Project Documentation (Work Compilation)

Created by: Preston Kelly email: preston_kelly1@baylor.edu
Reviewed by: Subash Kharel email: subash_kharel2@baylor.edu
Reviewed by: Scott Willis email: scott_willis1@baylor.edu

4/16/2022

This document is a culmination of the work that this group did over roughly the last ten weeks. This consisted of a planning phase, a concept phase, and a prototype phase. Here is the information regaridng this project.

1 Abstract-Problem & solution

This project was intended to be a group of two or three members to develop a web application using a database. An example of this would be a eShop, reservation system, booking, rental cars, etc.

For this project, the groups must use Java, Git, Maven, and a testing framework. Each group will also need to establish a git repositroy, with a website or wiki page, and an issue tracking or ticketing system such as GitHub.

For the data model, each team needs to have at least 7 entities/tables and have at least one one-to-many and one many-to-many relationships.

2 Introduction to the problem & Vision

This section contains the Project Vision for this project we felt were needed for this application to be considered ready.

2.1 Project Vision

For our group project, we will be creating a full stack library application. This application will have all the functionality of a standard library. There will be a collection of books, with the ability to add or remove books. Each book will have a unique identifier and will have th ability to be checked out by a student or removed by a librarian. Each student and librarian will be registered in our identity service. All books that are cheeked out will be marked as such and have a return date. Books can be marked as returned using the website. Overdue books will have a warning on the website, and an email will be sent out to the user who has checked out the overdue book. The numbers of copies of a book, expected return date of the books, and details about the book itself will all be easily visible in the User Interface (UI). Librarians and other employees will have a slightly different view in the UI, to alter and update the book collection as needed. Students and employees themselves can be added or removed from the system, as only active students can check out books, and only active employees can manage the collection.

3 Analysis

This section contains the information we felt was needed to be made clear for the project while analyzing our Vision and the requirements for this project.

3.1 Project Functional Requirements

Functional Requirements are key goals that we felt needed to be implemented into the project to be included for a functional application.

Functional Requirements List:
1. Users should be able register themselves for library access
2. Users should be able to login to library application after successful registration
3. Users passwords should be hashed when stored in database for security reasons
4. All users should be able to see all books in the library with the status of books with details
5. Administrative users will be able to add, remove, and update books.
6. Overdue books will send a notification to the associated email address that checked the book out.
7. Books can be marked as Available, Checked out, or Overdue.
8. All Book details are visible on the web page.

Non-functional Requirements are goals that are good to have, but not needed for the functionality of the application.

Non-Functional Requirements List:
1. Disclaimer page with rules and regulations

4 Design Concepts and Ideas

This section contains, at the time, thoughts and ideas for implementation, regarding this application.

4.0.1 Data Model

The data model is key for what will be the backbone for the relationship of the database. This determines key pairings and the relationships each entity will have to one another. This will also play a role when it comes to setting up the functions we wish to have in our application.

This document can be found here.

If you are not reading a digital version of this document, the link to the file will be located in the references section.

4.0.2 System Operations and Operation Contracts

The contract and System Operations is a section we wrote up to help us understand how we wanted to implement the various portions of our applications together and help us visualize individual portions of our code.

This document can be found here.

If you are not reading a digital version of this document, the link to the file will be located in the references section.

4.0.3 Implementation Consideration

During this time, we as team, decided that we would be separating the Application User Interface (UI) from the Database Back End. The reason for this would be to allow for pair development during the implementation face. We, as a group, felt this would be a better use of our time instead to allow for the parallel development and then reintegrate the portions together.

5 Design Choices

This section of the document is related to the design choices that were made when creating this application. This will cover the choices made regarding the front end and back end development.

5.1 Architecture Document

The architecture document was created to have a visual representation of our application structure. This is the ideal concept that we decided to move forward with. To see this document, it can be found here to find out more about Angular. A link will also be provided in the reference section of this document.

5.1.1 Front End Development

For the Front End User Interface (UI), our team went choose Angular. Angular is a modern web developer's platform that is able to be used to develop across multiple platforms.

The reason our team went with Angular is that the majority of our team has had some experience with Angular in the past, but the key reason for this was that it was capable of being used to quickly and efficiently stand up a web front end to be used for our application.

For more information, please visit here to find out more about Angular. A link will also be provided in the reference section of this document.

For the design of these front end, we wanted to create a simple yet easy to use UI. For this, we create a set of wire frames to be used as a guide to help with design.

The wire frames can be found here. A link will also be provided in the reference section of this document for this address.

