1. List the features that were implemented (table with ID and title).

Use Case that were implemented:

Use Case ID	Use Case Name
UC-1.1	Register
UC-1.2	Sign in
UC-1.3	Logout
UC-1.5	Browse Items
UC-1.6	Place Order
UC-1.9	Maintain Catalogue
UC-1.9.1	Add Item
UC-1.9.2	Edit Item
UC-1.10	Maintain User Accounts
UC-1.10.1	View List of Users in the system

2. List the features were not implemented from Part 2 (table with ID and title).

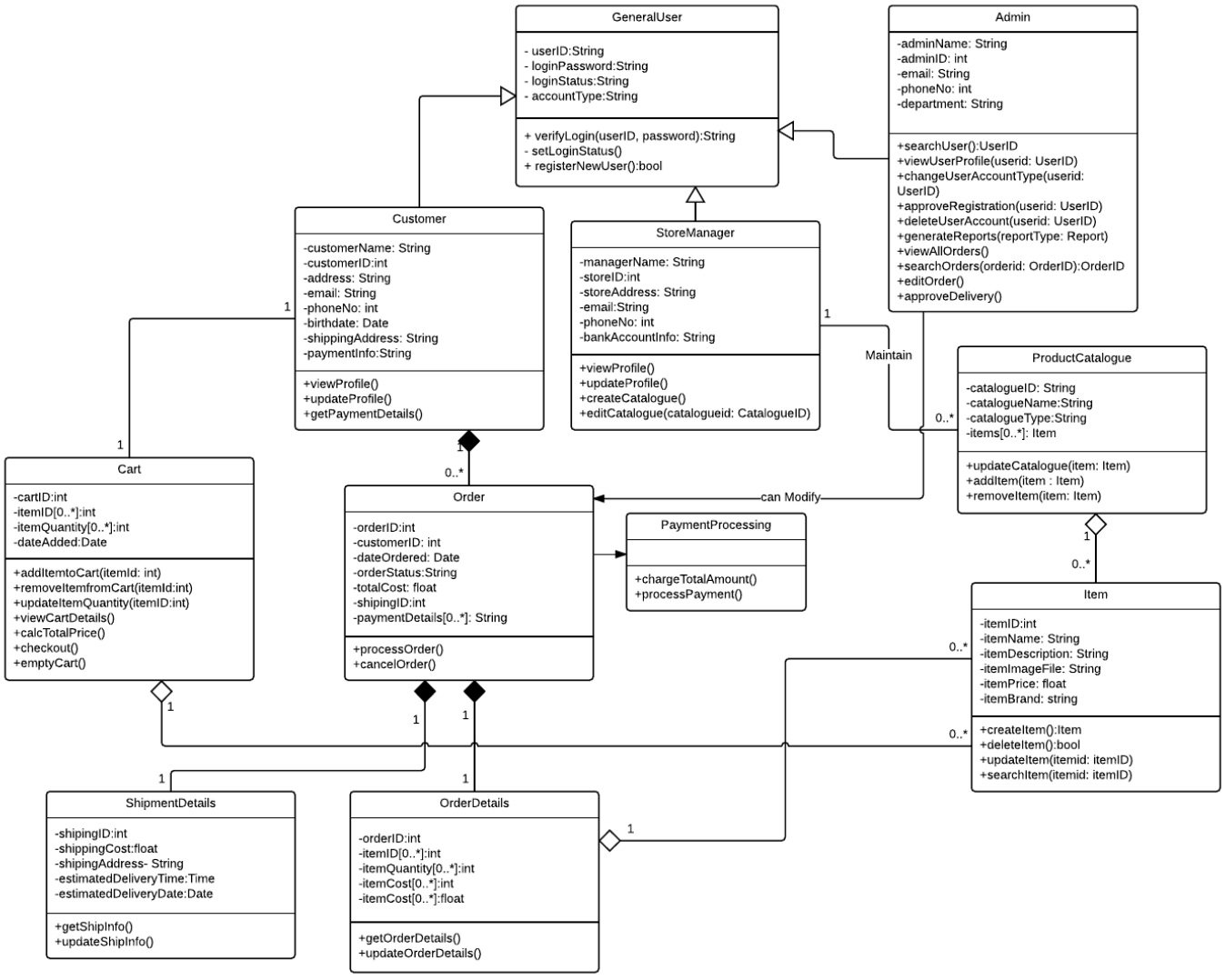
Use Case Documents:

Use Case ID	Use Case Name
UC-1.4	Maintain Personal Account
UC-1.4.1	View Account Details
UC-1.4.2	Modify Account Details
UC-1.4.3	Delete Account
UC-1.7	Cancel Order
UC-1.8	View Order History
UC-1.10.2	Delete a User Account
UC-1.10.3	Change User Account Type
UC-1.11	Approve Registration
UC-1.12	Edit Customer Orders
UC-1.12.1	Change Estimated Delivery Time
UC-1.12.2	Remove an Item from an order
UC-1.12.3	Add an item to an Order
UC-1.12.4	Change Item Quantity
UC-1.12.5	Cancel Order
UC-1.13	Authorize Delivery
UC-1.14	Report generation

