**CST-235 Java Application Programming Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | Milestone 6 | |
| **Date:** | August 16th, 2020 | |
| **Revision:** | 0.6 | |
| **Team:** | 1. Daniel Cender | |
| 2. Chance Anderson | |
| 3. Tim James | |
| 4. Marc Teixeira | |
| **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | *Home page design. Xml security config. loginError page. Login form basic j\_security setup* | *Tim James* | *4* | *0* | | *Documentation updates, spliced together video presentation, added Postman tests and collection* | *Daniel Cender* | *4* | *0* | | *Worked on registration form security. Added JavaDoc tags. Generated JavaDoc files.* | *Marc Teixeira* | *4* | *4* | |  | *Chance Anderson* | *0* | *0* | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | <https://github.com/DanielCender/CST-235-CLC.git> | |
| **Peer Review:** | *Y/N* | 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:**

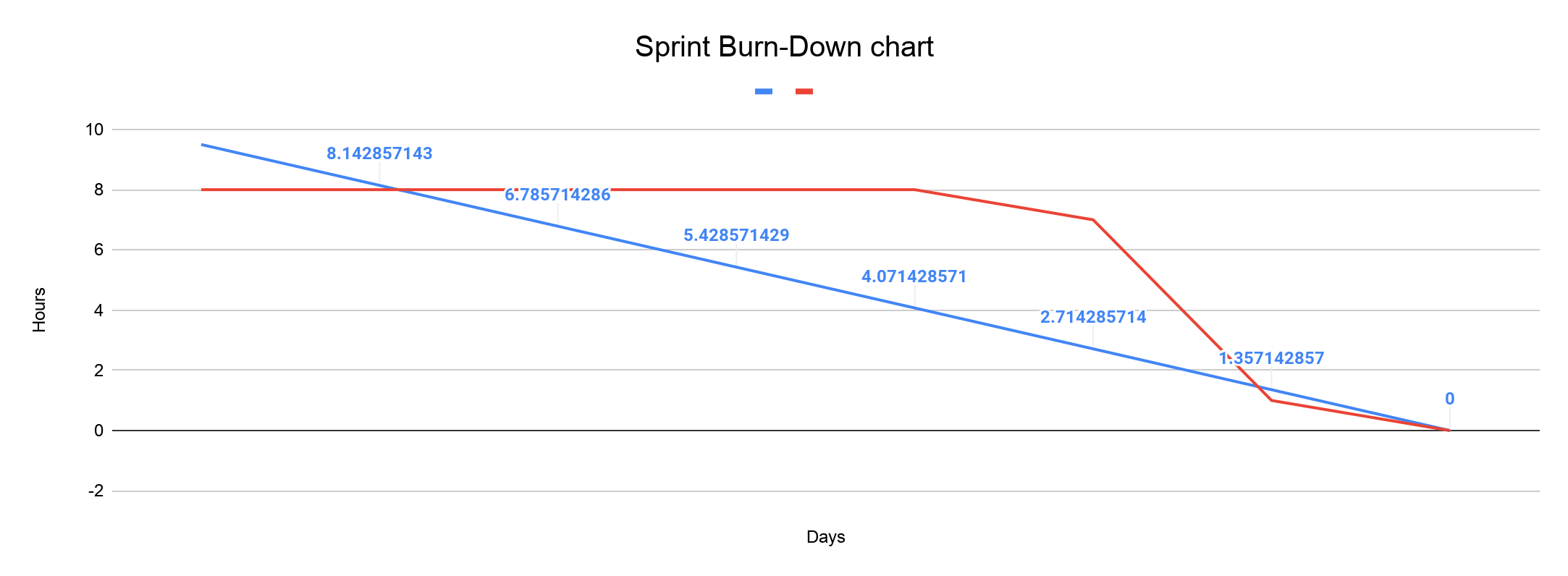
GitHub Project Backlog:<https://github.com/DanielCender/CST-235-CLC/projects/1#column-9863567>

**Agile Scrum Sprint Backlog:**

GitHub Sprint Backlog:

<https://github.com/DanielCender/CST-235-CLC/projects/1#column-9884199>

**Agile Scrum Burn Down Chart:**

****

**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 the URL or image file.*

|  |
| --- |
| **What Went Well** |
|  |
|  |
|  |

*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 the URL or image file.*

|  |  |  |
| --- | --- | --- |
| **What Did Not Go Well** | **Action Plan** | **Due Date** |
|  |  |  |
|  |  |  |
|  |  |  |

**Design Documentation**

**Domain / Product Write Up:**

For our final milestone, it is our goal to program a complete blogging website. As the domain will be a blog, the product will have a variety of features included in it. The blog will include title, author, content, date created, date edited, date deleted, and include tags as descriptors to increase searchability.

