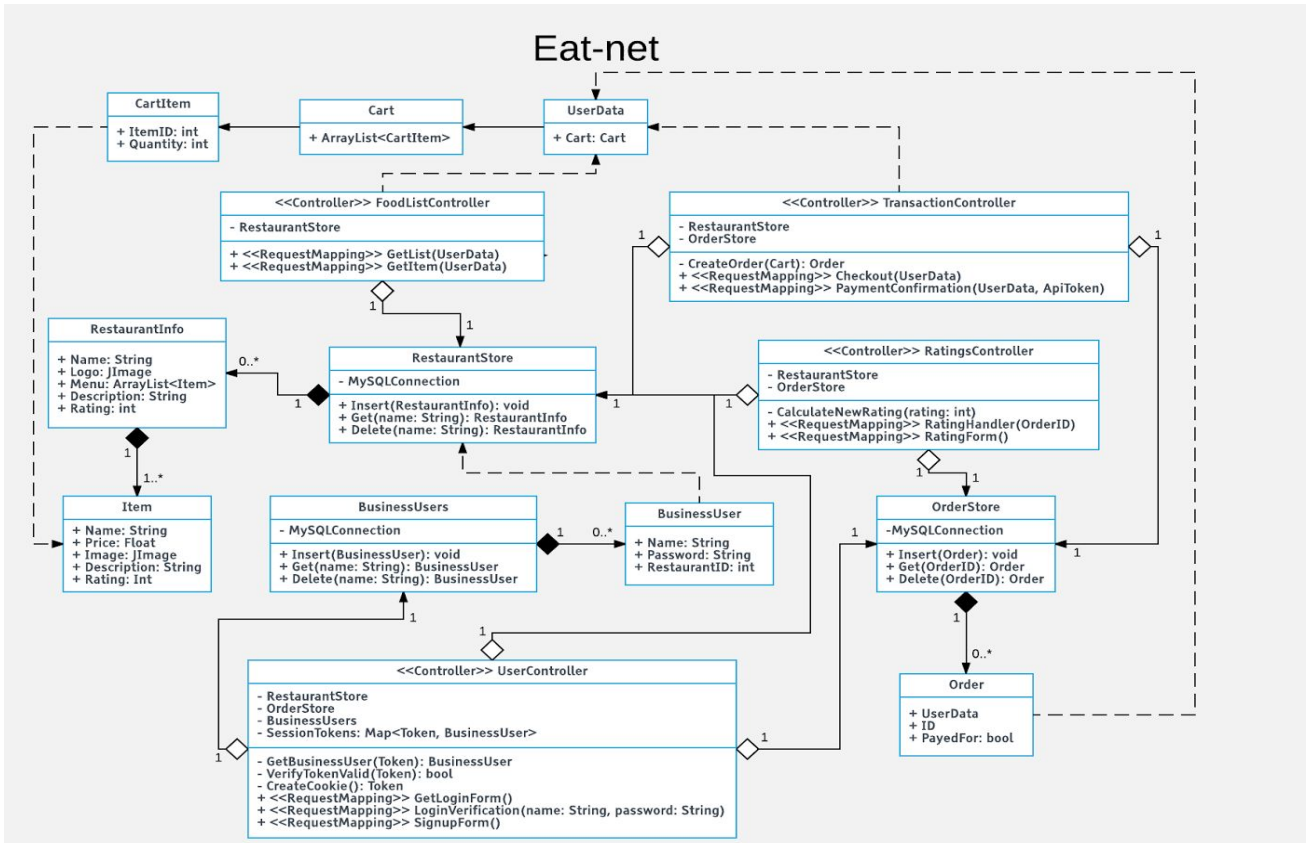
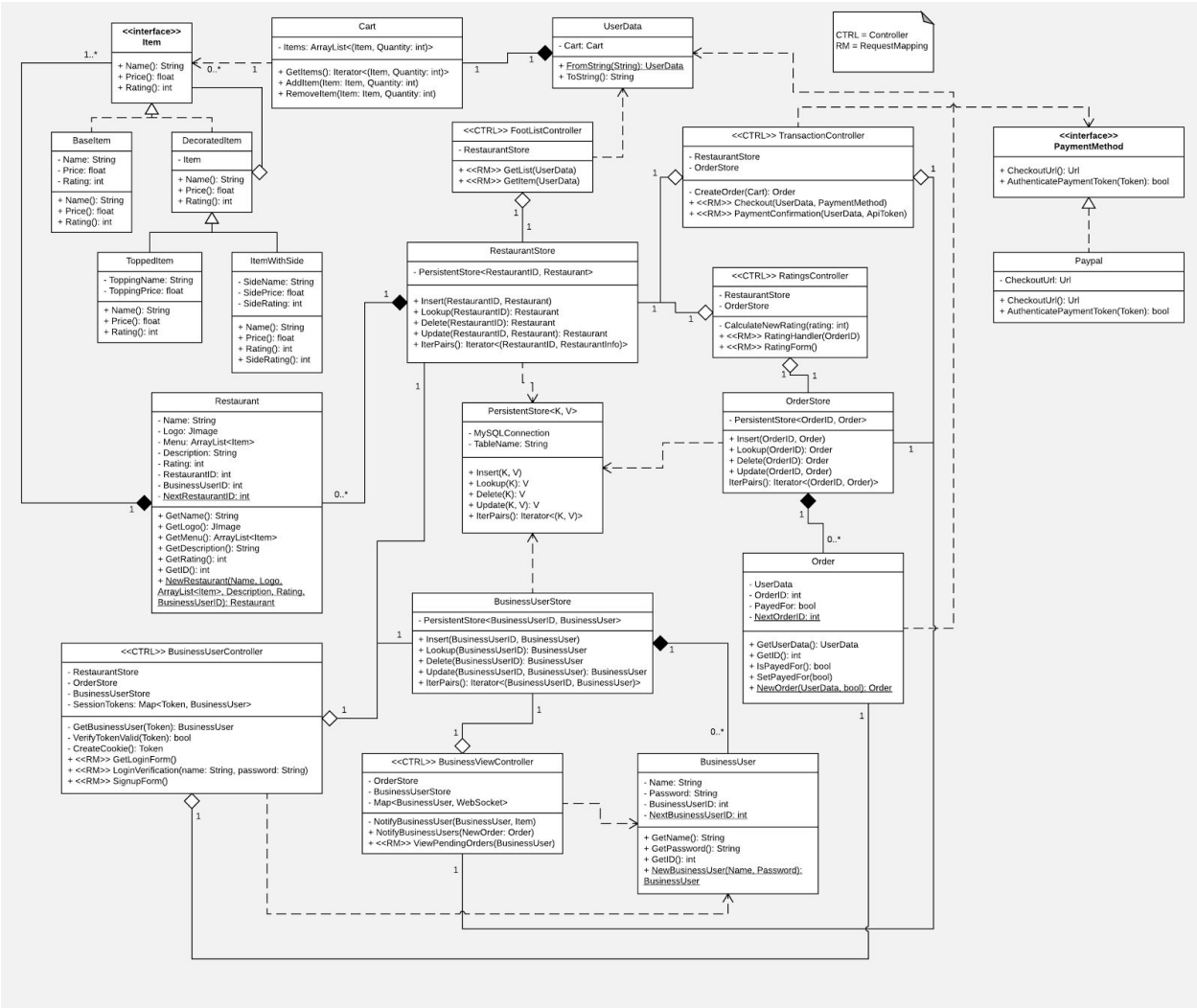

