LE/EECS4413 B - Building E-Commerce Systems (Fall 2023-2024)

EECS4413 E-Commerce Project Report

Shop for Style: Crafting an E-Commerce Experience

with Java Web Technologies

BY:

Antonio Cedrone (217798000)

Kevin Chau (217130741)

Gurban Jumagulyyev (216151268)

Bryce Cooke (217346354)

December 21, 2023

Table of Contents

Team Composition

The e-commerce project named Shop for Style brought together a dynamic and collaborative team, each member mentioned below contributing unique skills and expertise to ensure the success of our e-commerce platform.

- ★ Kevin Chau (217130741)
- ★ Bryce Cooke (217346354)
- ★ Gurban Jumagulyyev (216151268)
- ★ Antonio Cedrone (217798000)

Individual Contributions:

Kevin Chau (217130741)

- ➔ Kevin Chau primarily focused on front-end development and user interface design. He added sign-in and sign-up JSP/CSS files, which laid the groundwork for the initial user experience in accessing and utilizing the service. He also created JSP/CSS files for the cart home page and user accounts. The implementation of account JSP/CSS files and modifications made to the UserDao, UserDaoImpl, and Identity class to facilitate functions like retrieving all accounts (and related information), deleting accounts, and modifying existing accounts in the database. He created product cards to render information from the database in a clean and concise manner. On the administrative side, he developed admin panel JSP/CSS servlets and added functionality using AJAX and JSON to fetch existing user records from the database.

Bryce Cooke (217346354)

- ➔ Bryce Cooke coded the ItemDAO classes as well as most of the earlier classes in the model. He also implemented the sort and filter functions on the home page, as well as the functionality of the shopping cart. In addition, he implemented the function that allows users to update their profiles and for administrators to update the quantity of products.

Gurban Jumagulyyev (216151268)

- ➔ Actively contributed to the implementation of crucial functionalities, particularly those related to secure payment processing and checkout procedures. Gurban's expertise extended to crafting and refining Java Servlets to handle various back-end processes efficiently. In addition, Gurban was instrumental in creating a visually appealing and user-friendly front-end by contributing to the styling and layout of JSP files. Throughout the project, he demonstrated a strong commitment to collaboration and effective communication within the team.

Antonio Cedrone (217798000)

- ➔ Antonio Cedrone coded some of the DAO classes (AddressDAO, AddressDAOImpl, PoDAO, PoDAOImpl) and contributed to the others he did not (ItemDAO, ItemDAOImpl, UserDAO, UserDAOImpl) by adding additional methods. He also implemented the model classes Address, Amount, Identity (others did contribute some edits to this class, particularly credit card information), Item, and Order. He also edited the database creating the tables, PO, Address, and CustomerPO, and provided some edits on the remaining tables. He did very little work on the front-end part of the project, only working on a small amount of EL and JSTL to make data visible on the account page. He created the sign-in and sign-out functionality (SignoutServlet, SignupServlet, LoginServlet), PaymentServlet which contains the work necessary to reset the session variables and add user orders to the database, prepared the data source in Listener, created the ability to output images by including images in the database and creating the ImageRender class, created the functionality in the CheckoutServlet which does not let you proceed to checkout with items in quantities above availability. Also, he completed some of the work in AccountServlet for collecting some fields, as well as the work involved in updating the database with accurate user and address objects.

Communication and Collaboration:

Effective communication and collaboration were paramount to the success of our project. To facilitate seamless coordination, the team established a Discord server as our primary communication platform. Regular discussions, brainstorming sessions, and code sharing occurred on Discord, allowing team members to stay informed about ongoing developments. The server became a central hub for sharing progress, addressing challenges, and assigning tasks. Through collaborative efforts on the server, each team member gained insights into the elements contributed by others, fostering a holistic understanding of the project's intricacies.

This collaborative approach, coupled with open communication channels, enabled the Shop for Style team to leverage individual strengths, address challenges collectively, and create a well-rounded e-commerce platform.