Users will have the ability to create, edit, delete and tag their own posts. They will be able to have and change their own username and profile picture and update their personal page with information about themselves. Users will also be able to maintain a list of favorited/liked blogs and users.

**Install Instructions:**

Before running the web application, PostgreSQL must be set up and running. Once a PostgreSQL database has been established the create-db.sql file must be run to set up the proper schema and tables. The insert-db.sql file can then be used to add mock data to the application for testing. Once this has been done PostgreSQL’s connection string information can be set in the data/DataAccessInterface.java file. In the file you can change the PostgreSQL URL, username, and password to their local settings. The default settings assume a fresh installation of PostgreSQL, using the default ‘postgres’ database.

In order to authenticate with this application, the user’s database record will need a corresponding record in the JBoss server’s user set. In a Terminal (Mac) or Command line (Windows), navigate to the JBoss server’s location on the disk. “cd” into the /bin folder and run the add-user.sh script to add the new user to the Application users list. Do not specify the user for working with Remote services/connections.

**General Technical Approach:**

*Describe your approach and design here in your own words. You should also summarize any meeting notes, brainstorming sessions, and so forth that you want to retain throughout the design of your project.*

**Key Technical Design Decisions:**

*Any final technical design decisions, such as framework decisions and so forth, should be documented here. List the technology/framework, its purpose in the design, and why it was chosen.*

**Known Issues:**

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

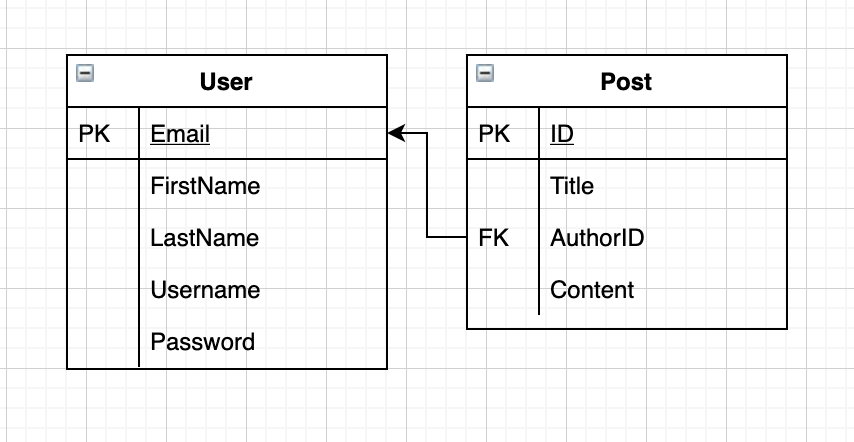
There’s one feature of this application which should be addressed soon as can be. When users view their own editable Post list, they are allowed to edit anyone’s Posts. Obviously, this is unacceptable in a consumer application. Greater security details must be implemented to create allowances for only the authors of a post to edit or delete it.

