LE EECS 4413
2017-2018 Winter Term
Project Report

Name	CSE Account	Student ID	GitHub
Jun Lin Chen	[redacted]	[redacted]	mc256
Chenxing Zheng	[redacted]	[redacted]	loveritsu929
Vishal Malik	[redacted]	[redacted]	vishal0027

Live Demo: https://citrus.yuri.moe

Table of Contents

Read Me

We are publicly hosting a demo site for this project on https://citrus.yuri.moe/
If you found any question about deploying this project or using the website, please contact one of the team members.

Program

Test User:

- Username: [redacted]
- Password:

Test Administrator

- Username: [redacted]
- Password:

The program source code will be publicly available on GitHub 24 hours after the submission deadline (or upon request). It will be available at the following addresses.
https://github.com/mc256/Citrus/
(Or https://github.com/YorkU-Citrus)

The program requires the following dependencies:

- `mariadb-java-client-2.2.3.jar`
- `jersey-bundle-1.19.1.jar`
- `javax.json-1.0.4.jar`
- `javax.json-api-1.1.2.jar`
- `javax.ws.rs-api-2.1.jar`
- `jstl-1.2.jar`
- `taglibs-standard-impl-1.2.5.jar`

For demonstration purpose, many application interfaces are publicly available to the open Internet. The tomcat is running on a virtual machine, in case the site got hacked, we can still restore the system easily.

The system structure of the demo site is shown below.

Database

We are hosting the database server for testing.

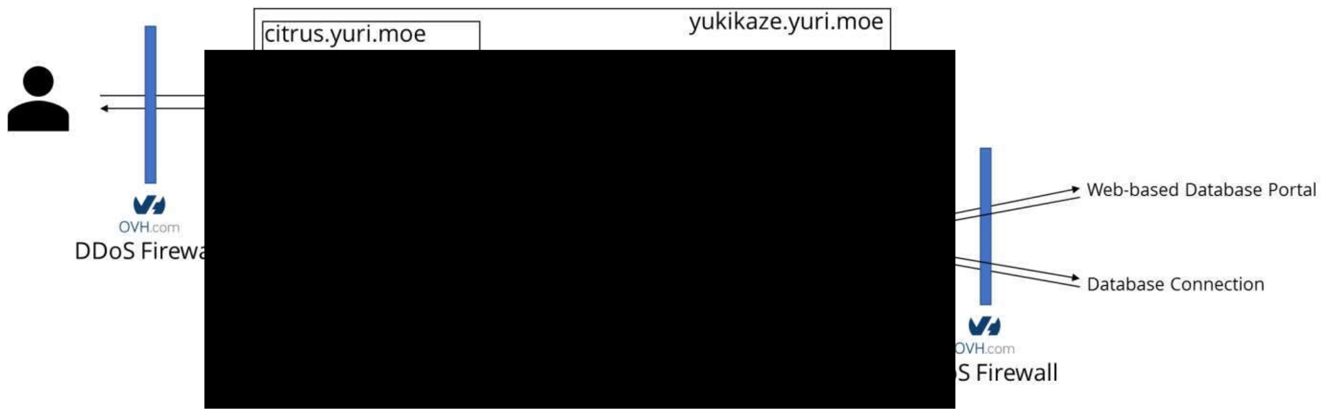
- Database Ho [redacted]
- Database Po
- User name:
- Password: P

To prevent potential attacks, we only allow the connection from the white-listed IP addresses. We have added York University (ASN802, IP range `199.212.64.0/24`) to the whitelist. You may access the database from the York University Keele campus.

The database is using MariaDB, which is the open source community version of MySQL. But it uses the same scheme (`MyISAM`) as MySQL. The required JDBC connector (`mariadb-java-client-2.2.3.jar`) is in the WAR package. You may need a different JDBC connector if you want to connect it to an Oracle MySQL server. Right now, mainstream Linux and BSD distribution include MariaDB instead of MySQL. It is better to support the open source community for using MariaDB.

