YQGDataPortal: Final Project Report

Presented by Het Patel and Nandini Patel

University of Windsor

Windsor. ON

August, 2020 | Summer 2020

Acknowledgement

The accomplishment of YQGDataPortal benefits of the help and direction from our professor - Dr. Ziad Kobti and a few online resources, which were of great help.

The project was done jointly by Het Patel and Nandini Patel. Het Patel was assigned the task to create the web pages (HTML, CSS, Bootstrap), debugging, and testing. Nandini Patel was assigned the task to develop the server side of the website (exchanging information between MVC), configurations, and testing.

Abstract

The problem our team aims to address with the YQGDataPortal project, a web application, is to provide an additional option to users where they are able to see tabled data without having to download it. Our project will not only save user's time and memory, it gives them convenience. This application has a list of datasets from which the user can choose from hence, it makes it easy for users to simply select any dataset of their choice. This software can also work with any dataset which has .csv extension or is stored into a database. To continue, there are some comparable products in existence today like data visualization tools and open data portals. Our application provides a mix of both online visualization and open data portals. This application is different because the City of Windsor does not offer this feature. In addition, our application is on the web unlike some applications where software needs to be installed in order to benefit from its service. Our team expects to create a functional website which is hosted on a domain where users can select their desired dataset, view its relevant information, and see the data without downloading it. Datasets will also be sorted into categories so users can find their data more easily.

Keywords

1. MVC model:

Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components Model, View, and Controller. MVC separates the business logic and the presentation layer. Model represents the data that is being transferred between the controller and other logical components. View represents the data; it gets the data from the model and represents it on the user interface. Controller controls the data of the model. It updates the model as new requests are made. In our project, the Model represents the Java files dataset.java and visualizeDbUtil.java, which deals with the data from the database and manipulates it. View represents all the HTML, CSS, and JSP files in the folder WebContent. Controller is the java file visualize.java which directs the requests from the view and commands the model to change data accordingly.

2. Real-time Retrieval

The basic data retrieval component used in the project is the MySQL database. A connector is used to make a connection to the database from Java. For this project, we have set up a database locally. Later a database can be set up on a cloud service to make it accessible to all the users. If a data has been updated in the database, the very next time the website is accessed, the view will show the updated changes. This makes the nature of the website dynamic.

3. Request/Response Cycle:

The website circulates between a request-response cycle. A request is made by a client. A client is a web-browser whose contents are made accessible by the server. A client interacts with the contents on the web-browser. When an activity is performed on the browser an HTTP request is made to the server to which server responds with the resource that it has. In the project election.html corresponds to the client which makes a request to the visualize.java to which visualize.java directs the request at appropriate place i.e. when the “View Table” button is clicked, an HTTP post request is made to the servlet, visualize.java, which is then directed to “show-data.jsp”.

4. Event-driven Programming

Event driven programming is a programming ideal in which the flow of program execution is determined by the user interactable events which are the redirection links and buttons of the website. In the project the transition from index.html to election.html, clicking “View Table” which displays a data table on a new page, are all events.

5. Dynamic Web Project:

A dynamic web project is built on different files such as Java, HTML, JSP, database, which can be inter-connected to each other. On the other hand, a static project is a project which is fully built on HTML. That is to say that if a data has to be updated in a static project the actual HTML file has to be updated. While the benefit of a dynamic project is that changing the data in a database can give us changes in the other files too. Our project is a dynamic web project in a way that a change in database can change the “show-table.jsp” file to represent the updated data. Our purpose of choosing a dynamic web project approach is to keep the user updated with the most recent data.

Introduction and Related Work

In order to view a dataset, users have to download it, which becomes a long process simply to view the data. With our new feature, users can directly get access to the data just by clicking the "View Table" button. This will allow the user to get an interactive view of the data on a new page without downloading the data file. The thought of this feature came since we all become bored with the chronology of opening the dataset, choosing an appropriate place to download the file, viewing it, and deleting it if it's not needed. With the motivation to solve that problem, we thought of replacing all these jobs to just one click of a button. Our objective is to assist the City of Windsor to develop the next generation's open data portal. In this report we have discussed various components of our project including background research, the approach we took to build our application, the process, final results, and what we plan to do in future.