Eat-net
CartItem
+ ItemID: int
+ Quantity: int
Cart
+ ArrayList<CartItem>
UserData
+ Cart: Cart
<<Controller>> FoodListController
- RestaurantStore
+ <<RequestMapping>> GetList(UserData)
+ <<RequestMapping>> GetItem(UserData)
<<Controller>> TransactionController
- RestaurantStore
- OrderStore
- CreateOrder(Cart): Order
+ <<RequestMapping>> Checkout(UserData)
+ <<RequestMapping>> PaymentConfirmation(UserData, ApiToken)
RestaurantInfo
+ Name: String
+ Logo: JImage
+ Menu: ArrayList<Item>
+ Description: String
+ Rating: int
RestaurantStore
- MySQLConnection
+ Insert(RestaurantInfo): void
+ Get(name: String): RestaurantInfo
+ Delete(name: String): RestaurantInfo
<<Controller>> RatingsController
- RestaurantStore
- OrderStore
- CalculateNewRating(rating: int)
+ <<RequestMapping>> RatingHandler(OrderID)
+ <<RequestMapping>> RatingForm()
Item
+ Name: String
+ Price: Float
+ Image: JImage
+ Description: String
+ Rating: Int
BusinessUsers
- MySQLConnection
+ Insert(BusinessUser): void
+ Get(name: String): BusinessUser
+ Delete(name: String): BusinessUser
BusinessUser
+ Name: String
+ Password: String
+ RestaurantID: int
OrderStore
-MySQLConnection
+ Insert(Order): void
+ Get(OrderID): Order
+ Delete(OrderID): Order
<<Controller>> UserController
- RestaurantStore
- OrderStore
- BusinessUsers
- SessionTokens: Map<Token, BusinessUser>
- GetBusinessUser(Token): BusinessUser
- VerifyTokenValid(Token): bool
- CreateCookie(): Token
+ <<RequestMapping>> GetLoginForm()
+ <<RequestMapping>> LoginVerification(name: String, password: String)
+ <<RequestMapping>> SignupForm()
Order
+ UserData
+ ID
+ PayedFor: bool



Changes made:

- Added PaymentMethod interface and concrete Paypal class (Strategy pattern)
- Added BusinessViewController, which has notify methods that the TransactionController uses to refresh Business Users's view (using an open websocket) whenever a new transaction is completed. (Observer pattern)
- Added topping and side dish composition using the decorator pattern
- Created PersistentStore class which implements SQL persistent data storage functionality. RestaurantStore, BusinessUserStore, and OrderStore have shim methods which delegate to the internal PersistentStore object.
- Renamed RestaurantInfo to Restaurant, BusinessUsers to BusinessUserStore
- Altered entity classes (BusinessUser, Restaurant, Order) to contain static constructor methods that generate unique ID's.