and users can search and reserve books immediately.

Deliverable: A refined requirements document with clear functional and non-functional requirements, along with supporting models.

C. C. 3. REQUIREMENTS VALIDATION

Ensuring that the refined requirements align with stakeholder needs and are feasible for implementation.

Activities:

- **Stakeholder Reviews:** Presented the refined requirements document and use case diagrams to administrators for approval.
- **Prototyping:** Developed low-fidelity prototypes for features such as the search interface and admin panel to gather feedback.
- **Checklist Validation:** Verified that requirements met the following criteria:
  - *Completeness:* All necessary functionalities were included (e.g., book search, notifications).
  - *Consistency:* No conflicts existed between user and admin requirements.
  - *Feasibility:* The requirements could realistically be implemented within the project scope.

Deliverable: A validated requirements document approved by stakeholders.

D. D. 4. REQUIREMENTS MANAGEMENT

Maintaining and updating requirements throughout the project lifecycle.

Activities:

- **Version Control:** Maintained a version-controlled repository for the requirements document to track changes.
- **Traceability:** Linked requirements to system components and implementation tasks.
  - *Example:* Traced the "Book Search" functionality to the frontend search interface and backend catalog.
- **Change Management:** Established a process to evaluate and approve requirement changes.
  - Example: If a new feature such as bulk book imports is requested, its impact on development timelines and resources is assessed.
- **Periodic Updates:** Conducted regular review meetings to ensure requirements remain relevant as the project progresses.

Deliverable: A living requirements document updated as needed to reflect changes and maintain relevance.

X. FUNCTIONAL REQUIREMENTS

The functional requirements define the core features of the **Library Management System**, detailing how the system should operate to meet its objectives:

A. USER AUTHENTICATION AND ROLE-BASED ACCESS

- Users will log in using their unique **username and password**.
- **Role-based access** will differentiate between Admin and User functionalities:
  - **Admin:** Full access to manage books, users, and system settings.
  - **User:** Limited access to search, borrow, and reserve books.

B. BOOK SEARCH AND BORROWING

- Users will search for books by title, author, or ISBN.
- Advanced search filters will allow sorting by category, publication date, and availability.
- Users will check the availability of books and borrow them directly through the system.

C. BOOK RESERVATIONS

- Users will reserve books that are currently unavailable.
- Reservations will automatically be canceled if not collected within the specified period.

D. USER NOTIFICATIONS

- The system will send notifications for:
  - Book availability (when a reserved book is ready for pickup).
  - Due date reminders for borrowed books.
  - Overdue notifications with applicable penalties.

E. CATALOG MANAGEMENT (ADMIN ONLY)

- Administrators will manage the catalog by adding, editing, and deleting books.
- Book categories will be maintained by the admin for accurate classification.

F. BORROWING AND RESERVATION REPORTS (ADMIN ONLY)

- Administrators will generate reports on:
  - Borrowing trends and overdue books.
  - Current reservations and user activity.

G. SYSTEM LOGS (ADMIN ONLY)

- The system will provide logs of user activities, including:
  - Login attempts.
  - Borrowings and reservations.
  - Catalog modifications.

H. SESSION MANAGEMENT

- The system will securely manage user sessions.
- Users will be automatically logged out after a period of inactivity to enhance security.

XI. NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements define the system's operational characteristics and constraints, ensuring usability, performance, and maintainability of the **Library Management System**.

A. PERFORMANCE REQUIREMENTS

- The system must support **at least 100 concurrent users** without performance degradation.
- Search functionality should return results within **5 seconds** for typical queries.
- The system must handle **database queries and updates** within **1 second** on average.

B. USABILITY REQUIREMENTS

- The interface should be **intuitive and user-friendly**, with minimal learning time for new users.
- Provide clear and concise error messages to guide users in resolving issues.

C. SECURITY REQUIREMENTS

- Use **HTTPS** for all communications to ensure secure data transfer.
- Implement **role-based access control** to restrict admin-specific functionalities from regular users.
- Perform **regular security audits** to identify and mitigate vulnerabilities.

D. SCALABILITY REQUIREMENTS

- The system must scale to support an **increasing number of users and book records** without significant performance drops.
- Cloud-based deployment should allow dynamic scaling based on usage demands.

E. AVAILABILITY REQUIREMENTS

- The system should maintain an **uptime of 99.9%**, ensuring minimal disruptions to service.
- Scheduled maintenance should be announced **at least 24 hours in advance**.

F. MAINTAINABILITY REQUIREMENTS

- The system must allow **modular updates** without affecting other functionalities.
- Use **version control** (e.g., Git) for tracking changes and collaborative development.
- Ensure that all code is **well-documented** for ease of future maintenance.

G. DATA INTEGRITY REQUIREMENTS

- Data entered into the system should be **validated** to prevent errors and inconsistencies.
- The system must implement **backup and recovery mechanisms** to prevent data loss in case of failure.

H. COMPLIANCE REQUIREMENTS

- Ensure the system complies with **GDPR** for handling user data securely and transparently.
- Follow relevant industry standards for data storage, such as **ISO/IEC 27001**.

XII. USER REQUIREMENTS

The user requirements define the essential features and functionalities required by all stakeholders of the Library Site, including both users (individuals searching for books) and administrators. These requirements ensure a secure, user-friendly, and efficient library management system.

A. 1. GENERAL REQUIREMENTS

These requirements are applicable to all system users.

- The system must be accessible via modern web browsers without requiring additional software installation.
- The user interface should be simple, intuitive, and easy to navigate.
- The system must implement a secure **username and password-based login mechanism** to protect user accounts.
- Users and administrators should only access functions within their respective roles through role-based access control (RBAC).

B. 2. USER REQUIREMENTS

These requirements define the functionalities needed by users to interact effectively with the system.

1) 2.1 Account Management

- Users can create an account using basic personal information (e.g., username and password).
- Users can update their profile information, such as contact details.
- Users can reset their passwords securely in case of forgotten credentials.

2) 2.2 Catalog Access

- Users can browse the library's book catalog, viewing details such as:
  - Title
  - Author
  - Genre
  - Availability status
- Users can search for books using keywords and apply filters, including:
  - Genre
  - Author
  - Publication year

3) 2.3 Notifications

- Users are notified of system events, including:
  - Responses to reservation requests.
  - Notifications about reserved books that have become available.
- Notifications are displayed on the user dashboard.

4) 2.4 Help and Support

- Users can access a FAQ section to resolve common queries.
- Users can contact library support directly via a built-in contact form for assistance.

C. 3. ADMINISTRATOR REQUIREMENTS

These requirements define the functionalities and tools necessary for administrators to manage and maintain the Library Site system effectively.

1) 3.1 Account and Role Management

- Administrators can create, update, and deactivate user accounts.
- Administrators have exclusive access to system management tools through role-based access control (RBAC).

2) 3.2 Catalog Management

- Administrators can add, update, or remove books from the catalog.
- Administrators can categorize books by genre, author, and publication year.
- Administrators can monitor and manage book availability statuses.

3) 3.3 Book Reservation Management

- Administrators can handle book reservation requests from users, including:
  - Approving or rejecting reservation requests.
  - Assigning books to specific users for pickup.
  - Cancelling reservations if necessary.
- Administrators can track overdue books and send reminders to users.

4) 3.4 Notifications Management

- Administrators can configure system-wide notifications for events such as:
  - System maintenance.
  - New book arrivals.
  - Policy updates.

5) 3.5 Helpdesk Support

- Administrators can monitor and respond to user support requests through a centralized helpdesk dashboard.

XIII. SYSTEM REQUIREMENTS

The system requirements for the Library Site project outline the technical, operational, and security specifications necessary for the system to function effectively. These requirements are derived from the project's objectives, user needs, and tasks undertaken during development.

A. 1. FUNCTIONAL REQUIREMENTS

The functional requirements define the key features and operations of the Library Site.

1) 1.1 User Management

- Users can register with a username and password.
- Administrators can create, update, or deactivate user accounts.
- Role-based access control (RBAC) ensures:
  - **Users:** Access catalog and request book reservations.
  - **Administrators:** Manage catalog, reservations, and users.

2) 1.2 Catalog Management

- Administrators can add, edit, or remove books in the catalog.
- Book details include:
  - Title
  - Author
  - Genre
  - Availability status
- Users can search the catalog using keywords and filters (e.g., genre, author, publication year).

3) 1.3 Reservation Management

- Users can request reservations for books marked as "Available."
- Administrators can:
  - Approve or reject reservation requests.
  - Assign reserved books for pickup.
  - Manage overdue reservations and send reminders.

4) 1.4 Notifications

- Notifications are displayed in the user dashboard for:
  - Reservation request updates.
  - Book availability changes.
- Administrators can configure global notifications for system events, such as:
  - Maintenance schedules.
  - New book arrivals.

5) 1.5 Helpdesk Support

- Users can access a FAQ section for common queries.
- Users can submit support tickets via a contact form.
- Administrators can respond to tickets and monitor user feedback.

B. 2. NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements define the system's performance, reliability, usability, and security standards.

1) 2.1 Performance Requirements

- The system must support up to 100 concurrent users without performance degradation.
- Page load times should not exceed 3 seconds for 90% of requests.

2) 2.2 Usability Requirements

- The user interface must be intuitive, requiring minimal training for first-time users.
- All forms and workflows must be validated to prevent user errors.

3) 2.3 Security Requirements

- The system must prevent unauthorized access through role-based permissions.
- All user inputs must be validated to prevent SQL injection or XSS attacks.

4) 2.4 Availability Requirements

- The system must maintain 99.9% uptime.
- Automated backups should be performed daily to prevent data loss.

5) 2.5 Scalability Requirements

- The system must be able to scale horizontally to accommodate future growth in user numbers and data volume.
- Database queries should be optimized to handle increased traffic.

6) 2.6 Maintainability Requirements

- The codebase must be modular and well-documented for future enhancements.
- Unit and integration tests must be written for all critical functionalities.

C. 3. SYSTEM CONSTRAINTS

The following constraints impact the design and implementation of the Library Site:

- The system must be developed using **ASP.NET** for backend logic and **MsSQL** for database management.
- The frontend must use a combination of **HTML**, **CSS**, **Razor**, and **Bootstrap** for responsive design.

D. 4. INTEGRATION REQUIREMENTS

- The system must integrate with existing library workflows, including inventory management and notifications.
- Embedded processes should be used for seamless interaction between frontend and backend systems.

XIV. SYSTEM MODELS

A. SCENARIOS

1) Scenario 1: User Registration and Login

Scenario Name: UserAuthentication

Participating Actors: User, System

1) A new user visits the Library Site and clicks the *"Signup"* button.
2) The system displays a registration form.
3) The user fills in the form with their username, password, and email address.
4) The system validates the form inputs and creates the user account.
5) The user is redirected to the login page.
6) The user enters their username and password to log in.
7) The system authenticates the credentials and grants access to the user dashboard.
8) If the credentials are invalid, the system displays an error message: *"Invalid username or password. Please try again."*

System Responses:

- Validate and securely store user data in the database.
- Send a confirmation email after successful registration.
- Handle invalid inputs with appropriate error messages.

2) Scenario 2: Search for Books

Scenario Name: BookSearch

Participating Actors: User, System

1) A user logs into the Library Site.
2) The user navigates to the *"Search"* page.
3) The user enters search keywords (e.g., title, author, genre, or ISBN) into the search bar.
4) The system processes the input and filters the book catalog based on the search criteria.
5) The system displays a list of matching results, including book details such as title, author, and availability status.
6) The user clicks on a book to view more details.
7) If no results are found, the system displays a message: *"No books match your search criteria."*

System Responses:

- Filter books dynamically based on user input.
- Display search results in a user-friendly format with pagination.
- Handle empty or invalid search queries with appropriate messages.

3) Scenario 3: Reserve a Book (Admin)

Scenario Name: AdminBookReservation

Participating Actors: Admin, User, System

1) A user contacts the library (e.g., via phone, email, or in-person) and requests a book reservation.
2) The admin logs into the Library Site using their credentials.
3) The admin navigates to the *"Manage Reservations"* page from the dashboard.
4) The admin searches for the requested book using filters like title, author, or ISBN.
5) The system displays the book details, including its availability status.
6) If the book is available, the admin selects the book and assigns it to the user by filling in their details (e.g., username, contact information).
7) The system processes the reservation, updates the book's status to *"Reserved"*, and links the reservation to the user.
8) The system sends an email notification to the user confirming the reservation: *"Your book has been successfully reserved and is ready for pickup."*
9) If the book is unavailable, the system notifies the admin: *"This book is currently reserved."*
10) The admin informs the user about the book's status and suggests alternatives if applicable.

System Responses:

- Update the book's status to *"Reserved"* in the database.
- Send a notification to the user with reservation confirmation.
- Display an error message to the admin if the book is already reserved.

4) Scenario 4: Manage Book Catalog (Admin)

Scenario Name: CatalogManagement

Participating Actors: Admin, System

1) The admin logs into the Library Site.
2) The admin navigates to the *"Manage Catalog"* page.
3) The admin adds a new book by entering details such as title, author, genre, ISBN, and availability status.
4) The system validates the inputs and updates the database with the new book information.
5) The admin edits an existing book entry to correct or update information.
6) The system processes the edits and reflects the changes in the catalog.
7) The admin deletes a book that is no longer available in the library.
8) The system removes the book from the database and updates the catalog.

System Responses:

- Validate admin inputs for book details.
- Synchronize database changes with the frontend.
- Display success or error messages after add, edit, or delete operations.

5) Scenario 5: Notifications for Due Dates

Scenario Name: NotificationSystem

Participating Actors: User, Admin, System

1) An admin reserves a book for a user, setting a due date in the system.
2) The system schedules a notification to be sent to the user 24 hours before the due date.

3) On the scheduled time, the system sends an email and dashboard notification reminding the user to return the book.
4) If the book is overdue, the system sends a follow-up reminder daily until the book is returned.
5) The admin can view overdue books and send additional reminders manually if needed.

System Responses:

- Automatically schedule and send notifications for due dates.
- Allow manual reminder notifications by the admin.
- Display all notifications in the user dashboard.

XV. SYSTEM BOUNDARIES FOR THE LIBRARY SITE

The Library Site System is an integrated platform designed to operate as a self-contained environment without reliance on external APIs. All functionalities are embedded within the system, with components working cohesively to deliver a seamless experience for both users and administrators.

A. INTERNAL SYSTEM COMPONENTS

The system comprises the following interconnected modules:

1) **User Management**
   - **Admin Role:** Responsible for managing the book catalog, user accounts, and processing book reservations.
   - **User Role:** Performs book searches, views details, and interacts with notifications for reservations and overdue books.
2) **Book Catalog Management**
   - Enables admins to add, update, and delete books.
   - Updates book availability status based on reservations.
3) **Search Module**
   - Provides a search interface for users to find books by title, author, or ISBN.
   - Filters results for relevance and displays real-time updates.
4) **Reservation Management**
   - Admins process all book reservations on behalf of users.
   - The database is updated with reservation details to maintain consistency in book availability.
5) **Notification System**
   - Sends reminders to users for due dates and overdue books.
   - Fully integrated within the backend logic for scheduling notifications.
6) **User Interface (UI)**
   - Designed using ASP.NET Razor pages to provide an intuitive and responsive experience.
   - Role-based interfaces display tailored features for admins and users.
7) **Database**
   - A centralized SQL Server database stores all persistent data, including user accounts, book catalog, and transaction logs.
   - Integrated with backend modules to support reliable operations.

B. INTERFACES

- **User Interface:** Accessible via a web browser, providing a secure login system and features based on user roles.
- **Database Interface:** Handles data storage and retrieval operations directly without relying on external services.

C. SYSTEM EXCLUSIONS

To ensure simplicity and focus, the following features are excluded:

- **External APIs:** All functionalities, including notifications and search, are embedded within the system.
- **Mobile Application:** The system is web-based and optimized for desktop browsers only.
- **Advanced Analytics:** Basic reporting is provided for admins, but complex analytics or visualizations are not included.

D. KEY CONSTRAINTS

- **Security:** Role-based access control ensures authorized use of features. Credentials are securely validated, and data operations are restricted to internal modules.
- **Performance:** Designed to handle up to 500 concurrent users with minimal latency.
- **Scalability:** Built to scale by increasing server resources as needed.
- **Reliability:** Implements robust error-handling to maintain uninterrupted operations.

E. OPERATIONAL SCOPE

The system's boundaries are confined to:

- Admins managing book catalogs, reservations, and notifications.
- Users searching for books and interacting with notifications.
- Data storage and processing confined to the centralized database.

This definition of system boundaries ensures clear operational guidelines, enabling efficient implementation and streamlined functionality.

XVI. RISK MANAGEMENT ANTICIPATED CHANGES AND PLAN FOR ADAPTATION

A. POTENTIAL CHANGES IN THE LIBRARY SITE SYSTEM

The Library Site system is designed with flexibility in mind; however, certain areas may require updates or enhancements based on feedback, scalability, or new requirements. Below is a detailed analysis of potential changes and our plan to address them.

1) 1. User Interface and Experience Enhancements

Potential Change: Feedback from admins or users may reveal areas needing improvement in navigation, layout, or usability. **Plan:**

- Conduct regular usability testing and gather stakeholder feedback.
- Ensure frontend codebase flexibility for iterative updates without disrupting core functionalities.

2) 2. Database Schema Modifications

Potential Change: New requirements, such as additional book details or updates to reservation logic, may necessitate changes to the database schema. **Plan:**

- Use version control for the database schema to manage changes effectively.
- Implement database migrations to handle schema updates without data loss.

3) 3. Role and Permission Adjustments

Potential Change: Additional roles (e.g., super admin, guest user) or more granular permissions may be needed. **Plan:**

- Design the role-based access control (RBAC) system with extensibility to accommodate future roles and permissions.

