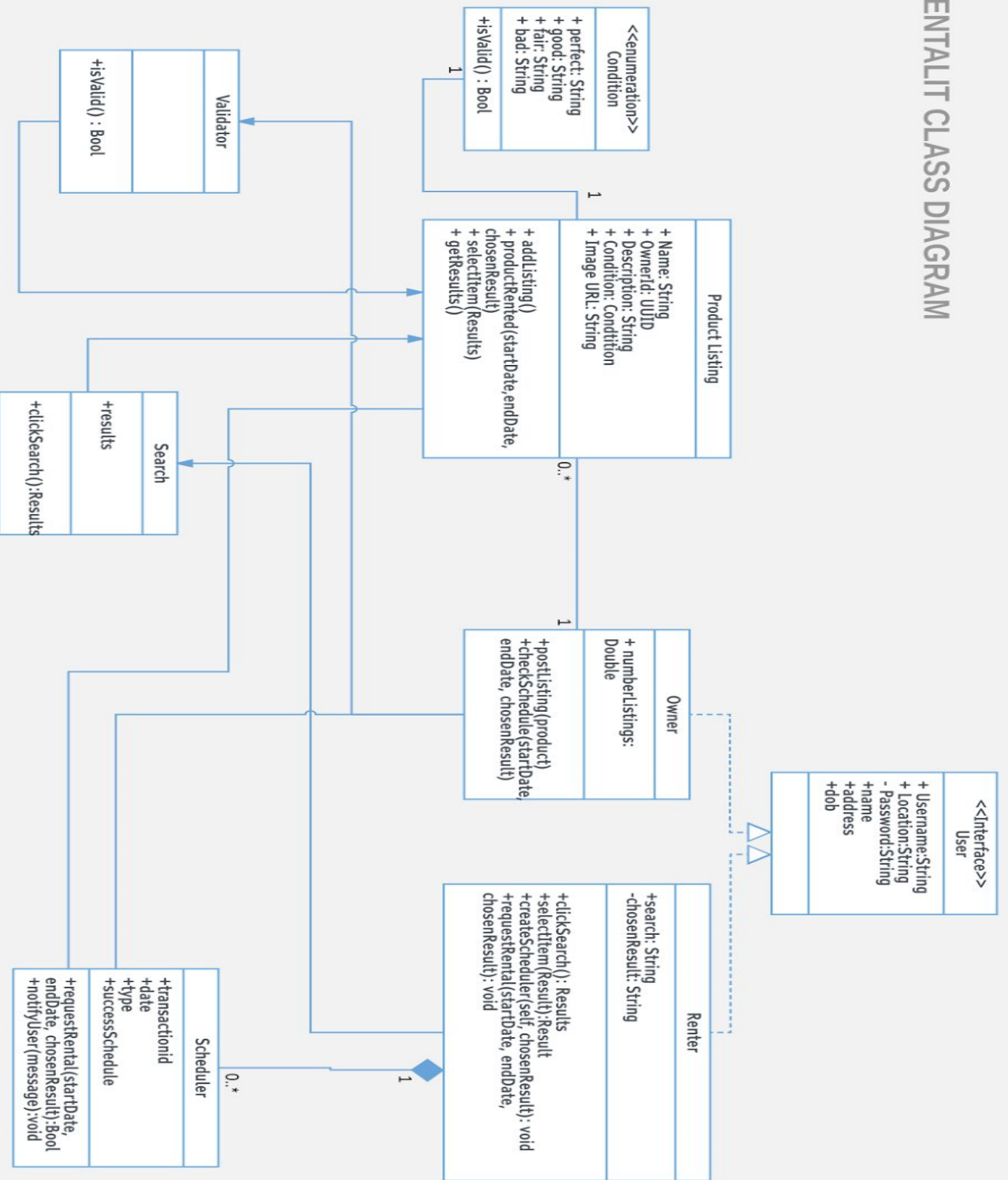# Project Part 3

# Team: 05 Rentalit

Nicholas Clement
Joel Marquez
Brennon Lee
Christopher Struckman
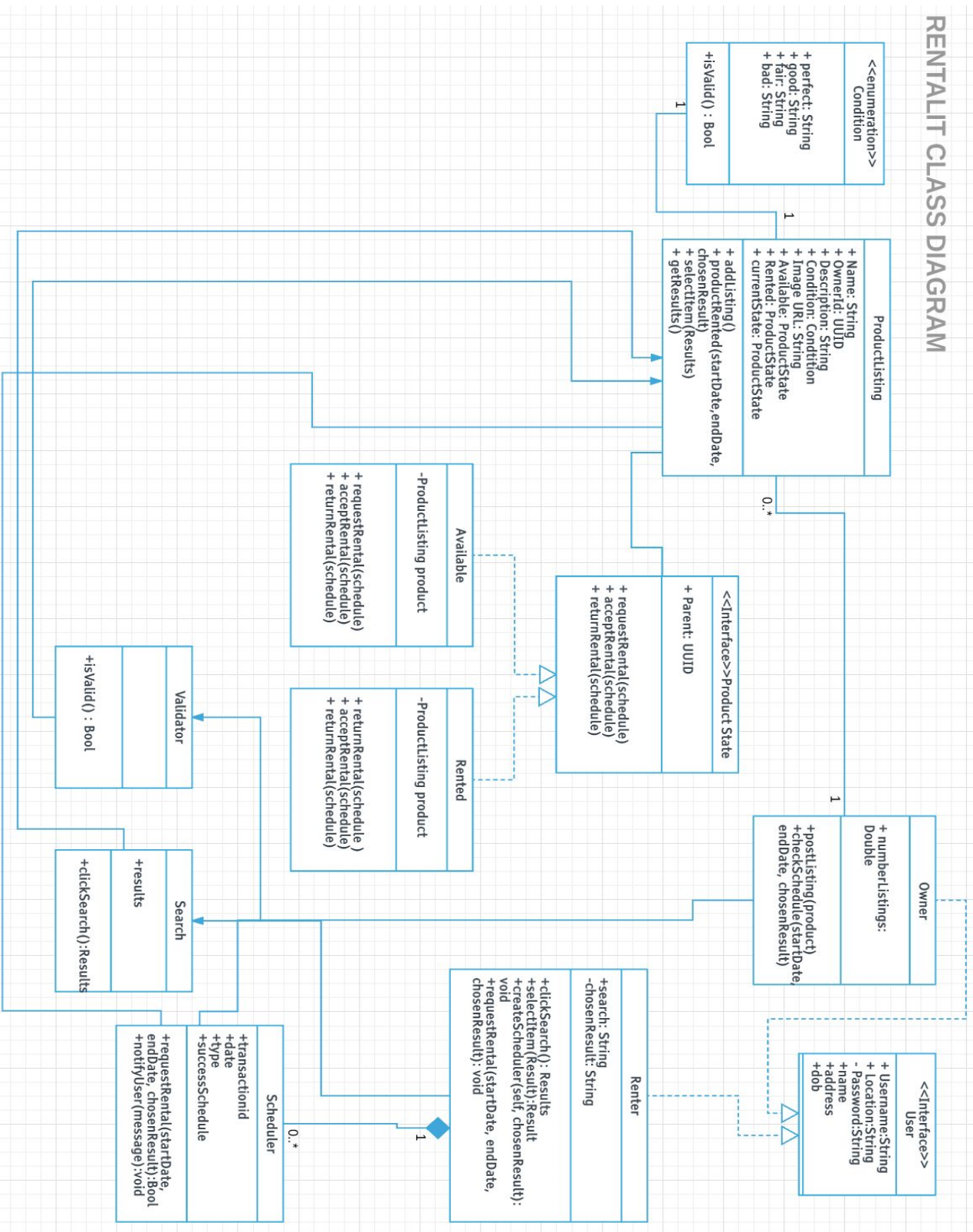
*A Rental Service*

Project Part 2 Class Diagram:

RENTALIT CLASS DIAGRAM

**<<enumeration>> Condition**
+ perfect: String
+ good: String
+ fair: String
+ bad: String

+isValid() : Bool

**Validator**

+isValid() : Bool

**Product Listing**
+ Name: String
+ OwnerId: UUID
+ Description: String
+ Condition: Condition
+ Image URL: String

+ addListing()
+ productRented(startDate, endDate, chosenResult)
+ selectItem(Results)
+ getResults()

**Search**
+results

+clickSearch():Results

**Owner**
+ numberListings: Double

+postListing(product)
+checkSchedule(startDate, endDate, chosenResult)

