S2T4 – BugBunny - BillBoard Evaluation

Hi everyone! I have some feedbacks on your D2 evaluations, please make changes accordingly.

Sequence Diagram Checklist

1. **Overall Structure**

The titles are clear and it's according to the purpose. The logical left-to-right flow is shown. The layout is readable and properly shown. Sequence diagrams also include explanations. You might add your use cases instead of explanations but that's fine too.

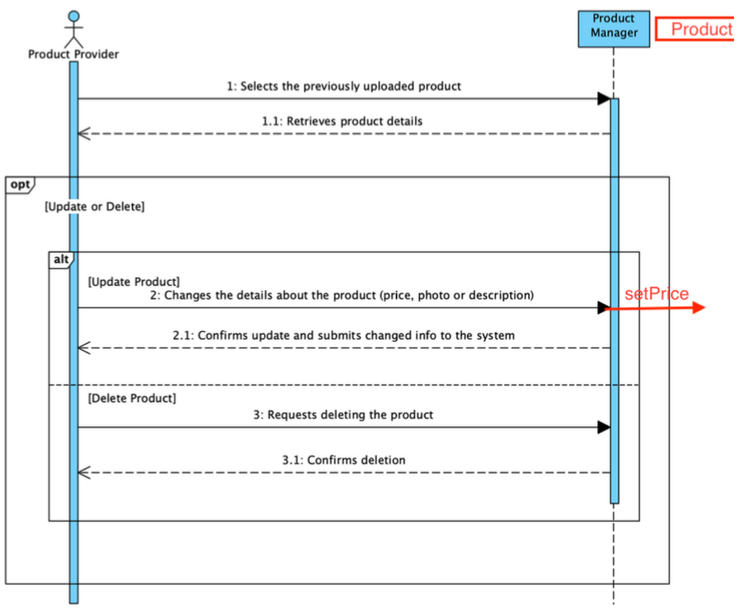
2. **Messages and Interactions**

Sender and receiver lifelines are correctly shown. Activation bars reflect the activity durations and control flow representation is done. Messages are also labeled with verbs and objects.

I could not find a state diagram for messages. The state diagram for messages and notifications would be nice to have.

Also, on the "Handle with Product" page, I find it interesting that the ProductManager does all the job without interacting with the Product. I would suggest adding an interaction with the Products too since updateProduct is likely to call Product.setPrice(Price priceOne) or Product.setPhoto(Photo img).

2.5. Handle with a Product


Product Provider
Product Manager
Product
1: Selects the previously uploaded product
1.1: Retrieves product details
opt
[Update or Delete]
alt
[Update Product]
2: Changes the details about the product (price, photo or description)
setPrice
2.1: Confirms update and submits changed info to the system
[Delete Product]
3: Requests deleting the product
3.1: Confirms deletion