At the beginning, we researched many other city's data portals and compared them with the City of Windsor's data portal. After brainstorming, we acknowledged the type of new feature that should be introduced to our website thus, we prepared a list of features that can be done in our scope. After finalizing our idea, we went onto preparing the design of the idea. We drew UML diagrams which were frequently used while coding to make sure that we are on the right track. After the second phase, we had developed a raw model of a visualization tool in Swing (Java) which visualizes data in the form of a graph. Soon before iteration II, we realized some complications with the visualization tool given the time frame therefore, considered the idea of representing the data on a web page. By the end of the third phase, we were able to develop a raw working website with the new feature.

Approach

Along with the MVC architectural pattern, the team decided to use five major design patterns to make the code and overall design resilient. The following GRASP patterns were used in this application: Creator, Expert, Low Coupling, Controller, High Cohesion. Diagrams are illustrated in Appendix A for visual purposes.

- 1.1: visualizeDbUtil is the creator class that is responsible for instantiating the Dataset class. visualizeDbUtil is an association with Dataset that is to say visualizeDbUtil is a creator of dataset objects. When visualizeDbUtil retrieves the name of the dataset from the user, it creates instances of class Dataset (entries) and passes it to the servlet.
- 1.2: visualize class in the information expert because it has the key information to fulfill its responsibilities getting the database name, passing it to visualize, and then passing database information to display it to the user. Visualize class does this by collaborating with other objects like visualizeDbUtil, show-data.jsp, and index.html to fulfill its responsibility.
- Our application has a great structure in terms of low dependency. All the classes and components interact with each other by using their functions but do not extend or implement other elements except for HttpServlet. Dataset names are retrieved from the user, automatically search in the database for relevant tables, and displays the content regardless of the number of fields or rows. This is beneficial because it increases reusability for other datasets.
- 1.3: visualize class is the servlet (controller), which coordinates all the system operations. Whenever a user opens the url, a request comes to visualize; then, the controller communicates with visualizeDbUtil (model) which returns data back to the controller. The controller then processes the data to show-data.jsp (view), which the user sees.
- 1.4: show-data.jsp (view) is created to be a highly cohesive file because it does a limited amount of work and independently. The view simply takes all the information from the servlet and displays it on the web browser.