5.1.2 Back End Development

For the Back End development, we created a simple Create, Read, Update, Delete (CRUD) function service that would interact with our database. These functions would allow for us to be called by the front end, and with some provided information, allow for the us to update our database.

For our database, our group chose to set up an in-memory database. This is also known as an H2 database. This type of database will spin up and preload defined information into our application. While this will not work for a production environment, this will allow for us to show a working concept.

For more information about H2 databases, that can be found here. A link will also be provided in the reference section of this document.

Another thing that our group integrated into the back end was Swagger. Swagger is used to simplify API development between users and teams to help develop API calls. This was implemented into our code to help refine our APi calls but also simply them and allow for the team to scale if this project was to suddenly grow.

For more information about Swagger, that can be found here. A link will also be provided in the reference section of this document.

6 Implementation

This section of the document details the implementation of Design choices for our project. This section focuses on the front end and back end implementation.

6.0.1 Front End Implementation

The Front End UI was created using Angular. This was originally built to be a stand alone application that could be stood up with out the back end integration.

To see the final code of the UI, you can see it here. A link will also be provided in the reference section of this document.

To see what the UI looks like after the before and after the back end integration, visit here. These images are of the various pages or components of our UI. A link will also be provided in the reference section of this document.

6.0.2 Back End Implementation

The Back End Implementation was developed in parallel with the Front End UI. This was a CRUD Database developed by our development team. For this backend, we also implemented Swagger configuration into it.

To see the final code of the UI, you can see it here. A link will also be provided in the reference section of this document.

7 Case Study - Our Use Cases

For our project, we came up with six use cases we felt were needed to be completed for our application needed to be able to do.

These use cases are as follows:
1. The ability to Register for the App
2. The ability to Sign into the App
3. The ability for a User to Checkout a Book
4. The ability for a User to Add a Book
5. The ability for a User to Update a Book
6. The ability for a User to Delete a Book

These six use cases can be considered the basic functionality required to have our function to work. For a more detailed information on the use cases, please go to the appendix of this document where you can read more about it.

8 Evaluations

In Table 1, you can see the overall information regarding the history of our git repository. Please note, when it comes to the amount of git commits, we will only be referring to the commits done within the final repository. This is due to the fact hat most of our work was done within this repository itself. This information is accurate as of 4/17/2022.

Category	Values
Number of Members	3
Issues Completed	52
Git Commits	118 and counting
Hours Worked	89

Table 1: Overview of tasks, git commits, and hours worked.

For a more detailed break down that includes the number of issues worked on, the amount of individual commits, and individual hours worked can be found here. A link will also be provided in the reference section of this document.

9 Conclusions

Overall, this was a very successful project. We were able to implement almost all of our functionality goals, all of our non-functional goals, and able to successfully complete all use cases.

The application User Interface (UI) is easy to read, use, and is fairly easy to be updated if it is decided that the application needs to have new functionality in the future.

The back end of the application was built from scratch, which helps to reduce the amount of white noise that could exist in the back end. Another thing to note, is that with the introduction of swagger configuration to the back end, we help to keep the api uniform moving forward. this will allow for the project to keep a effect structure moving forward.

In the end, the entire group who worked on this project have enjoyed the process and are happy to be able to showcase the work they have committed.

10 References

This section of the document contains url address for various componets we have refernced in this document. This includes repository address, urls for technology that was used, or information referenced urls.

10.1 Repository Addresses and Document Locations

The final repository address is as follows:

`https://github.com/willissa2121/library_checkout_proj_3`

The first repository we used is located here:

`https://github.com/willissa2121/library_checkout_proj`

The second repository we used is as followed:

`https://github.com/willissa2121/library_checkout_proj_3`

The Data Model diagram used for this application:

```
https://github.com/willissa2121/library_checkout_proj_3/blob/
    main/Previous%20Repo%20Work/Domain%20Models/Iteration%202/
    UML.png
```

The Wire Frame Images used during the design phase:

```
https://github.com/willissa2121/library_checkout_proj_3/tree/
    main/Previous%20Repo%20Work/Documenation/Wireframes/
    Iteration%201
```

The Systems Operations and Operation Contract for this application:

```
https://github.com/willissa2121/library_checkout_proj_3/blob/
    f73a4f039dedd2f664073461a1e1754665f06aa7/Previous%20Repo
    %20Work/Documenation/System%20Ops%20and%20Contracts/
    Iteration%201/systemops_contracts.pdf
```

Architecture Diagram:

```
https://github.com/willissa2121/library_checkout_proj_3/blob/
    main/architecture/app_architecture_guide.pdf
```

User Interface Source Code:

```
https://github.com/willissa2121/library_checkout_proj_3/tree/main/
    project-ui
```

Front End Post Back End Integration Images:

```
https://github.com/willissa2121/library_checkout_proj_3/tree/
    main/project-ui/src/app/UI%20Images/Post%20Backend%20
    Integration
```

Back End Database Integration:

```
https://github.com/willissa2121/library_checkout_proj_3/tree/
    main/ws-library-checkout
```

Work Evaluation Breakdown:

```
https://github.com/willissa2121/library_checkout_proj_3/blob/main/Work
    %20Review/Completed_Work_Review_Group_Project.pdf
```

10.2 Technology Used

The following are the address for various technology we implemented into our project.

