**CST-341 Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | CST-341-O500 Open Source Computing | |
| **Date:** | April 5, 2020 | |
| **Revision:** | v0.6.0 | |
| **Team:** | 1. Branden Manibusan 2. Tyler Rosenberry 3. Daniel Cender 4. David Pratt Jr. 5. Zachary Swoveland | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **HoursRemaining** | | *n/a* | Daniel Cender | *7.5* | *0* | | *n/a* | Zachary Swoveland | *n/a* | *0* | | *n/a* | David Pratt Jr | *3* | *0* | | *n/a* | Branden Manibusan | *0* | *0* | | *n/a* | Tyler Rosenberry | *0* | *0* | | |
| **GIT URL:** | <https://github.com/DanielCender/CST-341-O500-CLC.git> | |
| **Peer Review:** | Y | We acknowledge that our team has reviewed this report and we agree to the approach we are all taking. |

**Planning Documentation**

**Agile Scrum Product Backlog:**

The link below leads to the overall Backlog column on our group’s GitHub Project, which is tied to managing issues and code pull requests for our CLC project.

<https://github.com/DanielCender/CST-341-O500-CLC/projects/2#column-8179680>

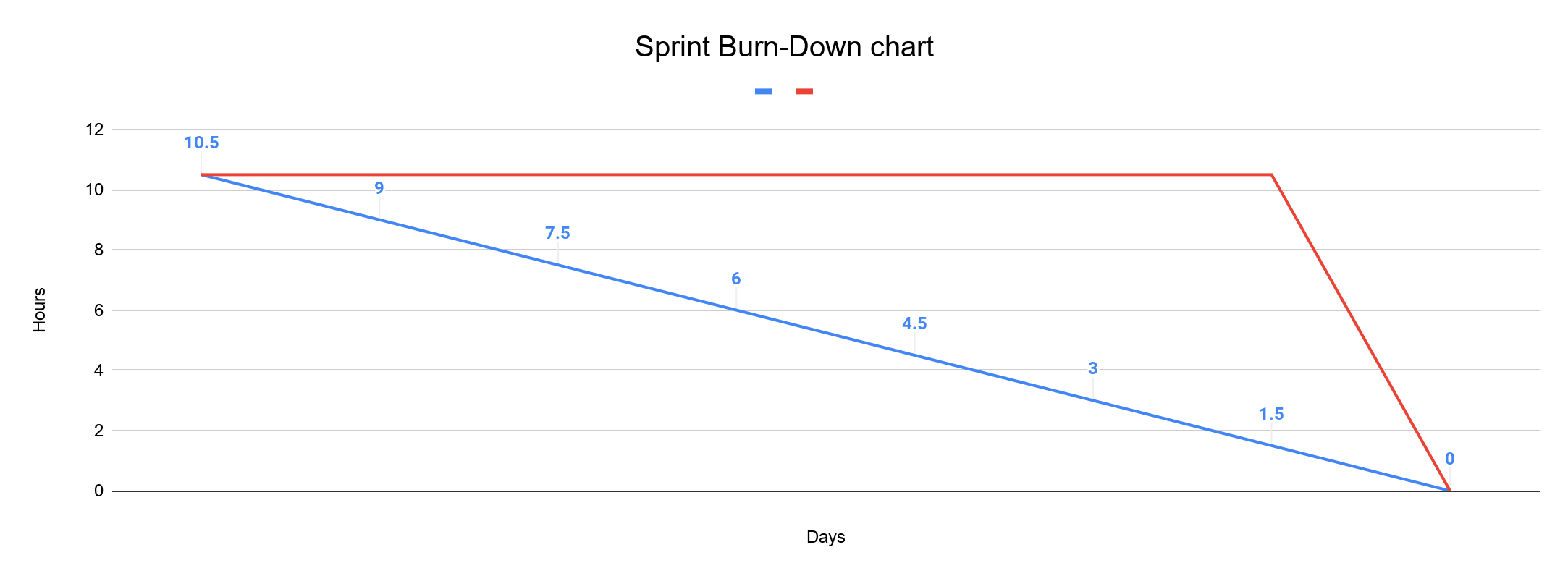
**Agile Scrum Sprint Backlog:**

The link below leads to the current sprint backlog column (called “To Do”) on our group’s GitHub Project.