Team Members Signature:

Kevin Chau: KC　　　　Bryce Cooke: BryceCooke

Gurban Jumagulyyev: GJ　　　　Antonio Cedrone: [illegible]

1. Introduction

The focus of our team project was to develop an online e-commerce website. The items we chose to make available on the website are different types and brands of clothes. We aimed to create a seamless and visually appealing user experience while ensuring the functionality of the website. The website includes features such as product listing, object filtering, account management, admin management, shopping cart, checkout, and payment functionality, with the main goal of providing an intuitive and user-friendly interface.

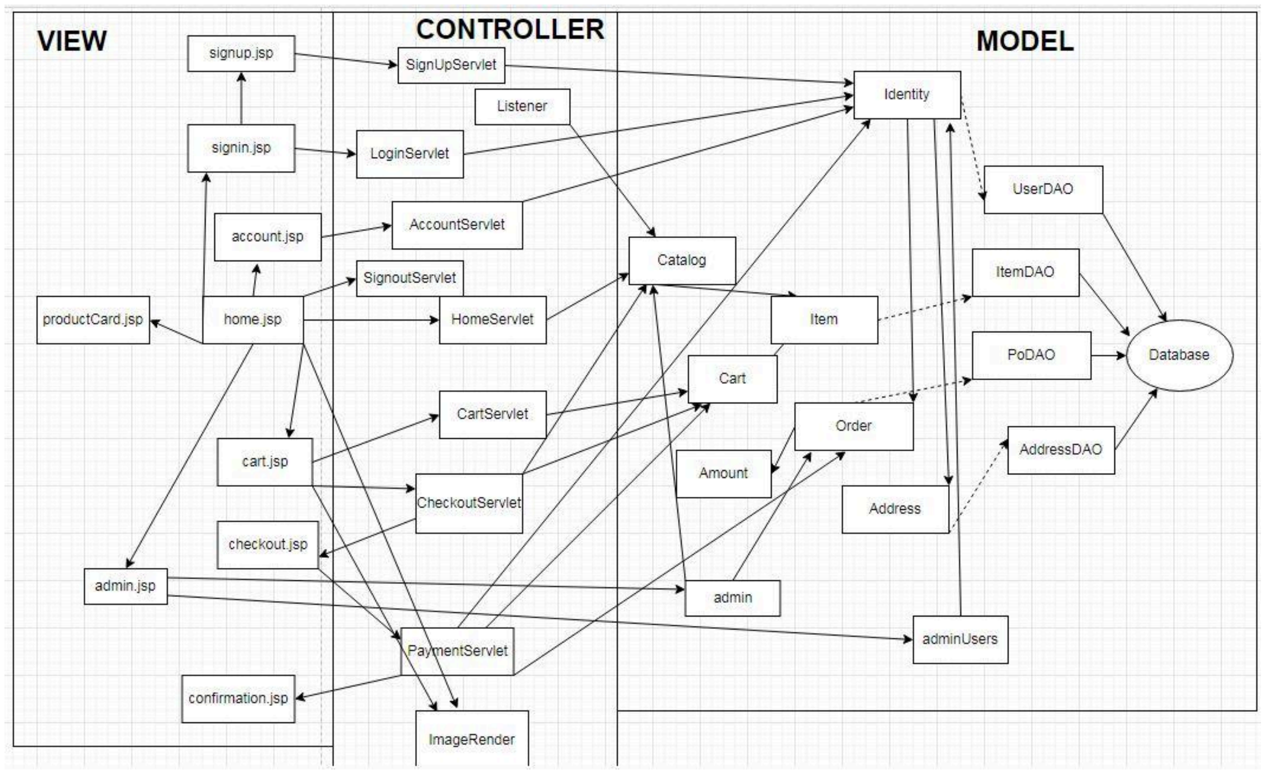
To accomplish this, we used design patterns and technologies learned in class. We heavily utilized DAO (Data Access Object) in the accessing of database information between application instances. We also used Model View Controller (MVC) to organize the general layout of the project and ensure separation between different elements, particularly the front-end, business logic, and back-end data representation. For the front-end display of information, we used JSP, Ajax, and JSON, and finally, for our business logic, we used Servlets, Listeners, and Filters.

