Group 70
Swap Your Ride

Hanna Loboda
Matt Weber

1. List the features that were implemented.

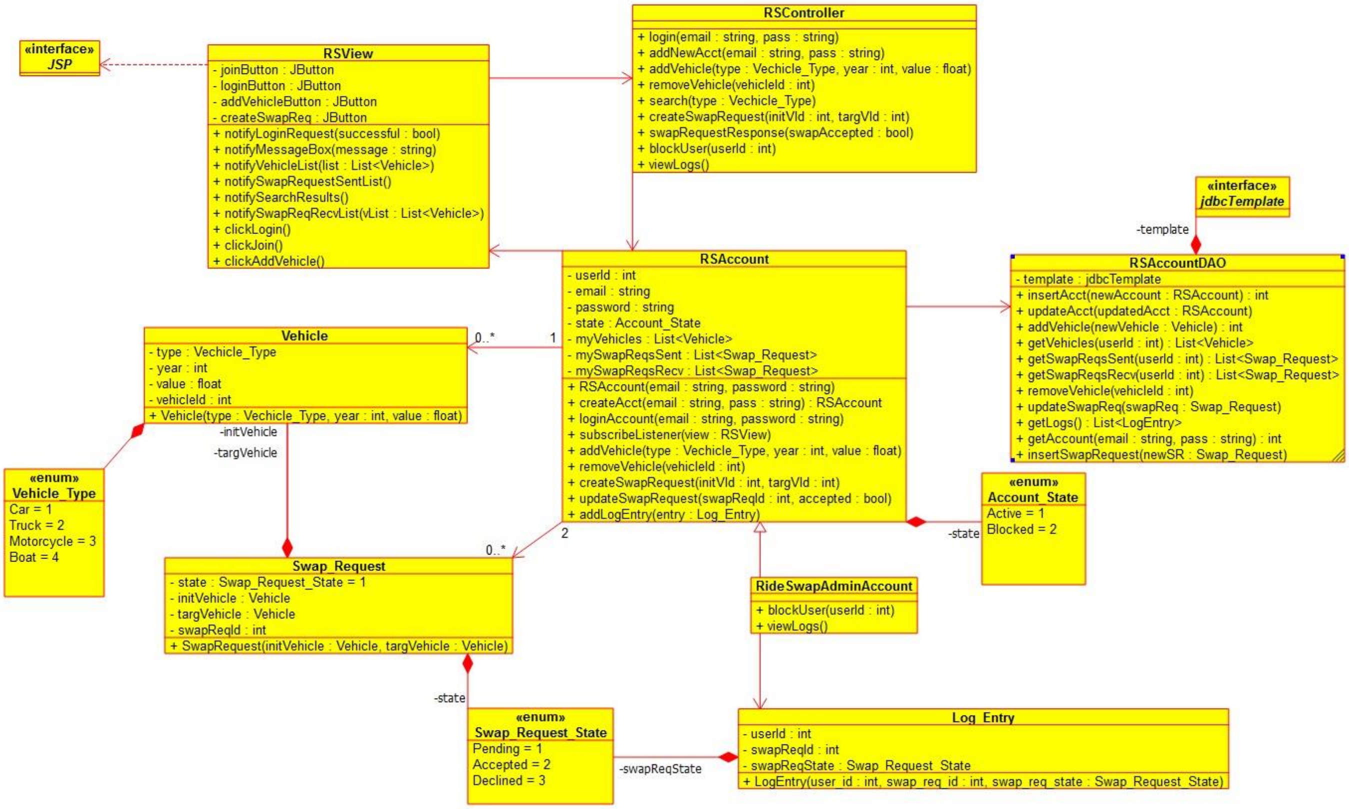
Implemented Features – User Requirements				
ID	**Requirement**	**Topic Area**	**Actor**	**Priority**
UR-01	Users can create a new account using email address	Login	User	Critical
UR-02	Users can login with email address	Login	User	Critical
UR-03	Users can add vehicles and vehicle data	Profile	User	Critical
UR-04	Users can remove vehicles	Profile	User	High
UR-05	Users can search for other vehicles	DB	User	High
UR-06	Users can create and send swap requests	DB	User	High
UR-07	Users can respond to pending swap requests	DB	User	High

