**CST-341 Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | CST-341-O500 Open Source Computing | |
| **Date:** | March 22, 2020 | |
| **Revision:** | v0.4.0 | |
| **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** | | *2,3,4* | Daniel Cender | *9* | *0* | | *1* | Zachary Swoveland | *3* | *0* | | *n/a* | David Pratt Jr | *n/a - busy with new baby* | *n/a* | | *n/a* | Branden Manibusan | *n/a* | *n/a* | | *n/a* | Tyler Rosenberry | *n/a* | *n/a* | | |
| **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:**

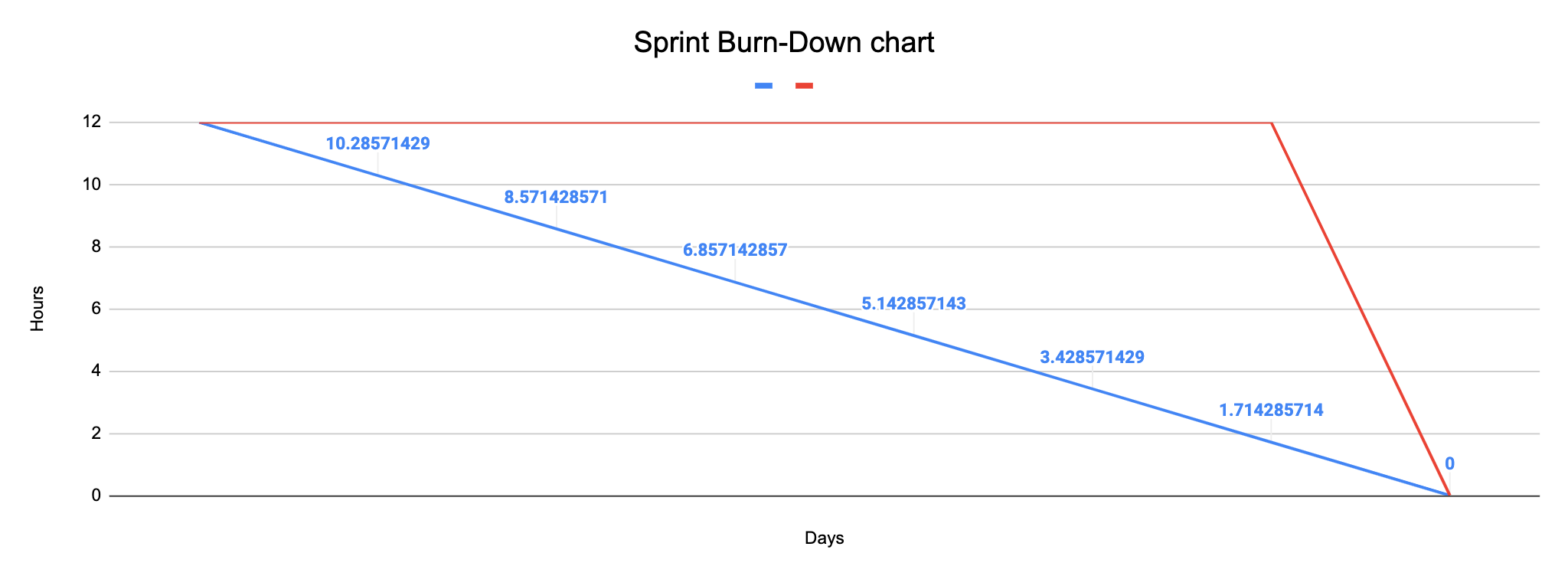
The link below leads to the current sprint backlog column (called “To Do”) on our group’s GitHub Project.

<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:

<https://docs.google.com/spreadsheets/d/1PNOYsftqeVIRJt-AYIHZD1wPgKpHpb5rbvtvKXAohWg/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** |
| Waiting until the work on our weekly Activities is finished to start researching and applying new tooling to our project. | We need to share more resources earlier in the week and do heavier research into the tooling outside of the course materials. | March 23, 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/>

***TODO -*** Consider switch to Apache Derby Database and updating documentation to reflect.

Using the application requires a running Microsoft SQL Server instance.

The DDL script should be run on this server to create the “Bookstore” database and tables.