3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

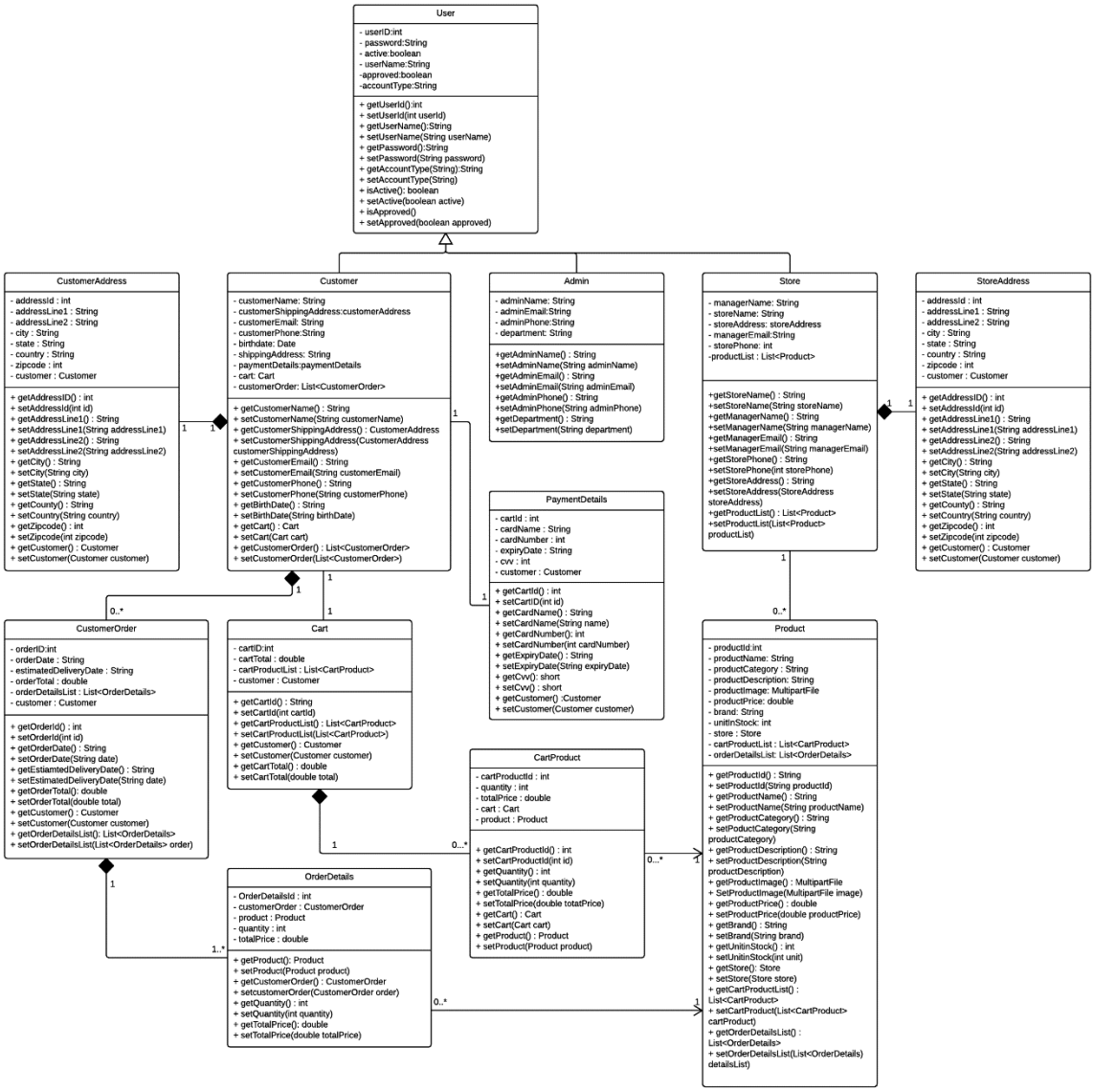
The final class diagram has changed significantly. Apart from the name changes of some attributes and classes, the final diagram now incorporated the MVC design pattern which was missing in the initial diagram.