4) 4. Notification System Expansion

Potential Change: Additional notification channels, such as SMS or push notifications, may be requested. **Plan:**

- Implement a modular notification system for easy integration of new channels.
- Use services like Twilio or Firebase for scalable notification delivery.

5) 5. Scalability and Performance Improvements

Potential Change: Increased user base or book catalog size may require system performance optimizations. **Plan:**

- Monitor system performance metrics and optimize backend queries.
- Use caching mechanisms, such as Redis, to reduce database load for frequently accessed data.

6) 6. Reporting and Analytics Enhancements

Potential Change: Admins may request advanced analytics or custom reports. **Plan:**

- Build a customizable reporting dashboard for new metrics.
- Integrate business intelligence tools like Power BI for advanced reporting if required.

7) 7. Legal and Compliance Updates

Potential Change: Updates to privacy laws or institutional policies may require changes in data storage and processing. **Plan:**

- Periodically review compliance with legal and institutional requirements.
- Implement data protection measures, such as anonymization and user data export tools.

8) 8. Integration of New Features

Potential Change: New features, such as digital lending or integration with external library networks, may be requested. **Plan:**

- Adopt a modular architecture for seamless addition of new features.
- Use design patterns like microservices if integration needs expand significantly.

B. CHANGE MANAGEMENT PLAN

1) **Change Request Process:**
   - Document all change requests and evaluate their impact on the timeline, budget, and scope.
   - Involve stakeholders in prioritizing changes.
2) **Version Control:**
   - Use Git for tracking changes in code, database schema, and documentation.
3) **Iterative Updates:**
   - Implement changes during designated sprint cycles to minimize disruption.
4) **Stakeholder Communication:**
   - Maintain regular communication with stakeholders to align on changes and priorities.

By planning for these changes, the Library Site system will remain adaptable, scalable, and aligned with user and stakeholder needs throughout its lifecycle.

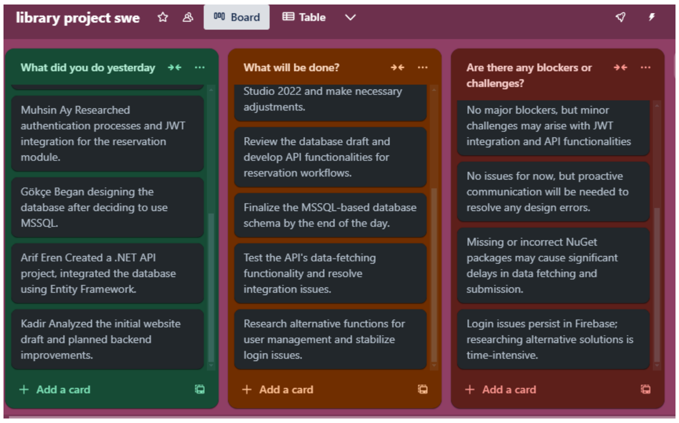
XVII. PROJECT MANAGEMENT TOOLS AND METHODS

Effective project management is essential for coordinating team efforts, tracking progress, and delivering a successful project. For the Library Site project, the team employed a combination of three tools—Trello, Google Docs, and GitHub—to manage tasks, facilitate communication, and handle code versioning. These tools provided a structured approach to project execution and ensured collaboration among team members.

A. TRELLO FOR SCRUM MANAGEMENT

Trello served as the primary tool for implementing Scrum methodology. It was used for task creation, assignment, and tracking, as well as for managing sprints. Each task was detailed with descriptions, assignees, and deadlines. Trello's visual Scrum boards allowed the team to monitor task progression across different stages, such as "To Do," "In Progress," and "Done."

Additionally, the tool provided burndown charts and sprint reports, which offered real-time insights into team performance and remaining workload. These metrics helped the team stay on track and adjust priorities when necessary.



library project swe

Board | Table

What did you do yesterday

Muhsin Ay Researched authentication processes and JWT integration for the reservation module.

Gökçe Began designing the database after deciding to use MSSQL.

Arif Eren Created a .NET API project, integrated the database using Entity Framework.

Kadir Analyzed the initial website draft and planned backend improvements.

+ Add a card

What will be done?

Studio 2022 and make necessary adjustments.

Review the database draft and develop API functionalities for reservation workflows.

Finalize the MSSQL-based database schema by the end of the day.

Test the API's data-fetching functionality and resolve integration issues.

Research alternative functions for user management and stabilize login issues.

+ Add a card

Are there any blockers or challenges?

No major blockers, but minor challenges may arise with JWT integration and API functionalities

No issues for now, but proactive communication will be needed to resolve any design errors.

Missing or incorrect NuGet packages may cause significant delays in data fetching and submission.

Login issues persist in Firebase; researching alternative solutions is time-intensive.

+ Add a card

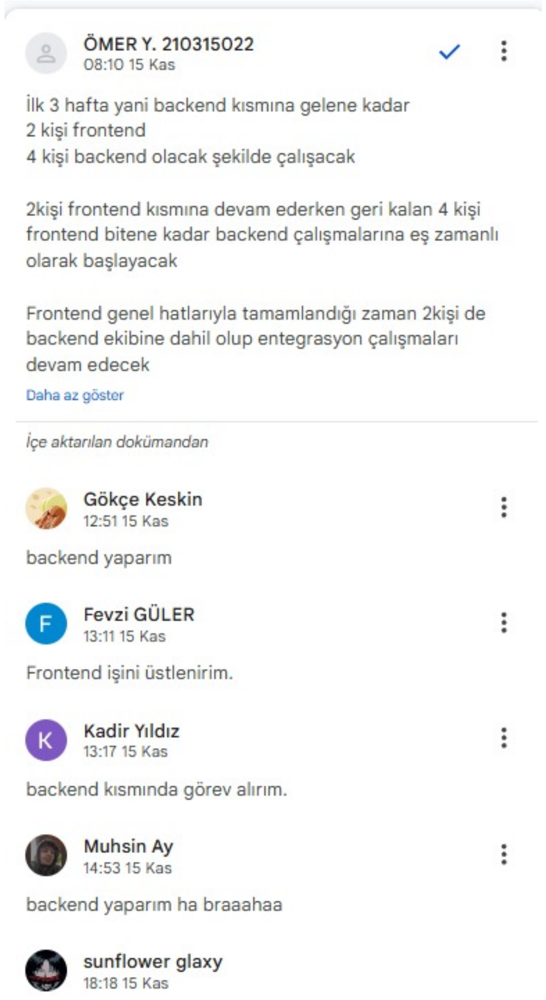
FIGURE 16. Trello Scrum Board for Task Management Example

B. GOOGLE DOCS FOR COMMUNICATION AND DOCUMENTATION

Google Docs was the central platform for team communication and documentation management. It facilitated real-time collaboration among team members, enabling multiple users to edit and review documents simultaneously. Key features utilized included:

- **Daily Scrum Updates:** Team members logged their daily progress, challenges, and next steps.
- **Announcements:** Important updates, such as changes in project scope or sprint goals, were shared through a centralized document.
- **Documentation:** All formal documents, including requirements, design specifications, and user stories, were created and maintained in Google Docs.

This approach ensured transparency and provided an easily accessible archive of project-related information for all team members.



ÖMER Y. 210315022
08:10 15 Kas

İlk 3 hafta yani backend kısmına gelene kadar
2 kişi frontend
4 kişi backend olacak şekilde çalışacak

2kişi frontend kısmına devam ederken geri kalan 4 kişi frontend bitene kadar backend çalışmalarına eş zamanlı olarak başlayacak

Frontend genel hatlarıyla tamamlandığı zaman 2kişi de backend ekibine dahil olup entegrasyon çalışmaları devam edecek

Daha az göster

İçe aktarılan dokümandan

Gökçe Keskin
12:51 15 Kas

backend yaparım

Fevzi GÜLER
13:11 15 Kas

Frontend işini üstlenirim.

Kadir Yıldız
13:17 15 Kas

backend kısmında görev alırım.

Muhsin Ay
14:53 15 Kas

backend yaparım ha braaahaa

sunflower glaxy
18:18 15 Kas

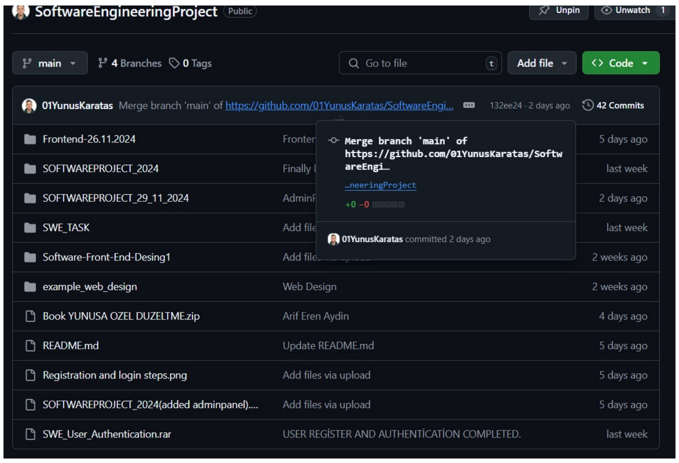
FIGURE 17. Google Docs for Communication and Documentation Sharing

C. GITHUB FOR CODE VERSIONING AND COLLABORATION

GitHub was employed to manage the project's codebase, ensuring efficient version control and collaboration. The repository was actively maintained with regular commits from team members, reflecting incremental progress and bug fixes. GitHub's features included:

- **Branching and Merging:** Team members worked on separate branches to implement new features or resolve issues, which were later merged into the main branch after review.
- **Commit History:** A total of 42 commits were logged, providing a transparent record of all code changes.
- **Pull Requests:** Code reviews and discussions were facilitated through pull requests, ensuring high-quality code integration.

GitHub also served as a backup for the project code, safeguarding against data loss and enabling collaborative problem-solving.



SoftwareEngineeringProject Public

Unpin · Unwatch 1

main · 4 Branches · 0 Tags · Go to file · Add file · Code

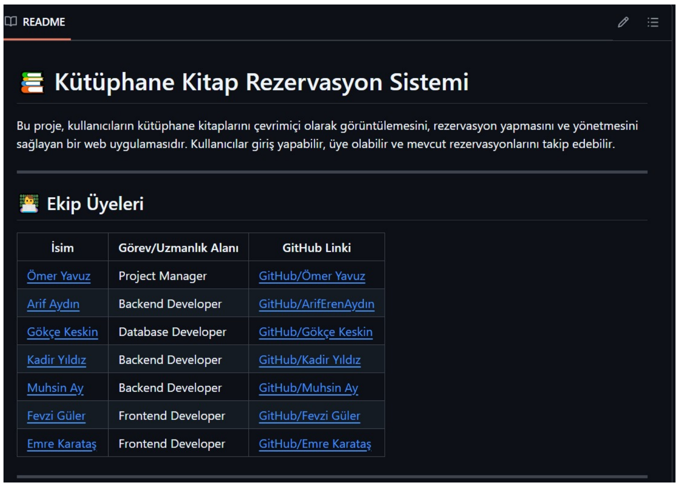
01YunusKaratas Merge branch 'main' of https://github.com/01YunusKaratas/SoftwareEngi... 132ee24 · 2 days ago · 42 Commits

Frontend-26.11.2024	Fronten	5 days ago
SOFTWAREPROJECT_2024	Finally I	last week
SOFTWAREPROJECT_29_11_2024	AdminF	2 days ago
SWE_TASK	Add file	last week
Software-Front-End-Desing1	Add file	2 weeks ago
example_web_design	Web Design	2 weeks ago
Book YUNUSA OZEL DUZELTME.zip	Arif Eren Aydin	4 days ago
README.md	Update README.md	5 days ago
Registration and login steps.png	Add files via upload	5 days ago
SOFTWAREPROJECT_2024(added adminpanel)....	Add files via upload	5 days ago
SWE_User_Authentication.rar	USER REGİSTER AND AUTHENTİCATİON COMPLETED.	last week

```
Merge branch 'main' of
https://github.com/01YunusKaratas/Softw
areEngi…
…neeringProject
+0 -0
```

01YunusKaratas committed 2 days ago

FIGURE 18. GitHub Repository for Code Management and Collaboration



README

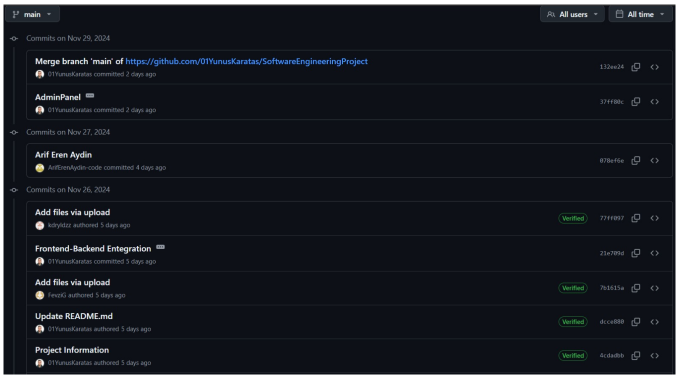
Kütüphane Kitap Rezervasyon Sistemi

Bu proje, kullanıcıların kütüphane kitaplarını çevrimiçi olarak görüntülemesini, rezervasyon yapmasını ve yönetmesini sağlayan bir web uygulamasıdır. Kullanıcılar giriş yapabilir, üye olabilir ve mevcut rezervasyonlarını takip edebilir.

Ekip Üyeleri

İsim	Görev/Uzmanlık Alanı	GitHub Linki
Ömer Yavuz	Project Manager	GitHub/Ömer Yavuz
Arif Aydın	Backend Developer	GitHub/ArifErenAydın
Gökçe Keskin	Database Developer	GitHub/Gökçe Keskin
Kadir Yıldız	Backend Developer	GitHub/Kadir Yıldız
Muhsin Ay	Backend Developer	GitHub/Muhsin Ay
Fevzi Güler	Frontend Developer	GitHub/Fevzi Güler
Emre Karataş	Frontend Developer	GitHub/Emre Karataş

FIGURE 19. GitHub Repository for Code Management and Collaboration



main · All users · All time

Commits on Nov 29, 2024

- Merge branch 'main' of https://github.com/01YunusKaratas/SoftwareEngineeringProject — 01YunusKaratas committed 2 days ago — 132ee24
- AdminPanel — 01YunusKaratas committed 2 days ago — 37ff80c

Commits on Nov 27, 2024

- Arif Eren Aydin — ArifErenAydin-code committed 4 days ago — 078ef6e

Commits on Nov 26, 2024

- Add files via upload — kdryldzz authored 5 days ago — Verified — 77ff097
- Frontend-Backend Entegration — 01YunusKaratas committed 5 days ago — 21e709d
- Add files via upload — FevziG authored 5 days ago — Verified — 7b1615a
- Update README.md — 01YunusKaratas authored 5 days ago — Verified — dcce880
- Project Information — 01YunusKaratas authored 5 days ago — Verified — 4cdadbb

FIGURE 20. GitHub Workflow for Code Management and Collaboration

D. INTEGRATED WORKFLOW

The integration of Trello, Google Docs, and GitHub created a seamless workflow that ensured efficiency and productivity. The interplay between these tools is summarized below:

- **Trello:** Centralized task management enabled structured sprint planning and real-time progress tracking.
- **Google Docs:** Provided a unified platform for documentation and communication, ensuring team alignment.
- **GitHub:** Ensured collaborative development with version control, allowing team members to work on features simultaneously.

Each tool addressed specific project needs, and their combined use ensured that every aspect of the project was systematically managed.

XVIII. CONCLUSION

The use of these three project management tools allowed for efficient communication, clear task distribution, and robust code collaboration. By leveraging Trello for task management, Google Docs for communication and documentation, and GitHub for code versioning, the team successfully executed the Library Site project while maintaining transparency and accountability across al

XIX. ARCHITECTURAL DESIGN FOR THE LIBRARY SITE

A. OVERVIEW

The architectural design of the Library Site system follows a three-tiered architecture that integrates the frontend, backend, and database layers into a cohesive framework. This approach ensures scalability, maintainability, and ease of integration. The system is implemented as an embedded application, removing the need for external APIs by combining all components into a single cohesive structure.

B. ARCHITECTURAL DESIGN DIAGRAM

C. COMPONENTS AND LAYERS

1) 1. Presentation Layer (Frontend)

This layer is responsible for the user interface and user experience. It interacts directly with users and provides functionalities based on their roles (admin or user).

Key Features:

- User-friendly interfaces developed using HTML, CSS, and Razor views within the ASP.NET MVC framework.
- Routing handled by embedded controllers to navigate seamlessly between pages.
- Forms for login, book search, reservation, and catalog management.
- Integrated session management to ensure secure and personalized user experiences.

2) 2. Application Layer (Backend)

This layer processes business logic and manages the interaction between the frontend and the database.

Key Features:

- Implemented using C# within the ASP.NET MVC framework.
- Role-based access control (RBAC) to distinguish admin and user functionalities.
- Embedded controllers like `BooksController` and `UsersController` for operations such as book search, reservations, and catalog updates.

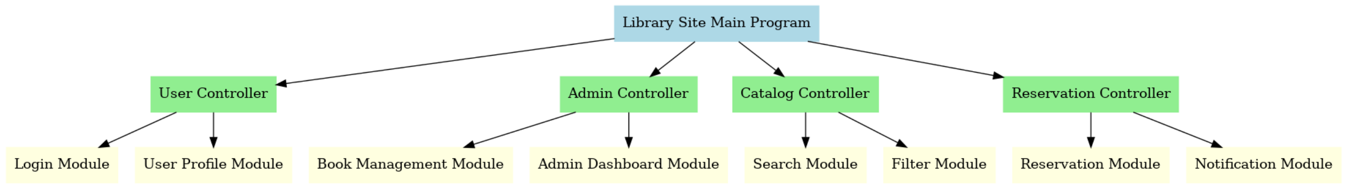


FIGURE 21. Library Site Architecture Diagram

- Embedded processes for direct communication between components, replacing external API calls.