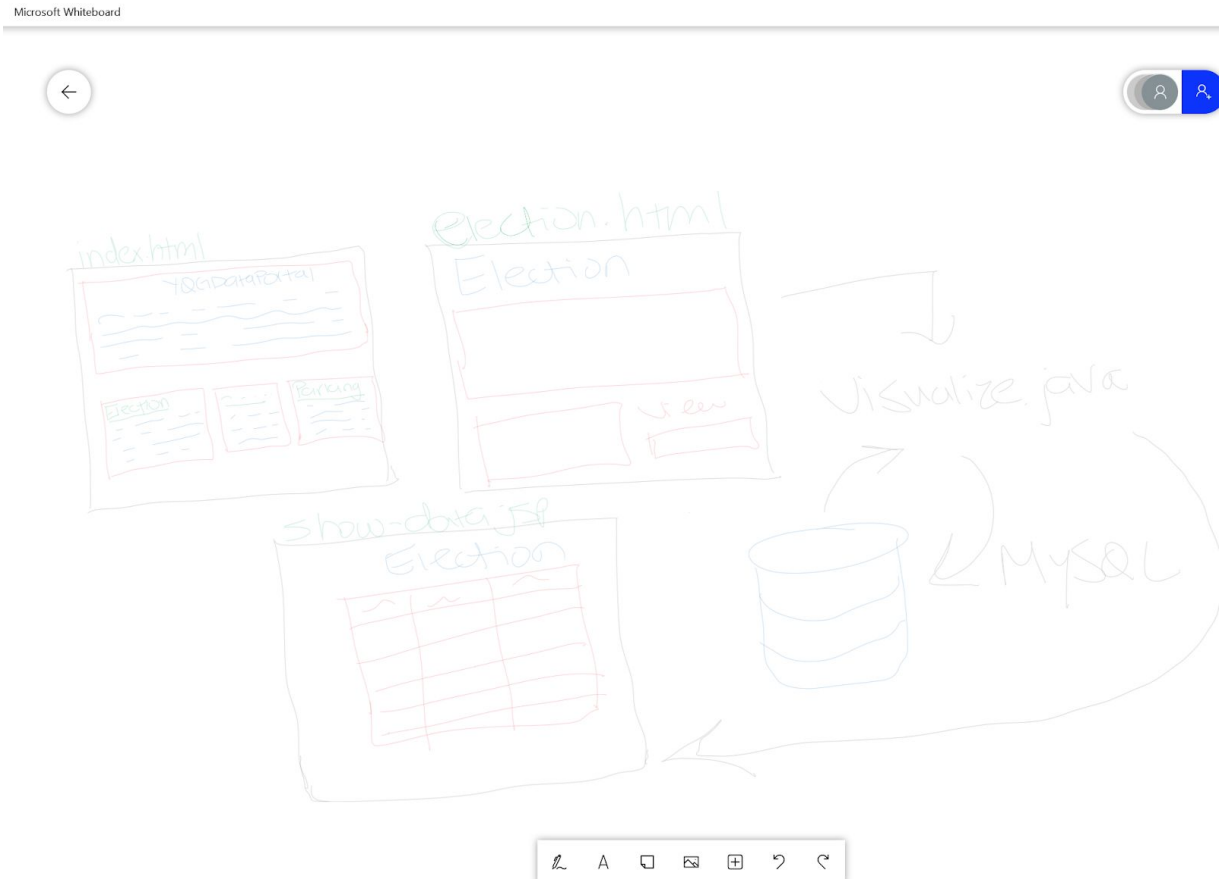
Demonstration

During the Inception phase, our team came up with the following use cases: filtering search by themes, visualizing datasets, and accessing APIs of a dataset. From these three cases, we decided to visualize datasets as that is the most uncommon feature seen on any data portal and implementing this feature gives convenience to users so, we planned to create a website which contains a list of datasets; users can click on their desired dataset, which would display them three options they can choose from: tables, maps, graphs. And this was our prototype. By the end of elaboration iteration I, we hoped to visualize all datasets in such manner.

Furthermore, during elaboration iteration I, we realized that our prototype could not be achieved given the two-week period for one iteration because there were more tasks involved than simply coding. We further broke down the task to having only one option for the user, which was to display a simple graph for a given dataset. The csv file containing data was converted to a text file, data was extracted using Java, and Swing (Java) was used for graphical user interface (GUI). By the end of this iteration, we successfully completed our task; the result can be viewed in Appendix A, 1.0. Using JUnit, we also tested our classes only to realize that testing was not much useful in our development process. We constantly tested our code with different inputs, checking for validations, and other small tests along the way, which we found to be more efficient. In regards to the final product of this iteration, we were still not satisfied with the result because our application did not support loose coupling. We had developed a local software that would take an input dataset containing three fields (columns) and create a simple dot graph accordingly but this application would not work on a large-scale. To an extent, it may require advanced programming to determine which field belongs to which axis with larger sets. In our application, the program was making that decision i.e. we took an election dataset as a sample to plot our data and it had the number of wards on x-axis, votes on y-axis, and the mayors were plotted accordingly. The thought of what will go on which axis was made by the team however, for another dataset like parking tickets, we saw that it had more than two fields. In such a case, we would need an algorithm to identify axes. In addition, graphs are not suitable for certain datasets hence it is challenging to determine which type of graph is needed for every dataset. Not

only did such planning give us many uncertainties, more importantly, we realized it would be time-consuming to gain knowledge and train ourselves to use high-level softwares.