Angular:

`https://angular.io/`

H2 Databases:

`https://www.h2database.com/html/main.html`

Swagger:

`https://swagger.io/`

Spring Boot:

`https://spring.io/projects/spring-boot`

Visual Paradigm Online (document Creation)

`https://online.visual-paradigm.com/`

Overleaf Latex Editor (PDF Creation)

`https://www.overleaf.com/`

10.3 Information referenced

CSS Tutorials:

`https://www.w3schools.com/`

Overleaf Guide

`https://faculty.buffalostate.edu/cunnindw/overleaf.pdf`

11 Appendix (List of Use Cases)

This section of the document contains the combine list of the use cases that were created and implemented for this project. These Use Cases include registering a user, signing in, checking a book out, adding a book, updating a book, and removing a book from the database.

11.1 Use Case UC1: Registration for BearBooks

11.1.1 Scope

User Registration for BearBooks Application

11.1.2 Level

User goal

11.1.3 Primary Actor

Application User (Book Borrower)

11.1.4 Stakeholders and Interests

Application User: wants fast, reliable, and accurate ability to view available books and check out the books of interest
Administration Users: Want a fast, reliable, and accurate application to manage the stock of rent able books
Development Company: Wants to accurately provide a safe and reliable application to borrow books from the business
Destination Owners: Wants a safe, reliable, and efficient way to rent books and keep stock organized and updated over time

11.1.5 Preconditions

Application User is not registered into the system
Application Front End is online.

Application Back End is online.

11.1.6 Success Guaranteed

11.1.7 Main Success Scenario

1.Application user launches the application with the ui and back end running.
2.Application User selects Sign In/Register tab in the header.
3.For registering, Application User fills in the information requested by the registration form.
4.Upon Submission, Application stores requested information in the database back end.

11.1.8 Extensions

*a. At any time the application fails
Restart the Application
1.If the UI is not working, restart the front end application portion. 2.If the Back end is not working, restart it
2a.Information that is stored in the database should load its contents for the applications
2b. System restarts new navigation
1. Load home page of application 1.a If failing to laod, check to ensure application is running.
1.b Restart application if it is not running.
1.c Reload Home page for application
2. Reload application if failed to load upon selecting the Sign In tab.
2.a Select the correct tab to load wanted page of application.
3. If failed to register, correct the incorrect field that is mentioned.
3.a Resubmit registration
4. If unable to register, but the UI is working, restart the back end to ensure the data base is up.
4.a Resubmit registration once the back end is up and running.

11.1.9 Special Requirements

-Responsive UI to allow for use across multiple devices -Application quick response when in use

11.1.10 Technology and Data Variations Lists

-Application must accept inputs from a touch screen entry

11.1.11 Frequency of Occurrence

Could be nearly continuous

11.1.12 Open Issues

-What to do if application is down due to issues or updates

11.2 Use Case UC2: Sign in for BearBooks

11.2.1 Scope

User Sign in for BearBooks Application

11.2.2 Level

User goal

11.2.3 Primary Actor

Application User (Book Borrower)

11.2.4 Stakeholders and Interests

Application User: wants fast, reliable, and accurate ability to view available books and check out the books of interest
Administration Users: Want a fast, reliable, and accurate application to manage the stock of rent able books
Development Company: Wants to accurately provide a safe and reliable application to borrow books from the business
Destination Owners: Wants a safe, reliable, and efficient way to rent books and keep stock organized and updated over time

11.2.5 Preconditions

Application User is registered into the system
Application Front End is online.
Application Back End is online.

11.2.6 Success Guaranteed

11.2.7 Main Success Scenario

1.Application user launches the application with the ui and back end running.
2.Application User selects Sign In/Register tab in the header.
3.For sign in, Application User fills in the information requested by for the sign in form.
4.Upon Submission, user will sign into the application.