On my (Daniel Cender’s) machine, this connection string was used to authenticate the JDBC driver with the instance:

*String connectionUrl = "jdbc:sqlserver://localhost:1433;databaseName=Bookstore;user=sa;password=Passw0rd1!;";*

The string **can be edited** to reflect the local user’s server credentials in the com.gcu.ApplicationConfiguration.java file, line 42.

*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.*

**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.

*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:**

* For an unknown reason, the form validation (using the @Valid annotation in the UserController) is not throwing errors for invalid UserModel property values. As such, if an email is invalid, it will create a record in the Users table with an empty Email column. It follows that since the password field is an empty string too, a user could just enter an empty registration form and then submit an empty login form to sign into the application.

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

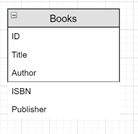
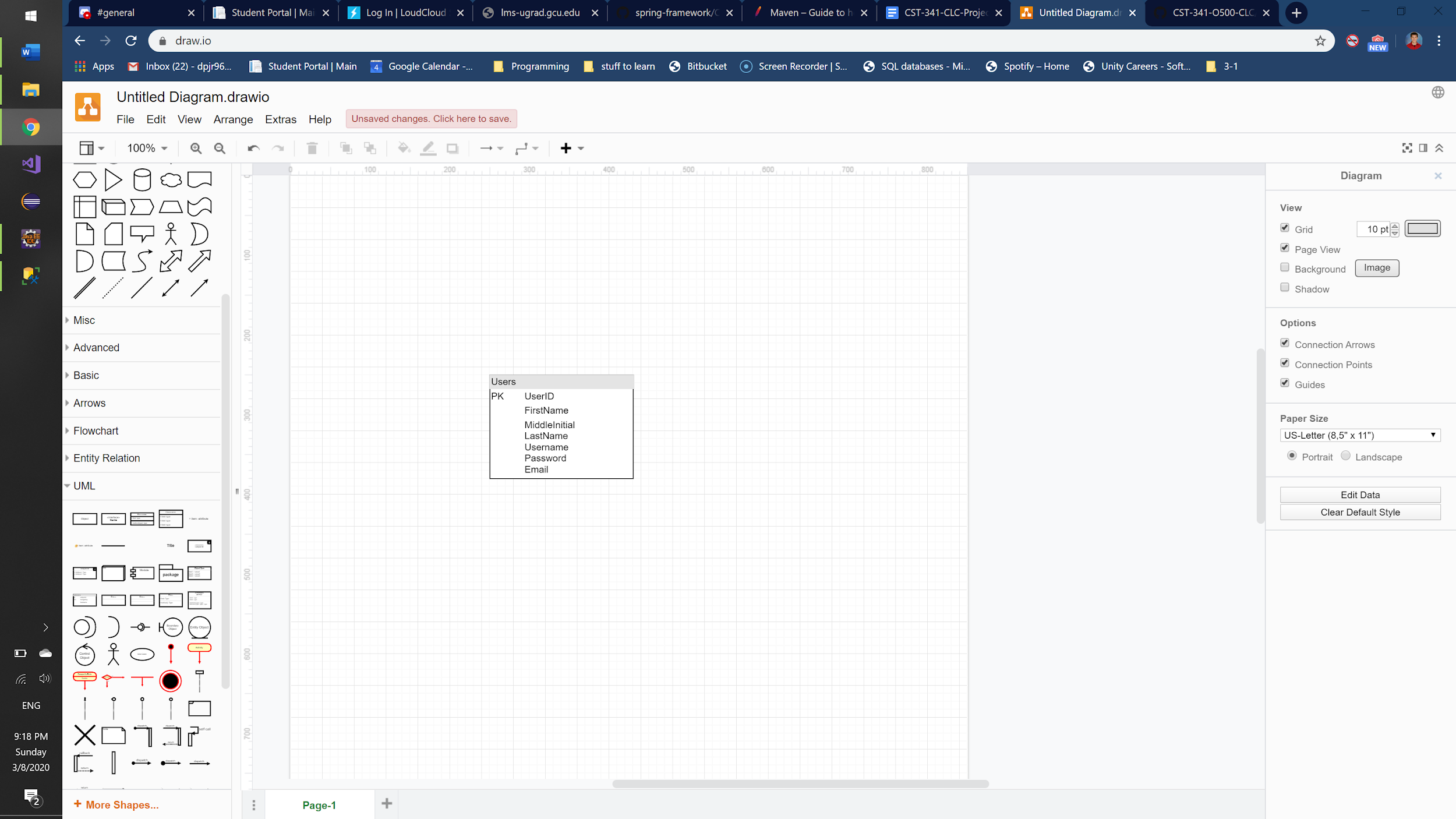
**Risks:**