3) 3. Data Layer (Database)

This layer handles data storage and retrieval.

Key Features:

- Relational database designed using Microsoft SQL Server.
- Well-defined schema to store user details, book records, and reservation history.
- Direct integration with the backend for efficient data access and updates.

D. DATA FLOW

1) **User Interaction:** Users (admins and general users) interact with the system through the presentation layer to perform tasks such as searching for books, managing reservations, and updating the catalog.
2) **Business Logic Processing:** The backend processes requests, validates data, and enforces business rules such as role-based access and availability checks.
3) **Data Storage and Retrieval:** The backend communicates with the database to fetch, update, or delete records based on user actions.

E. EXPLANATION OF EMBEDDED APPROACH

In contrast to API-based architectures, the Library Site uses an embedded approach to integrate frontend and backend components directly. This design ensures:

- Reduced latency as data is processed within the same application.
- Simplified maintenance with a unified codebase.
- Enhanced security by eliminating external API endpoints.

F. BENEFITS OF THE ARCHITECTURE

- **Scalability:** The system can handle increased user loads by deploying additional server instances.
- **Maintainability:** Clear separation of concerns between layers facilitates easier debugging and updates.
- **Performance:** Direct integration between layers reduces overhead associated with API communication.

XX. OBJECT MODEL

The object model for the Library Site illustrates the relationships between key entities, including:

- **Users:** Individuals who search for books and interact with the system.
- **Admins:** Roles responsible for managing the book catalog and reservations.
- **Books:** Entities containing details such as title, author, genre, and availability.
- **Reservations:** Links between users and the books they reserve.

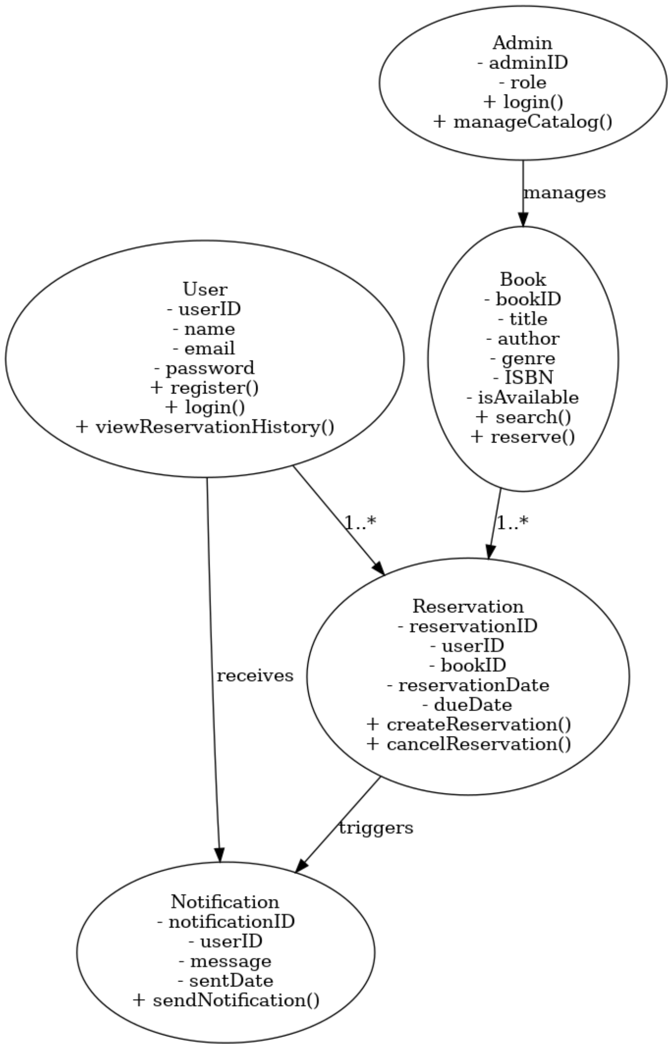


FIGURE 22. Object Model for Library Site

XXI. STATE DIAGRAMS

State diagrams provide a detailed view of the various states and transitions involved in key functionalities of the Library Site project. Below are the state diagrams for each core functionality:

A. ADMIN BOOK RESERVATION

This state diagram illustrates the workflow of the admin handling book reservations. The process includes states like checking book availability, reserving the book, and updating the system.

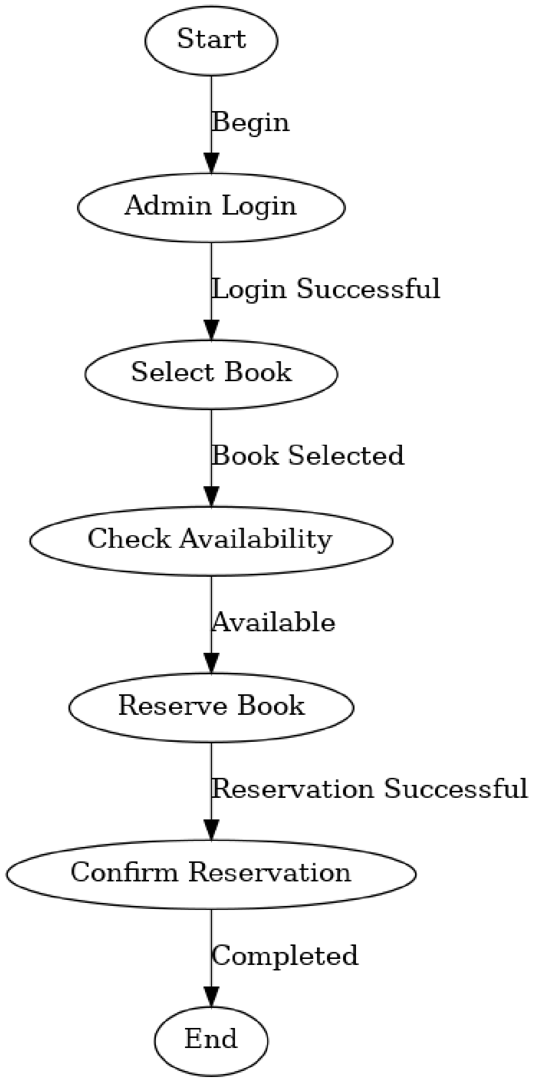


FIGURE 23. State Diagram for Admin Book Reservation

B. USER REGISTRATION AND LOGIN

This state diagram represents the flow of user registration and login, including account creation, validation, and session management.

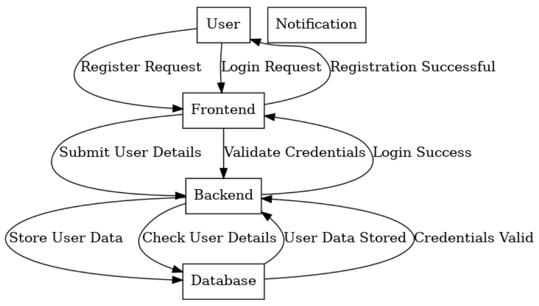


FIGURE 24. State Diagram for Registration and Login

C. SEARCH BOOK

This diagram outlines the states and transitions involved in the book search process. It includes input of search criteria, filtering results, and displaying detailed information.

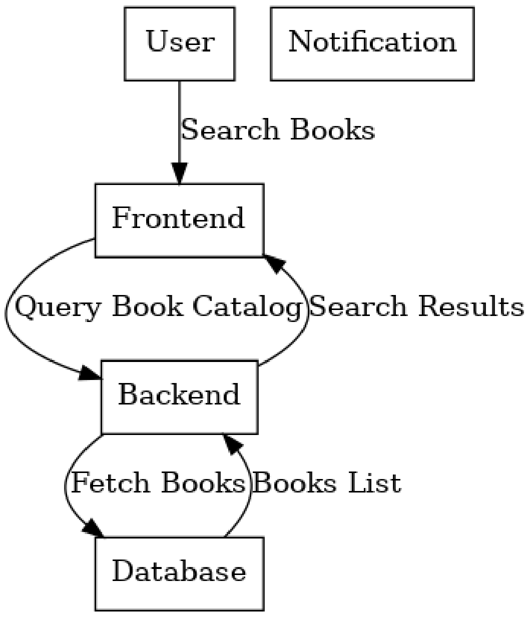


FIGURE 25. State Diagram for Search Book

D. MANAGE CATALOG

The manage catalog state diagram describes the admin's actions to add, update, or delete book entries in the catalog and the corresponding system updates.

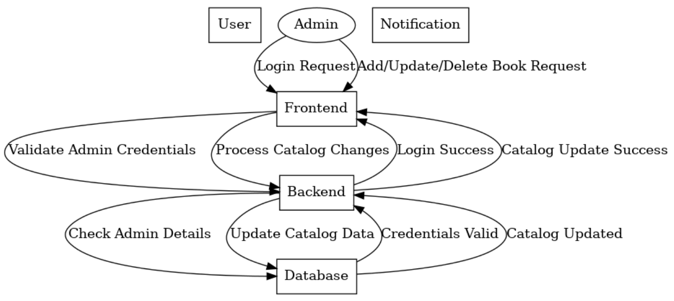


FIGURE 26. State Diagram for Manage Catalog

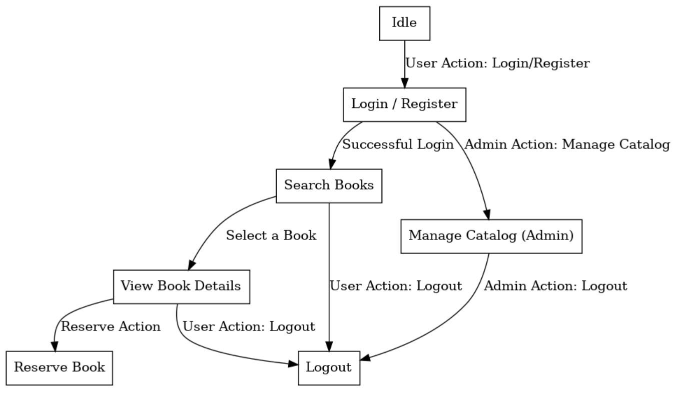


FIGURE 27. State Diagram for Book Entity in Library Site

XXII. USE CASE DIAGRAM EXPLANATION

This section provides a detailed explanation of the Use Case Diagram for the Library Application. The diagram outlines the primary actions performed by two main actors: **User/Customer** and **Administrator**.

A. USER/CUSTOMER USE CASES

The User/Customer actor represents the library's patrons who interact with the system for various purposes. Their primary actions include:

- **ManageAccount:** Allows users to create an account, update their profile, and reset their password.
  - **CreateAccount:** Enables users to register and create a new account.
  - **UpdateProfileInformation:** Allows users to update their existing account information.
  - **ResetPassword:** Provides functionality for users to reset their account password.
- **AccessCatalog:** Facilitates browsing and searching for books in the library catalog.
  - **SearchBooks:** Enables users to search for books by title, author, or genre.
  - **BrowseBookCatalog:** Allows users to explore the full catalog of available books.
- **MakeReservations:** Enables users to reserve books that are available.
  - **SendRequest:** Handles user requests for book reservations.
- **NotifiedOfSystemEvents:** Keeps users informed about system notifications.
  - **DisplayOnDashboard:** Displays notifications on the user's dashboard.
- **HelpAndSupport:** Offers support options for users.
  - **AccessFAQSection:** Provides access to frequently asked questions.
  - **ContactLibrarySupport:** Allows users to contact library support for assistance.

B. ADMINISTRATOR USE CASES

The Administrator actor represents library staff responsible for managing the system. Their primary actions include:

- **ManageCatalog:** Handles operations such as adding, updating, or deleting books in the catalog.
  - **CRUDUser:** Enables the creation, reading, updating, and deletion of user data.
  - **CRUDBooks:** Manages book data with similar CRUD functionalities.
  - **CategorizeBooks:** Organizes books into different categories.
  - **ManageAndMonitorBookAvailability:** Tracks and updates the availability status of books.
- **ManageBookReservation:** Oversees book reservations and ensures proper handling.
  - **TrackBooks:** Monitors borrowed books and their return statuses.
- **ManageNotifications:** Manages the creation and distribution of system notifications.
  - **SendReminder:** Sends reminders to users regarding upcoming due dates for borrowed books.
- **SupportHelpDesk:** Provides technical and operational support to users.

C. DIAGRAM INTERPRETATION

The Use Case Diagram highlights the responsibilities and interactions of each actor with the system. The inclusion of **include** relationships specifies how larger tasks are composed of smaller subtasks, aiding in understanding task dependencies and enhancing system clarity. This comprehensive structure ensures the system is user-friendly for customers while meeting the operational needs of administrators.The Access Catalog functionality allows users to search for books within the library system. The process begins with the user entering keywords or applying filters in the search bar. The system processes the input, queries the database for relevant results, and displays the matching book details to the user. This ensures that users can quickly and efficiently find books of interest.

For the Make Reservation functionality, both the user and the admin are involved. The user initiates the process by selecting a book and submitting a reservation request. The admin then reviews the request, checks the book's availability, and either approves or rejects the reservation. Upon approval, the system updates the book's status to "reserved" and notifies the user of the successful reservation. This interaction ensures proper control and tracking of book reservations.

The Manage Catalog functionality focuses on the admin's responsibilities in maintaining the library's book catalog. The admin logs into the system and accesses the catalog management interface to add, update, or remove book entries. After performing the desired operations, the system updates the database accordingly and confirms the changes to the admin. This functionality supports the efficient and organized management of the library's resources.

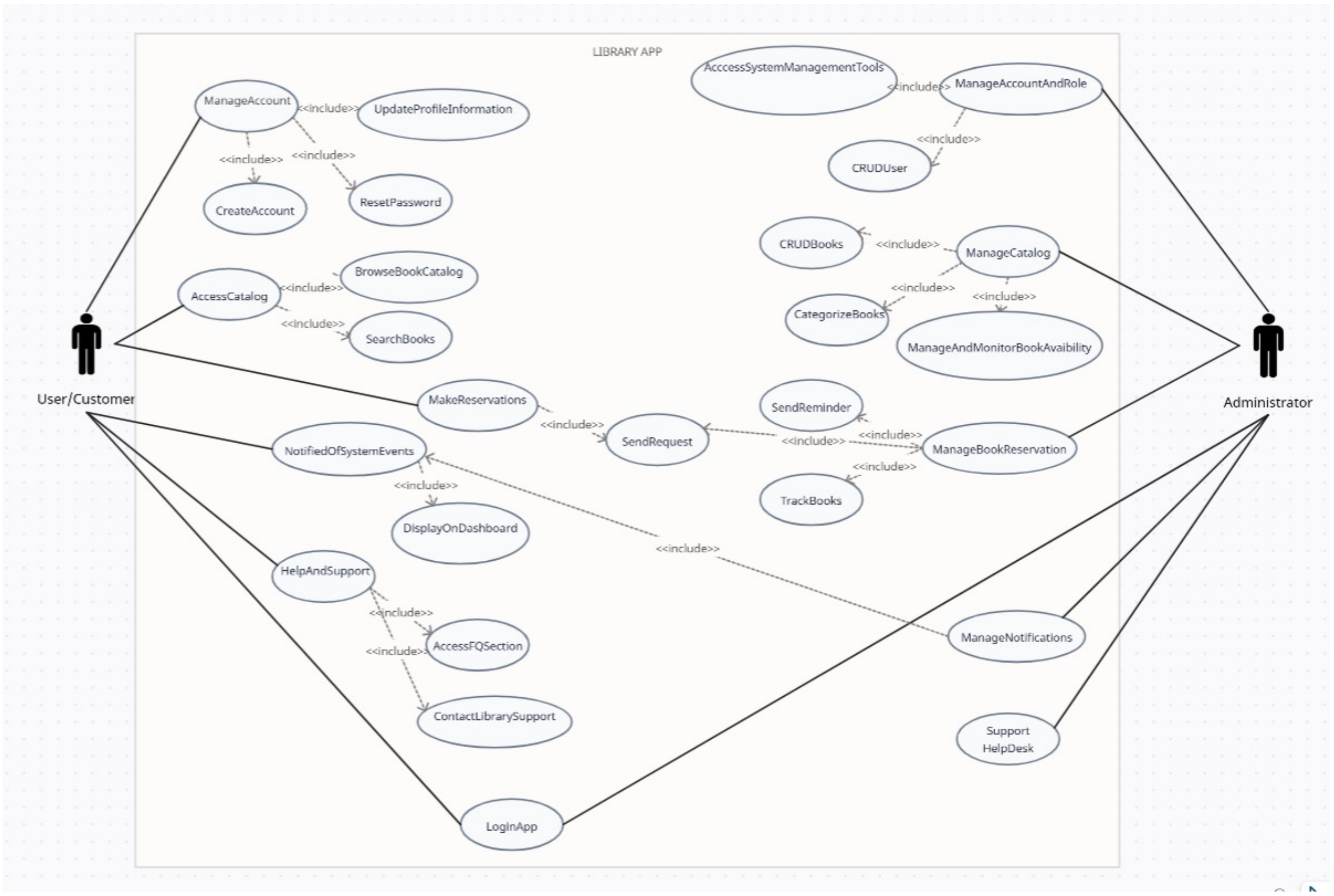


FIGURE 28. Use Case Diagram for the Library Application.

XXIII. DYNAMIC MODEL: SEQUENCE DIAGRAMS

This section presents the sequence diagrams for the three main functionalities of the Library Site system: **Make Reservation**, **Access Catalog**, and **Manage Catalog**. These diagrams highlight the dynamic interactions between actors and the system.

In the sequence diagrams, the **boundary** represents the interface through which users and administrators interact with the system. This boundary typically includes UI components or controllers that act as intermediaries between the actors (User/Admin) and the system's backend.

- **Boundary Component: LibraryApp Controller**
  - Serves as the primary boundary in this context.
  - Acts as the interface handling input from the User and Admin, as well as responses from the System.
  - Facilitates communication between the actors and the core application logic.