Moreover, In elaboration iteration II, we came up with a more clear and concise plan. Our team decided to create a basic website, which will include few datasets; when a user clicks on the one of their choice, it redirects them to a page which has more information about it, a section to download the dataset, and a link to view the table at realtime from a database. Below is an image of the team brainstorming ideas:



Our website was implemented by using MySQL to store datasets, Java to implement a controller and a model responsible for handling requests, exchanging information, and redirecting information to view, JSP file to display dataset content directly to the web, and HTML, CSS, and Bootstrap for website designing. By the end of this iteration, we successfully completed our task and were satisfied as it provided low-coupling. Not to mention, the final product was achieved after spending several extra hours and conducting additional meetings to achieve the high tasks.

Discussion

Initially, we had an application in our mind which would visualize data in three forms however, the final product is a website with the option to see live data. We decided to give our project a structure which resulted in a better transition from the previous phase. MVC model was chosen to be the base of our application. Evidently, the final result is excellent and far better than what was expected. In addition, data is dynamically loaded so if there are changes made to the dataset in the database, just with a single refresh, the user will be able to view them instantly. As we moved from one phase to another we noticed many similarities and changes. Our vision and business case:

> *Citizens can use City of Windsor's open data for research purposes or to improve their interaction with municipal services and facilities.*
>
> *To focus on making data more accessible dynamically and have interactive implementation of data sets.*

remained the same throughout this project and we achieved both components. We wanted to have visual implementation and dynamic access both of which were achieved by this. In terms of changes, there were many technical changes with tools & technologies as well as with our approach towards small tasks. We started with Java and Swing but later, used Java, JSP, MySQL, HTML, CSS, and Bootstrap. Very little code was carried from elaboration iteration I because of poor design. In elaboration iteration II however, we thoughtfully planned all the components of our application. Overall, what we planned for long term did not change but short term planning was excellent and helped altogether. To continue, many things happened behind the scenes from learning to verification and validation. Both team members spent time learning new tools i.e. GitHub (managing repositories, wiki, issues, project management), Eclipse (understanding configurations of MySQL, JSP file, HTML) and methods i.e. JSP-Servlet interaction, Servlet-Model interaction, Model-Database interaction. To add to that, the experiments and final result conveys that our application can be used for any dataset, regardless of the number of fields, stored in a database and display its content live on the website.

Conclusion and Future Work

Our set objective was to create a new open data access portal for City of Windsor with the ability to manage data, distribute it in various formats, and visualize it live on the site. Near the start, the team had made many assumptions without accounting for time needed to learn new tools and patterns. Overtime, we learned about project management, design patterns (GoF and GRASP), refactoring, and gained technical experience. Not only did all of this help our project, more importantly, we realized that by the end of each iteration, requirements became more clear, helping us take specific approaches to the project issues. By the end of the second elaboration iteration, we achieved to create a website that displays live data. It works under the assumption that users will select one of the datasets displayed on the site and since it's not hosted, the project needs to be on the user's machine. What we yet have to achieve is to host it on a domain such that users do not need to worry about any files or folders. Rather, they can simply go to a website using any device.

For future iterations, our team plans on doing three main tasks: hosting the site, we aim at categorizing all the datasets which will be more organized than the current list of the dataset, and include more datasets from City of Windsor's site. For the next iteration, we would focus on hosting on an online domain i.e AWS because it would make it convenient for us as developers as well as useful when getting feedback from users about the application. After finishing the elaboration iterations, we plan to develop this into a commercial service site. It will be a free site for anyone interested in City of Windsor's data where they can view all datasets more conviniently.

.

