

Milestone 5

Charles Henderson, Pengyu Yin



April 20, 2019

Grand Canyon University

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Java III Application Programming**  **Project Status and Design Report**   |  |  |  | | --- | --- | --- | | **Topic:** | *Milestone 5* | | | **Date:** | *April 20 2019* | | | **Revision:** | *5.0* | | | **Team:** | 1. *Charles Henderson* | | | 1. Pengyu Yin | | |  | | |  | | | **Weekly Team Status Summary:** | |  |  |  |  | | --- | --- | --- | --- | | **User Story** | **Team**  **Member** | **Hours**  **Worked** | **Date** | | *Milestone code was translated from a lot of the in-class activities.* | *Charles Henderson* | *3:30* | *Feb 8th 2019* | | *We again both worked in the library to accomplish milestone tasks. (Diagram Focus)* | *Pengyu Yin* | *1:00* | *Feb 8th 2019* | | *Failed to make a false DB with a user List(discussed in known issues).* | *Charles Henderson* | *3:00* | *Feb 9th 2019* | | *Class Diagram, Entity Relationships* | *Pengyu Yin* | *3:00* | *Feb 9th 2019* | | *Changed project report of Milestone 2 to match the changed Milestone 1.* | *Charles Henderson* | *5:00* | *Feb 16th 2019* | | *Refactoring Milestone 1 and Milestone 2 diagrams to keep updated to the current idea (books).* | *Pengyu Yin* | *5:00* | *Feb 16th 2019* | | *Created product class and idealized how to maneuver from the home page to product page.* | *Charles Henderson* | *1:30* | *Feb 23rd 2019* | | *Planned ahead for what was expected in the coming milestone, specifically wireframes and page design.* | *Pengyu Yin* | *1:30* | *Feb23rd 2019* | | *Attempted MySQL DB creation alongside DBeaver desktop application.* | *Charles Henderson* | *1:15* | *Mar 2nd 2019* | | *Attempted MongoDB and DBeaver, but both of us fell short to making these work.* | *Pengyu Yin* | *1:15* | *Mar 2nd 2019* | | *Coded the create product portion of the site with product controller & updated report. DB in works.* | *Charles Henderson* | *3:00* | *Mar 9th 2019* | | *Drew up several more diagrams including wireframes of product pages & updated UML diagram.* | *Pengyu Yin* | *3:00* | *Mar 9th 2019* | | *Continuing product manipulation and CRUD methodology, implementing business interface logic.* | *Charles Henderson* | *4:00* | *Mar 16th 2019* | | *Developing from the front end inward, moving from the view to match business/data functionality.* | *Pengyu Yin* | *4:00* | *Mar 16th 2019* | | *Database work and DAO creation in DB layer logic.* | *Charles Henderson* | *3:30* | *April 1st 2019* | | *Worked greater changes to UI to add functionality. Fixed bug in facescontext to put current product page* | *Pengyu Yin* | *3:30* | *April 1st 2019* | | *Implementing REST services, form validation and fixing various bugs with DB.* | *Charles Henderson* | *2:30* | *April 18th 2019* | | *Creating navbar to better conform to our original product design and polish some of the feel.* | *Pengyu Yin* | *2:30* | *April 18th 2019* | | *REST service created with ability to produce JSON on any webpage.* | *Charles Henderson* | *2:00* | *April 20th 2019* | | *Used the DAO properly to send out a list of products versus orders from the hacked REST service ICA.* | *Pengyu Yin* | *2:00* | *April 20th 2019* | | | | **GIT URL:** | https://github.com/agingdanger/CST-235 | | | **Peer Review:** | *Y* | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |   **Design Documentation**  **General Technical Approach:**  We are still maintaining the MVC and N-Layer designs by fleshing out each of their components. In this milestone we have implemented the CRUD services for each of the individual store products that we have. This was done by also finishing off the login/logout methodology in the previous milestone. Once the user is logged in and verified by the DB security check, the navbar and links associated with the user will then allow them to maneuver through our webpages and search the different products as they desire. The majority of the concepts that were generally used in the weekly activities have been reformed to contour to the milestone. This is absolutely including the REST activity.  **Key Technical Design Decisions:**  We are still designing everything in sitemaps, but our first design outlined in the sitemap will have an index page, product page, login, and registration. Each of these will fan out into a few more pages that will complete a fully-fledged sales website as outlined in the specifications. As far as framework decisions we are still a little wary, however, we are sure that we will at least be using the JSF component-based framework. CSS and .xhtml pages will be used to contour our design and create an operationally tasteful website. The user and registration beans will properly associate and talk with the respective controllers to send the proper view back to the customer. We have also created a search ID bean that allows us to place on the searched webpage the current context of the product. This is done through several different methods such as but are typically associated with the context class FacesContext. We also created a User Controller that would take the form data from the newly created Registration Form. Similarly, we’ve now created a Product Controller and Login Controller that route all of the product and user business logic to adhere to separations of design and implement CRUD methodology. This will eventually allow for a more refined transition from the registration page to the login page and product page to product creation. While the login information is displayed in the data grid (a hack from the assignment 3b) we have moved away from this approach and instead are now using the data grid to house the product list on the modify product page. This will be continuing with the usage of DBeaver and PostgreSQL and can only be seen once the user has logged in.  **Known Issues:**  We discussed how to cross reference the user variables with the registered user with Professor Shad but without a database this proved difficult. The original idea was to create a user list that would house everything that a registering customer would input. The problem with this was that every time the list is re-instantiated after closing the program it would no longer remember the data that was input prior. I originally thought the user List wasn’t working because of the scope of the variable declared. After trying @ApplicationScoped I came to the conclusion it was the issue of instantiation. We realized the importance of implementing a DB to reference information, otherwise the data will be hardcoded and otherwise useless. // We actually thought of a solution to this in which we would place items onto a txt file and use that as a temporary database. However, since we started actually coding the database, we thought it would be better to focus on it as it would be a solution much closer to what the end product will look like. Another problem we’re potentially having is how exactly to route previous orders the user has placed. We’re seeing necessity of more DB tables in the future.  We have since fixed the above issues since milestone 3, but have excitingly created new ones. One of our largest issues that we were able to fix was actually a problem in generating the product into a textbox based upon the search results of a specific id. Being on this bug for a couple days, we talked to Prof. Jackson and were able to rubber ducky out of ourselves an idea that would end up leading to the solution of this problem. This was also done by hacking one of the previous in-class activities to get usage of the current context product bean.  **Risks:**  We are worried that the scope of the project could potentially be too large and could perhaps overcomplicate if we go much beyond what is outlined in the sitemap. Time management is still an area of concern for us because of our housing situations limiting us from meeting up as often as we want. Another concern is our shift in project design, which has caused to have to work a little harder. However, we are confident that through this if we stay diligent we can possibly achieve a greater result then previously outlined.  To validate our previous concerns, we are having to rollback a few of our intended functionalities like the product cart. Instead, users will be able to browse all the products on our page and register to exclusive book selling content! One of the last major risks that we are running into is waiting to code our CSS. Our idea is that we want to wait until we have a finished product functionally before we apply some of our aesthetic design. The risk of course here is that we are saving a decent portion of work for the very last moment.  The REST services came about fairly easily after a few hours of work focusing on Bible-API.com. Utilizing string concatenation, we ran into issues processing the JSON so I just opted to have it print in the console.  **Entity Relationship Diagram:**    **Data Dictionary:**      **DDL Scripts:**  [**https://github.com/agingdanger/CST-235/blob/master/M2.sql**](https://github.com/agingdanger/CST-235/blob/master/M2.sql)  **Sitemap Diagram (Simplified Design):**    **User Interface Diagrams:**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **A screenshot of a cell phone  Description automatically generated**  **Class Diagrams (Of Current Build):**    **Security Design:**  Although there really isn’t a security design implemented yet, there are some basic security ideas that we’ve learned about so far that will probably be inside the program. Some of these include a basic authorization that will cross check session variables to ensure the user is who they have been verified to be. There are also coding regulations set in place through things like the MVC design which isolates concerns and is thus a key point of security to keep sensitive data away from a layer or section of code that is much more easily accessible.  The only added security design so far is a empty field error message to prevent unnecessary code from being typed into fields.  We have created DB’s with basic access from DBeaver and Postgres. Eventually, we will create user authentication and roles so that access to this will only be able to be done by proper DAO’s in the 6th milestone.  **Other Documentation: Project Run-Through (Moves with flow of data): Updated 4/20/19**  **JSON is placed into internet browser:**  **C:\Users\Pengyu\AppData\Local\Temp\Rar$DRa6180.1436\rest\Screen Shot 2019-04-18 at 14.54.13.png** |
| **Method in Rest Service class:** **C:\Users\Pengyu\AppData\Local\Temp\Rar$DRa6180.8408\rest\Screen Shot 2019-04-18 at 15.07.01.png** |
| **DAO method that converts what’s found in REST service to object:** **C:\Users\Pengyu\AppData\Local\Temp\Rar$DRa6180.24753\rest\Screen Shot 2019-04-18 at 15.07.33.png** |