A. SEQUENCE DIAGRAM: MAKE RESERVATION

- **Purpose:** Demonstrates the process for reserving a book.
- **Actors Involved:** User, Admin, System.
- **Steps:**
  - User requests a book reservation.
  - Admin verifies the request.
  - The system checks book availability and updates the database.
  - User receives confirmation of the reservation.

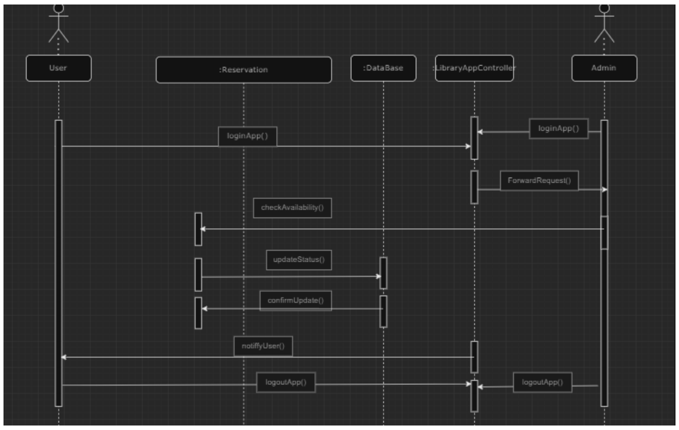


FIGURE 29. Sequence Diagram: Make Reservation.

B. SEQUENCE DIAGRAM: ACCESS CATALOG

- **Purpose:** Outlines the process for users to search and view books in the catalog.
- **Actors Involved:** User, System.
- **Steps:**
  - User enters search criteria.
  - The system processes the query and retrieves matching results from the database.
  - User views the search results and accesses book details.

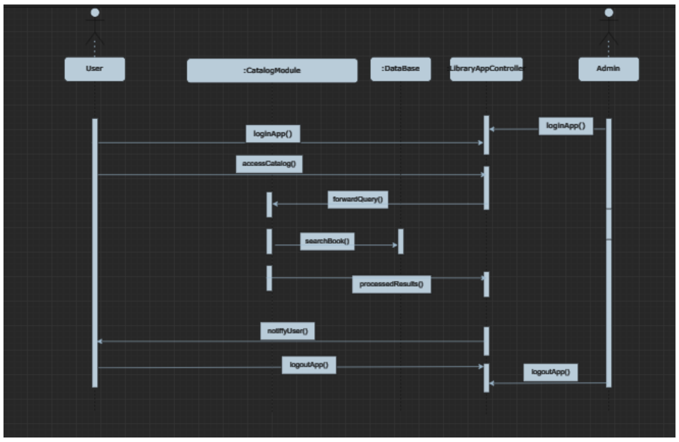


FIGURE 30. Sequence Diagram: Access Catalog.

C. SEQUENCE DIAGRAM: MANAGE CATALOG

- **Purpose:** Details the operations for administrators to manage the library catalog.
- **Actors Involved:** Admin, System.
- **Steps:**
  - Admin adds, updates, or deletes book records.
  - The system updates the database accordingly.
  - Admin receives confirmation of the successful operation.

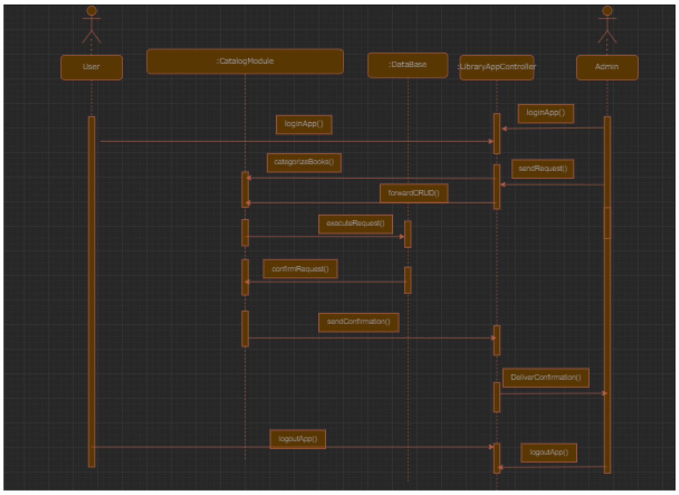


FIGURE 31. "Sequence diagram illustrating the process for administrators to manage the library catalog, including actions such as adding, updating, or deleting book records, and receiving confirmation of successful operations."

XXIV. CONCLUSION

The development of the "Book World" Library Management System emphasized a collaborative and user-focused approach, leveraging embedded systems and structured project management tools. By opting for a fully embedded architecture, the project successfully minimized dependencies on external APIs, ensuring a seamless and efficient integration of functionalities such as catalog access, book reservations, and catalog management.

A rigorous requirements engineering process laid the groundwork for the system's development. Starting with requirements elicitation and analysis, followed by validation and management, the team ensured the solution was tailored to meet both user and administrator needs. Supporting these phases were detailed UML diagrams, including use case and sequence diagrams, which clarified workflows and enhanced implementation precision.

The project was managed using the Scrum framework, complemented by Trello for task tracking and progress monitoring. Trello enabled us to break down tasks into manageable steps, assign responsibilities, and track progress visually, ensuring transparency and alignment across the team. Each day, team members recorded completed, ongoing, and blocked tasks, fostering a collaborative and proactive problem-solving environment.

Daily Scrum meetings, supplemented with Google Docs for updates and discussions, ensured clear communication. GitHub served as the central repository for version control, housing all code and documentation and streamlining collaboration among team members.

The embedded approach proved to be a robust solution, offering reliability, maintainability, and user-friendliness. The "Book World" system successfully addresses core library management needs while providing a solid foundation for future scalability. Challenges such as database synchronization and task allocation were effectively mitigated, reflecting the project's strong emphasis on planning and collaboration.

In conclusion, the "Book World" Library Management System highlights the benefits of integrating embedded systems with agile methodologies and collaborative tools like Trello. It stands as a comprehensive and adaptable solution for modern library operations.

XXV. ENHANCED SYSTEM DESIGN FEATURES

This document provides a concise overview of the software architecture for the **BookWorld Library Management System**. The design aims to create a reliable, user-friendly, and scalable platform to streamline library operations and enhance user experience. By introducing functionalities such as catalog management, user account handling, book reservations, and notification services, the system addresses challenges faced by traditional library setups, including inefficiencies and scalability issues.

The architecture is developed with constraints such as peak usage times, limited system resources, and diverse user needs in mind. Traceability information links architectural decisions to the requirements outlined in the **Requirements Analysis Document (RAD)**, ensuring consistency and alignment with the project's objectives. This report builds upon the analysis and findings presented in the RAD, using them as the foundation for the system's design and implementation.

XXVI. SYSTEM ARCHITECTURE AND SUBSYSTEMS

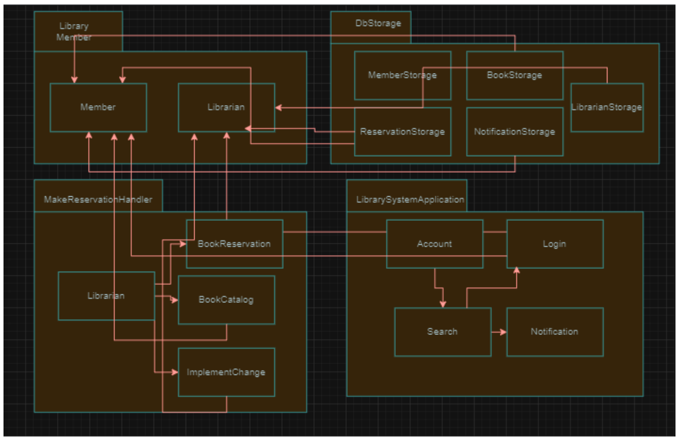


FIGURE 32. Proposed System Architecture for the Library Management System. This diagram illustrates the four primary subsystems

A. CLIENT-SERVER ARCHITECTURE FOR LIBRARY MANAGEMENT SYSTEM

In the Library Management System, a client-server architecture is utilized to efficiently divide responsibilities between client-side and server-side components. The client-side ensures a user-friendly interface for library members and administrators, while the server-side handles business logic, data processing, and security.

1) Client-Side Components

- Users interact with the system through a web interface to search for books, make reservations, and receive notifications about book availability and due dates.
- The interface provides real-time feedback for reservation status and catalog updates, enhancing user experience.

2) Server-Side Components

- The server manages core operations such as handling book reservations, validating user inputs, managing user accounts, and updating the catalog.
- A centralized database stores critical information, including user profiles, book details, reservations, and notifications.
- Security measures, such as user authentication, access control, and data validation, are enforced on the server to ensure data integrity and protection.

3) Advantages of Client-Server Architecture

- **Scalability:** The system can handle an increasing number of concurrent users and transactions by scaling server resources.
- **Centralized Data Management:** Ensures data consistency and integrity across the system.
- **Enhanced Security:** Sensitive operations like user authentication and catalog management are isolated to the server.

- **Streamlined Maintenance:** Centralized logic and database management simplify updates and debugging.

B. SUBSYSTEMS

1) 1. LibraryMember Subsystem

This subsystem represents the roles of library members and librarians interacting with the system.

Library Member:

- **Responsibilities:** Search for books
- **Functionality:** Offers an intuitive interface for members to access library resources and track their interactions.

Librarian:

- **Responsibilities:** Manage the catalog, approve reservations, and monitor system usage.
- **Functionality:** Facilitates administrative operations such as adding books and managing user accounts.

2) 2. LibrarySystemApplication Subsystem

This subsystem manages the primary functionalities of the Library Management System, including user authentication, catalog access, and notifications.

Account:

- **Responsibilities:** Handles user registration, profile updates, and account management.
- **Functionality:** Provides secure access to library features for registered users.

Login:

- **Responsibilities:** Validates user credentials and manages sessions.
- **Functionality:** Ensures secure user authentication.

Search:

- **Responsibilities:** Enables users to search for books in the catalog using various filters.
- **Functionality:** Provides dynamic search results based on title, author, or genre.

Notification:

- **Responsibilities:** Sends notifications for overdue books, reserved book availability, and system updates.
- **Functionality:** Keeps users informed about critical updates and actions.

3) 3. MakeReservationHandler Subsystem

This subsystem focuses on handling book reservations, catalog updates, and user feedback.

BookReservation:

- **Responsibilities:** Manages book availability and reservation status.
- **Functionality:** Allows library admins to reserve books and tracks their borrowing history.

BookCatalog:

- **Responsibilities:** Provides detailed information about available books.
- **Functionality:** Facilitates catalog browsing and search operations for users.

ImplementChange:

- **Responsibilities:** Handles updates to book availability and reservation requests.
- **Functionality:** Ensures real-time updates in the system database.

4) 4. DbStorage Subsystem

The DbStorage subsystem is responsible for securely managing persistent data across the system.

MemberStorage:

- **Responsibilities:** Stores user profiles and preferences.
- **Functionality:** Ensures secure and consistent management of member data.

BookStorage:

- **Responsibilities:** Stores details about books, including availability and metadata.
- **Functionality:** Provides accurate data for catalog browsing and reservations.

ReservationStorage:

- **Responsibilities:** Tracks reservation records, including active and past bookings.
- **Functionality:** Updates reservation status in real-time.

NotificationStorage:

- **Responsibilities:** Stores notification logs for user alerts and updates.
- **Functionality:** Ensures reliable delivery of notifications.

XXVII. HARDWARE-SOFTWARE MAPPING

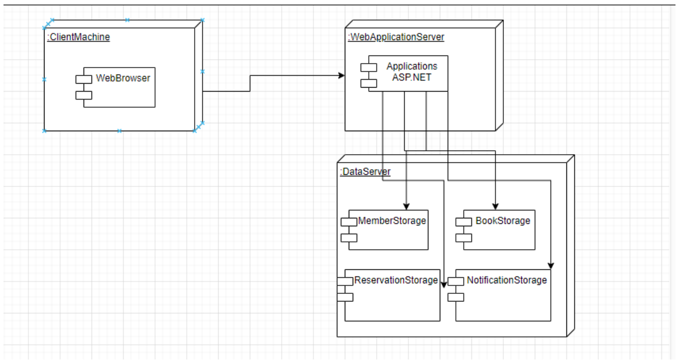


FIGURE 33. Hardware-Software Mapping for the Library Management System. This architecture showcases the interaction between the client machine (user interface) and the server-side subsystems, ensuring seamless operations and data processing.

The Library Management System employs a client-server architecture to distribute responsibilities effectively between hardware and software components. This architecture ensures scalability, reliability, and efficient data processing while maintaining a user-friendly experience for both library members and administrators.

A. CLIENT-SIDE

The **Client Machine** acts as the primary interface for users, including library members and librarians. The client-side software includes the web-based user interface developed using technologies like *HTML, CSS, and JavaScript (Razor with ASP.NET)*. Through this interface, users can perform operations such as:

- Searching the catalog.
- Managing reservations (by librarians).
- Receiving notifications about reserved book availability or overdue items.

The client machine hardware facilitates smooth interaction between users and the system via modern web browsers, ensuring accessibility and ease of use.

B. SERVER-SIDE

The **LibrarySystemApplication Server** handles core business logic, request processing, and system management. It consists of several critical subsystems:

- **MakeReservationHandler:** Manages book reservations, catalog updates, and librarian-initiated actions in real-time.
- **LibraryMember Subsystem:** Handles user roles, including library members and librarians, managing authentication and account interactions.
- **DbStorage Subsystem:** Manages persistent data storage securely. This subsystem includes:
  - **MemberStorage:** Stores user profiles and account details.
  - **BookStorage:** Maintains data about books, including availability and metadata.
  - **ReservationStorage:** Tracks all reservation records and their statuses.
  - **NotificationStorage:** Logs notifications for user alerts.

C. ADVANTAGES OF THE HARDWARE-SOFTWARE MAPPING

This mapping ensures:

- **Scalability:** The system can handle an increasing number of users and operations efficiently.
- **Data Security:** Sensitive data, such as user profiles and reservations, is managed centrally with secure access protocols.
- **Reliability:** The division of responsibilities across hardware and software ensures seamless operation.
- **User Accessibility:** Library members and librarians can access the system through modern web browsers without requiring additional software installations.

XXVIII. PERSISTENT DATA MANAGEMENT

In the **Library Management System**, persistent data management ensures efficient and secure storage of critical system information, enabling smooth operations for library members and administrators. The system employs a relational database to handle structured data, ensuring data consistency, scalability, and integrity across all subsystems.

1) LibraryMember Subsystem

The **LibraryMember Subsystem** securely stores user details such as names, emails, passwords, and account preferences in the database. This data is encapsulated within the *MemberStorage* table, enabling quick retrieval and management of member profiles. For librarians, additional attributes such as permissions, librarian IDs, and assigned tasks are managed via the *LibrarianStorage* table.

2) MakeReservationHandler Subsystem

The **MakeReservationHandler Subsystem** interacts with the database to manage real-time book reservations and catalog updates. The *ReservationStorage* table stores reservation details, including book IDs, reservation status, and member-librarian associations. The *NotificationStorage* table records system notifications, such as reserved book availability, overdue reminders, and administrative updates. This subsystem ensures that persistent data related to book reservations, catalog changes, and notifications is consistently updated and readily available for system use.

3) DbStorage Subsystem

The **DbStorage Subsystem** serves as the central repository for all system data. The relational database structure includes key tables such as:

- *MemberStorage*: Manages user profiles and account preferences.
- *BookStorage*: Stores book details, including titles, authors, genres, ISBNs, and availability status.
- *ReservationStorage*: Tracks active and completed book reservations, including reservation timestamps and statuses.
- *NotificationStorage*: Logs notifications sent to users and librarians regarding important events or updates.

These tables are designed to efficiently manage relationships between entities, such as members, librarians, books, and reservations.

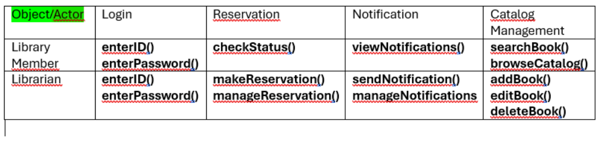
4) LibrarySystemApplication Subsystem

The **LibrarySystemApplication Subsystem** delivers user interactions and system responses by querying the database through *DbStorage*. Real-time data retrieval enables dynamic feedback to users, such as reservation confirmations, catalog search results, and system notifications. This connection ensures that all user actions are logged, processed, and reflected accurately in the database.

5) Summary

In summary, the persistent data management system in the **Library Management System** utilizes a robust relational database to centralize and secure critical information. Data encapsulation is achieved through well-structured tables, allowing the **LibraryMember**, **MakeReservationHandler**, and **LibrarySystemApplication** subsystems to interact seamlessly with the *DbStorage*. This architecture supports simultaneous access, real-time updates, and accurate feedback, ensuring a smooth and efficient user experience.

XXIX. ACCESS CONTROL AND SECURITY



Object/Actor	Login	Reservation	Notification	Catalog Management
Library Member	**enterID() enterPassword()**	**checkStatus()**	**viewNotifications()**	**searchBook() browseCatalog()**
Librarian	**enterID() enterPassword()**	**makeReservation() manageReservation()**	**sendNotification() manageNotifications**	**addBook() editBook() deleteBook()**

FIGURE 34. "Access Control Matrix: This matrix illustrates the specific actions that Library Members and Librarians are authorized to perform across various system functionalities, including Login, Reservation, Notification, and Catalog Management. It defines role-based permissions to ensure secure and efficient interaction with the library system."

The access control matrix establishes the permissions for two main actors—Library Member and Librarian—defining their interactions with critical system objects such as Login, Reservation, Notification, and Catalog Management.