Works Cited

MVC: Model, View, Controller. Code Academy. Retrieved https://www.codecademy.com/articles/mvc

Rao, Danya. *GRASP Design Patterns.* University of Colorado. Retrieved https://www.cs.colorado.edu/~kena/classes/5448/f12/presentation-materials/rao.pdf

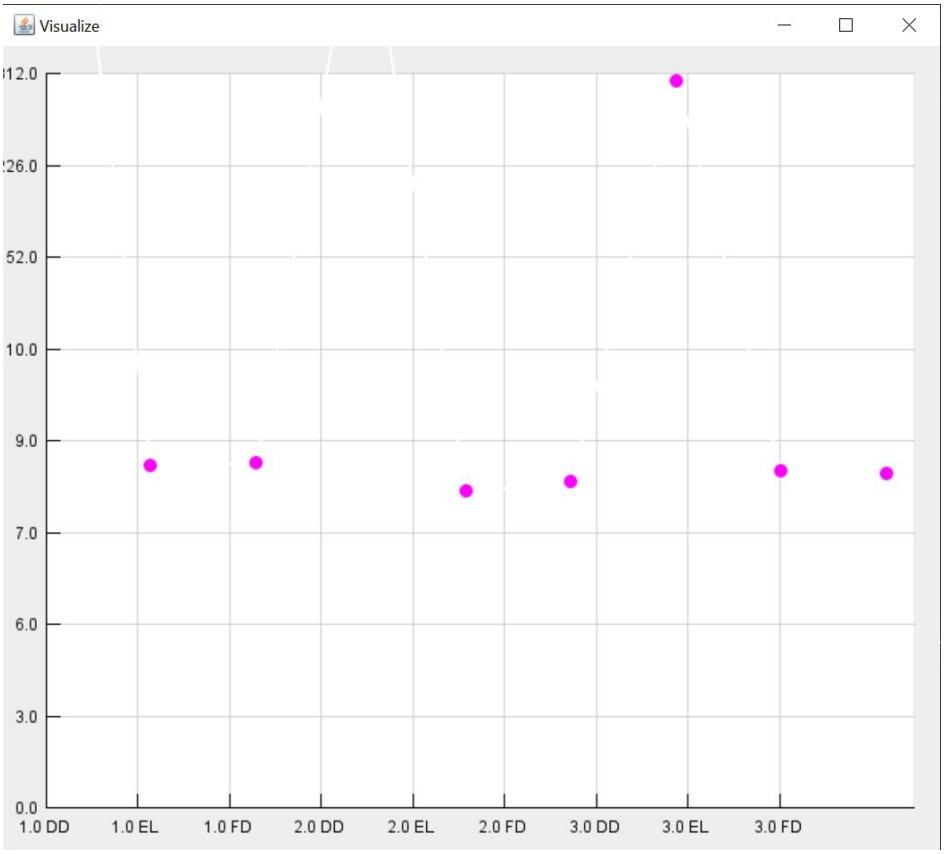
Strong, Jen. (2018, 06, 20). *The Request/Response Cycle of Web*. Medium. Retrieved from https://medium.com/@jen_strong/the-request-response-cycle-of-the-web-1b7e206e9047

Servlets | Servlet Tutorial. Javatpoint. Retrieved from https://www.javatpoint.com/servlet-tutorial

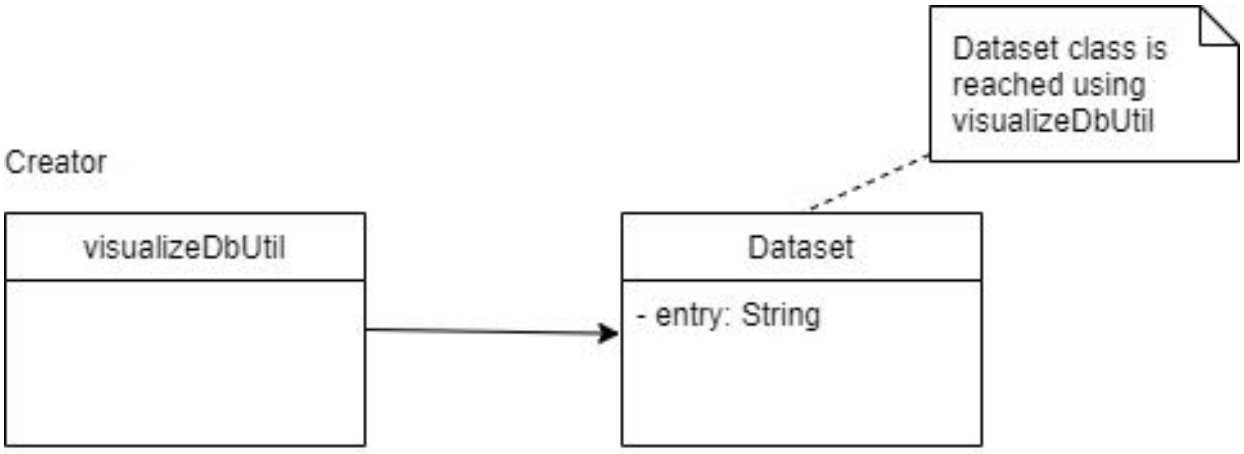
Vaidya, Neha. (2020, 07, 21). *Servlet and JSP Tutorial- How to Build Web Applications in Java?*. Edureka. Retrieved from https://www.edureka.co/blog/servlet-and-jsp-tutorial/