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 Users table in the Bookstore database:



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

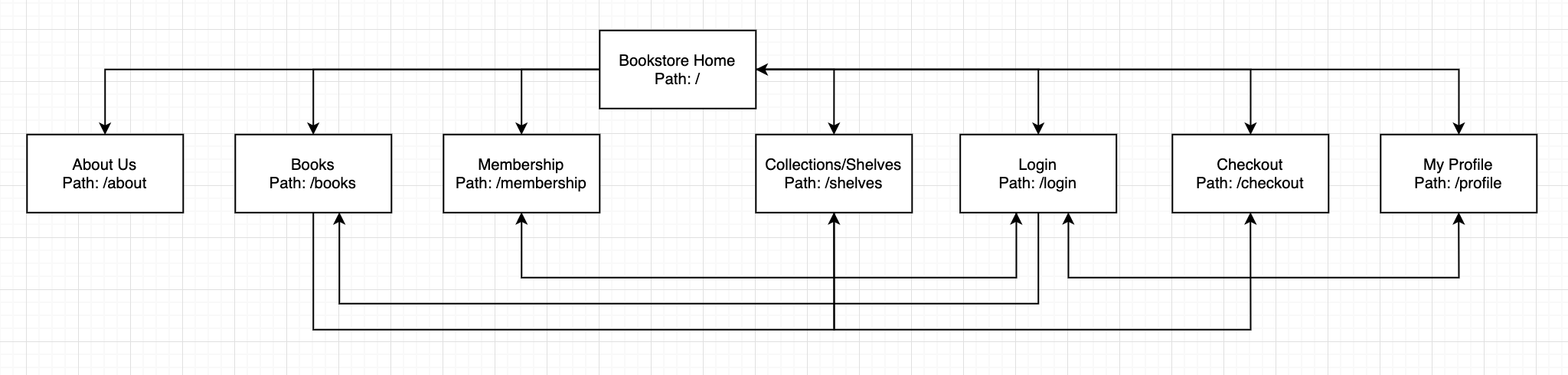
**DDL Scripts:**

<https://github.com/DanielCender/CST-341-O500-CLC/Milestone2/>

*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.