- **Library Members**: Library members authenticate securely using `enterID()` and `enterPassword()` for login, ensuring protected access to the system. They are authorized to perform actions such as `checkStatus()` for viewing reservation statuses, `viewNotifications()` for system updates, and operations like `searchBook()` and `browseCatalog()` to explore library resources.
- **Librarians**: Librarians authenticate using `enterID()` and `enterPassword()` and hold broader responsibilities. They can execute operations like `makeReservation()` for reserving books on behalf of members and `manageReservation()` to oversee and adjust reservations. In catalog management, librarians are empowered to add, edit, and delete books via operations like `addBook()`, `editBook()`, and `deleteBook()`. Additionally, they manage notifications through `sendNotification()` and `manageNotifications()` to keep users informed.

This structured approach to access control ensures that each actor can only access operations relevant to their roles, reducing the risk of unauthorized actions. Combined with encryption mechanisms and role-based access control (RBAC), this framework strengthens the system's security. Regular system monitoring, access audits, and secure authentication further enhance the protection of user data and ensure reliable interactions between users and the system.

XXX. BOUNDARY CONDITIONS

The boundary conditions of the Library Management System define the limits and constraints under which the system

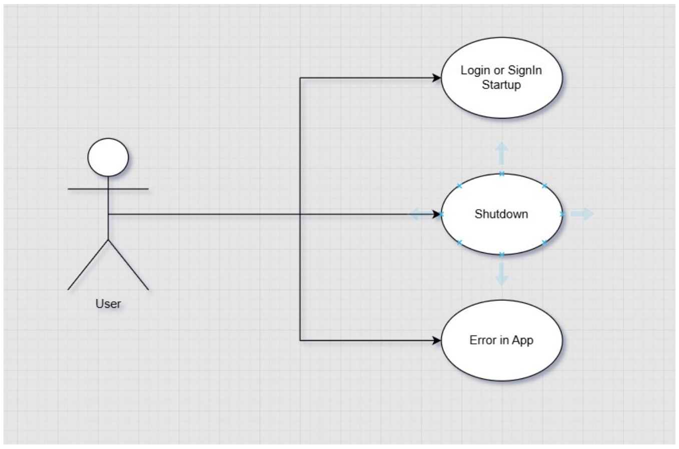


FIGURE 35. "Boundary Conditions for User Interaction - This diagram illustrates the boundary conditions for user interactions in the system, including login/sign-in during startup, app shutdown, and handling errors. It defines the critical states a user can encounter while interacting with the system, ensuring robust management of application lifecycle events."

operates. These conditions ensure seamless interaction between users and the system while maintaining data integrity, security, and efficiency.

A. SYSTEM INTERACTION BOUNDARIES

The Library Management System interfaces with its users (Library Members and Librarians) through a web-based application accessible via modern browsers. The interaction boundaries include:

- **Library Member Access:** Members can interact with functionalities such as login, reservation, notification, and catalog search.
- **Librarian Access:** Librarians can perform administrative tasks such as managing reservations, notifications, and the catalog.
- **Device Compatibility:** The system is designed to function across multiple devices, ensuring accessibility on desktops, tablets, and mobile devices.

B. DATA HANDLING BOUNDARIES

- **Data Storage:** The system utilizes a relational database for storing user data, book catalogs, reservations, and notification logs. Data is encapsulated and managed securely to prevent unauthorized access.
- **Reservation Constraints:** Users can only reserve available books, and reservation limits are applied based on system-defined policies.
- **Data Consistency:** Updates to the catalog or reservations are reflected across the system in real-time to ensure accurate information.

C. SECURITY BOUNDARIES

- **Role-Based Access Control (RBAC):** Permissions are strictly defined for Library Members and Librarians to prevent unauthorized access to sensitive functionalities.

- **Authentication:** Secure login mechanisms with encrypted passwords are implemented to safeguard user accounts.
- **Session Management:** User sessions are automatically terminated after a period of inactivity to enhance security.

D. SYSTEM PERFORMANCE BOUNDARIES

- **Scalability:** The system can handle up to 100 concurrent users without performance degradation.
- **Response Time:** The system ensures a maximum response time of 2 seconds for common operations such as login, search, and reservation.
- **Availability:** The system guarantees a 99.9% uptime, minimizing disruptions to user operations.

E. OPERATIONAL CONSTRAINTS

- **Maintenance Windows:** Scheduled maintenance will occur during off-peak hours, with prior notifications sent to users.
- **Language Support:** The system operates in a single default language, with potential for future multilingual expansion.
- **System Updates:** Feature updates or changes are implemented following stakeholder approval and are tested rigorously before deployment.

These boundary conditions ensure that the Library Management System operates within defined constraints, providing a secure, efficient, and user-friendly environment for managing library operations.

XXXI. SUBSYSTEM SERVICES

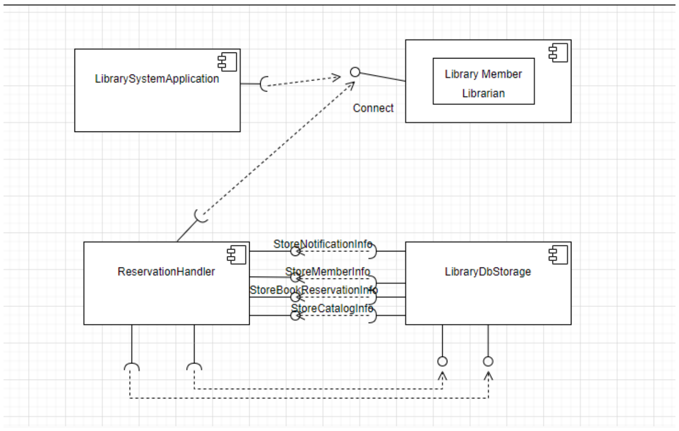


FIGURE 36. The diagram illustrates the Library Management System's Subsystem Services, showcasing the interactions between core components: the LibrarySystemApplication, Library Member and Librarian, ReservationHandler, and LibraryDbStorage. It highlights the data flow and service responsibilities across the system.

In our system architecture, the "Library Member and Librarian" subsystem plays a critical role in managing user interactions and administrative operations. It establishes communication between the "LibrarySystemApplication" and the "ReservationHandler" subsystems via the essential "Connect" service. This connection ensures user authentication, profile management, and seamless interaction between users and system functionalities.

The "ReservationHandler" subsystem enhances system operations by providing key services, including:

- **StoreNotificationInfo**: Logs and organizes notifications for members and librarians.
- **StoreMemberInfo**: Maintains secure storage of library member profiles and related data.
- **StoreBookReservationInfo**: Manages book reservation requests and updates availability in real time.
- **StoreCatalogInfo**: Tracks and updates the library catalog with detailed metadata.

This interconnected relationship between subsystems ensures a robust and efficient architecture, enabling real-time data flow and a seamless user experience. The "ReservationHandler" subsystem acts as a core component for reservation processing, catalog management, and notification delivery, while the "LibraryDbStorage" subsystem ensures secure and persistent data storage, supporting the system's scalability and reliability goals.

The accompanying diagram illustrates these interactions, showcasing the flow of data and responsibilities among the primary subsystems.

XXXII. TESTING PHASE

The testing phase is one of the most critical stages of the software development lifecycle. This phase ensures the accuracy, stability, and reliability of the developed system. The objective of the testing process is to detect potential errors early, enhance software quality, and meet user requirements effectively. During the testing phase, various types of tests were conducted to thoroughly evaluate the functionality of the system. Below is a summary of these tests and their applications:

A. UNIT TESTING

The smallest units of code (methods and classes) were tested using the xUnit framework. For instance, calculation functions were verified independently.

B. UI TESTING

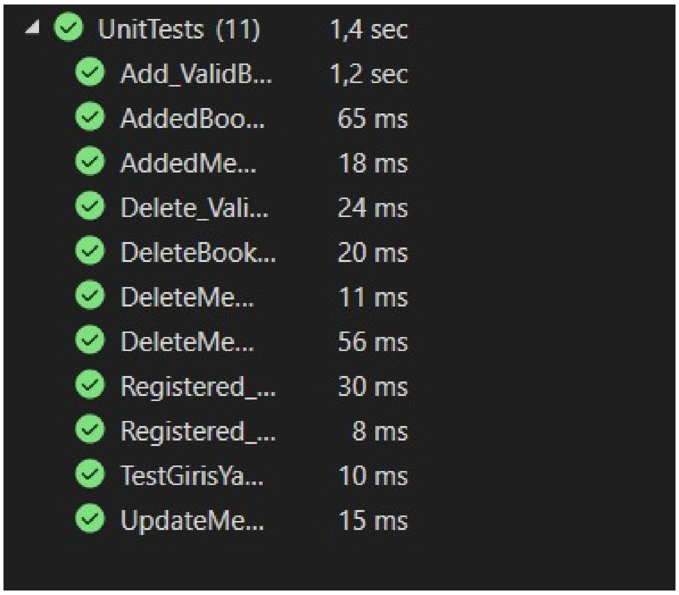
The user interface was validated using Selenium to ensure correct functionality. For example, the login form was tested in the browser.

C. FUNCTIONAL TESTING

The correct functioning of the application features was verified through manual testing. For instance, a file upload operation was successfully completed.

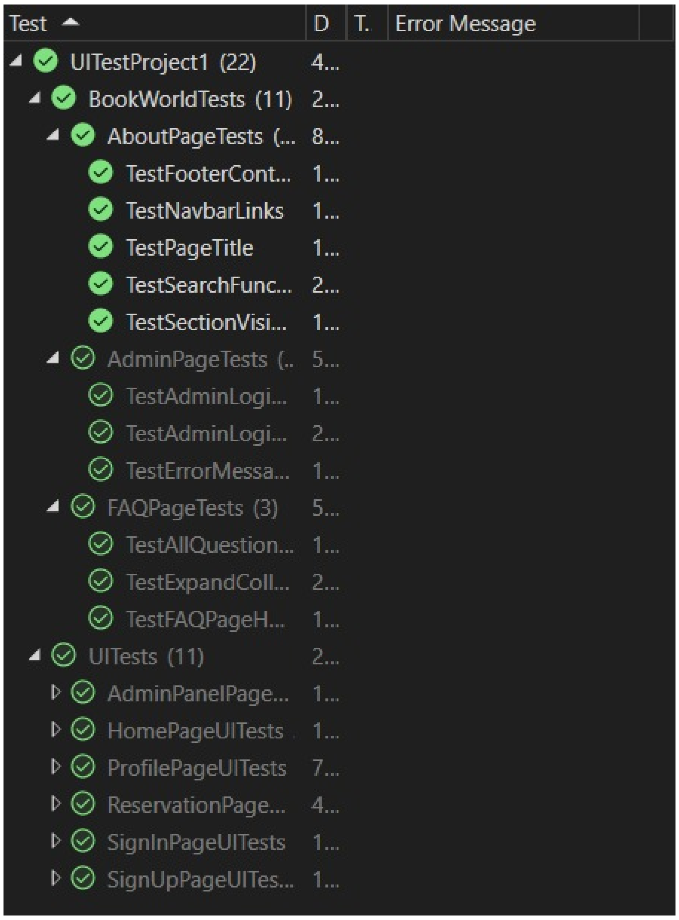
D. PERFORMANCE TESTING

Performance tests were conducted to measure the system's response time and behavior under load. Below are the details and results for each tested page:



```
◢ ✓ UnitTests (11)     1,4 sec
    ✓ Add_ValidB...    1,2 sec
    ✓ AddedBoo...       65 ms
    ✓ AddedMe...        18 ms
    ✓ Delete_Vali...    24 ms
    ✓ DeleteBook...     20 ms
    ✓ DeleteMe...       11 ms
    ✓ DeleteMe...       56 ms
    ✓ Registered_...    30 ms
    ✓ Registered_...     8 ms
    ✓ TestGirisYa...    10 ms
    ✓ UpdateMe...       15 ms
```

FIGURE 37. Unit Testing Results: Example of a unit test validating calculation functions.



```
Test ▲                        D    T.   Error Message
◢ ✓ UITestProject1 (22)       4...
  ◢ ✓ BookWorldTests (11)     2...
    ◢ ✓ AboutPageTests (...   8...
        ✓ TestFooterCont...   1...
        ✓ TestNavbarLinks     1...
        ✓ TestPageTitle       1...
        ✓ TestSearchFunc...   2...
        ✓ TestSectionVisi...  1...
    ◢ ✓ AdminPageTests (..    5...
        ✓ TestAdminLogi...    1...
        ✓ TestAdminLogi...    2...
        ✓ TestErrorMessa...   1...
    ◢ ✓ FAQPageTests (3)      5...
        ✓ TestAllQuestion...  1...
        ✓ TestExpandColl...   2...
        ✓ TestFAQPageH...     1...
  ◢ ✓ UITests (11)            2...
    ▷ ✓ AdminPanelPage...     1...
    ▷ ✓ HomePageUITests       1...
    ▷ ✓ ProfilePageUITests    7...
    ▷ ✓ ReservationPage...    4...
    ▷ ✓ SignInPageUITests     1...
    ▷ ✓ SignUpPageUITes...    1...
```

FIGURE 38. UI Testing Results: Selenium testing of the login form functionality.

HOME PAGE:

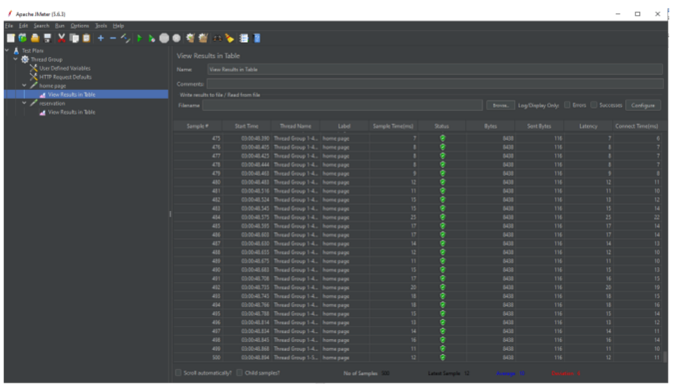


FIGURE 39. Performance Testing of HomePage Results

1) Test Configuration

- **Number of Threads:** 500 (simultaneous users accessing the application)
- **Ramp-up Time:** 10 seconds (total time for all users to access the application)
- **Loop Count:** 1 (number of times each user performs the operation)
- **Protocol:** HTTPS
- **Server Name or IP:** localhost
- **Port Number:** 7006

2) Home Page Performance

- **Number of Samples:** 500 (total requests sent)
- **Average Response Time:** 10 ms
- **Deviation:** 6 ms
- **Status:** 100% success

Analysis and Results:

- **Average Response Time:** The response time is 10 ms, which is considered fast (the lower, the better).
- **Deviation:** The deviation is 6 ms, indicating that response times are consistent (lower values mean more consistent responses).
- **Status:** All requests were successful.

3) Reservation Page Performance

- **Number of Samples:** 500 (total requests sent)
- **Average Response Time:** 9 ms
- **Deviation:** 10 ms
- **Status:** 100% success

Analysis and Results:

- **Average Response Time:** The response time is 9 ms, which is considered fast (the lower, the better).
- **Deviation:** The deviation is 10 ms, indicating that response times are fairly consistent (lower values mean more consistent responses).
- **Status:** All requests were successful.

4) Sign-In Performance

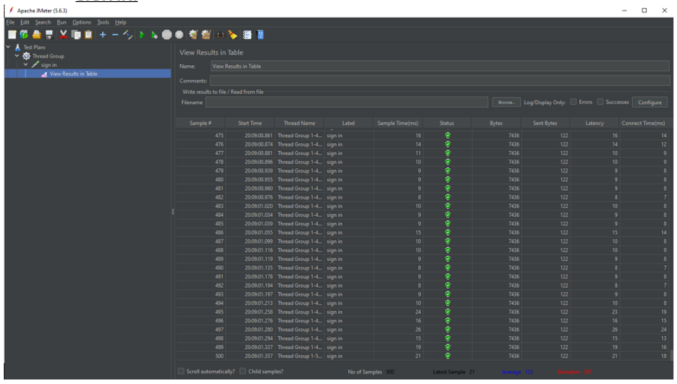
- **Number of Samples:** 500 (total requests sent)

- **Average Response Time:** 123 ms
- **Deviation:** 287 ms
- **Status:** 100% success

Analysis and Results:

- **Average Response Time:** The response time is 123 ms, which is relatively fast (the lower, the better).
- **Deviation:** The deviation is 287 ms, showing inconsistencies in response times (lower values indicate better consistency).
- **Status:** All requests were successful.

SIGN IN:



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename

Browse... Log/Display Only: Errors Successes Configure

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
475	20:09:00.861	Thread Group 1-4...	sign in	16	✓	7436	122	16	14
476	20:09:00.874	Thread Group 1-4...	sign in	14	✓	7436	122	14	12
477	20:09:00.881	Thread Group 1-4...	sign in	11	✓	7436	122	10	9
478	20:09:00.896	Thread Group 1-4...	sign in	10	✓	7436	122	10	9
479	20:09:00.939	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
480	20:09:00.955	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
481	20:09:00.960	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
482	20:09:00.976	Thread Group 1-4...	sign in	8	✓	7436	122	8	7
483	20:09:01.020	Thread Group 1-4...	sign in	10	✓	7436	122	10	8
484	20:09:01.034	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
485	20:09:01.039	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
486	20:09:01.055	Thread Group 1-4...	sign in	15	✓	7436	122	15	14
487	20:09:01.099	Thread Group 1-4...	sign in	10	✓	7436	122	10	8
488	20:09:01.116	Thread Group 1-4...	sign in	10	✓	7436	122	10	9
489	20:09:01.119	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
490	20:09:01.135	Thread Group 1-4...	sign in	8	✓	7436	122	8	7
491	20:09:01.178	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
492	20:09:01.194	Thread Group 1-4...	sign in	8	✓	7436	122	8	7
493	20:09:01.197	Thread Group 1-4...	sign in	9	✓	7436	122	9	8
494	20:09:01.213	Thread Group 1-4...	sign in	10	✓	7436	122	10	8
495	20:09:01.258	Thread Group 1-4...	sign in	24	✓	7436	122	23	19
496	20:09:01.276	Thread Group 1-4...	sign in	16	✓	7436	122	16	15
497	20:09:01.280	Thread Group 1-4...	sign in	26	✓	7436	122	26	24
498	20:09:01.294	Thread Group 1-4...	sign in	15	✓	7436	122	15	13
499	20:09:01.337	Thread Group 1-4...	sign in	19	✓	7436	122	19	16
500	20:09:01.357	Thread Group 1-5...	sign in	21	✓	7436	122	21	18

Scroll automatically? Child samples? No of Samples 500 Latest Sample 21 Average 123 Deviation 287

FIGURE 40. Performance Testing Results: System response analysis under 500 concurrent users using JMeter.