11.2.8 Extensions

*a. At any time the application fails
Restart the Application

1.If the UI is not working, restart the front end application portion. 2.If the Back end is not working, restart it
2a.Information that is stored in the database should load its contents for the applications
2b. System restarts new navigation
1. Load home page of application 1.a If failing to laod, check to ensure application is running.
1.b Restart application if it is not running.
1.c Reload Home page for application
2. Reload application if failed to load upon selecting the Sign In tab.
2.a Select the correct tab to load wanted page of application.
3. If failed to sign in, correct the incorrect field that is mentioned.
3.a Resubmit sign in
4. If unable to sign in, but the UI is working, restart the back end to ensure the database is up.
4.a Resubmit sign in once the back end is up and running.

11.2.9 Special Requirements

-Responsive UI to allow for use across multiple devices -Application quick response when in use

11.2.10 Technology and Data Variations Lists

-Application must accept inputs from a touch screen entry

11.2.11 Frequency of Occurrence

Could be nearly continuous

11.2.12 Open Issues

-What to do if application is down due to issues or updates

11.3 Use Case UC3: Checkout Books using BearBooks Application

11.3.1 Scope

User checks out a Book from BearBooks Application

11.3.2 Level

User goal

11.3.3 Primary Actor

Application User (Book Borrower)

11.3.4 Stakeholders and Interests

Application User: wants fast, reliable, and accurate ability to view available books and check out the books of interest
Administration Users: Want a fast, reliable, and accurate application to manage the stock of rent able books
Development Company: Wants to accurately provide a safe and reliable application to borrow books from the business
Destination Owners: Wants a safe, reliable, and efficient way to rent books and keep stock organized and updated over time

11.3.5 Preconditions

Application User is registered into the system
Application User is signed into the application
Application Front End is online.
Application Back End is online.

11.3.6 Success Guaranteed

11.3.7 Main Success Scenario

a.Assume user is already signed in to the application and the application is running
1.User selects Checkout Book tab in the header.
2. Search through available books in the catalog shown on this tab.
3. Click the checkout book button inline with the the book you wish to checkout.
4. Page will update and the book you checkout will no longer be visible on this page.

11.3.8 Extensions

*a. At any time the application fails
Restart the Application
1.If the UI is not working, restart the front end application portion. 2.If the Back end is not working, restart it
2a.Information that is stored in the database should load its contents for the applications
2b. System restarts new navigation
1. Load the Book Checkout tab 1.a If failing to load, check to ensure application is running.
1.b Restart application if it is not running.
1.c Reload Home page for application and restart process
2. Reload application if failed to load upon selecting the Book Checkout In tab.
2.a Select the correct tab to load wanted page of application.
3. If failed to check out, ensure that the back end is running.
3.a Reattempt to checkout the book

11.3.9 Special Requirements

-Responsive UI to allow for use across multiple devices -Application quick response when in use

11.3.10 Technology and Data Variations Lists

-Application must accept inputs from a touch screen entry

11.3.11 Frequency of Occurrence

Could be nearly continuous

11.3.12 Open Issues

-What to do if application is down due to issues or updates

11.4 Use Case UC4: Add Books to data base for BearBooks

11.4.1 Scope

User adds a Book to BearBooks Application

11.4.2 Level

User goal

11.4.3 Primary Actor

Application User (Book Borrower)

11.4.4 Stakeholders and Interests

Application User: wants fast, reliable, and accurate ability to view available books and check out the books of interest
Administration Users: Want a fast, reliable, and accurate application to manage the stock of rent able books
Development Company: Wants to accurately provide a safe and reliable application to borrow books from the business
Destination Owners: Wants a safe, reliable, and efficient way to rent books and keep stock organized and updated over time

11.4.5 Preconditions

Application User is registered into the system
Application User is signed into the application
Application Front End is online.
Application Back End is online.

11.4.6 Success Guaranteed

11.4.7 Main Success Scenario

a.Assume user is already signed in to the application and the application is running
1.User selects Admin tab in the header.
2. Click the add book button from this page.
3. Fill in content regarding the new book.
4. Submit new book information to the server.

11.4.8 Extensions

*a. At any time the application fails
Restart the Application
1.If the UI is not working, restart the front end application portion. 2.If the Back end is not working, restart it
2a.Information that is stored in the database should load its contents for the applications
2b. System restarts new navigation
1. Load the Admin tab
1.a If failing to load, check to ensure application is running.
1.b Restart application if it is not running.
1.c Reload Home page for application and restart process
2. Reload application if failed to load upon selecting the Admin tab.
2.a Select the correct tab to load wanted page of application.
3. If failed to add book, ensure that the back end is running.
3.a Reattempt to add the book
4. If failing to submit to back end, ensure that the back end is working.
4.a If still failing to upload, ensure the submitted values are correct for the new book.