*

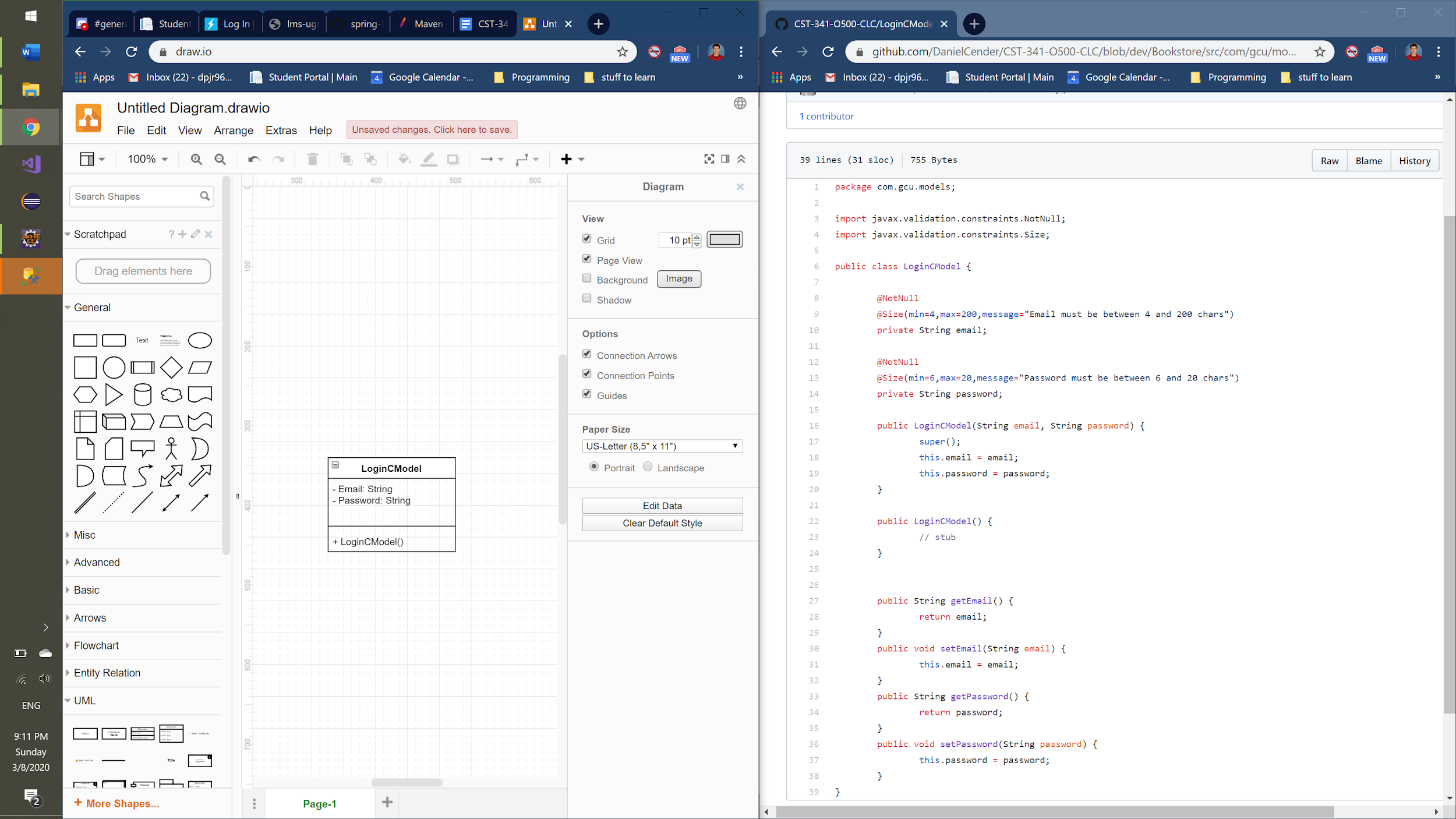
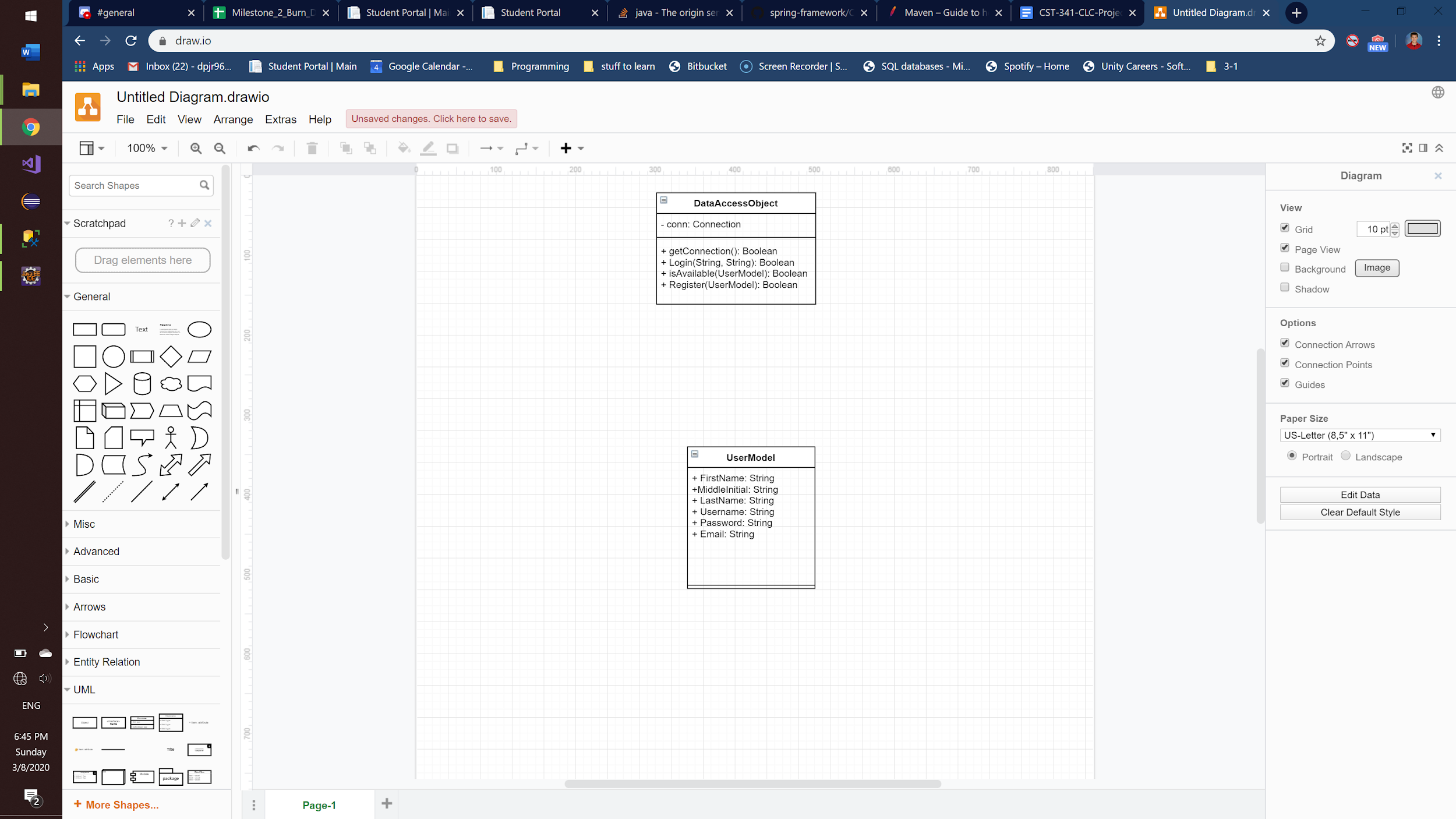
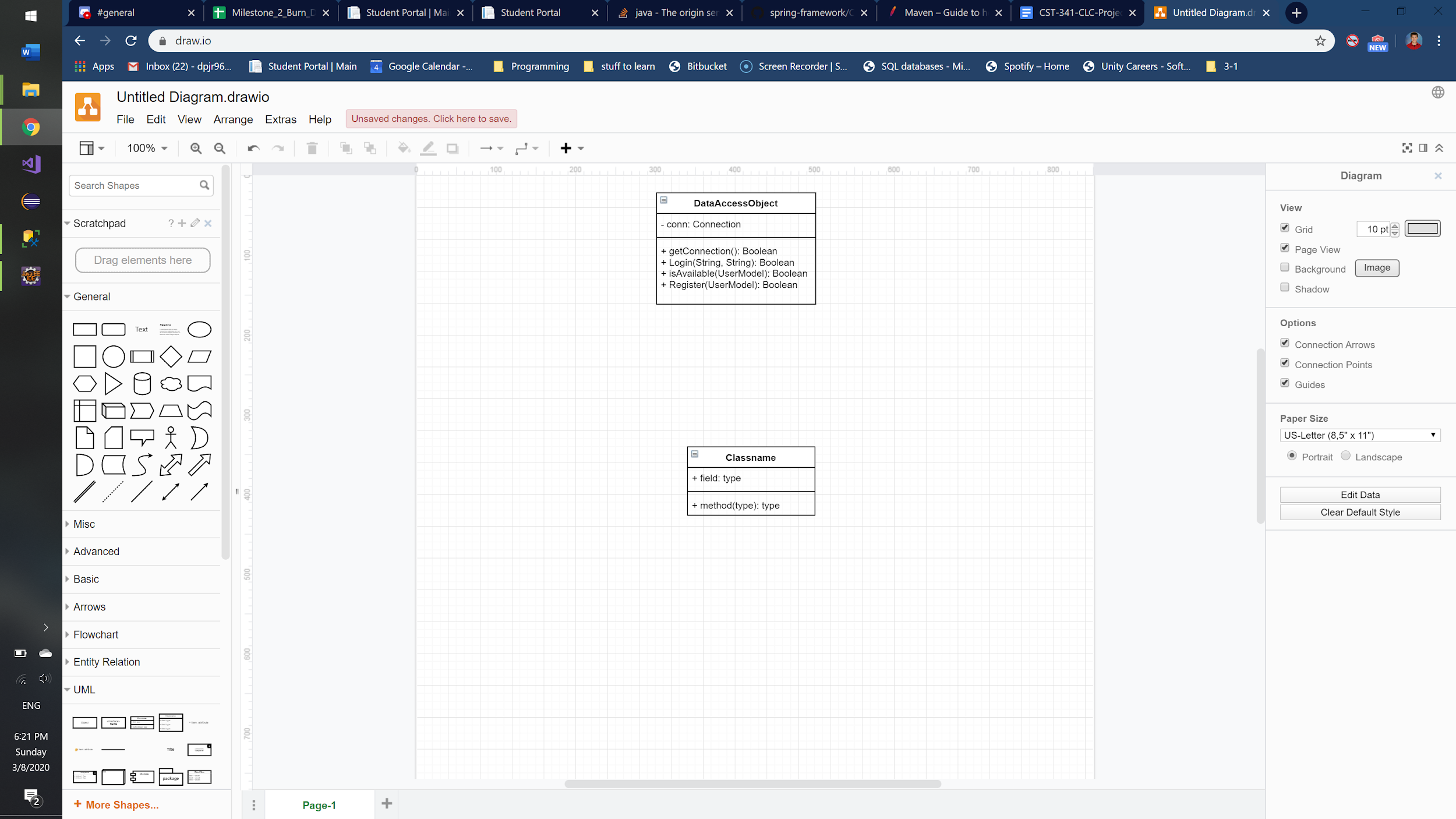
****