The main strength of the design, which took advantage of organizational patterns like MVC, is that there is a separation between different parts of the project. This increases things like maintainability and extensibility as well as allows for team members to work on different parts of the project concurrently without worrying about how other parts of the project end up being implemented.

The main weakness of the implementation is that there is no connection between customer information stored in the profile and that which is entered at checkout. We planned to connect the two forms and even implement autofill for the information but were unable to due to time constraints.

2. Architecture Description

The architecture of our program is three tiered: It has a presentation tier, a business logic tier, and a data tier. The JSP pages are used for presentation since they are what the end-user sees, the servlets and listener make up the business logic tier because they are logical Java programs that mediate between the JSP pages and the data tier, and the model and dao packages, combined with the database make up the data tier since all of them either store or retrieve data. There is a clear separation between the front-end and back-end since each file (JSP or Java file) has a specific purpose that corresponds to one of the tiers.



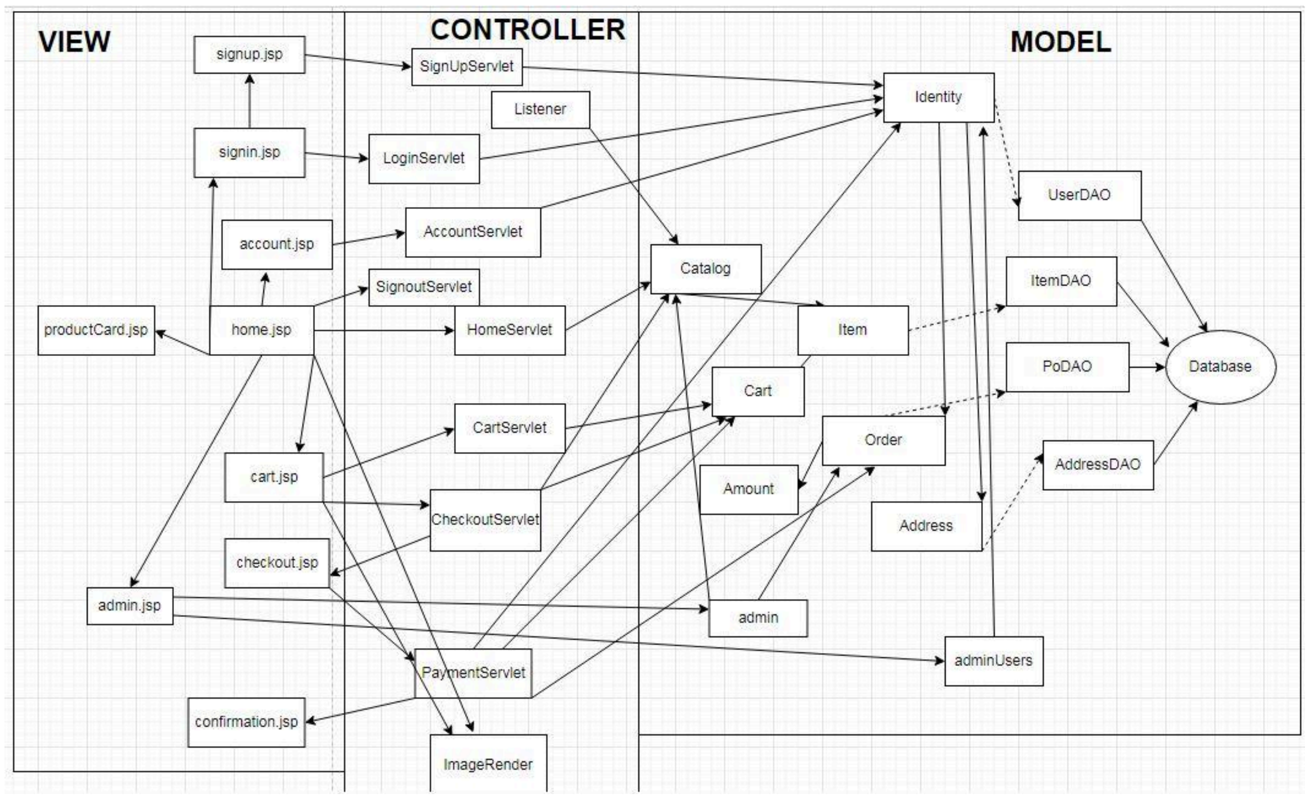
One example of a use case is "Sign In". The user clicks the sign-in button on the homepage and is redirected to the sign-in page, which asks for a username and password. When the sign-in button is clicked, the user and password are sent to the LoginServlet. This servlet uses the UserDAO to access the Customer table in the database. If no customer is found with the exact username and password, a message saying "User name or password is incorrect!" is displayed on the page. Otherwise, the user's identity is stored as an attribute in the session and the home page is displayed.