E. SECURITY TESTING

The security testing phase was conducted using the OWASP ZAP tool to identify potential vulnerabilities. This tool categorized alerts into four levels:

- **Informational Priority Alert:** General information.
- **Low Priority Alert:** Suggestions for minor improvements.
- **Medium Priority Alert:** Important issues with moderate urgency (e.g., XSS).
- **High Priority Alert:** Critical issues requiring immediate attention (e.g., SQL Injection).

Our project generated a total of 17 alerts: 3 Medium, 5 Low, and 9 Informational. Below are the Medium and Low priority alerts, their explanations, and suggested solutions.

1) Medium Priority Alerts

a: 1. Content Security Policy (CSP) Header Not Set

Description: The CSP header specifies which sources are allowed to load content in the application. Its absence can enable attackers to inject malicious JavaScript or perform XSS attacks.

Solution: Add a Content-Security-Policy header in the application middleware to restrict sources to trusted ones. This can be done by configuring the application to include a directive such as "`default-src 'self'`".

b: 2. Cross-Domain Misconfiguration

Description: The server's CORS settings allow unrestricted access (`Access-Control-Allow-Origin: *`), potentially exposing sensitive data to third-party domains.

Solution: Update the CORS configuration to allow access only from trusted domains by specifying the allowed origins and restricting methods and headers to secure values.

c: 3. Missing Anti-Clickjacking Header

Description: Without the `X-Frame-Options` header, the application is vulnerable to clickjacking attacks, where malicious sites embed the application in an iframe.

Solution: Add an `X-Frame-Options` header with the value "`SAMEORIGIN`" to prevent the application from being loaded in an iframe on external websites.

2) Low Priority Alerts

a: 1. Cookie Without Secure Flag

Description: Cookies without the Secure flag can be transmitted over unencrypted HTTP connections, exposing them to MITM attacks.

Solution: Ensure that all cookies include the Secure and HttpOnly flags to prevent their transmission over unencrypted connections and to protect them from being accessed via client-side scripts.

b: 2. Cross-Domain JavaScript Source File Inclusion

Description: The application includes JavaScript files from untrusted third-party sources (e.g., https://cdn.jsdelivr.net/), which can host malicious content.

Solution: Host JavaScript files locally on the application server instead of relying on external sources to ensure their integrity and security.

c: 3. Server Leaks Version Information via "Server" HTTP Response Header

Description: The "Server" header reveals the server type and version, making it easier for attackers to exploit known vulnerabilities.

Solution: Remove or obscure the "Server" header to avoid disclosing unnecessary information about the server.

d: 4. Strict-Transport-Security (HSTS) Header Not Set

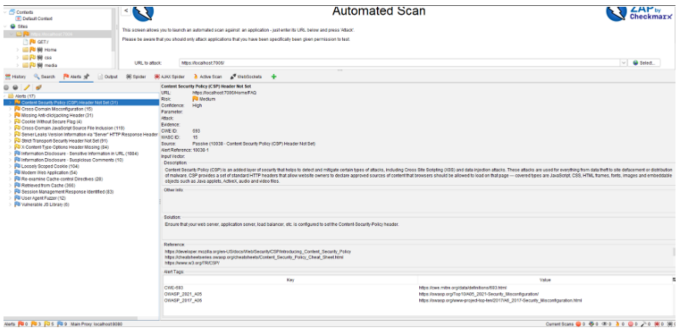
Description: Without HSTS, browsers may use HTTP instead of HTTPS, exposing connections to potential MITM attacks.

Solution: Enable HSTS in the server configuration to enforce HTTPS connections and prevent the use of unencrypted HTTP.

e: 5. X-Content-Type-Options Header Missing

Description: The absence of this header allows browsers to MIME-sniff and misinterpret content types, potentially leading to security risks.

Solution: Add the `X-Content-Type-Options: nosniff` header in the server configuration to prevent MIME-sniffing.



This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: https://localhost:7006/

Alerts (17)
- Content Security Policy (CSP) Header Not Set (31)
- Cross-Domain Misconfiguration (15)
- Missing Anti-clickjacking Header (31)
- Cookie Without Secure Flag (4)
- Cross-Domain JavaScript Source File Inclusion (119)
- Server Leaks Version Information via "Server" HTTP Response Header
- Strict-Transport-Security Header Not Set (31)
- X-Content-Type-Options Header Missing (84)
- Information Disclosure - Sensitive Information in URL (1884)
- Information Disclosure - Suspicious Comments (10)
- Loosely Scoped Cookie (104)
- Modern Web Application (54)
- Re-examine Cache-control Directives (28)
- Retrieved from Cache (366)
- Session Management Response Identified (83)
- User Agent Fuzzer (12)
- Vulnerable JS Library (6)

Content Security Policy (CSP) Header Not Set
URL: https://localhost:7006/Home/FAQ
Risk: Medium
Confidence: High
Parameter:
Attack:
Evidence:
CWE ID: 693
WASC ID: 15
Source: Passive (10038 - Content Security Policy (CSP) Header Not Set)
Alert Reference: 10038-1
Input Vector:

Description:
Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Other Info:

Solution:
Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Reference:
https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://www.w3.org/TR/CSP/

Alert Tags:

Key	Value
CWE-693	https://cwe.mitre.org/data/definitions/693.html
OWASP_2021_A05	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
OWASP_2017_A06	https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html

Alerts 0 3 5 9 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0

FIGURE 41. Security Testing Results: Analysis of vulnerabilities such as SQL Injection using OWASP ZAP.

F. ACCESSIBILITY TESTING WITH AXE-DEVTOOLS

The results from the accessibility testing are categorized based on the severity of their impact on user experience:

- **Critical:** Issues that completely prevent users from using the page.
- **Serious:** Issues that make the user experience difficult but not impossible.
- **Moderate:** Issues that could be improved but do not negatively affect users.
- **Minor:** Minor details or suggestions for improvement.

Below are the detailed results and solutions for each page.

1) Home Page

a: 1. Text Contrast Issue

The text inside the `<p>` tag does not have sufficient contrast with the background color. This makes it harder for users to read the text.
Solution: Increase the contrast by either darkening the background color (e.g., `background-color: #0056b3`) or changing the text color to a lighter shade (e.g., `color: #E0E0E0`).

b: 2. Navbar Dropdown Icon

The icon does not include a description, making it unclear for users.
Solution: Add an `aria-label` or `title` attribute to the icon to provide descriptive text.

c: 3. FAQ Icon

The FAQ icon lacks a description, which may confuse users about its purpose.
Solution: Use an `aria-label` attribute to provide a clear description.

2) Sign-Up Page

a: 1. Navbar Drop Link

The link does not have descriptive text for screen readers and only includes an icon.
Solution: Add an `aria-label` attribute to explain the link's function.

b: 2. FAQ Link

Similar to the Navbar Drop Link, the FAQ link also lacks descriptive text.
Solution: Add an `aria-label` or `title` attribute to provide clarity.

3) Sign-In Page

The same issues identified on the Sign-Up Page were also present on the Sign-In Page. The solutions are identical.

4) Reservation Page

a: Navbar Issue

The navbar dropdown does not have descriptive text, making it inaccessible for screen readers.
Solution: Add an `aria-label` attribute to explain the purpose of the dropdown.

5) Profile Page

a: 1. Edit Profile Name Button

The button is unclear for screen readers, making it difficult for users to understand its purpose.
Solution: Add an `aria-label="Edit Profile Name"` attribute to describe the button's function.

b: 2. Edit Email Address Button

The button lacks descriptive text for screen readers.
Solution: Add an `aria-label="Edit Email Address"` attribute to the button.

c: 3. Edit Password Button

The button does not provide enough information for screen readers to convey its purpose.
Solution: Add an `aria-label="Edit Password"` attribute.

d: 4. Edit Phone Number Button

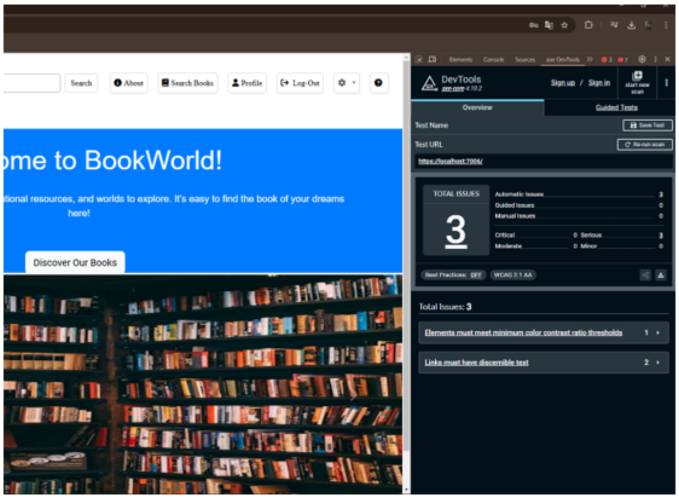
The button is unclear for screen readers, similar to other profile buttons.
Solution: Add an `aria-label="Edit Phone Number"` attribute.

e: 5. Navbar Dropdown

The navbar dropdown lacks descriptive text for screen readers.
Solution: Add an `aria-label="Settings"` attribute to clarify its purpose.

G. DATABASE TESTING

The accuracy of database operations was validated using Entity Framework In-Memory testing. CRUD operations and data consistency were successfully tested.



Test Name
Test URL
https://localhost:7006/

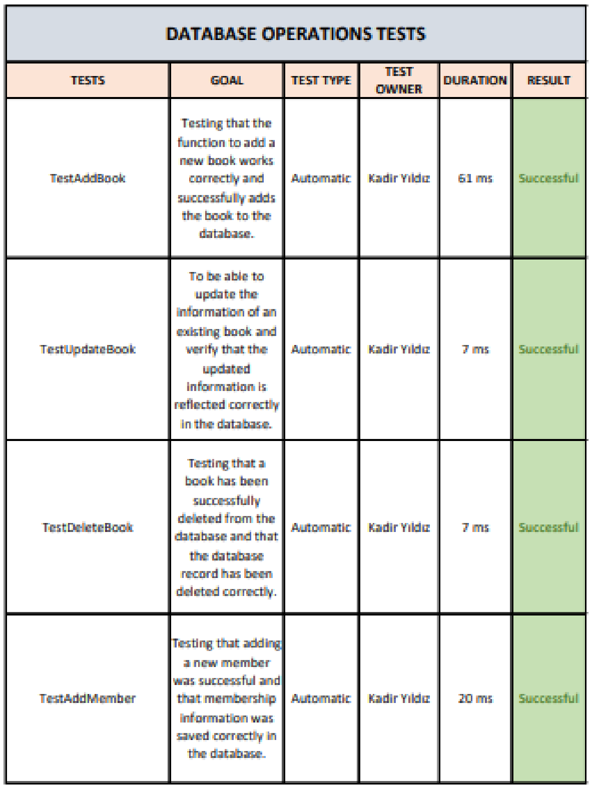
TOTAL ISSUES	
3	
Automatic Issues	3
Guided Issues	0
Manual Issues	0
Critical	0
Serious	3
Moderate	0
Minor	0

Best Practices: OFF WCAG 2.1 AA

Total Issues: 3

Elements must meet minimum color contrast ratio thresholds	1
Links must have discernible text	2

FIGURE 42. Accessibility Axe-Devtools: Testing Results: Verification of keyboard accessibility using axe-core and WAVE tools.

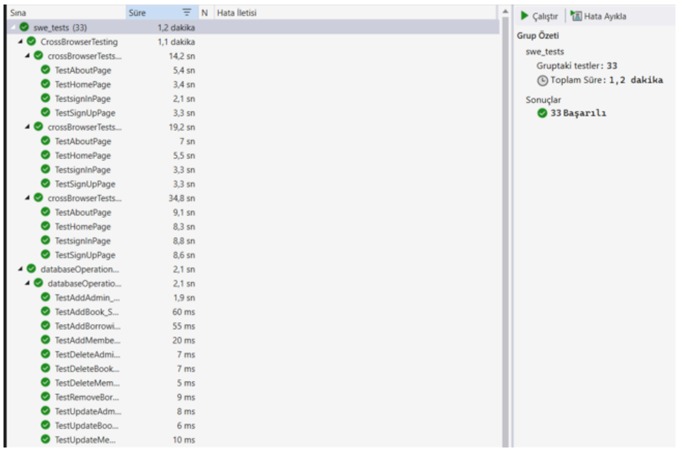


DATABASE OPERATIONS TESTS					
TESTS	GOAL	TEST TYPE	TEST OWNER	DURATION	RESULT
TestAddBook	Testing that the function to add a new book works correctly and successfully adds the book to the database.	Automatic	Kadir Yildiz	61 ms	Successful
TestUpdateBook	To be able to update the information of an existing book and verify that the updated information is reflected correctly in the database.	Automatic	Kadir Yildiz	7 ms	Successful
TestDeleteBook	Testing that a book has been successfully deleted from the database and that the database record has been deleted correctly.	Automatic	Kadir Yildiz	7 ms	Successful
TestAddMember	Testing that adding a new member was successful and that membership information was saved correctly in the database.	Automatic	Kadir Yildiz	20 ms	Successful

FIGURE 43. Database Testing Results: Successful testing of CRUD operations and data consistency using Entity Framework In-Memory.

H. CROSS-BROWSER TESTING

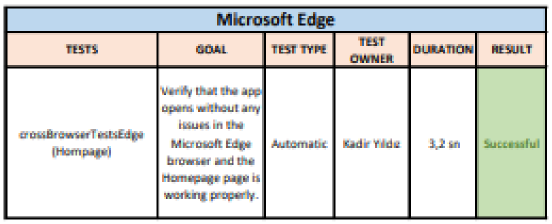
Cross-browser testing was conducted to ensure the application functions consistently across multiple web browsers. This process verified that key features such as the Home page, About page, Sign-in page, and Sign-up page work seamlessly without issues, regardless of the browser.

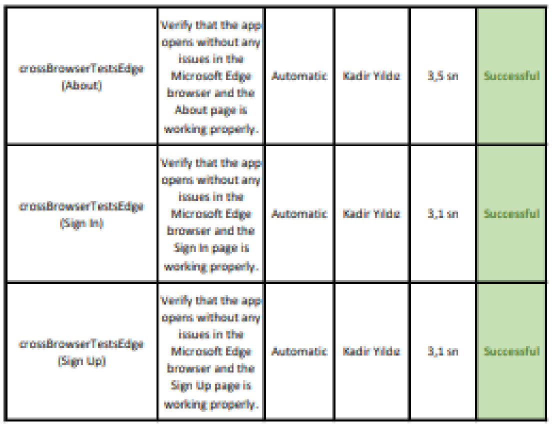


Sına	Süre	N	Hata İletisi
swe_tests (33)	1,2 dakika		
CrossBrowserTesting	1,1 dakika		
crossBrowserTests...	14,2 sn		
TestAboutPage	5,4 sn		
TestHomePage	3,4 sn		
TestsignInPage	2,1 sn		
TestSignUpPage	3,3 sn		
crossBrowserTests...	19,2 sn		
TestAboutPage	7 sn		
TestHomePage	5,5 sn		
TestsignInPage	3,3 sn		
TestSignUpPage	3,3 sn		
crossBrowserTests...	34,8 sn		
TestAboutPage	9,1 sn		
TestHomePage	8,3 sn		
TestsignInPage	8,8 sn		
TestSignUpPage	8,6 sn		
databaseOperation...	2,1 sn		
databaseOperatio...	2,1 sn		
TestAddAdmin_...	1,9 sn		
TestAddBook_S...	60 ms		
TestAddBorrowi...	55 ms		
TestAddMembe...	20 ms		
TestDeleteAdmi...	7 ms		
TestDeleteBook...	7 ms		
TestDeleteMem...	5 ms		
TestRemoveBor...	9 ms		
TestUpdateAdm...	8 ms		
TestUpdateBoo...	6 ms		
TestUpdateMe...	10 ms		

Grup Özeti
swe_tests
Gruptaki testler: 33
Toplam Süre: 1,2 dakika
Sonuçlar
33 Başarılı

FIGURE 44. Cross-Browser Testing Results.