<https://github.com/DanielCender/CST-341-O500-CLC/projects/2#column-8179685>

**Agile Scrum Burn Down Chart:**

The below link and embedded image are from our Google Sheets burn down chart:



The link to view the entire sheet of hour-task breakdowns is here: <https://docs.google.com/spreadsheets/d/1yviqXmRQJthjw6ahv8QQVoOw5mhavmffmAOrsVywxEg/edit?usp=sharing>

**Agile Retrospective Results:**

*The following table should be completed after each Retrospective on Things That Went Well (Keep Doing). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool, you must include a URL or Image File.*

|  |
| --- |
| **What Went Well** |
| * Using Discord chat again as a team worked well to foster speedy communication. |
| * Using the Google office suite (Docs and Sheets) helped the group collaborate efficiently, even with limited time to communicate among one another. |

*The following table should be completed after each Retrospective on Things That Didn’t Go Well (Stop Doing) and What Would Be Done Differently Next Time with an Action Plan to Improve (Try Doing and Continuous Improvement). An alternative to the following table is to use a Mind Mapping tool such as Coggle. If you use a Mind Mapping tool, you must include a URL or Image File.*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
| * Using Discord chat worked well, but ultimately this week spawned the least contributor interaction so far, even compared to last week | * It may help to share more concrete examples and document the code classes/interfaces earlier in the week so no contributors feel left behind by missing weeks of work | April 6, 2020 |

**Design Documentation**

**Project Description:**

The project our group will build is an online bookstore, similar to a Goodreads (<https://www.goodreads.com/>) with added ability of purchasing the books users have selected.

The website will include pages for each of these core details/features: home/root, about us, registration, login, profile, book browsing, bookshelf managing, and checkout.

Users will sign up for an account using a basic email/password authentication scheme, after which they will be allowed to use the full site’s functionality.

Users may browse for books on a dedicated page (/books), then either add a book to a Bookshelf (just a collection of books) or a shopping cart.

Users view their profile information and update their personal data, even deleting their account if they’d like.

Users can create as many Bookshelves for their account as they’d like, then add as many books to each of them as they wish. We’ll have to design our data scheme to account for this kind of scalable nature.

Books will consist of basic information needed to order the physical material (ISBN code, title, author’s name, publisher and version).

**Install Instructions:**

*Requirements: Git CLI, Java IDE*

1. **Pull Source:**
   1. Open your MacOS/Linux/Windows Terminal or command line interface.
   2. Run ‘git clone <https://github.com/DanielCender/CST-341-O500-CLC.git>’ to pull the source code from the project repository
   3. Run ‘cd CST-341-O500-CLC’ or ‘chdir CST-341-O500-CLC’ (Unix vs Windows commands)
2. **Open in Eclipse/IntelliJ IDEA/NetBeans**
   1. Open a Project from a local directory and import the directory downloaded by Git in the prior step
   2. (for Eclipse) Right-click the com.gcu.controllers.BaseController and select Run As > **Run on Server**.
   3. Open the in-IDE web browser and navigate to: <http://localhost:8080/Bookstore/>

*Include step-by-step instructions for setting up your database, configuring, and deploying/installing your application. This section should also include detailed instructions for what configuration files are required by your application, what configuration settings need to be adjusted for various runtime (development or production) environments, and where the files need to be deployed to. This section should also contain detailed instructions for how to clone your application source code from BitBucket and deploy the application to an externally hosted site.*

**Development Instructions:**

*Requires: Eclipse IDE, Git, Maven installation*

**Startup**

1. Open the /Bookstore directory as a Maven project in Eclipse.
2. RIght-click project, click Maven -> Update Project (will create /target dir)
3. Right-click project, click Maven -> Download Sources
4. Right-click project, click Run As -> Maven Build
5. Right-click project, click Run As -> Run on Server

The Tomcat server may need to be cleaned and restarted to update it’s referenced .war SNAPSHOT for the project.

**General Technical Approach:**

This program will be written using Spring MVC and utilizing Java EE as the base language. The Spring MVC framework will provide our visual page elements, styled according to the Bootstrap CSS framework with help from the jQuery JavaScript library.