Another use case is "Add to cart". On the homepage, the client can select the quantity of an item and then click the "Add to cart" button, which redirects to the cart page that shows all items in the shopping cart. When this button is clicked, the request is sent to CartServlet, which gets the shopping cart which is stored as a session attribute. CartServlet will create an item object with the specific item ID and quantity that the user requested and will add it to the cart. The cart object will increase the quantity of this type of product if it is already in the cart or will add it normally otherwise. Once the cart is updated, the CartServlet will update the session cart and total price, which will be displayed on the cart.jsp when called.

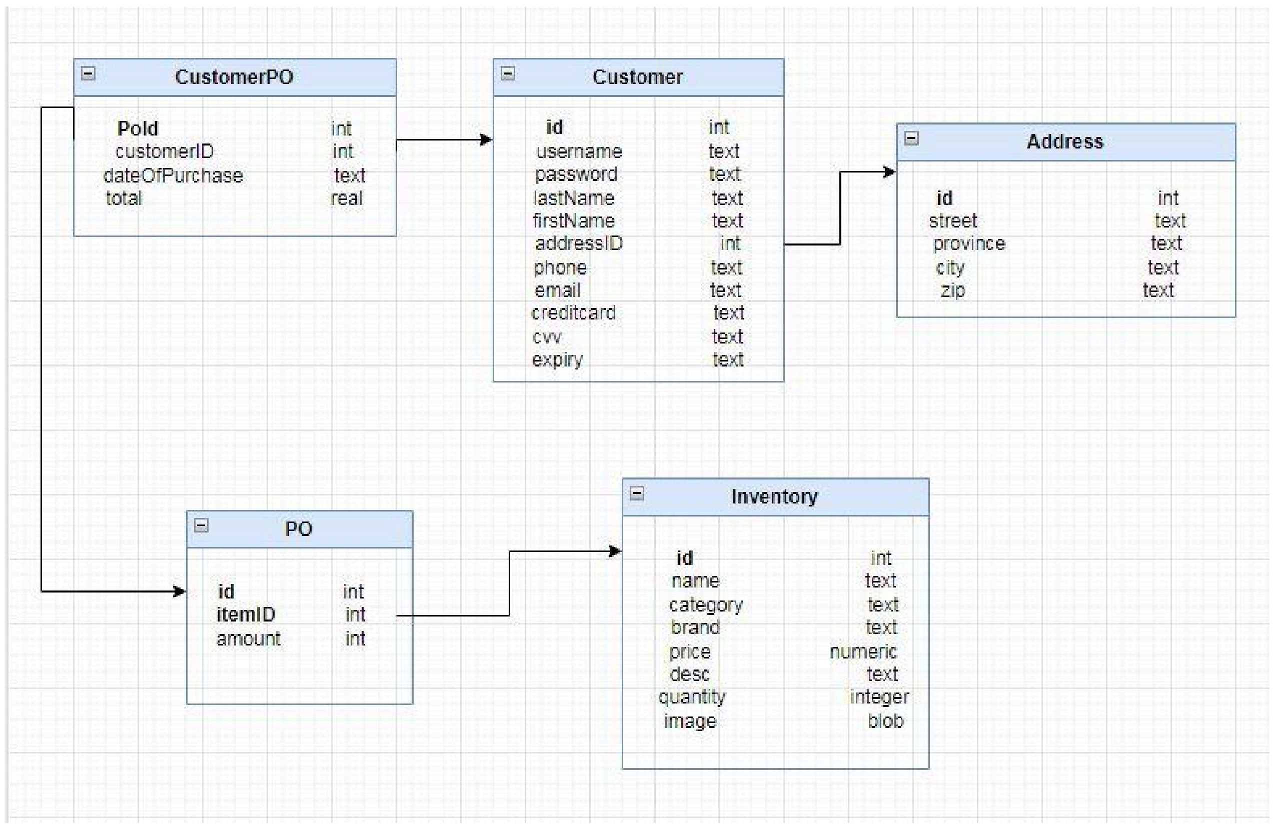
3. Design Description

The main design patterns used are Model View Controller and Data Access Object. Both patterns allow for a separation of front-end and back-end, with the model view controller separating the class representations of table elements from the controller that collects view data and controls the flow of the program and from the view that outputs model data. The Data Access Object then separates the controller, which includes business logic, from database access and the running of queries. In our project, the view was the JSP, CSS, and other related files, which due to the model view controller pattern, did not contain business logic, only the needed code to output data and produce a user interface. The controller contained servlets and listeners which were used to complete the business logic needed to operate the program, such as getting field data and passing control between JSP pages. The model was where information from the database was stored. This section contained only classes that represented table entities and allowed for their fields to be exposed for information output in other parts of the MVC design. The Data Access Object in our project separated the business logic in the controller from database calls, thus making the classes in the controller call methods and not need the details of database access exposed to them. Due to these two patterns, the separation between front-end and back-end is clear with classes that have only one job divided into different groups that pass information to each other as described previously rather than doing everything (like having business logic or DAO method calls in JSP files).

Present the UML package/class diagram for main components, showing how different models are related. An example is shown below.



Also present the database schema diagrams showing the database tables you designed, and how different database tables are related.