11.4.9 Special Requirements

-Responsive UI to allow for use across multiple devices -Application quick response when in use

11.4.10 Technology and Data Variations Lists

-Application must accept inputs from a touch screen entry

11.4.11 Frequency of Occurrence

Could be nearly continuous

11.4.12 Open Issues

-What to do if application is down due to issues or updates

11.5 Use Case UC5: Update Books to data base for BearBooks

11.5.1 Scope

User updates a Book to BearBooks Application

11.5.2 Level

User goal

11.5.3 Primary Actor

Application User (Book Borrower)

11.5.4 Stakeholders and Interests

Application User: wants fast, reliable, and accurate ability to view available books and check out the books of interest
Administration Users: Want a fast, reliable, and accurate application to manage the stock of rent able books
Development Company: Wants to accurately provide a safe and reliable application to borrow books from the business
Destination Owners: Wants a safe, reliable, and efficient way to rent books and keep stock organized and updated over time

11.5.5 Preconditions

Application User is registered into the system
Application User is signed into the application
Application Front End is online.
Application Back End is online.

11.5.6 Success Guaranteed

11.5.7 Main Success Scenario

a.Assume user is already signed in to the application and the application is running
1.User selects Admin tab in the header.
2. Click the Update book button for a book row from this page.
3. Fill in content regarding the current book.
4. Submit updated book information to the server.

11.5.8 Extensions

*a. At any time the application fails
Restart the Application
1.If the UI is not working, restart the front end application portion. 2.If the Back end is not working, restart it
2a.Information that is stored in the database should load its contents for the applications
2b. System restarts new navigation
1. Load the Admin tab
1.a If failing to load, check to ensure application is running.
1.b Restart application if it is not running.
1.c Reload Home page for application and restart process
2. Reload application if failed to load upon selecting the Admin tab.
2.a Select the correct tab to load wanted page of application.
3. If failed to update, ensure that the back end is running.
3.a Reattempt to update the book
4. If failing to submit to back end, ensure that the back end is working.
4.a If still failing to upload, ensure the submitted values are correct for the new book.

11.5.9 Special Requirements

-Responsive UI to allow for use across multiple devices -Application quick response when in use

11.5.10 Technology and Data Variations Lists

-Application must accept inputs from a touch screen entry

11.5.11 Frequency of Occurrence

Could be nearly continuous

11.5.12 Open Issues

-What to do if application is down due to issues or updates

11.6 Use Case UC6: Delete Books from the data base for BearBooks

11.6.1 Scope

User deletes a Book to BearBooks Application

11.6.2 Level

User goal

11.6.3 Primary Actor

Application User (Book Borrower)

11.6.4 Stakeholders and Interests

Application User: wants fast, reliable, and accurate ability to view available books and check out the books of interest
Administration Users: Want a fast, reliable, and accurate application to manage the stock of rent able books
Development Company: Wants to accurately provide a safe and reliable application to borrow books from the business
Destination Owners: Wants a safe, reliable, and efficient way to rent books and keep stock organized and updated over time

11.6.5 Preconditions

Application User is registered into the system
Application User is signed into the application
Application Front End is online.
Application Back End is online.

11.6.6 Success Guaranteed

11.6.7 Main Success Scenario

a.Assume user is already signed in to the application and the application is running
1.User selects Admin tab in the header.
2. Click the delete book button for a book row from this page.
3. Confirm wanting to delete book.
4. Remove book information from the back end.

11.6.8 Extensions

*a. At any time the application fails
Restart the Application
1.If the UI is not working, restart the front end application portion. 2.If the Back end is not working, restart it
2a.Information that is stored in the database should load its contents for the applications
2b. System restarts new navigation
1. Load the Admin tab
1.a If failing to load, check to ensure application is running.
1.b Restart application if it is not running.
1.c Reload Home page for application and restart process
2. Reload application if failed to load upon selecting the Admin tab.
2.a Select the correct tab to load wanted page of application.
3. If failed to delete, ensure that the back end is running.
3.a Reattempt to delete the book
4. If failing to submit to back end, ensure that the back end is working.
4.a If still failing to delete, ensure the that the book has not already been deleted.

11.6.9 Special Requirements

-Responsive UI to allow for use across multiple devices -Application quick response when in use

11.6.10 Technology and Data Variations Lists

-Application must accept inputs from a touch screen entry

11.6.11 Frequency of Occurrence

Could be nearly continuous

11.6.12 Open Issues

-What to do if application is down due to issues or updates