Project Part 6

Team: 05 Rentalit

Nicholas Clement
Joel Marquez
Brennon Lee
Christopher Struckman

A Rental Service

Features that were implemented:

ID	Title
BR-004	All postings must be stored in the database
UR-003	Users must be able to search for products they want to rent
UR-004	Users must be able to rent products from other users
UR-005	Users must be able to post items that can be rented
UR-007	Users must be able to track items currently rented out
UR-008	Users must be able to set a rental period

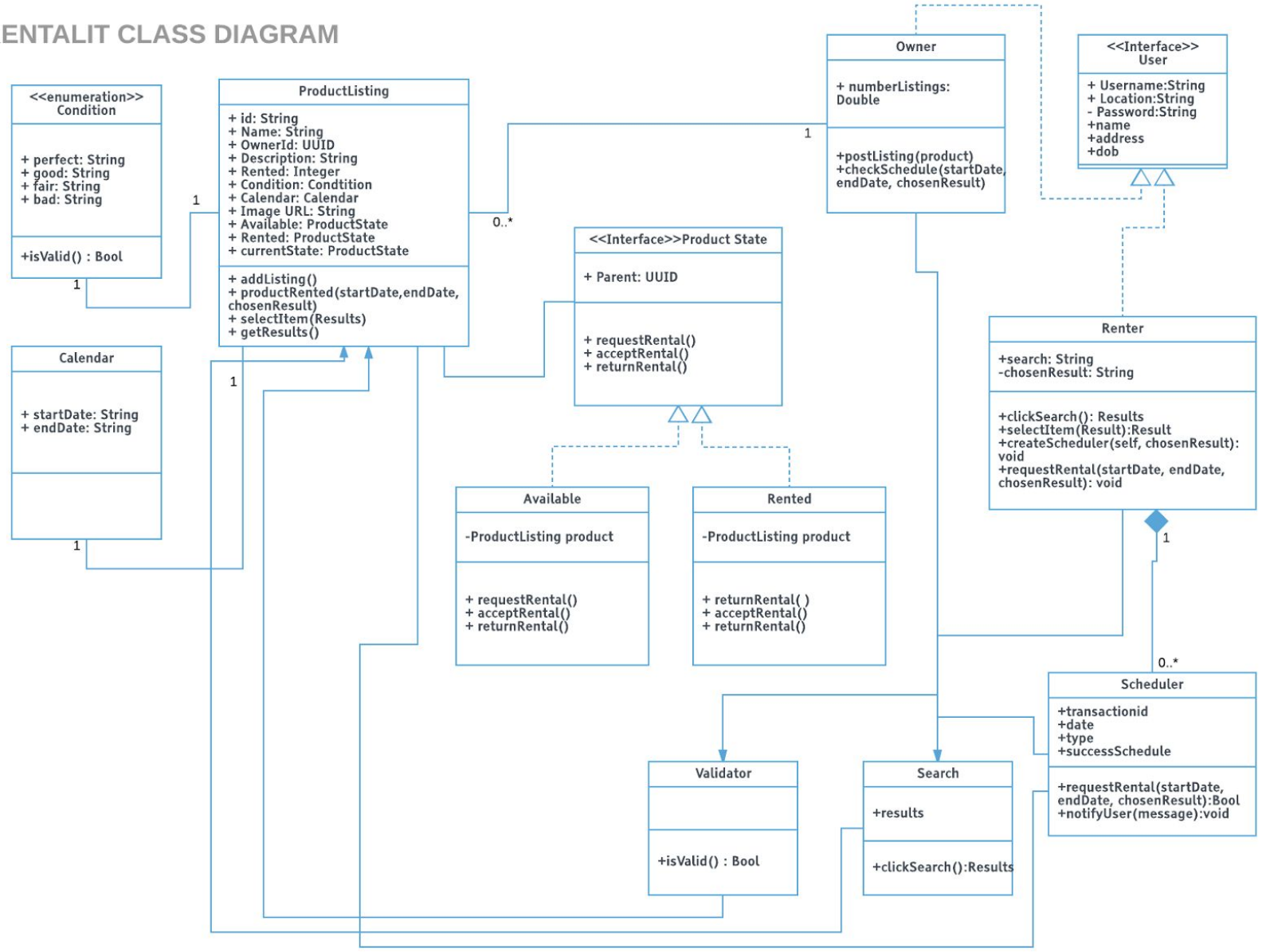
Features that were NOT implemented:

ID	Title
UR-001	Users can create an account
UR-002	Admins and Users can login
UR-009	Users must be able to view previous items they rented
UR-010	Users can see ratings
UR-011	Admins can search for customers by name
BR-002	All users must have a password
BR-001	All users must have a login username consisting of first letter firstname + lastname

Project Part 2 Class Diagram:

Final Class Diagram:

RENTALIT CLASS DIAGRAM

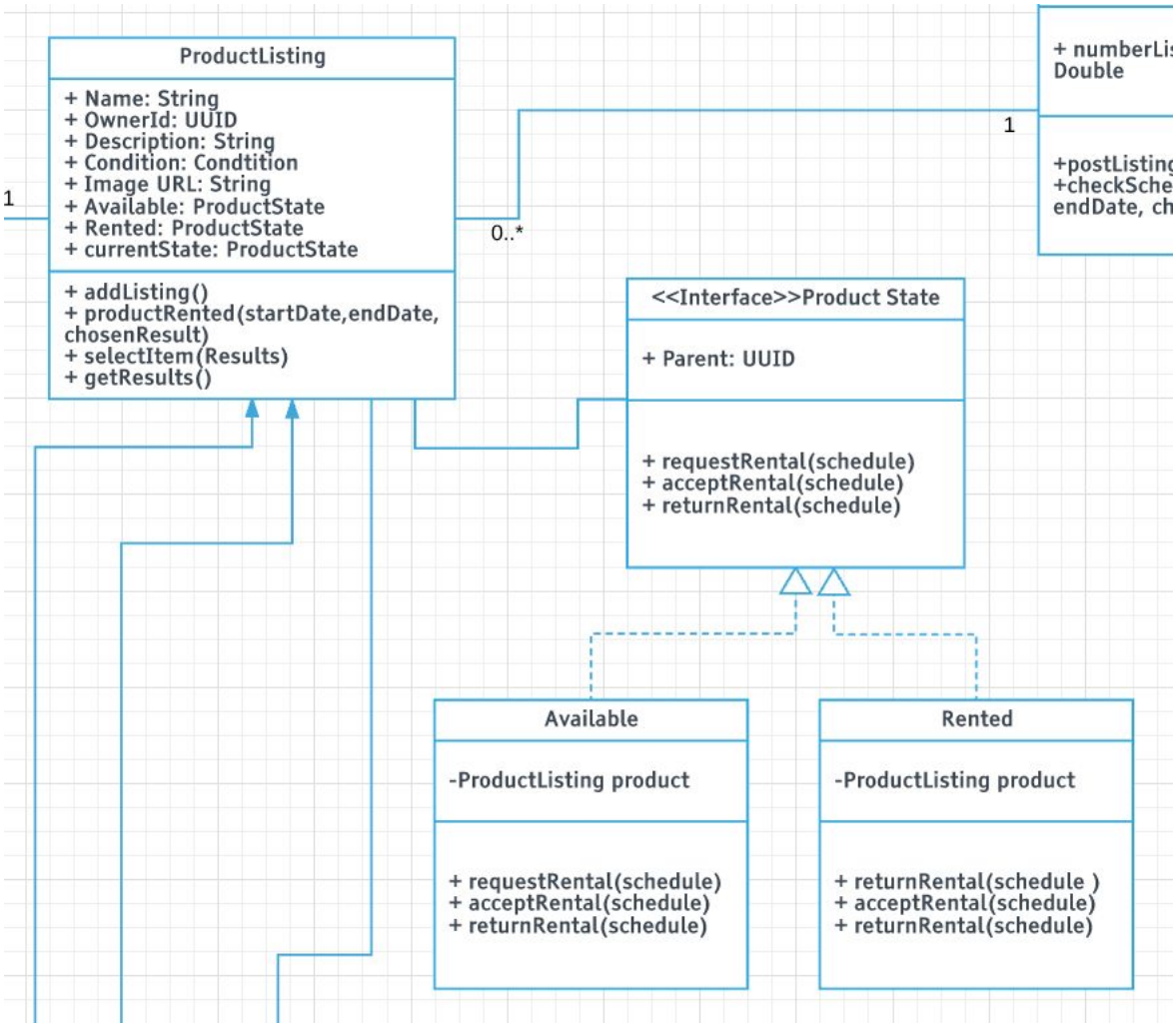


Changes

1. Product Listing:
    a. In our final class diagram, we added additional attributes to the Product Listing object to better handle the operations on the object. First, we added an integer depicting whether the listing is rented or available. Using this bit, we were able to set the current state on deserialization.
    b. Second, we introduced a calendar object which was then used as an attribute for the product listing class and which tracked the rental period for an item
2. Product State:
    a. There was a minor change in the final version of the product state's methods. Originally, we designed them to take in a Schedule object which would hold the desired rental period, however, in the final version we realized that it was unnecessary since the calendar attribute would already be set by this point.

Implemented Design Patterns

During the "Design Pattern Refactoring" opportunity we had in class, we discovered that the State design pattern suited our rental project idea. The State design pattern allows objects to change their behavior based off of the current internal state. For our project, this mapped over to each rentable Product object within our rental system. Each Product would have its own state of "rented" or "available". And given a product's current state, there would be specific methods available.



Takeaways

One of the biggest takeaways from going through this experience has been the realization of how much easier the implementation of a project becomes when there is a good design. By having spent so much of our time designing our system, refactoring and applying design patterns where possible; the implementation became much simpler, less buggy and overall a more enjoyable experience.

Another key takeaway from this project and the course as a whole was anti-patterns. It was very helpful to plan out our design patterns prior to coding, otherwise it is easy to use an anti-pattern as a quick solution to a common problem. Designing our system prior to coding it also allowed us to have an efficient development cycle, and plan out features accordingly. It helped us to divy up the workload easily, and concurrently develop features that were independent.