The web interface for the database (`PhpMyAdmin`) is also available online. You can access this page using:

[redacted]



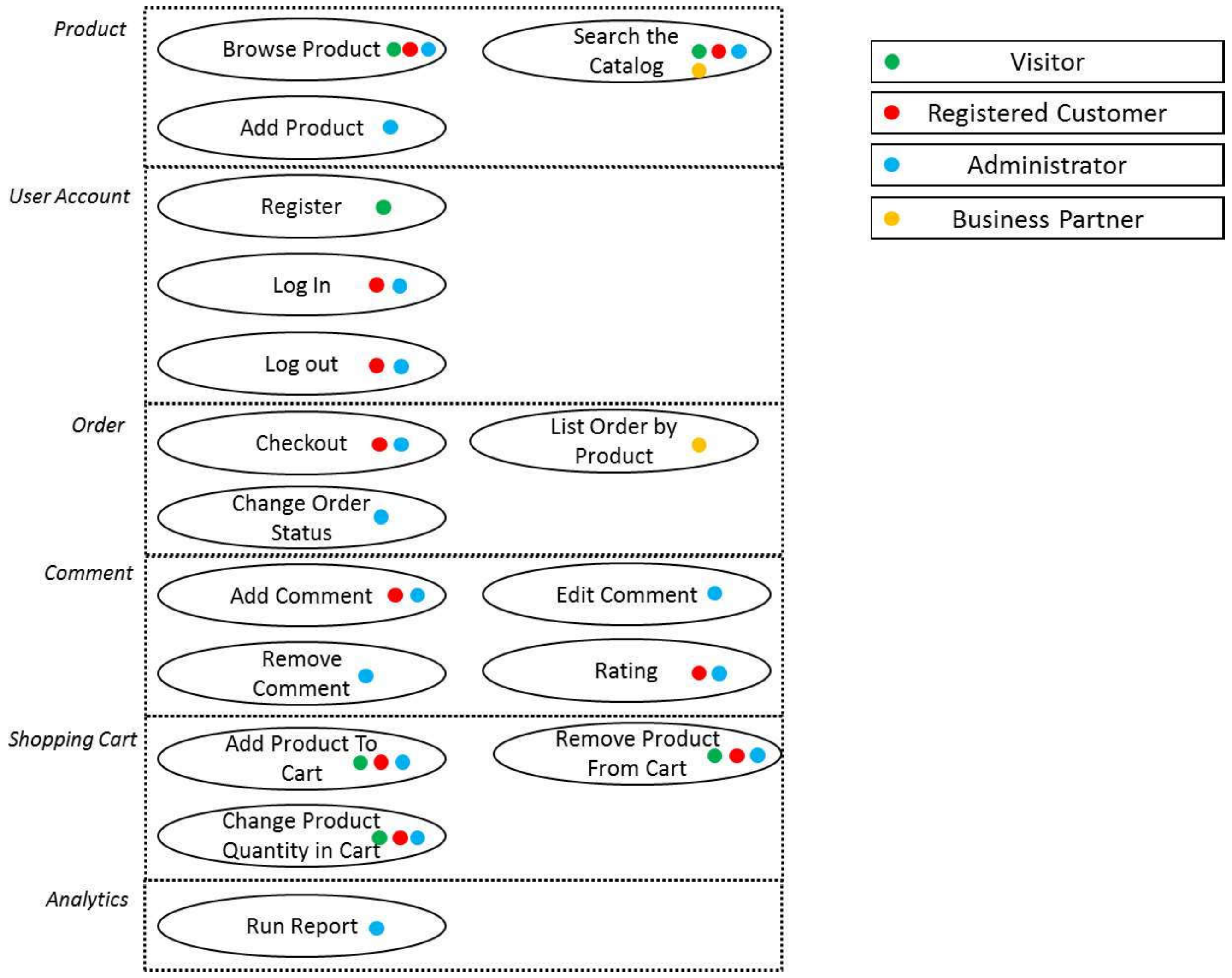
[redacted]

Architecture

Program Structure

UML Use case

Because we have too many stakeholders and features, we use color to match the use cases and the stakeholders instead of using lines to connect them.