2. List the features that were not implemented.

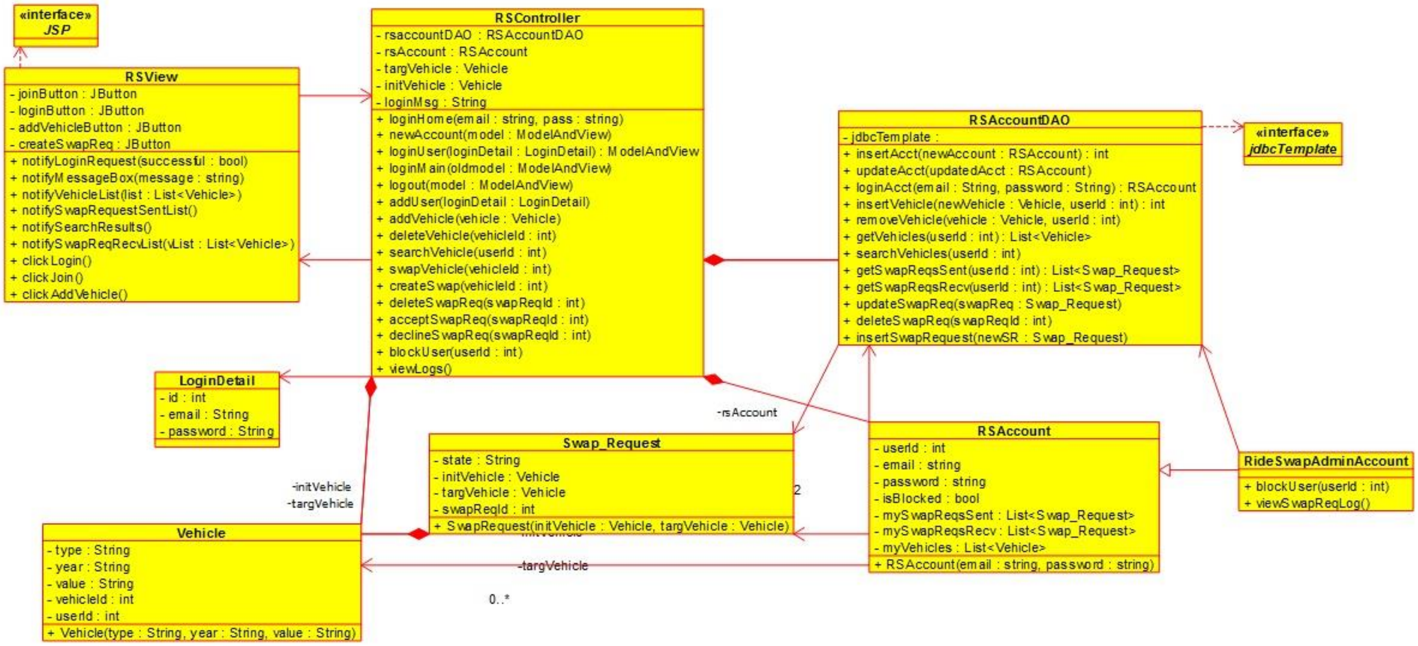
Unimplemented Features – User Requirements				
ID	**Requirement**	**Topic Area**	**Actor**	**Priority**
UR-08	Users can contact other users	DB	User, Admin	Low
UR-09	Admin can block users	DB	Admin	High
UR-10	Admin can review contact logs	Logging	Admin	Medium

3. Show your Part2 Class diagram and your final class diagram.

Part2 Class Diagram:


«interface»
JSP
RSView
- joinButton : JButton
- loginButton : JButton
- addVehicleButton : JButton
- createSwapReq : JButton
+ notifyLoginRequest(successful : bool)
+ notifyMessageBox(message : string)
+ notifyVehicleList(list : List<Vehicle>)
+ notifySwapRequestSentList()
+ notifySearchResults()
+ notifySwapReqRecvList(vList : List<Vehicle>)
+ clickLogin()
+ clickJoin()
+ clickAddVehicle()
RSController
+ login(email : string, pass : string)
+ addNewAcct(email : string, pass : string)
+ addVehicle(type : Vechicle_Type, year : int, value : float)
+ removeVehicle(vehicleId : int)
+ search(type : Vehicle_Type)
+ createSwapRequest(initVId : int, targVId : int)
+ swapRequestResponse(swapAccepted : bool)
+ blockUser(userId : int)
+ viewLogs()
«interface»
jdbcTemplate
-template
RSAccountDAO
- template : jdbcTemplate
+ insertAcct(newAccount : RSAccount) : int
+ updateAcct(updatedAcct : RSAccount)
+ addVehicle(newVehicle : Vehicle) : int
+ getVehicles(userId : int) : List<Vehicle>
+ getSwapReqsSent(userId : int) : List<Swap_Request>
+ getSwapReqsRecv(userId : int) : List<Swap_Request>
+ removeVehicle(vehicleId : int)
+ updateSwapReq(swapReq : Swap_Request)
+ getLogs() : List<LogEntry>
+ getAccount(email : string, pass : string) : int
+ insertSwapRequest(newSR : Swap_Request)
RSAccount
- userId : int
- email : string
- password : string
- state : Account_State
- myVehicles : List<Vehicle>
- mySwapReqsSent : List<Swap_Request>
- mySwapReqsRecv : List<Swap_Request>
+ RSAccount(email : string, password : string)
+ createAcct(email : string, pass : string) : RSAccount
+ loginAccount(email : string, password : string)
+ subscribeListener(view : RSView)
+ addVehicle(type : Vechicle_Type, year : int, value : float)
+ removeVehicle(vehicleId : int)
+ createSwapRequest(initVId : int, targVId : int)
+ updateSwapRequest(swapReqId : int, accepted : bool)
+ addLogEntry(entry : Log_Entry)
0..*
1
Vehicle
- type : Vechicle_Type
- year : int
- value : float
- vehicleId : int
+ Vehicle(type : Vechicle_Type, year : int, value : float)
-initVehicle
-targVehicle
«enum»
Vehicle_Type
Car = 1
Truck = 2
Motorcycle = 3
Boat = 4
2
0..*
Swap_Request
- state : Swap_Request_State = 1
- initVehicle : Vehicle
- targVehicle : Vehicle
- swapReqId : int
+ SwapRequest(initVehicle : Vehicle, targVehicle : Vehicle)
«enum»
Account_State
Active = 1
Blocked = 2
-state
RideSwapAdminAccount
+ blockUser(userId : int)
+ viewLogs()
-state
«enum»
Swap_Request_State
Pending = 1
Accepted = 2
Declined = 3
-swapReqState
Log_Entry
- userId : int
- swapReqId : int
- swapReqState : Swap_Request_State
+ LogEntry(user_id : int, swap_req_id : int, swap_req_state : Swap_Request_State)

Final Class Diagram:


«interface»
JSP
RSView
- joinButton : JButton
- loginButton : JButton
- addVehicleButton : JButton
- createSwapReq : JButton
+ notifyLoginRequest(successful : bool)
+ notifyMessageBox(message : string)
+ notifyVehicleList(list : List<Vehicle>)
+ notifySwapRequestSentList()
+ notifySearchResults()
+ notifySwapReqRecvList(vList : List<Vehicle>)
+ clickLogin()
+ clickJoin()
+ clickAddVehicle()
RSController
- rsaccountDAO : RSAccountDAO
- rsAccount : RSAccount
- targVehicle : Vehicle
- initVehicle : Vehicle
- loginMsg : String
+ loginHome(email : string, pass : string)
+ newAccount(model : ModelAndView)
+ loginUser(loginDetail : LoginDetail) : ModelAndView
+ loginMain(oldmodel : ModelAndView)
+ logout(model : ModelAndView)
+ addUser(loginDetail : LoginDetail)
+ addVehicle(vehicle : Vehicle)
+ deleteVehicle(vehicleId : int)
+ searchVehicle(userId : int)
+ swapVehicle(vehicleId : int)
+ createSwap(vehicleId : int)
+ deleteSwapReq(swapReqId : int)
+ acceptSwapReq(swapReqId : int)
+ declineSwapReq(swapReqId : int)
+ blockUser(userId : int)
+ viewLogs()
RSAccountDAO
- jdbcTemplate :
+ insertAcct(newAccount : RSAccount) : int
+ updateAcct(updatedAcct : RSAccount)
+ loginAcct(email : String, password : String) : RSAccount
+ insertVehicle(newVehicle : Vehicle, userId : int) : int
+ removeVehicle(vehicle : Vehicle, userId : int)
+ getVehicles(userId : int) : List<Vehicle>
+ searchVehicles(userId : int)
+ getSwapReqsSent(userId : int) : List<Swap_Request>
+ getSwapReqsRecv(userId : int) : List<Swap_Request>
+ updateSwapReq(swapReq : Swap_Request)
+ deleteSwapReq(swapReqId : int)
+ insertSwapRequest(newSR : Swap_Request)
«interface»
jdbcTemplate
LoginDetail
- id : int
- email : String
- password : String
-rsAccount
-initVehicle
-targVehicle
Swap_Request
- state : String
- initVehicle : Vehicle
- targVehicle : Vehicle
- swapReqId : int
+ SwapRequest(initVehicle : Vehicle, targVehicle : Vehicle)
2
RSAccount
- userId : int
- email : string
- password : string
- isBlocked : bool
- mySwapReqsSent : List<Swap_Request>
- mySwapReqsRecv : List<Swap_Request>
- myVehicles : List<Vehicle>
+ RSAccount(email : string, password : string)
RideSwapAdminAccount
+ blockUser(userId : int)
+ viewSwapReqLog()
Vehicle
- type : String
- year : String
- value : String
- vehicleId : int
- userId : int
+ Vehicle(type : String, year : String, value : String)
-targVehicle
0..*