**Risks:**

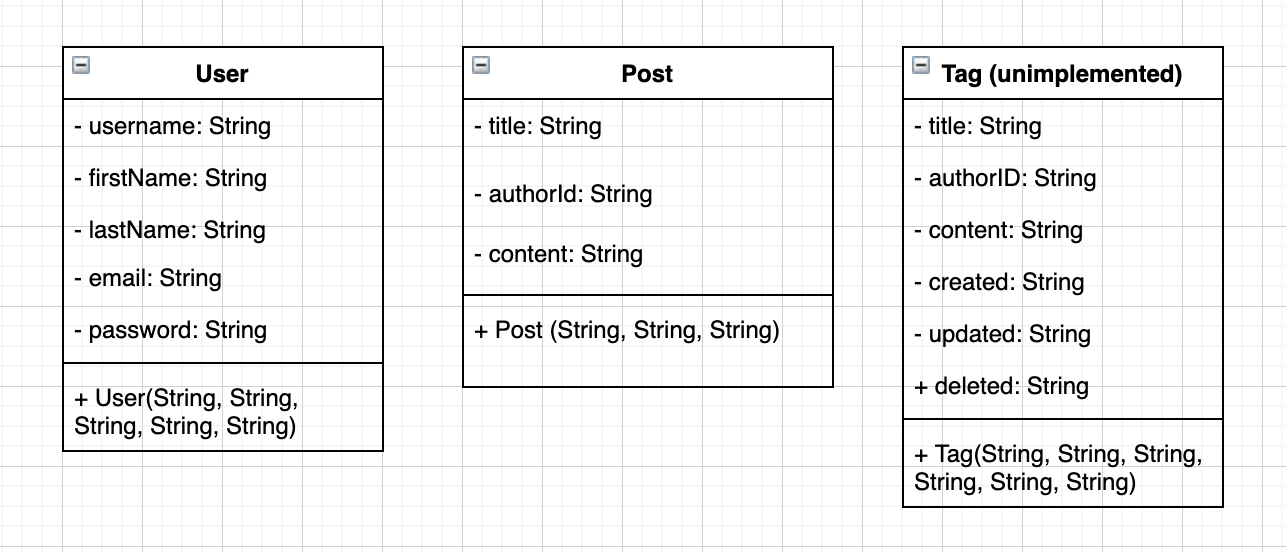
*Any risks, unknowns, or general project elements that should be tracked for risk management should be documented here.*

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Details | Strategy for Risk **Avoidance** | Strategy for **Mitigation** |
| My project design is too ambitious. | Features such as shopping carts, customer comments, appointment scheduling, date matching, accounting and inventory systems will take too long to develop in one semester. | Study the required elements in the project needed to finish the course | Keep scope limited to the design presented in the tutorials. |
| Computer failure | Lose data, code corrupted, hardware failure, computer too slow for required applications (JBOSS) | Purchase new computer. | Backup all code via Git to a central GitHub repository. Don’t save significant edits offline without committing to a feature branch. |
| Personal emergency | Sickness. Vacations. Work schedule is difficult | Realize timing is never perfect for these online classes.  Plan out personal work availability at the beginning of week and alert team to any weeks when availability will be a problem. | Work ahead when I have free time. Keep weekends open for homework, without sacrificing all sanity just for a grade. |
| Learning curve too steep | I will spend too much time re-learning the basics of Java programming that was taught in previous classes. | Use StackOverflow | Use YouTube and StackOverflow |
| Internet service fails | Cannot submit work at 11:59 p.m. on Sunday due to a technical problem. | Upgrade to a reliable service. | Work at Starbucks. Use phone hotspot. Or alert team members to issue and have another one submit work. |
| Partner doesn't do his/her share | Project milestones are missed due to late or poor quality of work from team members. | This class isn’t formatted to allow pre-evaluation of work partners.  Have a frank conversation about your expectations and work habits...nothing else can be done really. | Do all the work yourself to avoid incomplete results.  There are no other possible work partners; we are stuck with our current team no matter what, essentially.  Just an unavoidable issue at times. You can’t both trust your team implicitly and plan for their possible failures, it’s oxymoronic. |
| Not sure how to accomplish a specific coding task | Questions to professor don’t get answered immediately. No response after 10:00 p.m. | Plan on finishing work one day before the deadline.    Get explanations from online documentation and tutorials. | Submit questions with fully documented problems: copy of source code and run-time results – errors or unexpected output. |

**ER Diagram:**

****

**UML Diagrams:**

****

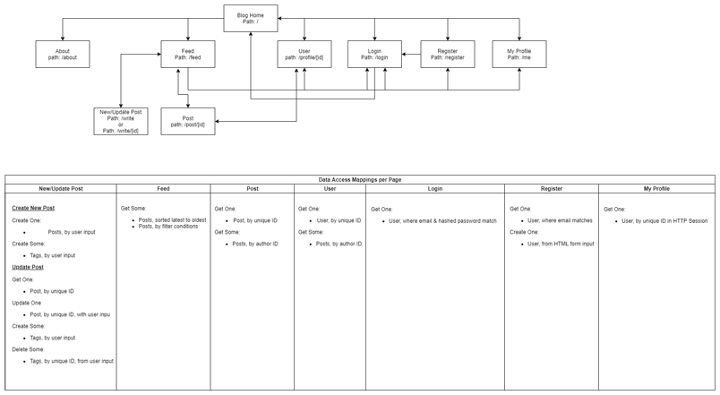
**DDL Scripts:**

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

Create DB Tables: <https://github.com/DanielCender/CST-235-CLC/blob/dev/create-db.sql>