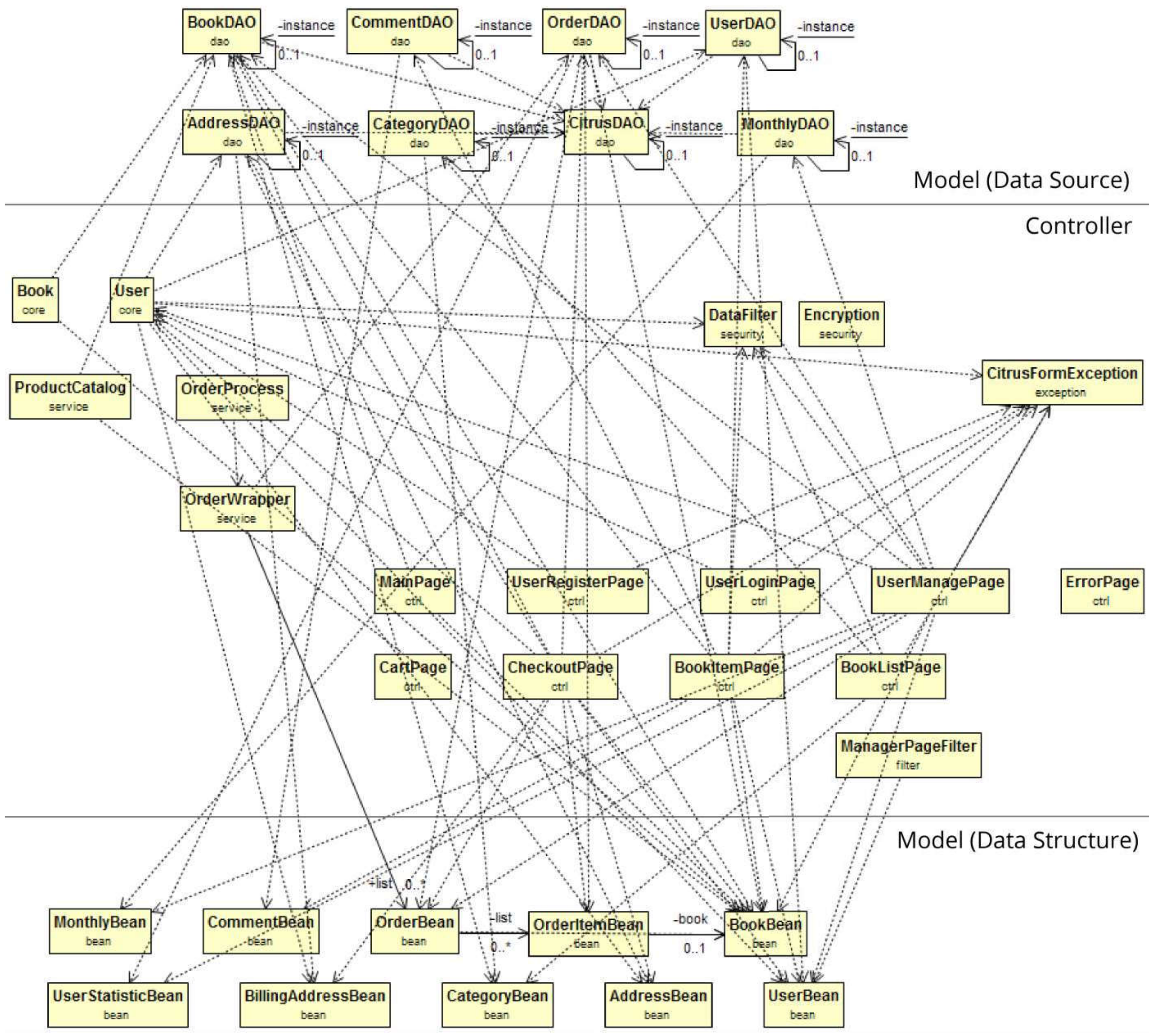
Class Diagram

Descriptions of all the packages:

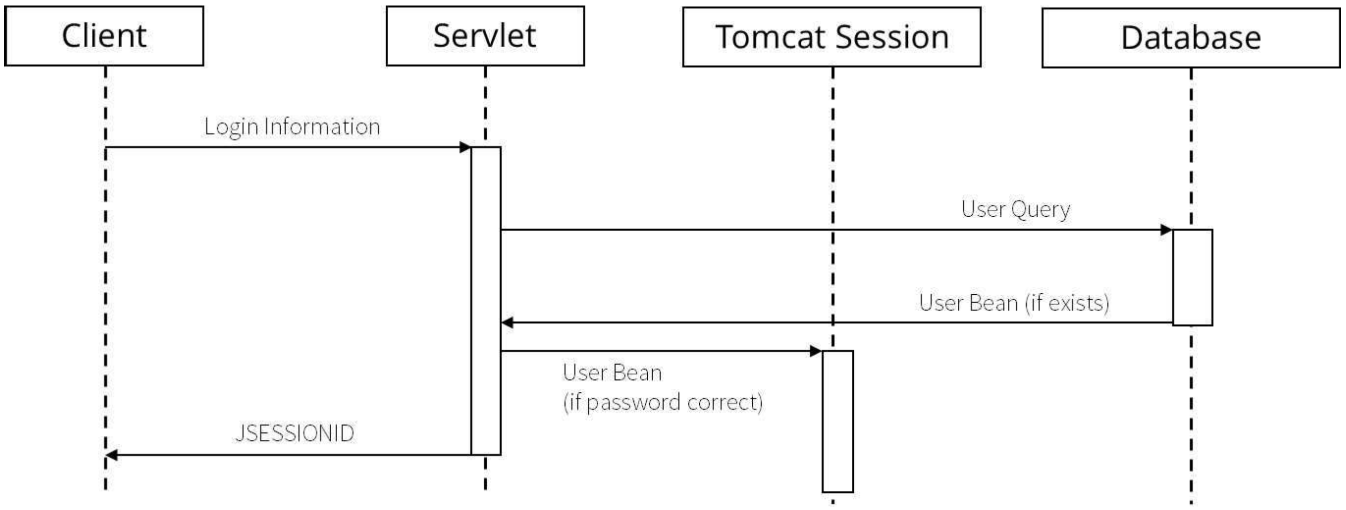
- **bean**
  This package contains the fundamental data structures for a data object in this project. All the data handling methods exchange these objects.
- **core**
  This package contains the REST APIs. It mainly built for the web page. Therefore, we decided to JSON instead of XML to reduce the overhead. It is also a part of the internal controller of this e-commerce system.
- **ctrl**
  All the Java classes are servlets. It gathers necessary components and data for the web page. It is the controller of this e-commerce system.

- **dao**
  This package contains all the data access objects. All the objects are singletons so that we do not have duplicate connections, which reduce the number of database connections. All the database operations are also atomic which avoid race conditions. It is the model of this system.
- **exception**
  This package contains an exception class specifically for this project.
- **filter**
  This package contains some filter to filter out illegal access to the web service.
- **security**
  This package takes care of all the encryption and anti-injection in this class.
- **service**
  This package is also the API. But they are mainly built for the project requirement B, C and I.

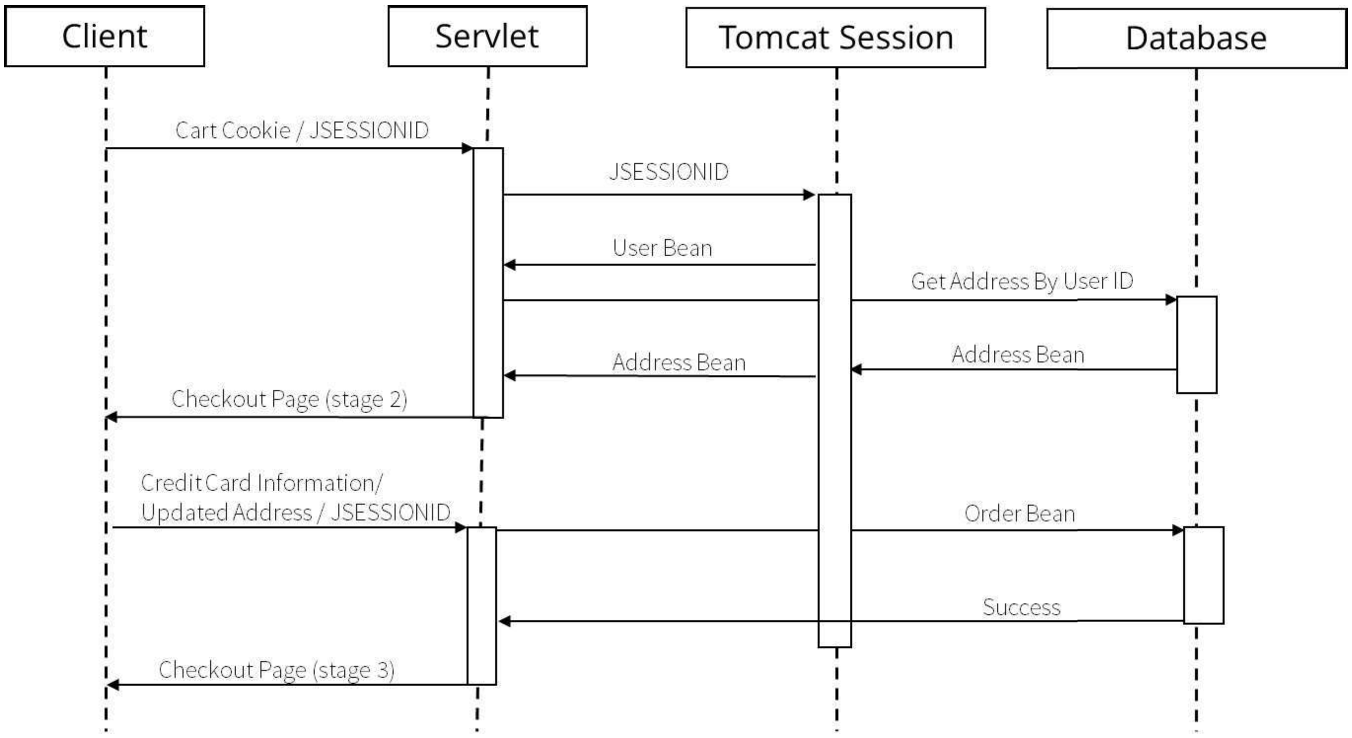
The unlisted part in this class diagram are the Views, which are the JSP files.