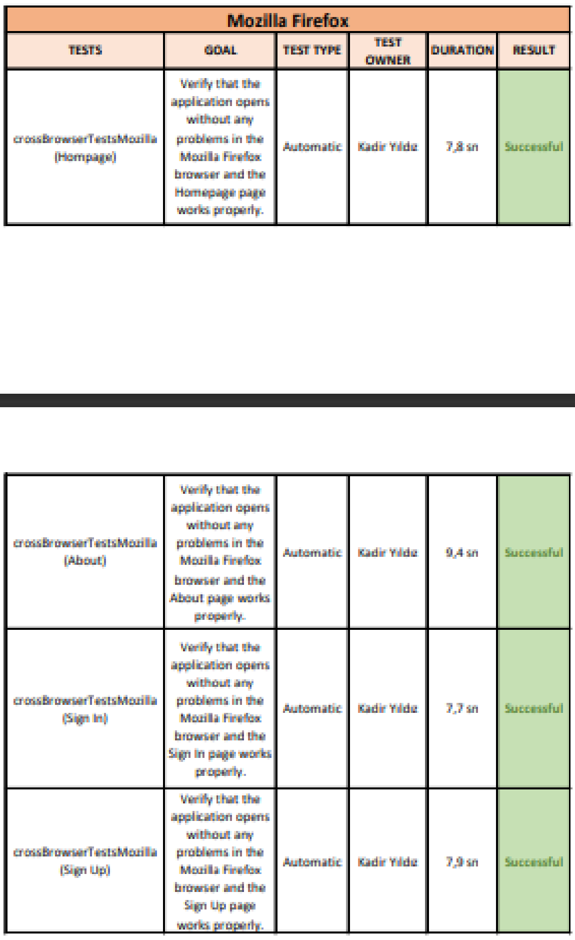


Microsoft Edge					
TESTS	GOAL	TEST TYPE	TEST OWNER	DURATION	RESULT
crossBrowserTestsEdge (Hompage)	Verify that the app opens without any issues in the Microsoft Edge browser and the Homepage page is working properly.	Automatic	Kadir Yildiz	3,2 sn	Successful
crossBrowserTestsEdge (About)	Verify that the app opens without any issues in the Microsoft Edge browser and the About page is working properly.	Automatic	Kadir Yildiz	3,5 sn	Successful
crossBrowserTestsEdge (Sign In)	Verify that the app opens without any issues in the Microsoft Edge browser and the Sign In page is working properly.	Automatic	Kadir Yildiz	3,1 sn	Successful
crossBrowserTestsEdge (Sign Up)	Verify that the app opens without any issues in the Microsoft Edge browser and the Sign Up page is working properly.	Automatic	Kadir Yildiz	3,1 sn	Successful

FIGURE 45. Cross-Browser Testing Results: Microsoft Edge.

XXXIII. FINAL REMARKS AND DOCUMENTATION

The final part of this report includes the documentation of the daily Scrum meetings and workflow diagrams, which



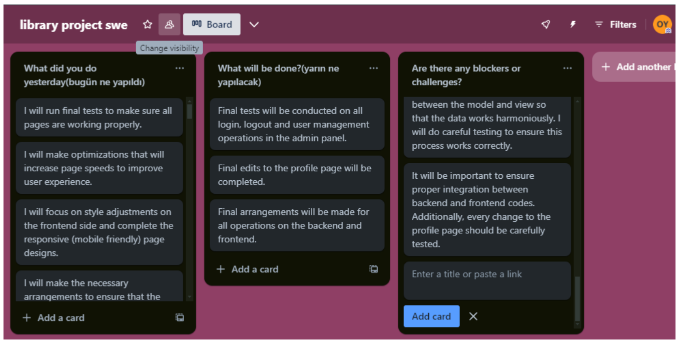
Mozilla Firefox					
TESTS	GOAL	TEST TYPE	TEST OWNER	DURATION	RESULT
crossBrowserTestsMozilla (Hompage)	Verify that the application opens without any problems in the Mozilla Firefox browser and the Homepage page works properly.	Automatic	Kadir Yıldız	7,8 sn	Successful
crossBrowserTestsMozilla (About)	Verify that the application opens without any problems in the Mozilla Firefox browser and the About page works properly.	Automatic	Kadir Yıldız	9,4 sn	Successful
crossBrowserTestsMozilla (Sign In)	Verify that the application opens without any problems in the Mozilla Firefox browser and the Sign In page works properly.	Automatic	Kadir Yıldız	7,7 sn	Successful
crossBrowserTestsMozilla (Sign Up)	Verify that the application opens without any problems in the Mozilla Firefox browser and the Sign Up page works properly.	Automatic	Kadir Yıldız	7,9 sn	Successful

FIGURE 46. Cross-Browser Testing Results: Mozilla Firefox.

showcase the progression and completion of tasks throughout the project.

A. DAILY SCRUM SCREENSHOTS

Below are the screenshots from our daily Scrum meetings, documenting task updates, challenges, and team coordination:



What did you do yesterday(bugün ne yapıldı)

I will run final tests to make sure all pages are working properly.

I will make optimizations that will increase page speeds to improve user experience.

I will focus on style adjustments on the frontend side and complete the responsive (mobile friendly) page designs.

I will make the necessary arrangements to ensure that the

What will be done?(yarın ne yapılacak)

Final tests will be conducted on all login, logout and user management operations in the admin panel.

Final edits to the profile page will be completed.

Final arrangements will be made for all operations on the backend and frontend.

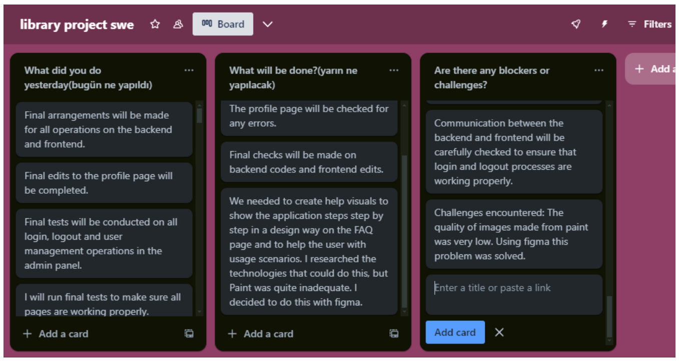
Are there any blockers or challenges?

between the model and view so that the data works harmoniously. I will do careful testing to ensure this process works correctly.

It will be important to ensure proper integration between backend and frontend codes. Additionally, every change to the profile page should be carefully tested.

Enter a title or paste a link

FIGURE 47. Daily Scrum Meeting: 30.11



What did you do yesterday(bugün ne yapıldı)

Final arrangements will be made for all operations on the backend and frontend.

Final edits to the profile page will be completed.

Final tests will be conducted on all login, logout and user management operations in the admin panel.

I will run final tests to make sure all pages are working properly.

What will be done?(yarın ne yapılacak)

The profile page will be checked for any errors.

Final checks will be made on backend codes and frontend edits.

We needed to create help visuals to show the application steps step by step in a design way on the FAQ page and to help the user with usage scenarios. I researched the technologies that could do this, but Paint was quite inadequate. I decided to do this with figma.

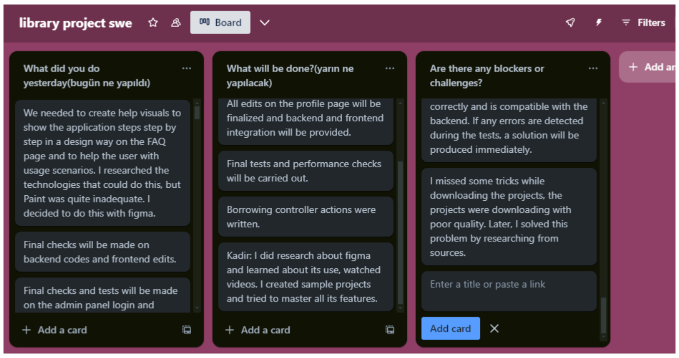
Are there any blockers or challenges?

Communication between the backend and frontend will be carefully checked to ensure that login and logout processes are working properly.

Challenges encountered: The quality of images made from paint was very low. Using figma this problem was solved.

Enter a title or paste a link

FIGURE 48. Daily Scrum Meeting: 01.12



What did you do yesterday(bugün ne yapıldı)

We needed to create help visuals to show the application steps step by step in a design way on the FAQ page and to help the user with usage scenarios. I researched the technologies that could do this, but Paint was quite inadequate. I decided to do this with figma.

Final checks will be made on backend codes and frontend edits.

Final checks and tests will be made on the admin panel login and

What will be done?(yarın ne yapılacak)

All edits on the profile page will be finalized and backend and frontend integration will be provided.

Final tests and performance checks will be carried out.

Borrowing controller actions were written.

Kadir: I did research about figma and learned about its use, watched videos. I created sample projects and tried to master all its features.

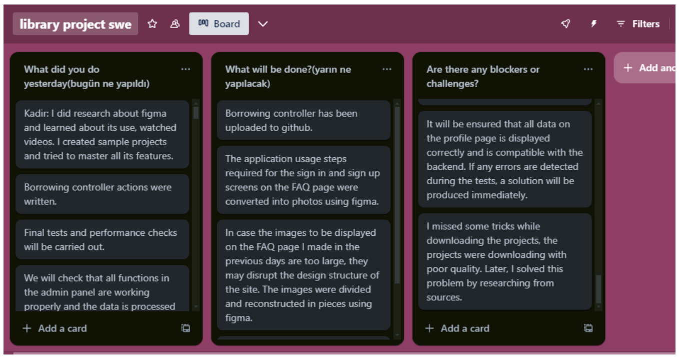
Are there any blockers or challenges?

correctly and is compatible with the backend. If any errors are detected during the tests, a solution will be produced immediately.

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

Enter a title or paste a link

FIGURE 49. Daily Scrum Meeting: 02.12



What did you do yesterday(bugün ne yapıldı)

Kadir: I did research about figma and learned about its use, watched videos. I created sample projects and tried to master all its features.

Borrowing controller actions were written.

Final tests and performance checks will be carried out.

We will check that all functions in the admin panel are working properly and the data is processed

What will be done?(yarın ne yapılacak)

Borrowing controller has been uploaded to github.

The application usage steps required for the sign in and sign up screens on the FAQ page were converted into photos using figma.

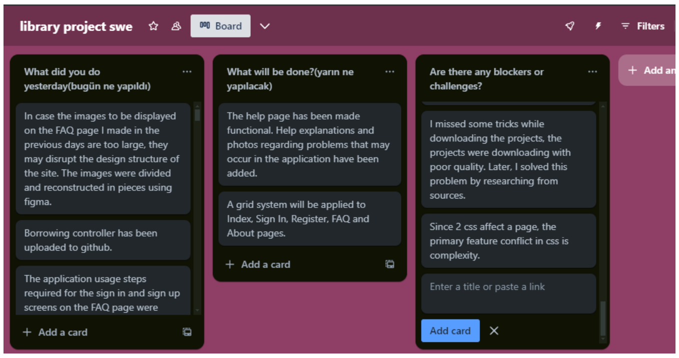
In case the images to be displayed on the FAQ page I made in the previous days are too large, they may disrupt the design structure of the site. The images were divided and reconstructed in pieces using figma.

Are there any blockers or challenges?

It will be ensured that all data on the profile page is displayed correctly and is compatible with the backend. If any errors are detected during the tests, a solution will be produced immediately.

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

FIGURE 50. Daily Scrum Meeting: 03.12



What did you do yesterday(bugün ne yapıldı)

In case the images to be displayed on the FAQ page I made in the previous days are too large, they may disrupt the design structure of the site. The images were divided and reconstructed in pieces using figma.

Borrowing controller has been uploaded to github.

The application usage steps required for the sign in and sign up screens on the FAQ page were

What will be done?(yarın ne yapılacak)

The help page has been made functional. Help explanations and photos regarding problems that may occur in the application have been added.

A grid system will be applied to Index, Sign In, Register, FAQ and About pages.

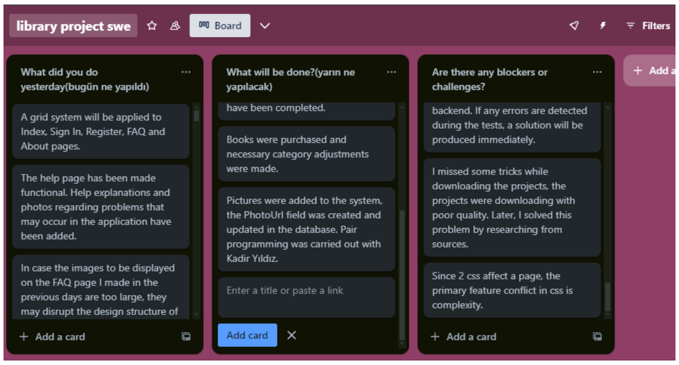
Are there any blockers or challenges?

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

Enter a title or paste a link

FIGURE 51. Daily Scrum Meeting: 04.12



What did you do yesterday(bugün ne yapıldı)

A grid system will be applied to Index, Sign In, Register, FAQ and About pages.

The help page has been made functional. Help explanations and photos regarding problems that may occur in the application have been added.

In case the images to be displayed on the FAQ page I made in the previous days are too large, they may disrupt the design structure of

What will be done?(yarın ne yapılacak)

have been completed.

Books were purchased and necessary category adjustments were made.

Pictures were added to the system, the PhotoUrl field was created and updated in the database. Pair programming was carried out with Kadir Yıldız.

Enter a title or paste a link

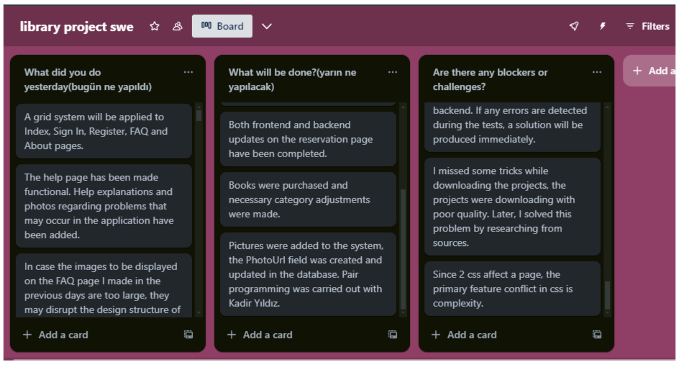
Are there any blockers or challenges?

backend. If any errors are detected during the tests, a solution will be produced immediately.

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

FIGURE 52. Daily Scrum Meeting: 13.12 Screenshot 1



What did you do yesterday(bugün ne yapıldı)

A grid system will be applied to Index, Sign In, Register, FAQ and About pages.

The help page has been made functional. Help explanations and photos regarding problems that may occur in the application have been added.

In case the images to be displayed on the FAQ page I made in the previous days are too large, they may disrupt the design structure of

What will be done?(yarın ne yapılacak)

Both frontend and backend updates on the reservation page have been completed.

Books were purchased and necessary category adjustments were made.

Pictures were added to the system, the PhotoUrl field was created and updated in the database. Pair programming was carried out with Kadir Yıldız.

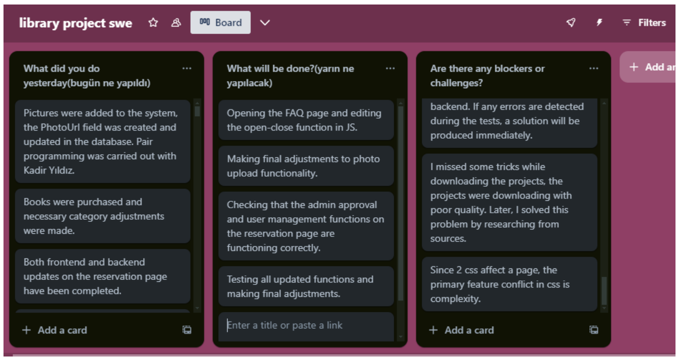
Are there any blockers or challenges?

backend. If any errors are detected during the tests, a solution will be produced immediately.

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

FIGURE 53. Daily Scrum Meeting: 13.12 Screenshot 2



What did you do yesterday(bugün ne yapıldı)

Pictures were added to the system, the PhotoUrl field was created and updated in the database. Pair programming was carried out with Kadir Yıldız.

Books were purchased and necessary category adjustments were made.

Both frontend and backend updates on the reservation page have been completed.

What will be done?(yarın ne yapılacak)

Opening the FAQ page and editing the open-close function in JS.

Making final adjustments to photo upload functionality.

Checking that the admin approval and user management functions on the reservation page are functioning correctly.

Testing all updated functions and making final adjustments.

Enter a title or paste a link

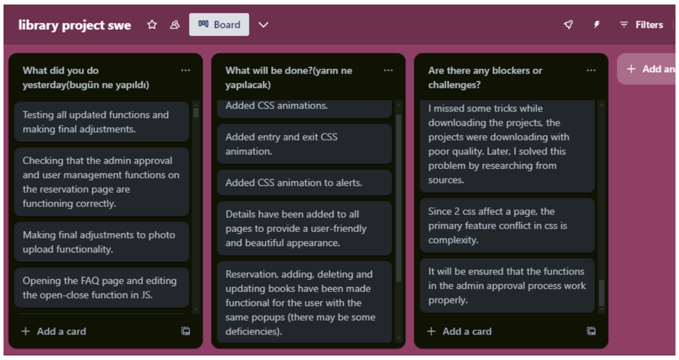
Are there any blockers or challenges?

backend. If any errors are detected during the tests, a solution will be produced immediately.

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

FIGURE 54. Daily Scrum Meeting: 14.12



What did you do yesterday(bugün ne yapıldı)

Testing all updated functions and making final adjustments.

Checking that the admin approval and user management functions on the reservation page are functioning correctly.

Making final adjustments to photo upload functionality.

Opening the FAQ page and editing the open-close function in JS.

What will be done?(yarın ne yapılacak)

Added CSS animations.

Added entry and exit CSS animation.

Added CSS animation to alerts.

Details have been added to all pages to provide a user-friendly and beautiful appearance.

Reservation, adding, deleting and updating books have been made functional for the user with the same popups (there may be some deficiencies).

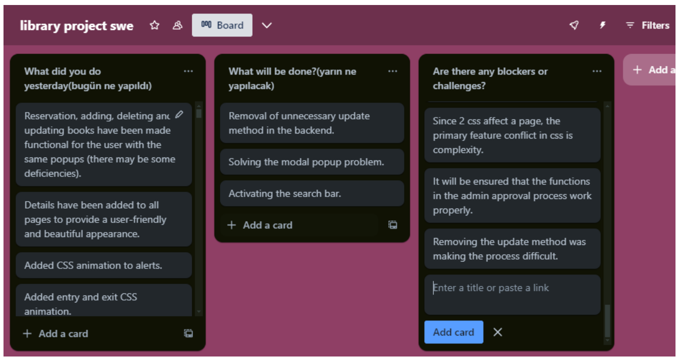
Are there any blockers or challenges?