Class Diagrams Checklist

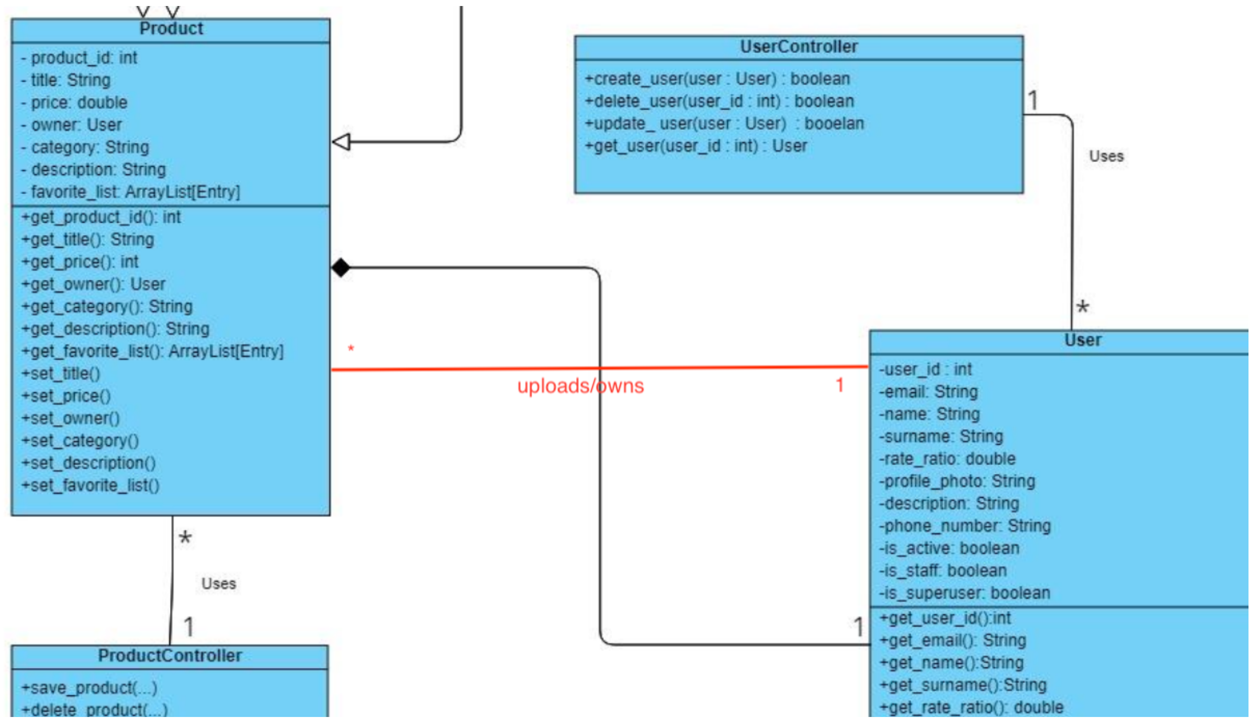
1. Classes

All necessary classes are present and clearly named. Class names are singular and capitalized. There isn't any redundant or ambiguous class names. Elements are named properly.

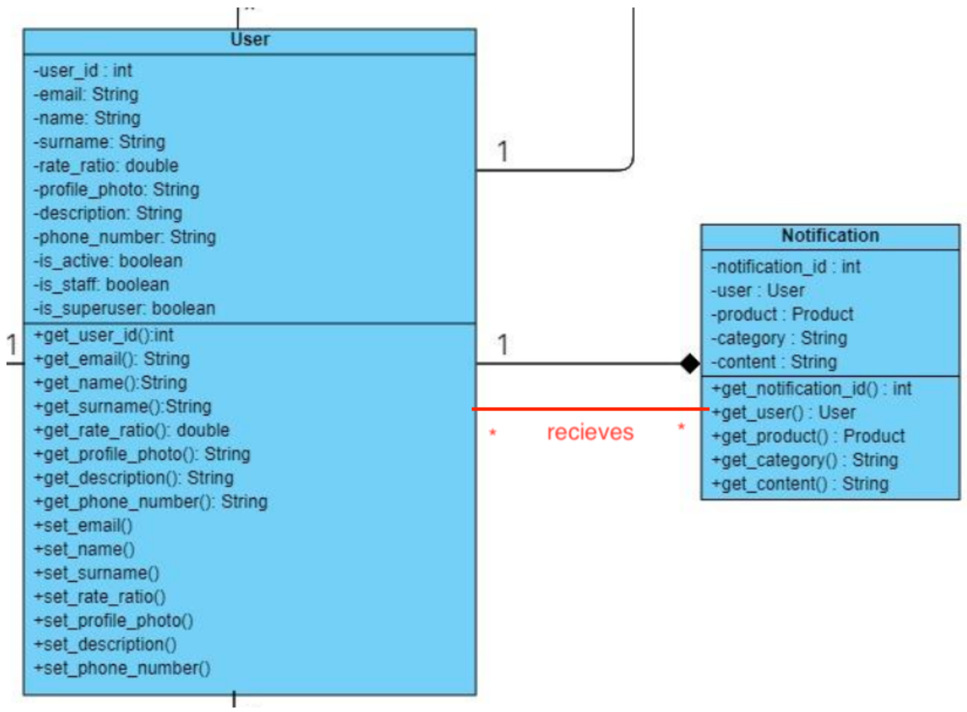
2. Associations

Associations are readable and recognizable. However, I have spotted some issues on some of the aggregations. Why does the user and the product have an aggregate relationship? The same thing goes for the user and the notifications.