Insert DB Tables: <https://github.com/DanielCender/CST-235-CLC/blob/dev/insert-data.sql>

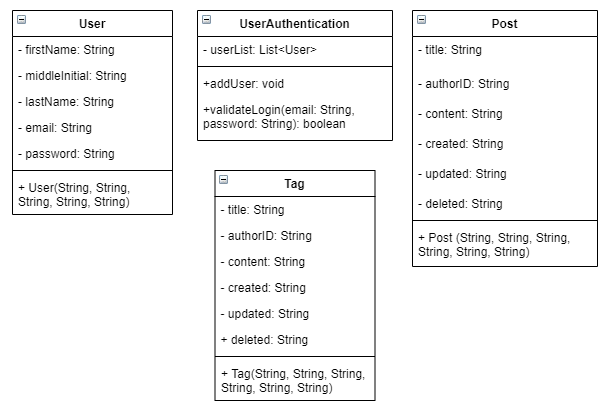
**Sitemap Diagram:**

****

**User Interface Diagrams:**

****

**Class Diagrams:**

****

**Service API Design:**

*This section should fully document any Third Party Service Interface APIs being consumed or application specific Service APIs 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 a third party developer to integrate with the service and API.*

**There are only a few routes offered to 3rd-party API consumers:**

* /rest/posts/{count}

This route has a URL param of “count” which specifies the max number of items to return. This parameter must be specified for a successful response.

* /rest/posts/post/{id}

This route has the “id” URL param for retrieving a specific Post record by ID.

* /rest/users/{count}

Returns users, limited to the specified number of records by the “count” URL parameter, much like the /posts/{count} route.

* /rest/users/user/{email}

Returns a specific user by their email (also serving as their ID) if a valid record corresponds to the URL “email” parameter’s value.

**Sample Route Responses:**

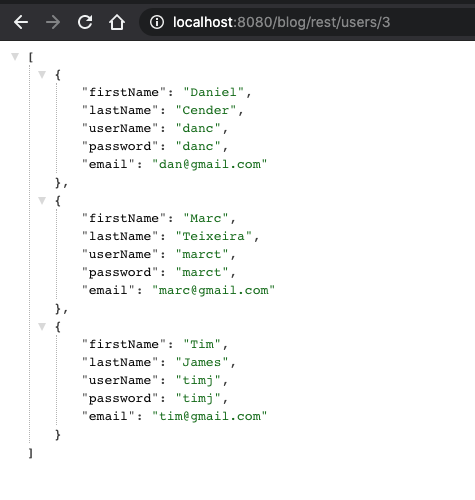
* /rest/posts/3



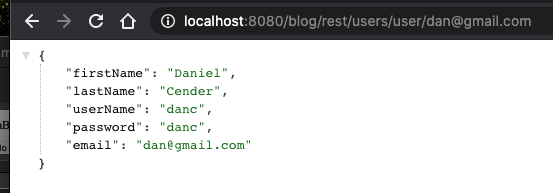
* /rest/posts/post/2



* /rest/users/3



* /rest/users/user/[dan@gmail.com](mailto:dan@gmail.com)



**Testing:**

To see the routes of this API in action, download the CST-235-CLC Blog App.postman\_collection.json file from the latest milestone (Milestone 6) of the GitHub repository and import that into Postman. Each of the four routes should include a very minor test suite which should validate correct responses from the API. *The tests assume that the local PostgreSQL database contains a post with an ID of 2 and at least 3 Posts and Users each.*