**Key Technical Design Decisions:**

We decided, at this point to design a homepage that links to registration and login pages

By requirements for this project, we are committed to designing our application in a layered architecture to support development purely using the Spring MVC framework.

Although the project started its use of database interacting with a connection to a locally-hosted MS SQL Server instance, that proved to be too unwieldy for most class contributors to handle using functionally. As a result of that, significant extra time was put in during the week 5 milestone to switch the project over to using an embedded Apache Derby database instead.

When the project is served from Tomcat, the Spring bean initializes the database with raw tables and populates some seed data from a script (*src/com/gcu/insert-data.sql*).

*Any final technical design decisions, (e.g., framework decisions) should be documented here. List the technology/framework, its purpose in the design, and why it was chosen.*

**Known Issues:**

* Some of the form validation is not tightly coupled to the proper fields in the UI views. Some time will be needed during polishing to refine how validation is performed upon form submissions for products and user logins/registrations.

*Any anomalies or known issues in the code or functionality should be documented here.*

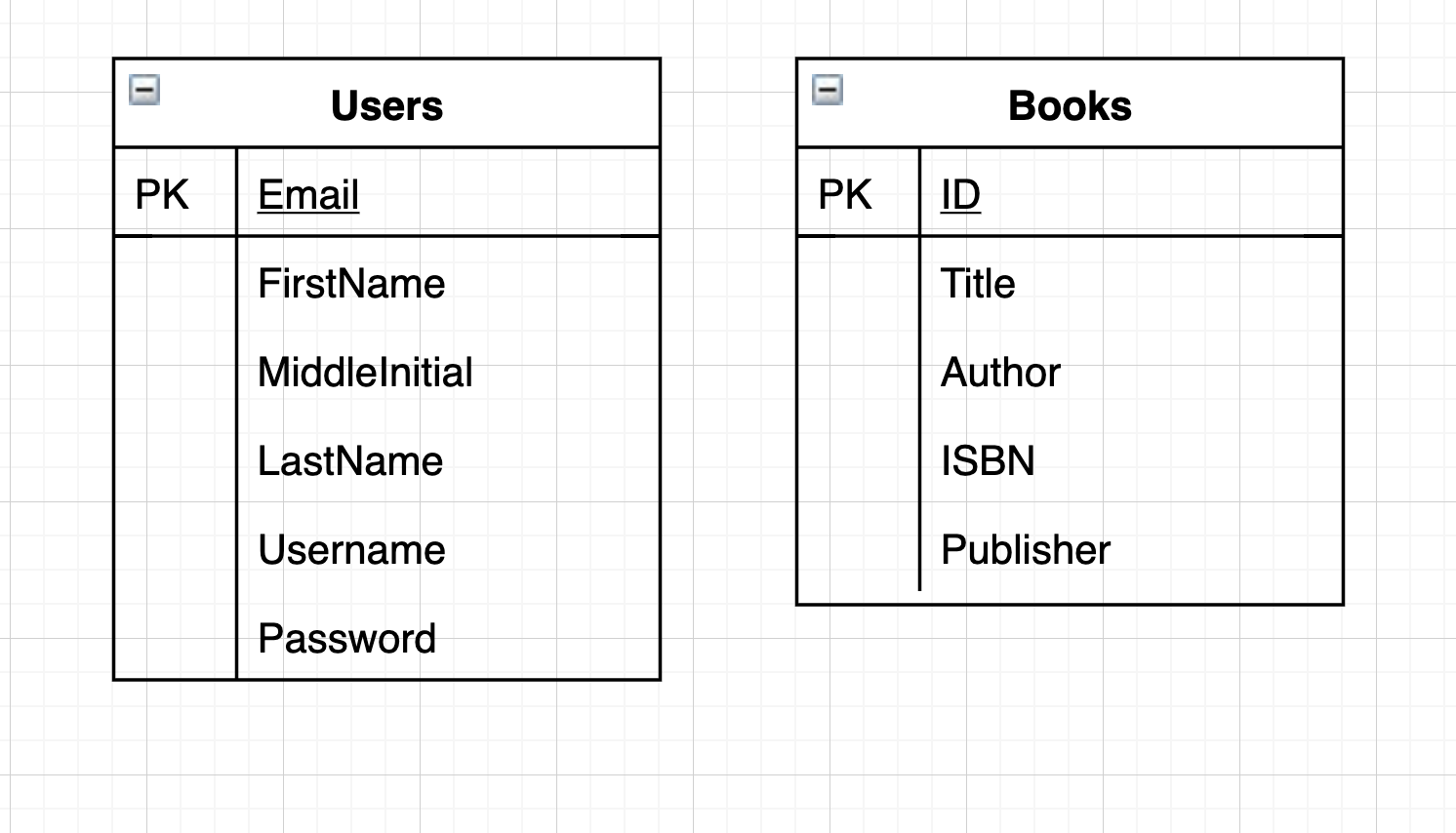
**Risks:**

At this point, there are no risks, besides the gaping security vulnerability described above.

*Any risks, unknowns, or general project elements that should be tracked for risk management should be documented here.*

**ER Diagram:**

The diagram below is for the tables in the Bookstore database:



*Include an image file of your ER database diagram.*

**DDL Scripts:**

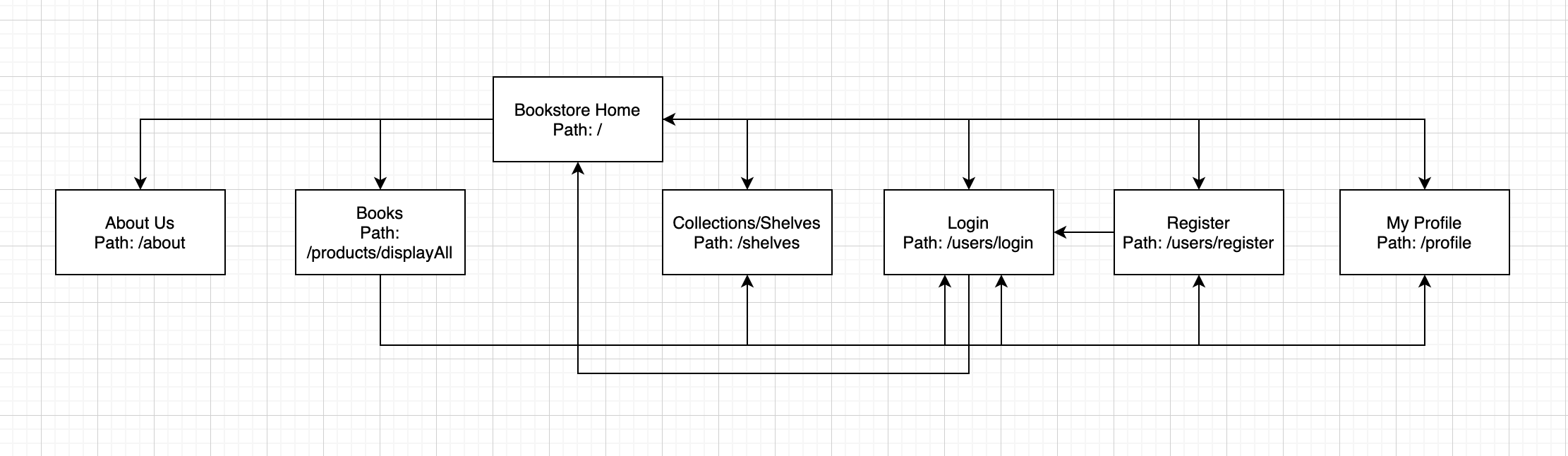
<https://github.com/DanielCender/CST-341-O500-CLC/tree/master/Bookstore/src/com/gcu>

The DDL Scripts, both for table creation and data seeding, will be located in the src/com/gcu package directory. The “create-db.sql” script defines the schema, and “insert-data.sql” seeds the tables.