I would suggest revising your aggregations. The association should be defined instead of a composition. User can exist on their own, so you should not think of a strong composition for this.



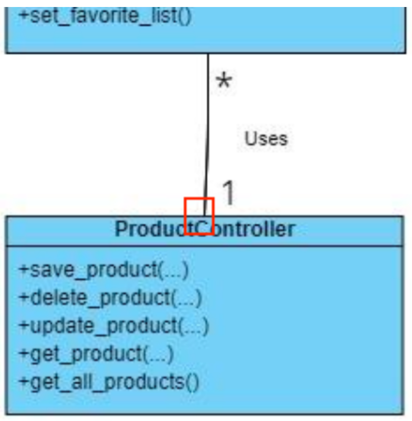
The same thing goes for the notifications and entries. Why does user have a composite relationship with the notifications and entries? The notifications and the user should have a many-to-many relationship. A notification can go to multiple users and a user might receive multiple notifications.



Why does the user class have a composite association with the entry class?

I would suggest thinking about the usage of the composition in this section since it is not meaningful that the User class is dependent on 3 other classes. It seems like if the notifications are dead, it means users are also dead too. Composition is like a "death relationship". So, does this mean a User cannot be on its own? Should the user be defined with notifications, entries, and products? It seems very irrelevant to me. Might as well as define them in an association, not in a composite relationship.

Additionally, for the controllers' part, I believe you might add an aggregation instead of an association (suppose I draw a diamond, not a square).



Since the ComplaintController is a third-party application, I am not including that. However, the UserController and ProductController should have an aggregate relationship.

Can the product controller exist on its own without the need for products? The products can exist on their own, however, I don't think it's true for ProductController. It needs products, otherwise, it is kind of useless.

There isn't any 1-1 to associations which is comprehensible. Multiplicity is specified on both ends of the diagrams.

Activity Diagram Checklist

1. Overall Structure

The titles and purpose are clearly defined. The sequence is logical, and it is mostly top-to-bottom.

2. Actions and Control Flow

Actions and activities are clearly labeled. The decision points are correctly used. Control flows are effectively used and accurately connected. The placement of loop and merge nodes is appropriate.

State Diagram Checklist

1. Overall Structure

State diagrams have a clear title and a purpose. Logical organization of states and transitions used. The layout and flow of direction are proper. State notations are appropriate.

2. States and Transitions

The labeling of the states is clear. Transition names are descriptive and understandable. The initial and final states are accurately represented. Entry and exit conditions are proper. The depiction of composite states is correct. However, the composite states are not shown. Deletion might be a composite state on its own.

3.3. Product Post/Update/Delete

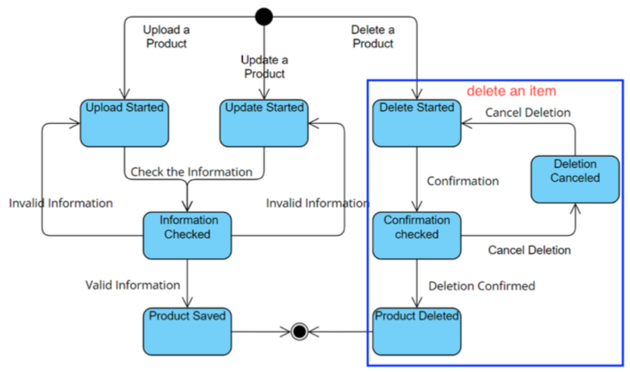


Figure 15: State diagram of product upload, update, and delete process

Mockup Evaluation Checklist

1. **Overall Design**

The overall design is understandable and it’s clear and user friendly. Local navigation and flow between screens are understandable. The font, colors, and other design elements are consistent. Appropriate white space is used for readability.

2. **Content & Information**

Each mockup screen has relevant information regarding its function. The labeling, headers, and instructions are clear on the pages. The usage of icons and images is relevant, however, there isn’t any image of the products. It might be good to add one for several images.

3. **Overall Evaluation**

User feedback is not added in the mockups. You should add a mockup on a page where a label says, “Give us feedback!” or something like that since it is required for the evaluation. You might also add a screen for the complaint page since it would be different from other pages. Additionally, you might show the notifications in your mockups since there is only the image of the notification. You did not show what’s inside the notification.