Appendix A

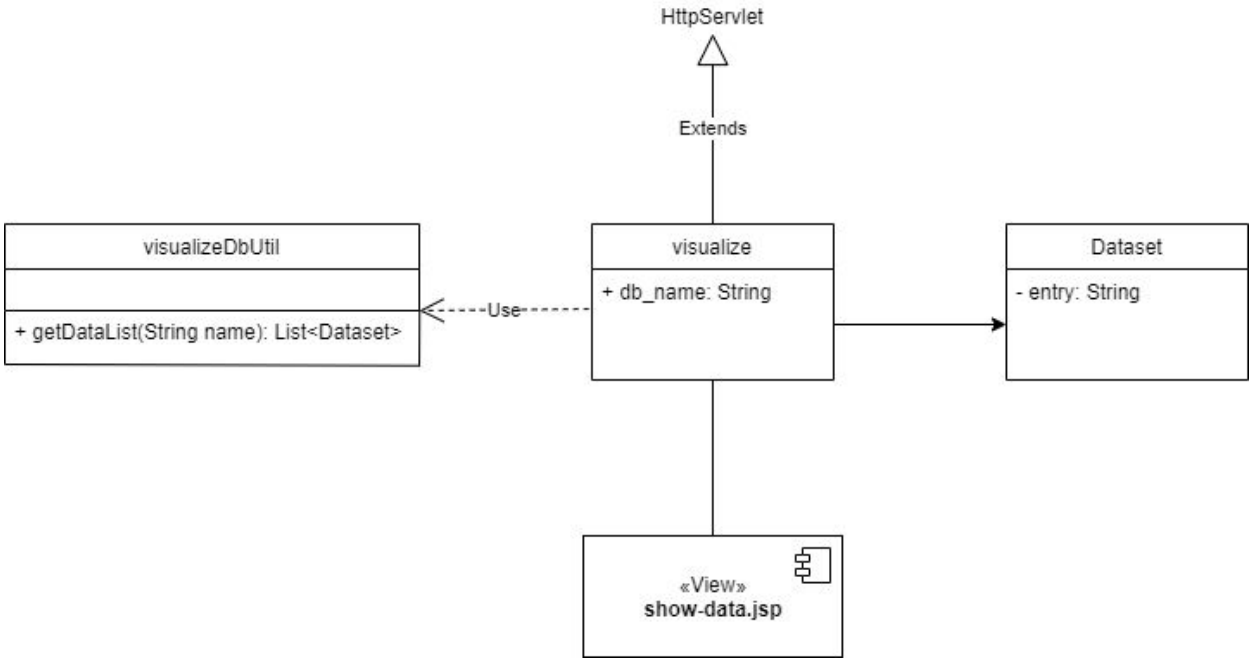
1.0



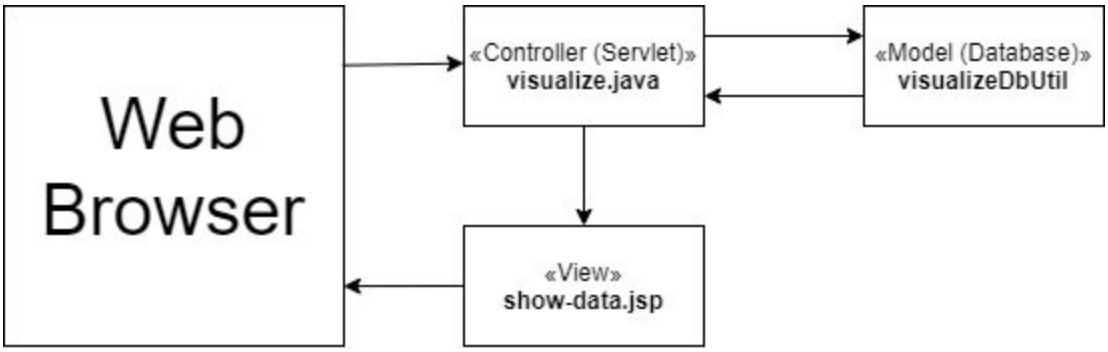