*This should contain a link to Bitbucket from where the DDL script can be downloaded.*

**Sitemap Diagram:**

*Include an image file of your Sitemap diagram.*

****

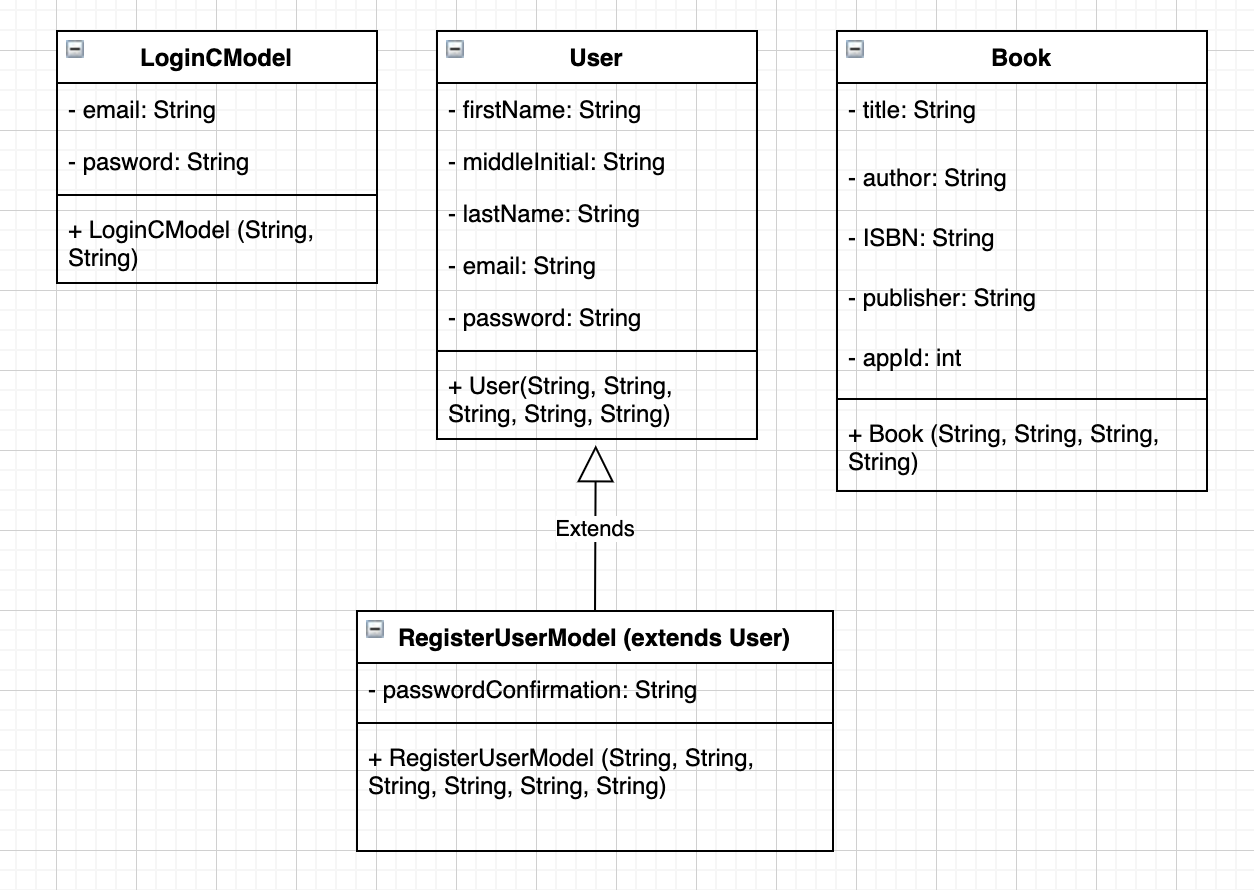
**User Interface Diagrams:**

Although the registration, login, main, and browse pages have been constructed roughly, no final designs have been white-boarded or decided upon quite yet.