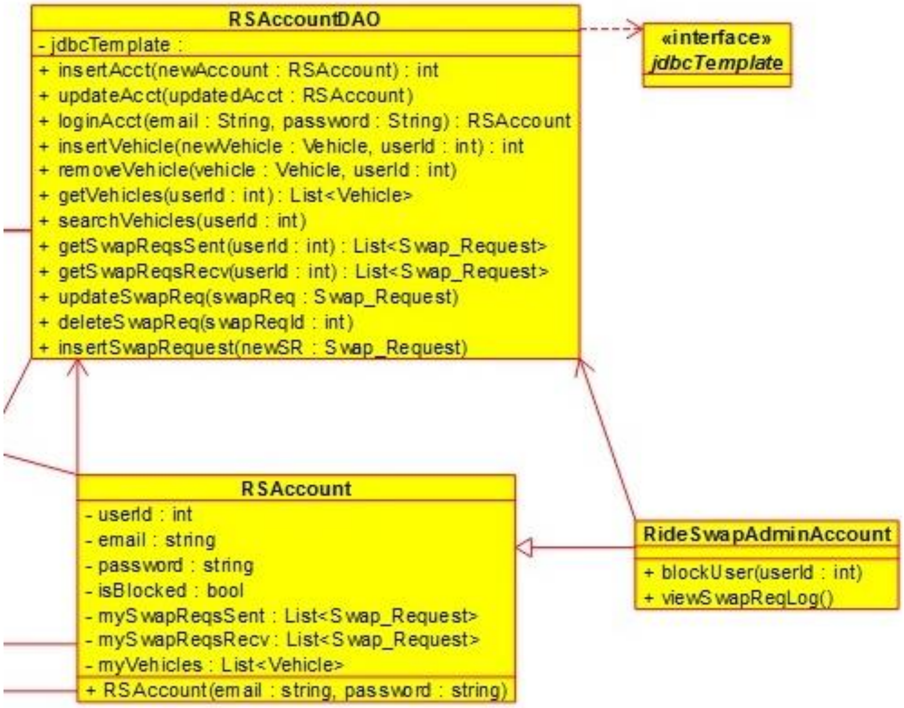
3. What changed? Why?

A lot of functionality that was in RSAccount class was moved to RSController. It made more sense for the controller class to interact directly with the RSAccountDAO then go through the model classes. A lot of the changes were more due to streamlining the design to fit within an MVC architecture. We also removed

Enums as these do not work well passing between DB and JSP pages. Log_Entry class was also removed due to too much redundant code that was already contained in the other model classes.

4. **Did you make use of any design patterns in the implementation of your final prototype?**

   We looked at the various design patterns and decided the only one we could really apply to our project was the Factory design pattern. Rather than expose the RSController class to both RSAccount and RSAdminAccount, we made the RSAccountDAO act as a Factory and determine (based on data stored in the database) which class type to instantiate and pass back to the controller class.


RSAccountDAO
- jdbcTemplate :
+ insertAcct(newAccount : RSAccount) : int
+ updateAcct(updatedAcct : RSAccount)
+ loginAcct(email : String, password : String) : RSAccount
+ insertVehicle(newVehicle : Vehicle, userId : int) : int
+ removeVehicle(vehicle : Vehicle, userId : int)
+ getVehicles(userId : int) : List<Vehicle>
+ searchVehicles(userId : int)
+ getSwapReqsSent(userId : int) : List<Swap_Request>
+ getSwapReqsRecv(userId : int) : List<Swap_Request>
+ updateSwapReq(swapReq : Swap_Request)
+ deleteSwapReq(swapReqId : int)
+ insertSwapRequest(newSR : Swap_Request)
«interface»
jdbcTemplate
RSAccount
- userId : int
- email : string
- password : string
- isBlocked : bool
- mySwapReqsSent : List<Swap_Request>
- mySwapReqsRecv : List<Swap_Request>
- myVehicles : List<Vehicle>
+ RSAccount(email : string, password : string)
RideSwapAdminAccount
+ blockUser(userId : int)
+ viewSwapReqLog()

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?**

   We have learned that analysis and design is a critical aspect of creating any software project, and spending more time during this process creates a better overall product. The ability to recognize and apply design patterns is hugely beneficial, especially when being able to apply them early in the design process and not have to refactor a ton of code to do so. Another main lesson we learned is to also become familiar with the architecture of the project; we were not familiar with Spring MVC and spent a great deal of time trying to get our design to fit within the parameters of Spring MVC.