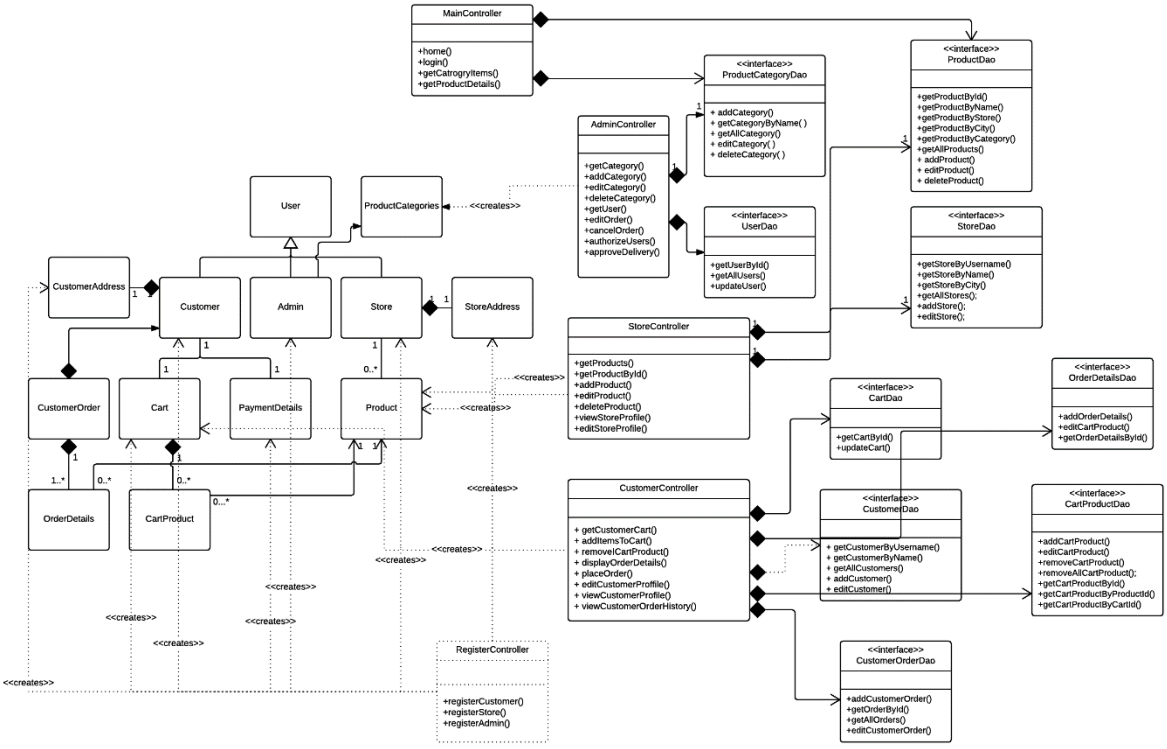
Part 2 class diagram



Final Class Diagram

The final class diagram has been divided into two part. The first part shows only the entities, their attributes and methods. It does not show the controller and Data Access Objects to keep the diagram less cluttered. The 2nd part of the diagram shows the controller, Data access objects along with these entities (attributes and methods of these entities are not show). We have not shown a class for view because our view consists of JSP pages.





The class diagram changed significantly because of the MVC architecture. Instead of implementing methods (to manipulate data) inside the classes, we used MVC during implementation which helped in decoupling and makes future changes possible. Using MVC helped us to test easily and add functionalities without changing the flow. Also, we made different controllers instead of one. This helped in delegation as well as it was to maintain and understand the flow in our project.

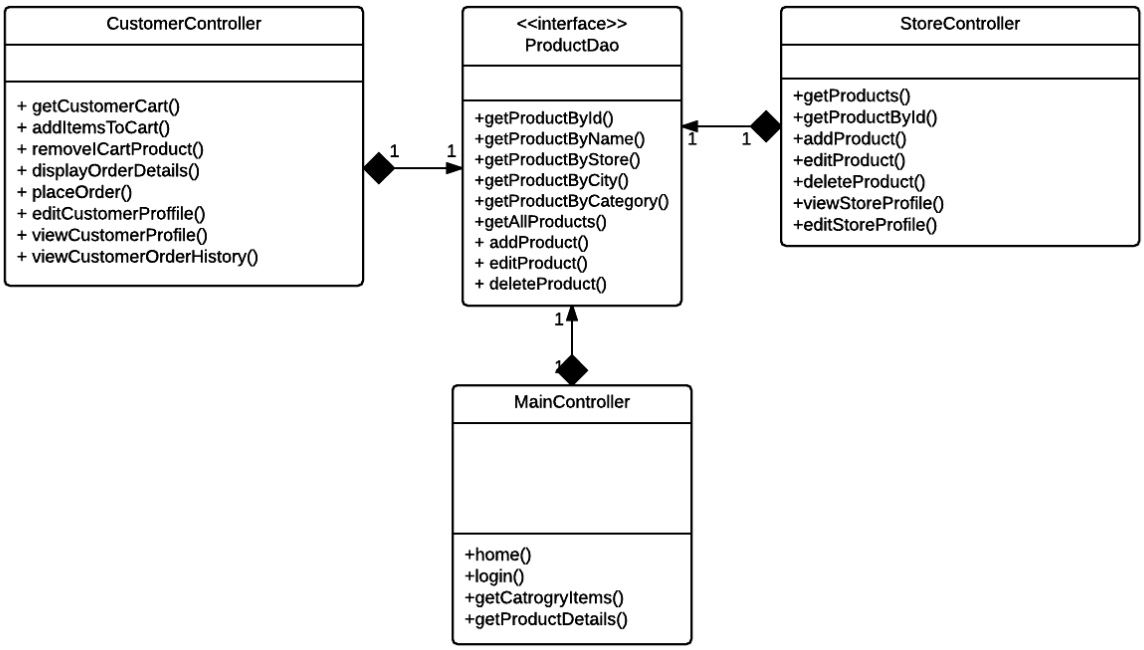
4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF). If not, where could you make use of design patterns in your system? Show a class diagram of how you could implement each design pattern and compare how it would change from your current class diagram.