1.1 (Creator)



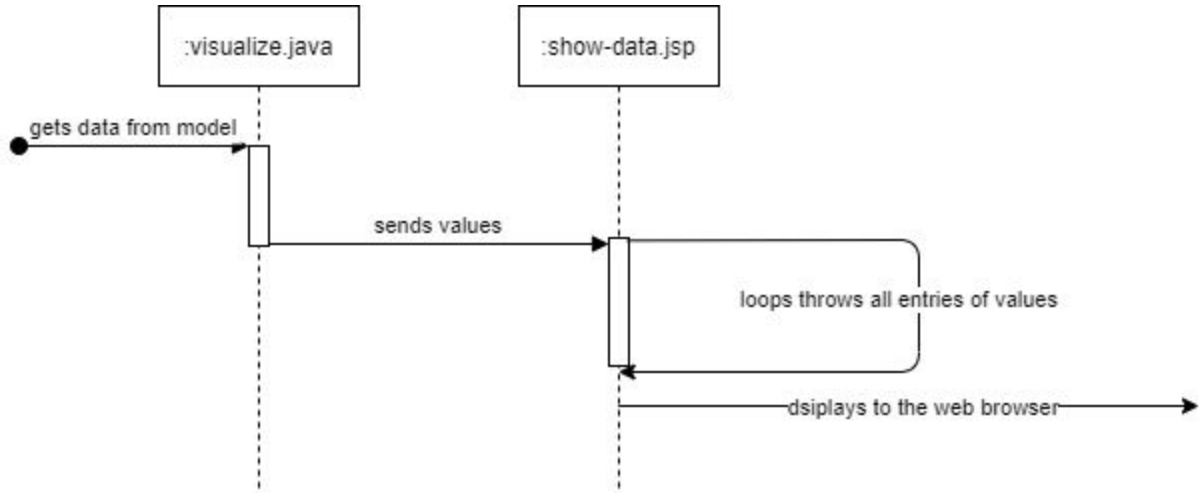
1.2 (Expert, Class Diagram)