**Security Design:**

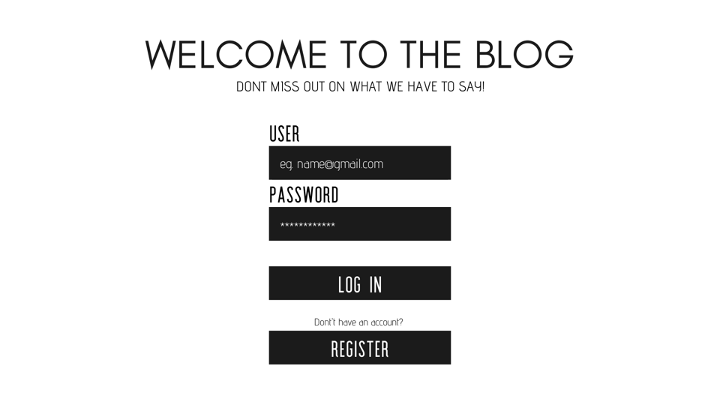
*This section should also contain all of the roles and privileges that are supported by the design.*

The security design of this application is a simple application of the JavaEE Security framework. Only “Application” users created on the running JBoss EAP server in the “BlogUser” authentication group are allowed as valid logins.

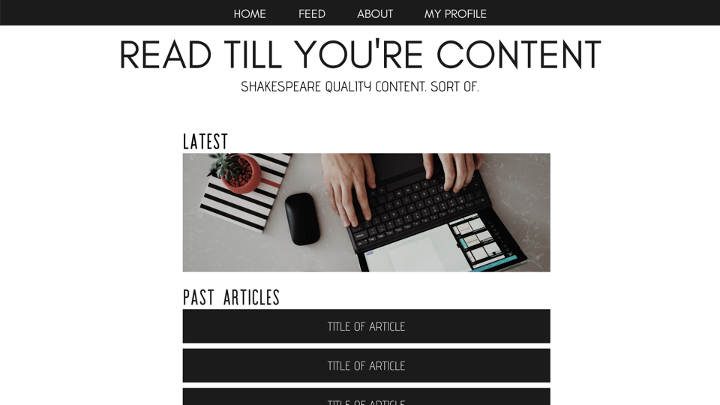
Because of that use of the Security framework, a user can register a database user with the /registration.xhtml page, but they won’t be able to log in until an app administrator creates their account on the JBoss server with the /bin/add-user.sh script.

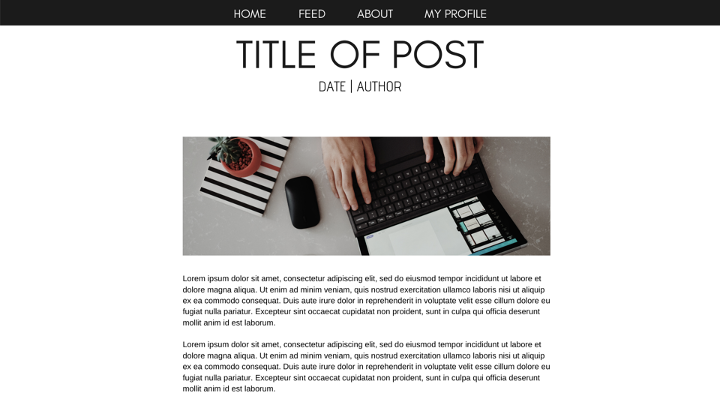
**Other Documentation:**

*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 why you are leaving this section as N/A.*

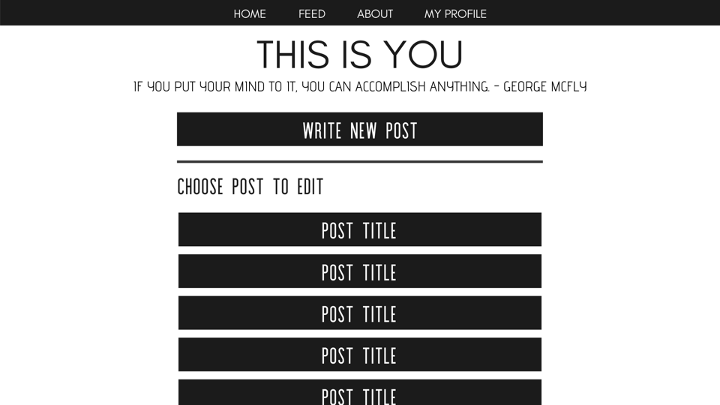
**

**

**

**

**

**

**