4. Advanced and Distinguished Features (Beyond the Requirements)

One additional feature that we added beyond the basic requirements of the project has to do with security. To prevent SQL injection attacks, all of the queries used in the project that take in user input are parameterized queries. Another additional feature used in the project was the use of the Listener classes, which are an expression of the Observer design pattern. This listener class was used when the servlet context was initialized to create a DataSource connected to the database and pass it along to the rest of the application at a servlet context attribute.

On our webstore, we made use of AJAX and JSON to enhance the functionality of the admin panel. AJAX allows for asynchronous communication with the server, meaning we were able to update parts of the application without having to reload the whole page. We utilized JSON to structure the user data that was being sent between the server and the webpage. By combining AJAX and JSON, we were able to create a seamless experience in the admin panel. Admins were able to get real-time updates about users and inventory.

5. Implementation

For the front-end, the technologies that were used were those focused on in class, with some of the others presented in tutorials only being considered. While Antonio did review the React tutorial posted on the e-class page, due to time constraints it did not seem feasible to implement the front-end of the project using this technology. For the back-end, Antonio tried to use Spring Boot and did some testing with SQLite. Once again, due to time constraints, producing a finished product seemed unfeasible, and thus, the technology used was what was taught in class. The technologies decided upon for the final project were JSP, EL, HTML, CSS, and Ajax for the front-end and Servlet and SQLLite for the back-end and data persistence.

One benefit of these choices was the ease of implementation as they were taught in class, and the group already had practical experience implementing web projects with them. Also, this ease of use produced another benefit, the ability to work quickly on implementation without further difficulties than would have existed if a new technology was introduced.

The issues with this choice are things such as the fact that the back-end does not adhere to REST protocols and cannot be accessed from a URL. Other issues involve the benefits of using other powerful technologies like Spring Boot, which allows for things like dependency injection that serve to positively impact the ability to maintain and extend the final product.