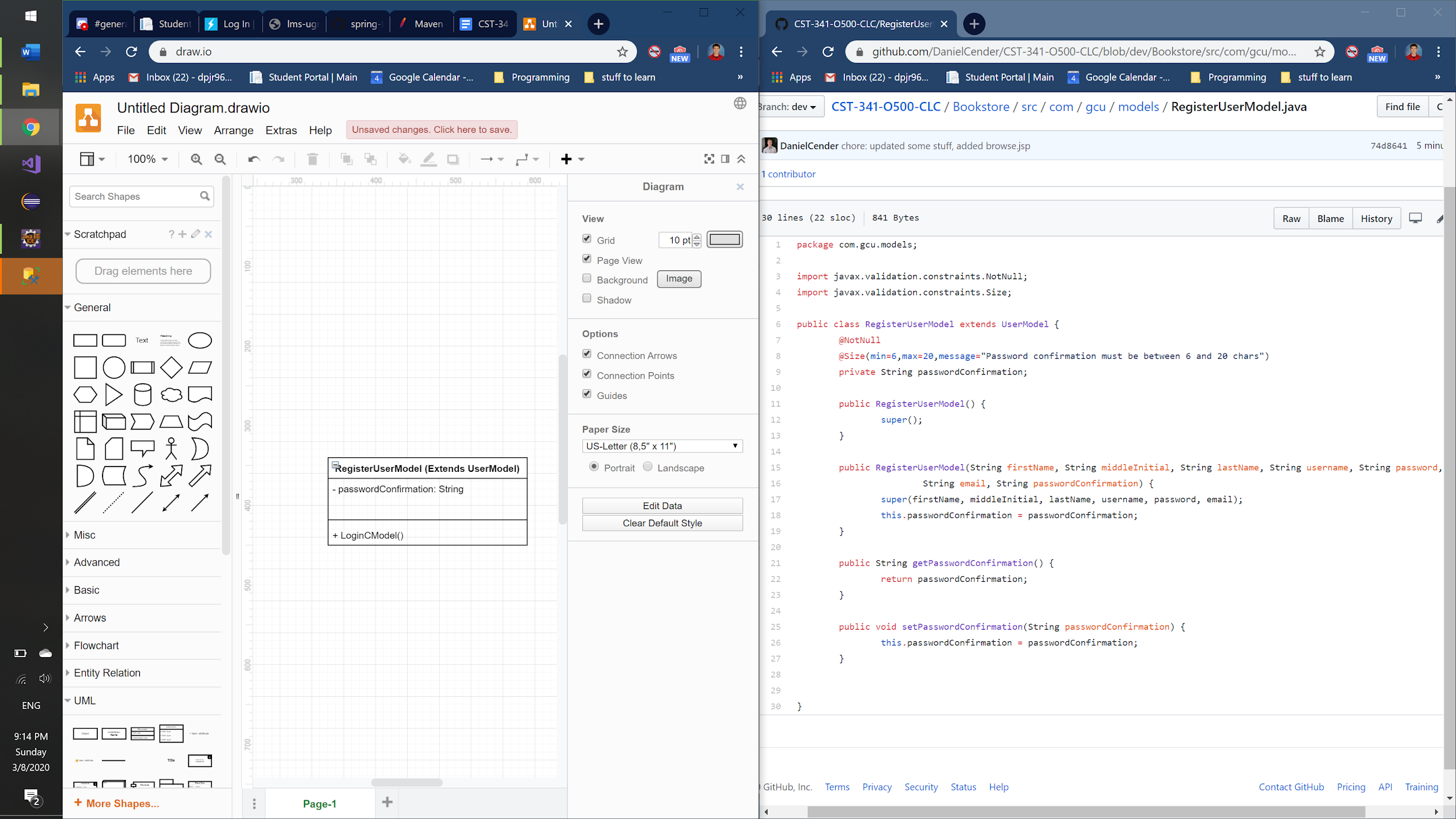
**User Interface Diagrams:**

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:**





*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:**

Security, at this point in the application, includes a login page which allows registered users to login and eventually will be the security behind which all secured pages are protected.

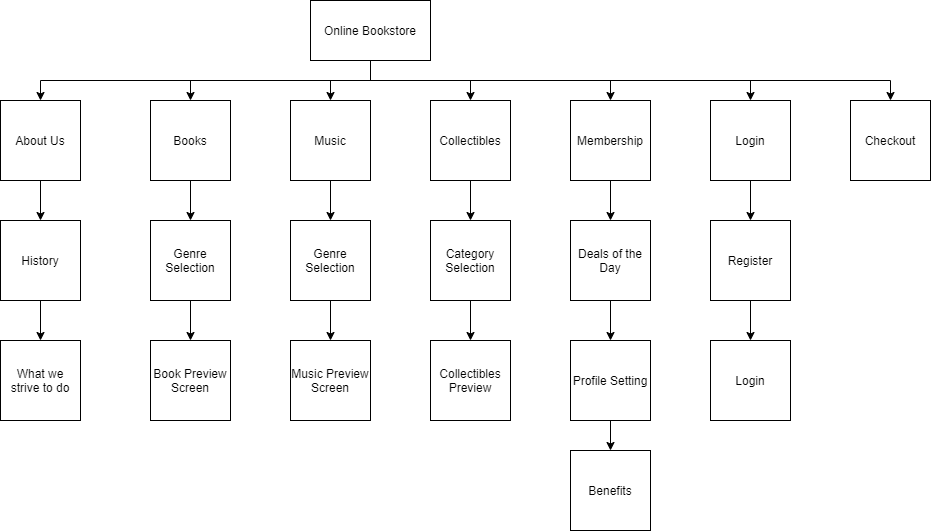
Security for the project users will be enforced via a login account scheme. Users will log into our application and have delegated access to their own book selections (“shelves”) and shopping carts (if applicable).

There may be an administration login role which can be applied to “super users,” such as developers or administrators who will require elevated permissions to edit or test the application.