Sequence Diagram for a successful user login.



Sequence Diagram for a successful user checkout.



Design Decision

1. Error Handling

There are many considerations about the error handling. The exception message breaks the program flow which makes the program's behavior more difficult to predict. It also makes some part of the code difficult to cover when we are doing unit testing. In some recent design, for example, Apple drops the support for throwing exceptions in their latest programming language Swift. We are also trying to think how to use the exception mechanism efficiently.

In this project, we try to handle all the exceptions on the stage of the controller. Therefore, we can somehow display the reason for the exception on the page. For other classes, we only throw or re-throw the exceptions (except the unit test written in the main function). We make this as the first design decision because it is very important to have a concise programming style.

2. Event-driven Design

As the UML class diagram shows, all the models are highly modelized. We are using the event-driven design for this e-commerce system. All the operations are motivated by an event starting from the servlet or REST APIs.

3. Service Aspect

This program has different stakeholders. Different stakeholder requires different features. We can group the service into the following aspects. We are using Servlet, Jersey, Apache Axis as the interfaces to the user, but behind this, we are using uniform components for handling the data and communicate with the database.

4. Special Requirement Highlights

Fulfilling special, odd, weird and unreasonable requests from the clients, are the important and essential part in real business life. In the project requirement PDF, we found two special implementations. To reduce the impacts on our origin system structure, we do some special workaround for these points.

- “hard code that every 3rd request is denied on your website.”
  We did hard code this in the checkout page class, using the session to count the number of requests.
- “provide ammonized reports with user buying statistics.” (as an adminstrator)
  In my point of view, the administrator should know the customer’s name so that that the book can be sent to the client. It makes no sense to hide user’s name in this report. We define an extra method in the user bean to generate the asterisks on the username.

5. HTML5 Web Page Design

In the limited amount of time, and we are not allowed to use any web front-end CSS libraries or JavaScript libraries, we decided to drop the support for the outdated browsers. Currently, the project’s stakeholders are young and clever technicians. They usually use an up-to-date browser and using the cutting-edge device. Hence, we decided to use HTML5 with JSP.

We are also using a responsive design for all the pages. All the pages are crafted carefully using CSS. Our site supports most of the modern browsers on smartphones. Furthermore, the book catalog page loads the list of books using AJAX as you scroll down the page. All the operations are smooth and elegant.

6. Design for High Concurrency

We are using MariaDB (MySQL), which is one of the most popular databases in open source community. We rarely do update operation on the database; we merely do insert which may reduce the locking time on the database. When we want to read the data, we always selecting the latest record. This decision increase the reading time but decrease the writing time, which is good if you have multiple slave database servers.

All the database operations in our data access objects classes are atomic. It avoids creating race condition in the program scope.

7. Separate static and dynamic component of the pages.

It is possible to set up reverse caching proxy for the static pages. Those pages (web resources) rarely change, it can be distributed to a CDN to reduce the loading delay.