The reason why SQLite was chosen for this project is that it allowed for the database to be present within the project such that if deployment was not possible given time constraints (it was not), there would be no concerns regarding database access and use. Also, due to the group having experience using software like DB Browser, which allowed for easy access and modification of the database file, SQLite seemed like the solution that would allow anyone in the group to properly understand and work on the database.

6. Deployment efforts

Due to the tutorials and when they were presented in eclass, the main solution considered for the deployment of the project was Docker. Antonio had downloaded Docker Desktop, found the tomcat image available on the platform, and began to review tutorials provided in the application on how to deploy a project. After watching the tutorials and videos on YouTube, there was also interest within the team to deploy in Amazon AWS. Unfortunately, due to a lack of time, we found ourselves in a situation where we needed to prioritize implementing the project's core features rather than deployment.

7. Conclusion

At the end of our Shop for Style e-commerce project, we made use of a correct strategic combination between architecture, design pattern and technologies to create an interactive online shopping atmosphere. In addition, the use of Model-View-Controller (MVC) made it easy to separate out responsibilities and was thus helpful in making our system modular and maintainable. The use of Data Access Object (DAO) patterns made communication between business logic and the database seamless, speeding up data retrieval. JSP, HTML, CSS and Java Servlets were all brick-and-mortar technologies that solidly underpinned the frontend display and backend operations. The strengths of our design and implementation lie in the clarity of separation between frontend and backend components, providing a foundation for maintainability and scalability. The team's commitment to collaboration and effective communication, facilitated by platforms like Discord, allowed us to navigate the challenges and complexities of the project efficiently. Additionally, the security measures implemented, such as parameterized queries to prevent SQL injection, added an advanced layer of protection to our application.

However, challenges emerged in the form of incomplete deployment and a constrained time frame that hindered the realization of all planned features. The inability to connect customer profiles with checkout information represents a weakness in the current implementation. Learning from these challenges, we acknowledge the importance of prioritizing deployment considerations early in the development process.

The things that went well in the team project were related to group work and collaboration. Despite the limited time available for the group to complete the project, everyone in the group worked hard to contribute something to its development, and among the group members, ideas were shared quite smoothly. Also, everyone was cooperative and assisted each other in the completion of different elements of the project without needing to be chased after for work. The things that went wrong were that the project could not be completed in its entirety with some elements of the project's requirements missing. Another thing that went wrong was that the project could not be deployed due to time constraints. A final thing that went wrong was that the group did not get the chance to explore a technology not covered in class due to time constraints.

What has been learned from this project is the importance of collaboration and splitting up work to allow for individual strengths to shine in the final product. In terms of collaboration, given the time constraints, everyone in the group needed to select work that they would do and communicate this information to everyone else so no one was wasting time completing work that others were better equipped to do or had already started. Throughout the project, each member learned to be clear about what they were working on and what they thought they could complete, and as a result, the division of work was never in dispute, and there was no conflict. We also gained an understanding of the importance that data management plays in enhancing system performance and stability. Through the use of patterns such as DAO, we managed data efficiently, which in turn led to improved overall performance of our store. This experience also highlighted the importance of code reusability. By implementing design patterns like MVC, we were able to make our code more reusable, and in doing so, we not only simplified the development process but also made it easier to add features to other aspects of the project. Lastly, the importance of testing and debugging was another key aspect to take away from this project. Learning these things was crucial in streamlining the user experience and improving our journey as software developers.

An advantage of completing the project in teams is that the scope of the project can be larger than if everyone is working individually. This allows for a project of a reasonable size to be created rather than just a piece of a project with no features. Another advantage of working in a team is that those without a large amount of experience or interest in one particular area can work on another part of the project that they are interested in. One drawback of working in a team is that if a minority in the group wants to use a different technology than everyone else, they must concede if the rest of the group is adamant. Particularly in a project like this where different technologies were introduced so late in the term. If learning is a journey, the project can be said to have offered everyone an ideal opportunity for education through practice.