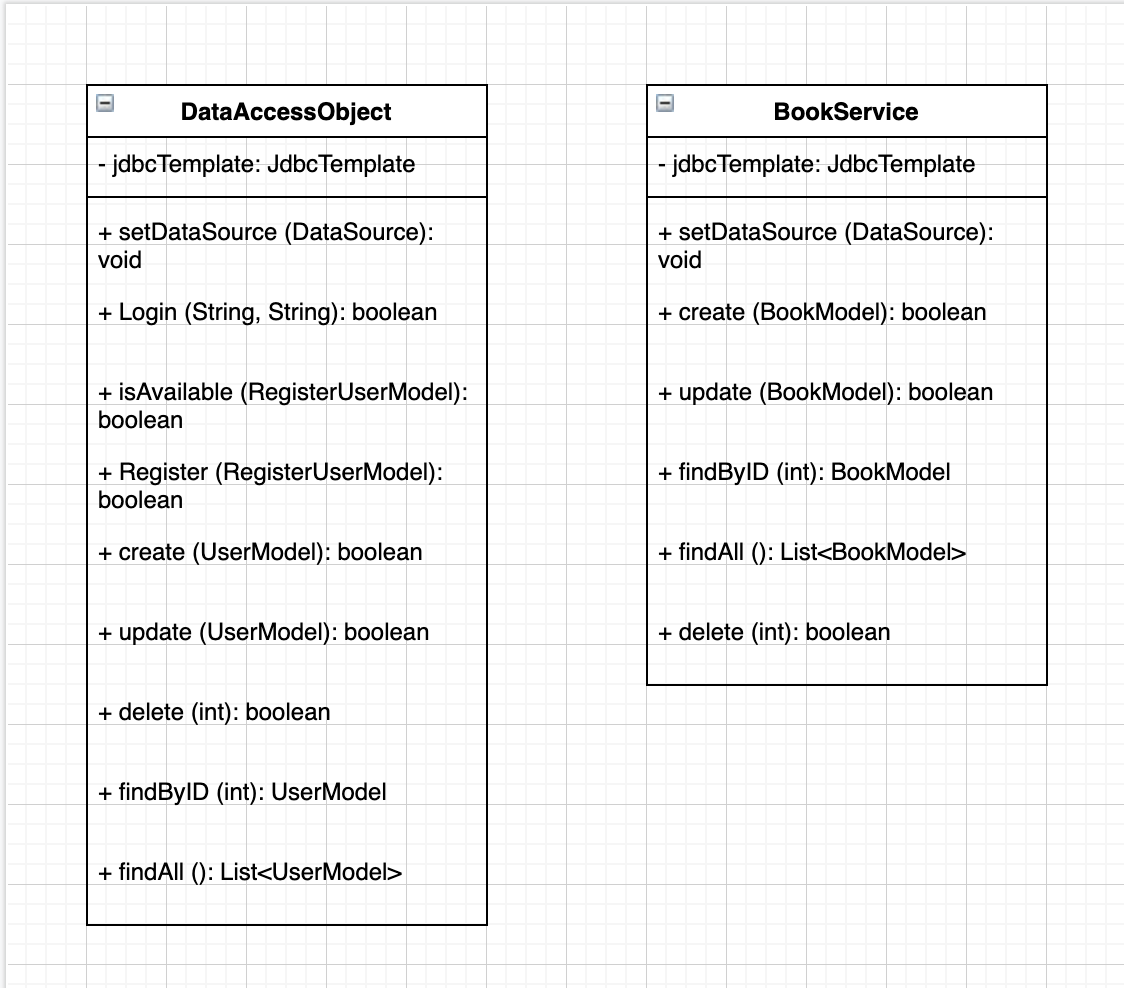
*You should insert any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for labeling this section N/A.*

**Class Diagrams:**

Data Model Class Diagrams:



Data Access Object Class Diagrams:



*You should insert any class diagrams here. Your class diagrams should be drawn correctly with the three appropriate class compartments, + and – minus to indicate accessibility, and the data types for the state/properties as well as method arguments and return types. If you have no supporting documentation, please explain the rationale for labeling this section N/A.*

**Service API Design:**

No API’s are being consumed at this time.

*This section should fully document any Third Party Service Interface API’s being consumed or application specific Service API’s being published, how to access the service, what parameters are required by the API, and the detailed JSON data format specification that could be used by athird party developer to integrate with the service and API.*

**Security Design:**

Pages are restricted based on a simple authentication scheme. When a user attempts to access an authenticated route, a check is performed for a valid user session. If no session is found, an alert is shown and the user is redirected to the /users/login page. If a valid session exists, the user may access and view pages and data without interruption.

*This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.*

**Other Documentation:**

*You should insert any additional drawings, storyboards, white board pictures, project schedules, tasks lists, and so forth that support your approach, design, and project. If you have no supporting documentation, please explain the rationale for labeling this section N/A.*