Implementation

A. Data Access Component

This implementation includes two solutions. It uses the JDBC connection pool when it runs on Tomcat with the context configured. When you run the program as a Java Application for debugging or developing purpose, it automatically falls back to a direct database connection.

All the DAO classes relay on the CitrusDAO class. You can check the CitrusDAO class for more detail.

B. Product Catalog Component/ Service

This part is simply generated by the Eclipse Java EE using Apache Axis. Because all the components are highly modularized, the actual coding for this service only takes one line.

Demo URL: https://citrus.yuri.moe/Citrus/services/ProductCatalog

C. Order Process Component/ Service

We are using Jersey for this implementation. Our XML scheme is different than the scheme that the project document provides to fit our design. It is also available online (https://citrus.yuri.moe/order-list.xsd).

In our design, the "part number" is the "book ID." This service returns the list of orders with contains a specific book. You may also provide the "offset" and "limit" parameter to get a specific range of result which is similar to a SQL query with the LIMIT keyword.

Demo URL: https://citrus.yuri.moe/REST/XML/

D. Session Controller

We use the default session service provided by Tomcat. To make things simple, we use event-driven design, eliminate the use of cookies on the client side and the use of memory on the server side. The session information (e.g., billing address, shipping addresses) will only load to the session controller when it is needed.

E. Book Store Main Page

For the main page, we planned to implement a big slider underneath the navigation bar, but we do not have enough time to achieve this objective, it is just a big image for now. After this big image, we list the top four sellers. And there is a box for content.

We use AJAX and REST to implement the requests in the category pages and search result page and using JSON as the data format. The web page automatically loads more books when you scroll down the page.

Demo:

- UC M1: https://citrus.yuri.moe/list?category=1
- UC M2: https://citrus.yuri.moe/item?id=15
- UC M3: (Please login as user "test") https://citrus.yuri.moe/item?id=3
  To post a comment for a book, you must have a completed order which contains the book. You must change the "ostatus" in "citrus_order" table to "COMPLETED".
- UC M4: (Please use the search input box on any page)
- UC M5: (Please click on the "Add to cart" button)
- UC M6: (It is on the right end of the menu bar.)

F. Shopping Cart Page

The shopping cart is implemented on the client side using a cookie. The data is stored in a JSON format of text in a cookie. The JavaScript on the page analyses the cookie and update the web page. A tiny red bubble on top of the shopping cart icon indicates how many items are in the shopping cart.

Demo:

- UC C1: https://citrus.yuri.moe/cart
- UC C2: (On the same page as above) When the quantity decreases to zero, it will not disappear unless you refresh the page. This design prevents unwanted removes.
- UC C3: Payment button is on the right-hand side of the page.

G. Payment Page

Sessions and cookies carefully protect the access to the payment (checkout) pages. The entire process can be separated into three stages.

The first stage concentrates on verifying user's identity. Starting from the shopping cart, the non-login visitor enters the first stage asking for login or register. The login visitor skips the first stage and directly jump to the second stage. The second stage focuses on the shipping address, billing address, and the billing information. The page loads user's information from the database if they have been provided. If they have not yet provided. The third stage is the final stage. It shows whether the order is processed.

Demo Page:

- UC P1-4:
  You cannot access these pages directly. Please start from https://citrus.yuri.moe/cart

H. Analytics Page

This page is only available to the administrator. In the user table, we have a field to identify the role of a registered user. And we use a filter to check whether the user's role is administrator or not.

Demo Page:

- UC A1-3: https://citrus.yuri.moe/manage?type=analytics
  (Please login as a "MANAGER." You can set your "urole" in table "citrus_user" to "MANAGER")

I. Web Service

We did slightly change the scheme. The new scheme is online (https://citrus.yuri.moe/order-list.xsd).

Despite this web service as required on C, we have also implemented many different REST services. But those server returns JSON for the webpage. We choose to use JSON because compared with XML, JSON has less overhead. We were planning to use REST and AJAX on all the forms. But we don't have enough time to implement them.