Our project is based on MVC architecture and the design pattern we implemented are singleton and strategy. Using MVC provides an important advantage as it separates the user interaction from application part.

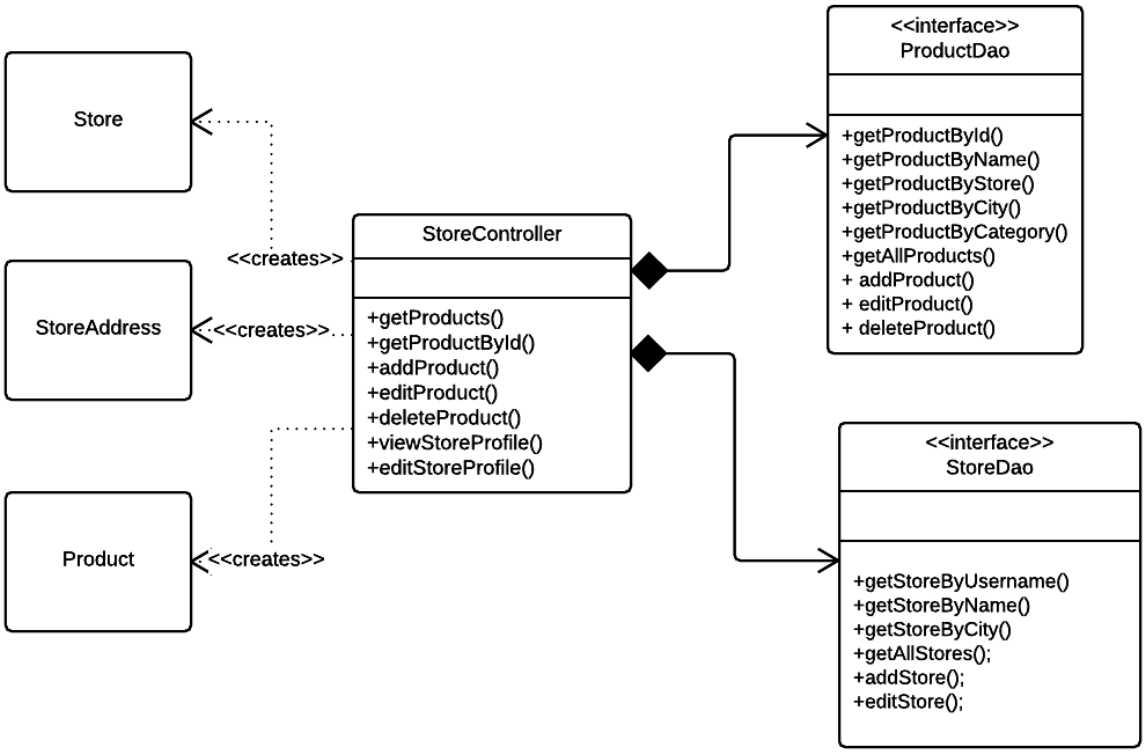
Singleton design pattern was implemented by making use of DAO (data access objects) which are responsible for manipulating the data from the database. Our project consists of DAO as an interface which then implemented by DAO classes to perform CRUD operations. The controllers

make use of these DAO interface to do all the CRUD operation. Since, the interface cannot be instantiated and can only refer the objects which implemented these interface, Singleton pattern is followed.

Singleton Pattern



Strategy design pattern was implemented through the use of controllers and view resolver (Spring). The view resolver delegates to controller for any decisions about the interface behavior.



5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

It was a good learning experience and we learned how important it is to learn about architecture and think about design patterns while doing a project. Using and developing on these implemented ideas makes the implementation smooth. We understood how important it is to analyze well in advance and design properly. We also learned how to think in advance and think what all user will want and how we as system developer would like to move ahead. We got stuck in few situations and we were able to see why planning ahead is good and how it makes the development easy and smooth.