I missed some tricks while downloading the projects, the projects were downloading with poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

FIGURE 55. Daily Scrum Meeting: 15.12



What did you do yesterday(bugün ne yapıldı)

Reservation, adding, deleting and updating books have been made functional for the user with the same popups (there may be some deficiencies).

Details have been added to all pages to provide a user-friendly and beautiful appearance.

Added CSS animation to alerts.

Added entry and exit CSS animation.

What will be done?(yarın ne yapılacak)

Removal of unnecessary update method in the backend.

Solving the modal popup problem.

Activating the search bar.

Are there any blockers or challenges?

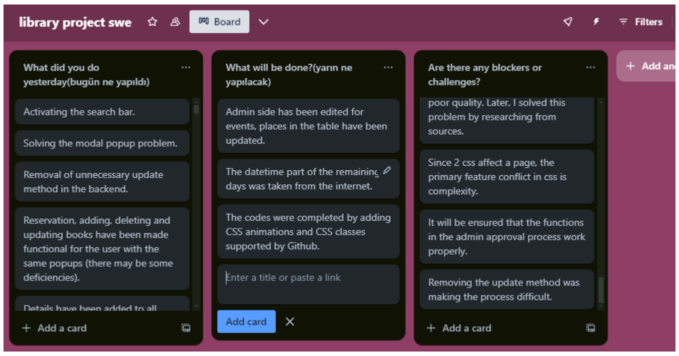
Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

Enter a title or paste a link

FIGURE 56. Daily Scrum Meeting: 16.12



What did you do yesterday(bugün ne yapıldı)

Activating the search bar.

Solving the modal popup problem.

Removal of unnecessary update method in the backend.

Reservation, adding, deleting and updating books have been made functional for the user with the same popups (there may be some deficiencies).

Details have been added to all

What will be done?(yarın ne yapılacak)

Admin side has been edited for events, places in the table have been updated.

The datetime part of the remaining days was taken from the internet.

The codes were completed by adding CSS animations and CSS classes supported by Github.

Enter a title or paste a link

Are there any blockers or challenges?

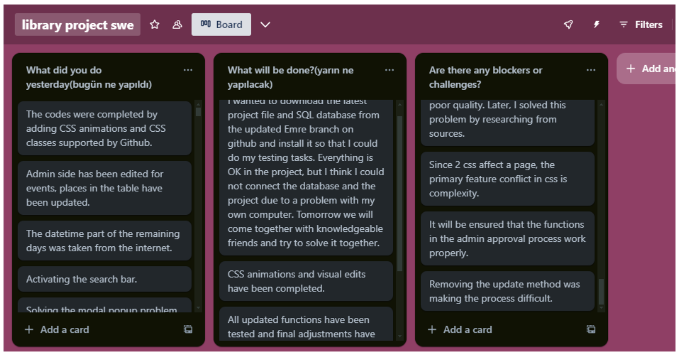
poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

FIGURE 57. Daily Scrum Meeting: 17.12



What did you do yesterday(bugün ne yapıldı)

The codes were completed by adding CSS animations and CSS classes supported by Github.

Admin side has been edited for events, places in the table have been updated.

The datetime part of the remaining days was taken from the internet.

Activating the search bar.

Solving the modal popup problem.

What will be done?(yarın ne yapılacak)

I wanted to download the latest project file and SQL database from the updated Emre branch on github and install it so that I could do my testing tasks. Everything is OK in the project, but I think I could not connect the database and the project due to a problem with my own computer. Tomorrow we will come together with knowledgeable friends and try to solve it together.

CSS animations and visual edits have been completed.

All updated functions have been tested and final adjustments have

Are there any blockers or challenges?

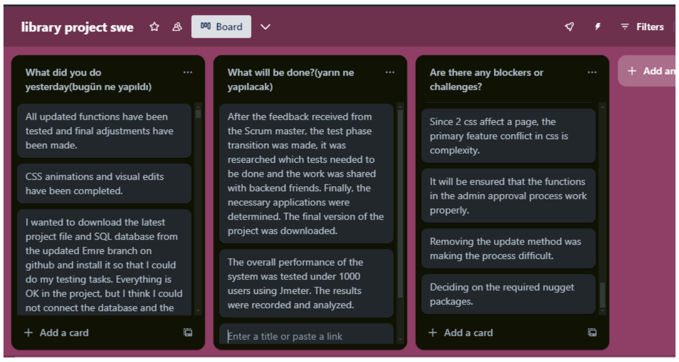
poor quality. Later, I solved this problem by researching from sources.

Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

FIGURE 58. Daily Scrum Meeting: 18.12



What did you do yesterday(bugün ne yapıldı)

All updated functions have been tested and final adjustments have been made.

CSS animations and visual edits have been completed.

I wanted to download the latest project file and SQL database from the updated Emre branch on github and install it so that I could do my testing tasks. Everything is OK in the project, but I think I could not connect the database and the

What will be done?(yarın ne yapılacak)

After the feedback received from the Scrum master, the test phase transition was made, it was researched which tests needed to be done and the work was shared with backend friends. Finally, the necessary applications were determined. The final version of the project was downloaded.

The overall performance of the system was tested under 1000 users using Jmeter. The results were recorded and analyzed.

Enter a title or paste a link

Are there any blockers or challenges?

Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

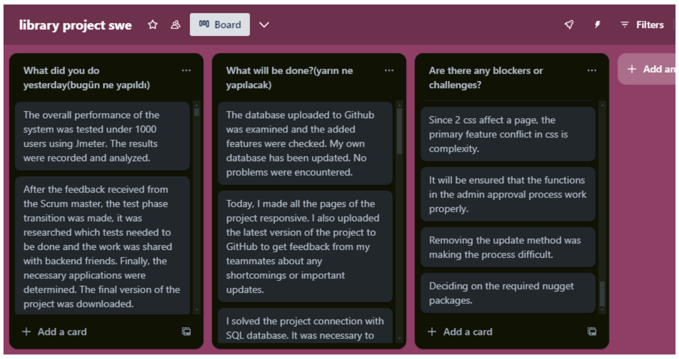
Deciding on the required nugget packages.

FIGURE 59. Daily Scrum Meeting: 19.12

XXXIV. XXVIII. REFERENCES

REFERENCES

[1] K. Schwaber and J. Sutherland, “The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game,” Scrum.org, 2020. [Online]. Available:



What did you do yesterday(bugün ne yapıldı)

The overall performance of the system was tested under 1000 users using Jmeter. The results were recorded and analyzed.

After the feedback received from the Scrum master, the test phase transition was made, it was researched which tests needed to be done and the work was shared with backend friends. Finally, the necessary applications were determined. The final version of the project was downloaded.

What will be done?(yarın ne yapılacak)

The database uploaded to Github was examined and the added features were checked. My own database has been updated. No problems were encountered.

Today, I made all the pages of the project responsive. I also uploaded the latest version of the project to GitHub to get feedback from my teammates about any shortcomings or important updates.

I solved the project connection with SQL database. It was necessary to

Are there any blockers or challenges?

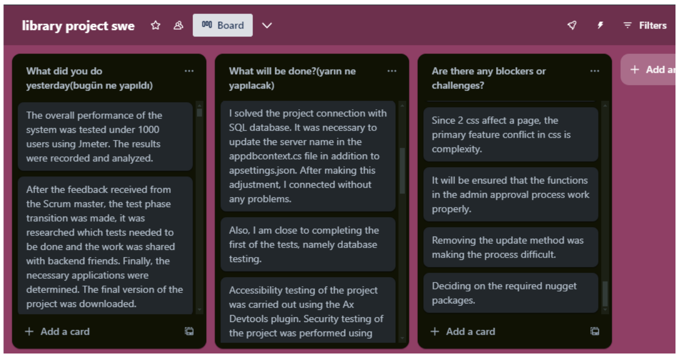
Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

Deciding on the required nugget packages.

FIGURE 60. Daily Scrum Meeting: 20.12 Screenshot 1



What did you do yesterday(bugün ne yapıldı)

The overall performance of the system was tested under 1000 users using Jmeter. The results were recorded and analyzed.

After the feedback received from the Scrum master, the test phase transition was made, it was researched which tests needed to be done and the work was shared with backend friends. Finally, the necessary applications were determined. The final version of the project was downloaded.

What will be done?(yarın ne yapılacak)

I solved the project connection with SQL database. It was necessary to update the server name in the appdbcontext.cs file in addition to apsettings.json. After making this adjustment, I connected without any problems.

Also, I am close to completing the first of the tests, namely database testing.

Accessibility testing of the project was carried out using the Ax Devtools plugin. Security testing of the project was performed using

Are there any blockers or challenges?

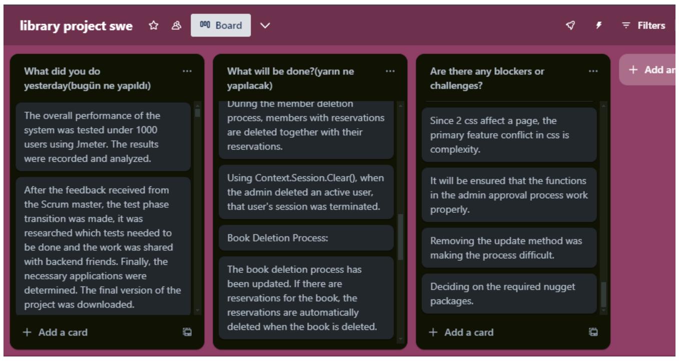
Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

Deciding on the required nugget packages.

FIGURE 61. Daily Scrum Meeting: 20.12 Screenshot 2



What did you do yesterday(bugün ne yapıldı)

The overall performance of the system was tested under 1000 users using Jmeter. The results were recorded and analyzed.

After the feedback received from the Scrum master, the test phase transition was made, it was researched which tests needed to be done and the work was shared with backend friends. Finally, the necessary applications were determined. The final version of the project was downloaded.

What will be done?(yarın ne yapılacak)

During the member deletion process, members with reservations are deleted together with their reservations.

Using Context.Session.Clear(), when the admin deleted an active user, that user's session was terminated.

Book Deletion Process:

The book deletion process has been updated. If there are reservations for the book, the reservations are automatically deleted when the book is deleted.

Are there any blockers or challenges?

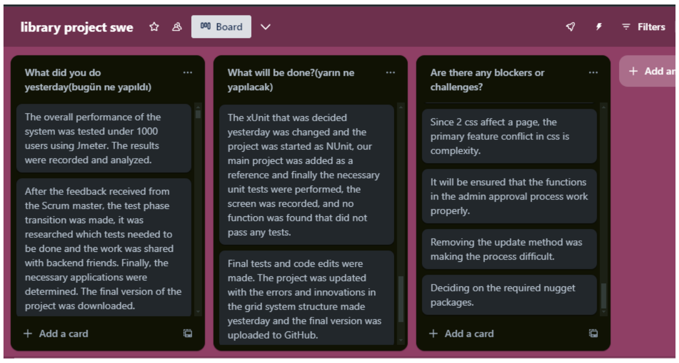
Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

Deciding on the required nugget packages.

FIGURE 62. Daily Scrum Meeting: 20.12 Screenshot 3



What did you do yesterday(bugün ne yapıldı)

The overall performance of the system was tested under 1000 users using Jmeter. The results were recorded and analyzed.

After the feedback received from the Scrum master, the test phase transition was made, it was researched which tests needed to be done and the work was shared with backend friends. Finally, the necessary applications were determined. The final version of the project was downloaded.

What will be done?(yarın ne yapılacak)

The xUnit that was decided yesterday was changed and the project was started as NUnit, our main project was added as a reference and finally the necessary unit tests were performed, the screen was recorded, and no function was found that did not pass any tests.

Final tests and code edits were made. The project was updated with the errors and innovations in the grid system structure made yesterday and the final version was uploaded to GitHub.

Are there any blockers or challenges?

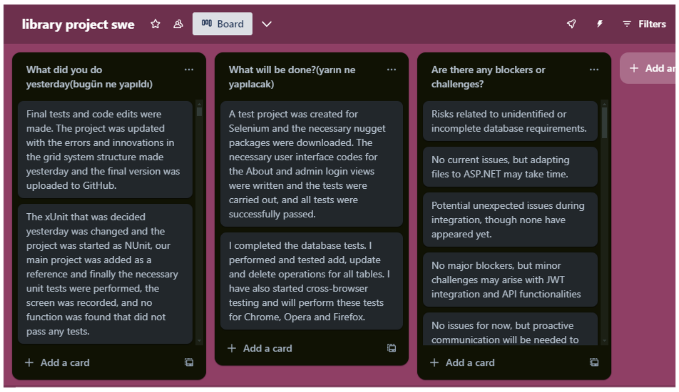
Since 2 css affect a page, the primary feature conflict in css is complexity.

It will be ensured that the functions in the admin approval process work properly.

Removing the update method was making the process difficult.

Deciding on the required nugget packages.

FIGURE 63. Daily Scrum Meeting: 20.12 Screenshot 4



What did you do yesterday(bugün ne yapıldı)

Final tests and code edits were made. The project was updated with the errors and innovations in the grid system structure made yesterday and the final version was uploaded to GitHub.

The xUnit that was decided yesterday was changed and the project was started as NUnit, our main project was added as a reference and finally the necessary unit tests were performed, the screen was recorded, and no function was found that did not pass any tests.

What will be done?(yarın ne yapılacak)

A test project was created for Selenium and the necessary nugget packages were downloaded. The necessary user interface codes for the About and admin login views were written and the tests were carried out, and all tests were successfully passed.

I completed the database tests. I performed and tested add, update and delete operations for all tables. I have also started cross-browser testing and will perform these tests for Chrome, Opera and Firefox.

Are there any blockers or challenges?

Risks related to unidentified or incomplete database requirements.

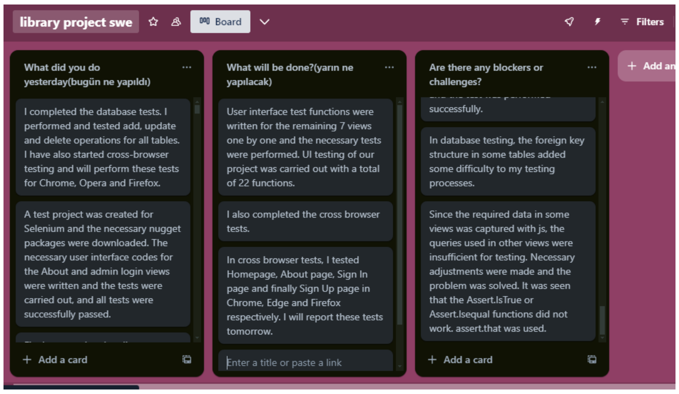
No current issues, but adapting files to ASP.NET may take time.

Potential unexpected issues during integration, though none have appeared yet.

No major blockers, but minor challenges may arise with JWT integration and API functionalities

No issues for now, but proactive communication will be needed to

FIGURE 64. Daily Scrum Meeting: 21.12



What did you do yesterday(bugün ne yapıldı)

I completed the database tests. I performed and tested add, update and delete operations for all tables. I have also started cross-browser testing and will perform these tests for Chrome, Opera and Firefox.

A test project was created for Selenium and the necessary nugget packages were downloaded. The necessary user interface codes for the About and admin login views were written and the tests were carried out, and all tests were successfully passed.

What will be done?(yarın ne yapılacak)

User interface test functions were written for the remaining 7 views one by one and the necessary tests were performed. UI testing of our project was carried out with a total of 22 functions.

I also completed the cross browser tests.

In cross browser tests, I tested Homepage, About page, Sign In page and finally Sign Up page in Chrome, Edge and Firefox respectively. I will report these tests tomorrow.

Are there any blockers or challenges?

successfully.

In database testing, the foreign key structure in some tables added some difficulty to my testing processes.

Since the required data in some views was captured with js, the queries used in other views were insufficient for testing. Necessary adjustments were made and the problem was solved. It was seen that the Assert.IsTrue or Assert.Isequal functions did not work. assert.that was used.

FIGURE 65. Daily Scrum Meeting: 22.12

https://scrumguides.org
[2] Trello, “Trello: Project management and task tracking software,” Accessed: Nov. 22, 2024. [Online]. Available: https://trello.com
[3] GitHub, “GitHub: Collaborative development platform,” Accessed: Nov. 22, 2024. [Online]. Available: https://github.com
[4] Google, “Google Docs: Online document collaboration tool,” Accessed: Nov. 22, 2024. [Online]. Available: https://docs.google.com
[5] G. Booch, J. Rumbaugh, and I. Jacobson, “The Unified Modeling Language User Guide,” 2nd ed., Addison-Wesley, 2005.
[6] R. E. Fairley, “Managing and Leading Software Projects,” Hoboken, NJ, USA: Wiley, 2009.
[7] M. Cohn, “Succeeding with Agile: Software Development Using Scrum,” Boston, MA, USA: Addison-Wesley, 2009.
[8] I. Sommerville, “Software Engineering,” 10th ed., Pearson Education, 2015.
[9] A. van Lamsweerde, “Requirements Engineering: From System Goals to UML Models to Software Specifications,” Chichester, UK: Wiley, 2009.
[10] K. E. Kendall and J. E. Kendall, “Systems Analysis and Design,” 10th ed., Pearson, 2019.
[11] C. Coronel, S. Morris, and P. Rob, “Database Systems: Design, Implementation, and Management,” 13th ed., Cengage Learning, 2020.
[12] E. Freeman and E. Robson, “Head First HTML and CSS,” 2nd ed., Sebastopol, CA, USA: O’Reilly Media, 2012.
[13] L. Williams and R. Kessler, “Pair Programming Illuminated,” Boston, MA, USA: Addison-Wesley, 2002.
[14] Team Members of “Book World Project,” “Library management system project,” Internal documentation, 2024.
[15] J. Smith, “Library Management Systems: A Practical Guide,” New York, NY, USA: Springer, 2018.
[16] T. Noergaard, “Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers,” 2nd ed., Burlington, MA, USA: Newnes, 2012.