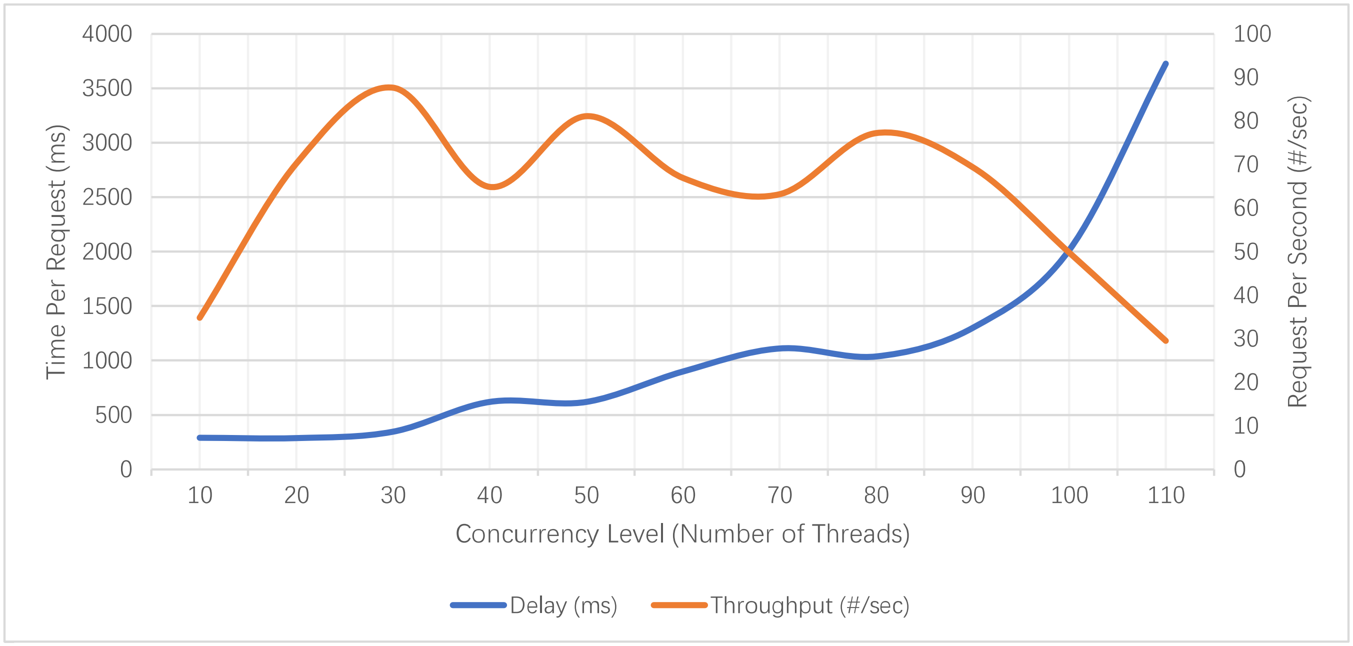
JSON REST service:

- https://citrus.yuri.moe/REST/user/register : POST – register a new user.
- https://citrus.yuri.moe/REST/user/login : POST – user login.
- https://citrus.yuri.moe/REST/book/ : POST – add new book to the catalog.
- https://citrus.yuri.moe/REST/book/id/{id}: GET – get the book by id.
- https://citrus.yuri.moe/REST/book/category/{id}/{offset}: GET – get the list of books by category IDs
- https://citrus.yuri.moe/REST/book/all/{offset}: GET – list all the books
- https://citrus.yuri.moe/REST/book/search/{keyword}/{offset}: GET – search books by keywords

H. Performance and Scalability (Performance Testing Report)

This project is design to be highly scalable. As we have mentioned in the Design Decisions, we have made our best effort to separate static and dynamic content.

We have set up the environment for testing (https://citrus.yuri.moe/). As the graph shown in Read Me. The server is hosting on a KVM virtual machine on OVH's infrastructure in Ottawa. We have allocated 1 Core 3.2 GHz, 2GB RAM and 1GB SWAP for this virtual machine. The pressure testing class is in the `Citrus_Test.war` file in the main package. We also tested this program using Apache Bench to test this server. The error rate is close to zero in this experiment, which means the server does not drop any request. The result as the graph shown below.



The error rate goes up as the concurrency level increase. The ISP has also implemented a DDoS firewall which might drop the request. We don't know whether the request is dropped by Tomcat, Nginx or the ISP. Hence, we did not include that part of data.