**<<Interface>> User**
+ Username:String
+ Location:String
- Password:String
+name
+address
+dob

**Renter**
+search: String
-chosenResult: String

+clickSearch(): Results
+selectItem(Result):Result
+createScheduler(self, chosenResult): void
+requestRental(startDate, endDate, chosenResult): void

**Scheduler**
+transactionid
+date
+type
+successSchedule

+requestRental(startDate, endDate, chosenResult):Bool
+notifyUser(message):void

1

1

1

0..*

1

0..*

Project Part 3 Class Diagram:

RENTALIT CLASS DIAGRAM

**<<enumeration>> Condition**
+ perfect: String
+ good: String
+ fair: String
+ bad: String

+isValid() : Bool

**ProductListing**
+ Name: String
+ OwnerId: UUID
+ Description: String
+ Condition: Condition
+ Image URL: String
+ Available: ProductState
+ Rented: ProductState
+ currentState: ProductState

+ addListing()
+ productRented(startDate,endDate, chosenResult)
+ selectItem(Results)
+ getResults()

**Available**
-ProductListing product

+ requestRental(schedule)
+ acceptRental(schedule)
+ returnRental(schedule)

**Rented**
-ProductListing product

+ returnRental(schedule)
+ acceptRental(schedule)
+ returnRental(schedule)

**<<Interface>>Product State**
+ Parent: UUID

+ requestRental(schedule)
+ acceptRental(schedule)
+ returnRental(schedule)

**Validator**
+isValid() : Bool

**Search**
+results

+clickSearch():Results

**Owner**
+ numberListings: Double

+postListing(product)
+checkSchedule(startDate, endDate, chosenResult)

**Renter**
+search: String
-chosenResult: String

+clickSearch() : Results
+selectItem(Result):Result
+createScheduler(self, chosenResult):
void
+requestRental(startDate, endDate,
chosenResult): void

**Scheduler**
+transactionid
+date
+type
+successSchedule

+requestRental(startDate,
endDate, chosenResult):Bool
+notifyUser(message):void

**<<Interface>> User**
+ Username:String
+ Location:String
- Password:String
+name
+address
+dob

1    1    0..*    1    0..*    1

<u>Design Patterns:</u>

After digging deeper into design patterns in this course, we noticed the State design pattern suited our project idea perfectly. 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.

For example a rented product would have a "returnRental" method whose behavior would intuitively be a renter returning an item back to the owner. This method would take in a "schedule" parameter which is used to access the current internal state of the Product.

Following the diagram of the State Pattern provided in the lecture slides:

# State Pattern - Structure
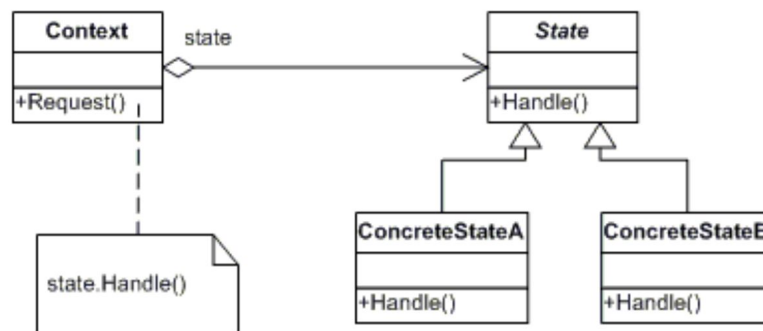


Our structure mapped over similarly where our "Context" is the "Product Listing" and each Product has its own "Product State". Based off of the current internal state of the Product, methods in "Available" or "Rented" are used to change the behavior of each Product.

**ProductListing**

+ Name: String
+ OwnerId: UUID
+ Description: String
+ Condition: Condtition
+ Image URL: String
+ Available: ProductState
+ Rented: ProductState
+ currentState: ProductState

+ addListing()
+ productRented(startDate,endDate, chosenResult)
+ selectItem(Results)
+ getResults()

1

0..*

1

+ numberLis
Double

+postListing
+checkSche
endDate, ch

**<<Interface>>Product State**

+ Parent: UUID

+ requestRental(schedule)
+ acceptRental(schedule)
+ returnRental(schedule)

**Available**

-ProductListing product

+ requestRental(schedule)
+ acceptRental(schedule)
+ returnRental(schedule)

**Rented**

-ProductListing product

+ returnRental(schedule )
+ acceptRental(schedule)
+ returnRental(schedule)