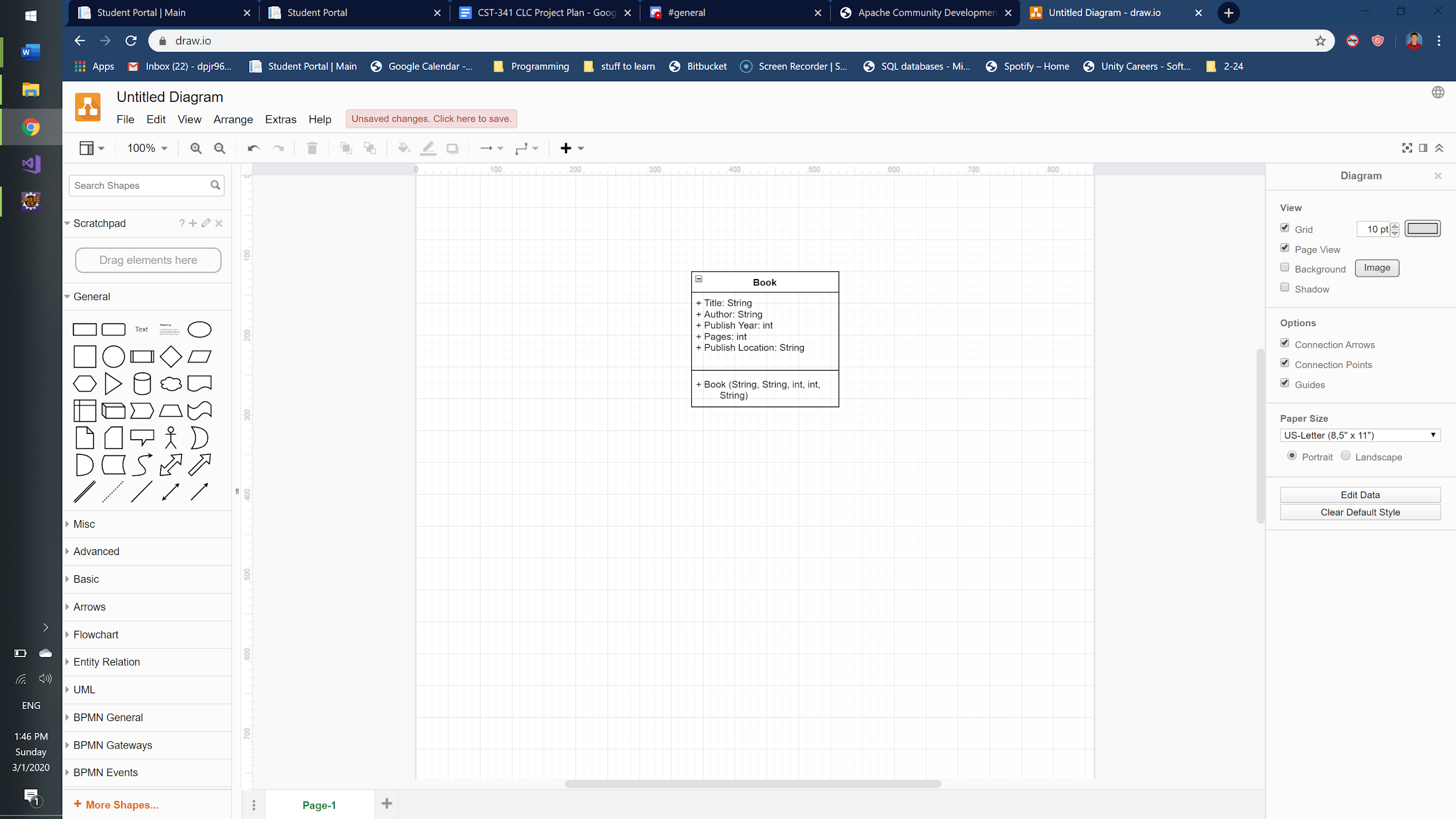
*This section should outline the design for how authentication and authorization was supported. This section should also contain all of the roles and privileges that are supported by the design.*

**Other Documentation:**

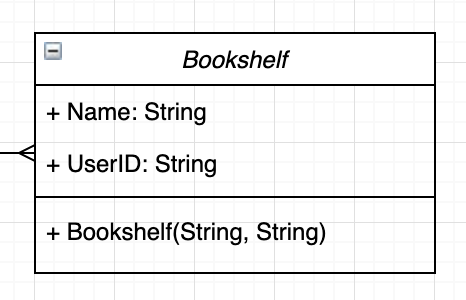
Below is a mapping of potential features that have been brainstormed for a handle of the pages in this application.



Below is a UML of our basic “product” which, for our bookstore, is books:



A UML diagram for a basic Bookshelf class, which users may create for storing books they’d like to purchase or read:



*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.*