K. Security

This project supports SSL. We are currently using LetsEncrypt SSL certificate. Thanks for this free SSL solution, we do not need to pay a penny for enjoying a more secure Internet environment. We are also using Nginx for an extra layer for this project. The password is not passed in plain text because we are using SSL for encryption.

For demonstration purpose, we did not limit the request source for any of the web services. But we can limit it easily. We are using Nginx as a reverse proxy for this project, and we can add some rule on that reverse proxy to protect the Tomcat server. Nginx also provide a handy rate limiting option for us.

To Be Continued

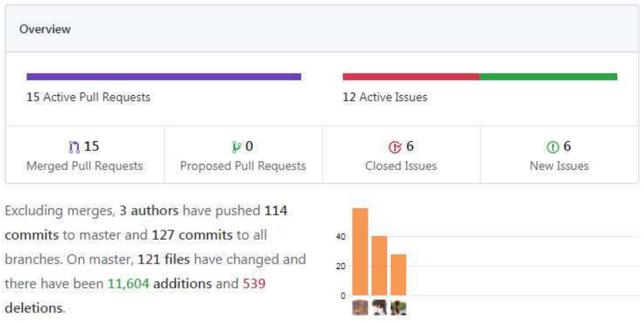
There is still some unfinished work. These items are not a part of the project requirement. We will continue working on it to improve this project.

- Change password
- Change Account role
- Change the status of the order.

Contribution

We collaborate using GitHub. Each team member uses a GitHub account. We don't meet often, but we use GitHub to collaborate.

Jun Lin Chen
mc256
Chenxing Zheng
loveritsu929
Vishal Malik
vishal0027


Overview

15 Active Pull Requests

12 Active Issues

15 Merged Pull Requests	0 Proposed Pull Requests	6 Closed Issues	6 New Issues

Excluding merges, **3 authors** have pushed **114 commits** to master and **127 commits** to all branches. On master, **121 files** have changed and there have been **11,604 additions** and **539 deletions**.

Jun Lin Chen

I am the carry for this project and mainly focus on the front end and some parts of the backend. In this project, I understand why Java EE was a successful commercial standard. Compared with PHP or Nodejs, Java is a strongly typed language, which forces the programmer to think more about what the most proper way to implement such a method or object is. Especially in a big company, it increases the readability of the code in some sense. If it were in PHP, the source code might be difficult to maintain if an unskilled PHP programmer writes it. However, this also comes with drawbacks. The development speed in Java EE is not as fast as PHP or Nodejs. This project gave me a chance to practice my web application programming skill in a language that I have never tried. And it adds an extra development option when I need to develop a web application next time. I have also learned some skill for writing sophisticated database queries from other team members.

Chenxing Zheng

In this project, I mainly focus on the data layer. Instead of using the provided database schema, we have our schema that best fit the design of our website. The work involves designing the tables, finding relations among the tables and providing an appropriate interface (the Bean and DAO classes) of the database. Besides the general query methods, I also take security and encryption into consideration. The database is immune to SQL injections, and I protect user’s password using salt and hash function. I keep modifying the implementation of those JDBC methods along with the iteration of the project, based on the work done by my team members and discussion with them. (e.g., What should the proper argument of this method, user ID, username or a user bean? Should this functionality be done in the database, or in the Java program?) This project enhances my understanding of SQL, concurrency control, encryption, etc. I learned a lot from my team’s work as well (e.g., responsive web design, REST web service which I did not quite understand in class, scalability and so on).

Vishal Malik

In this Project, I mainly focused on the JavaScript, Form Validation and Filter aspect of project. While doing the data validation at Servlet I also focused doing it by JavaScript at User level. The work involved was verification of data by use of regex and by throwing appropriate exception. To protect the website from cross-side scripting we used JavaScript to remove any html tag and content inside <> tags so that the data which actually displayed on website is Pure text. I kept modifying JavaScript and developed it accordingly such that the chances any possible exploit decreases. In addition to this I also used JavaScript to do Country validation by matching user input to a dataset of all ISO countries. This project enhanced my understanding in JavaScript, Servlets. Discussion with team mates enhanced my overall understanding of Ecommerce web app. My team mate Chenxiang Zheng enhanced my knowledge towards the Database aspect of this project whereas Jun Lin Chen enhanced my knowledge of design semantics of any web application and (REST web service).

Signatures [redacted]
Proudly su
April 25, 2018