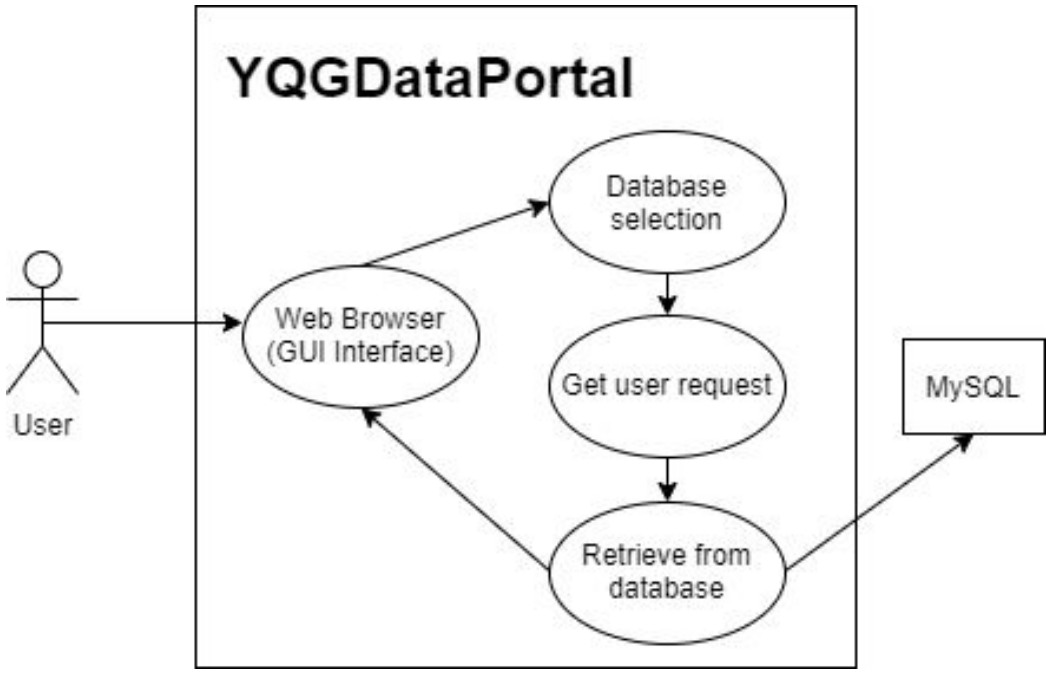
1.3 (Controller)



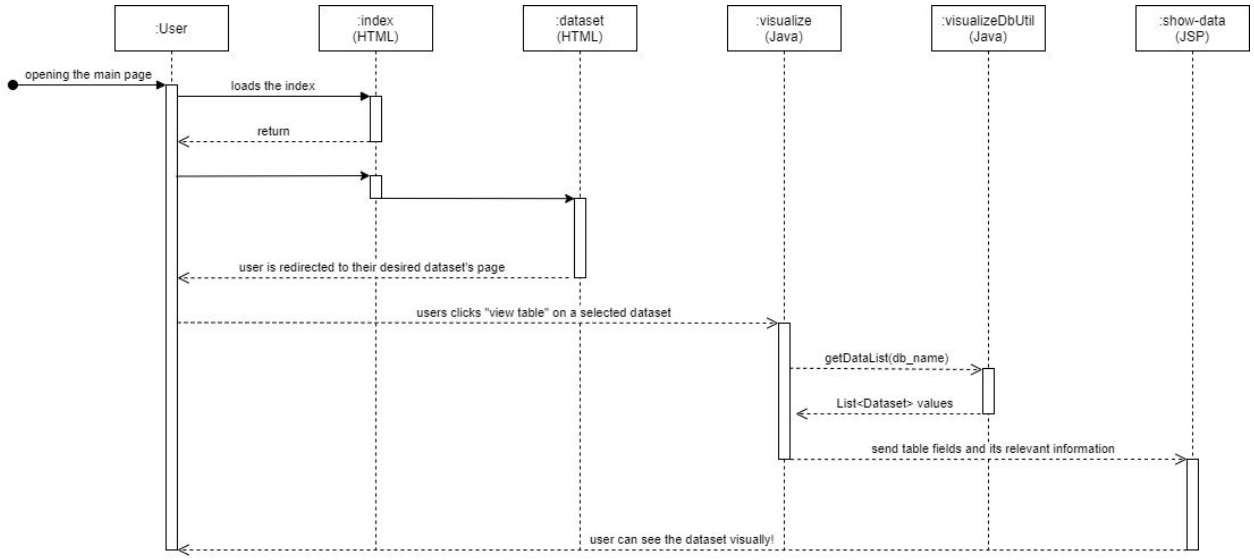
1.4 (High Cohesion)



1.5 (Use Case)



1.6 (Sequence Diagram)



1.7 (Design Model)