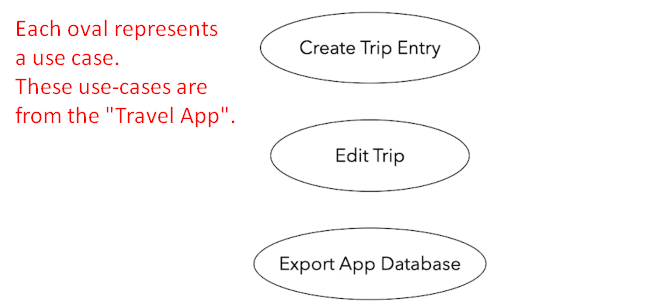
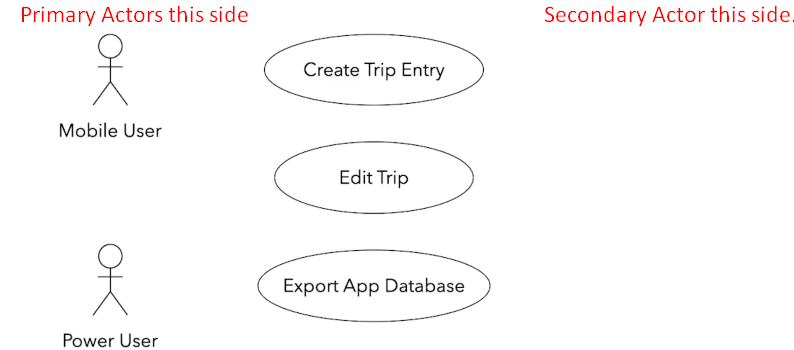
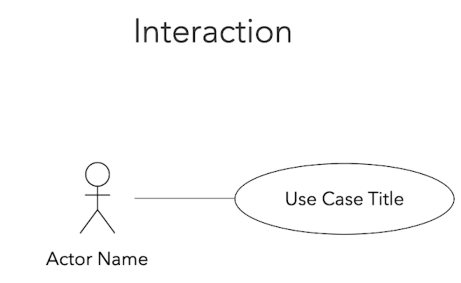
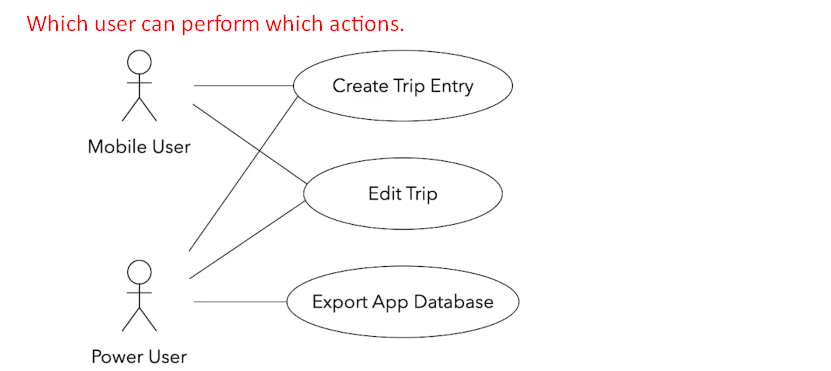
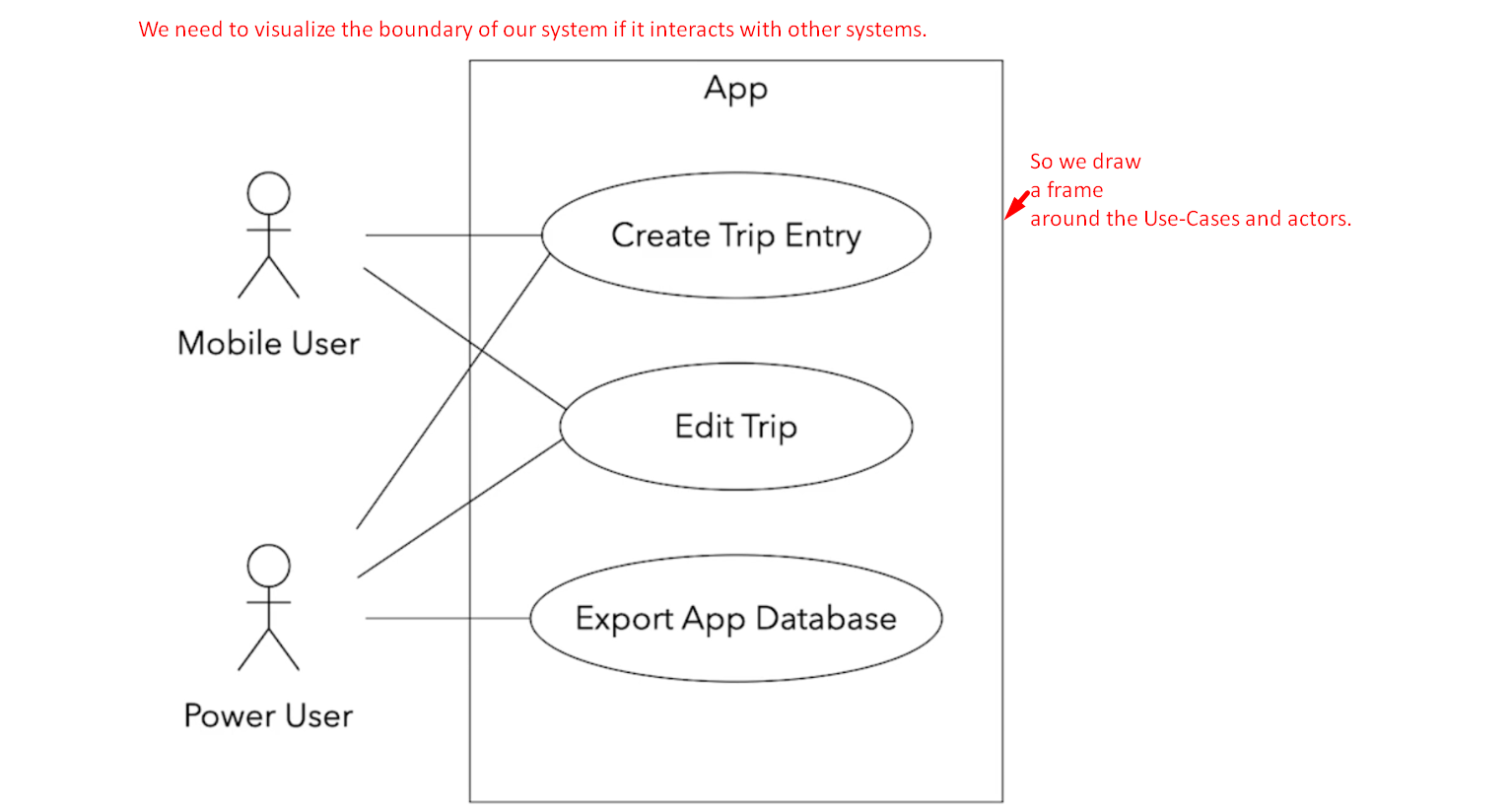
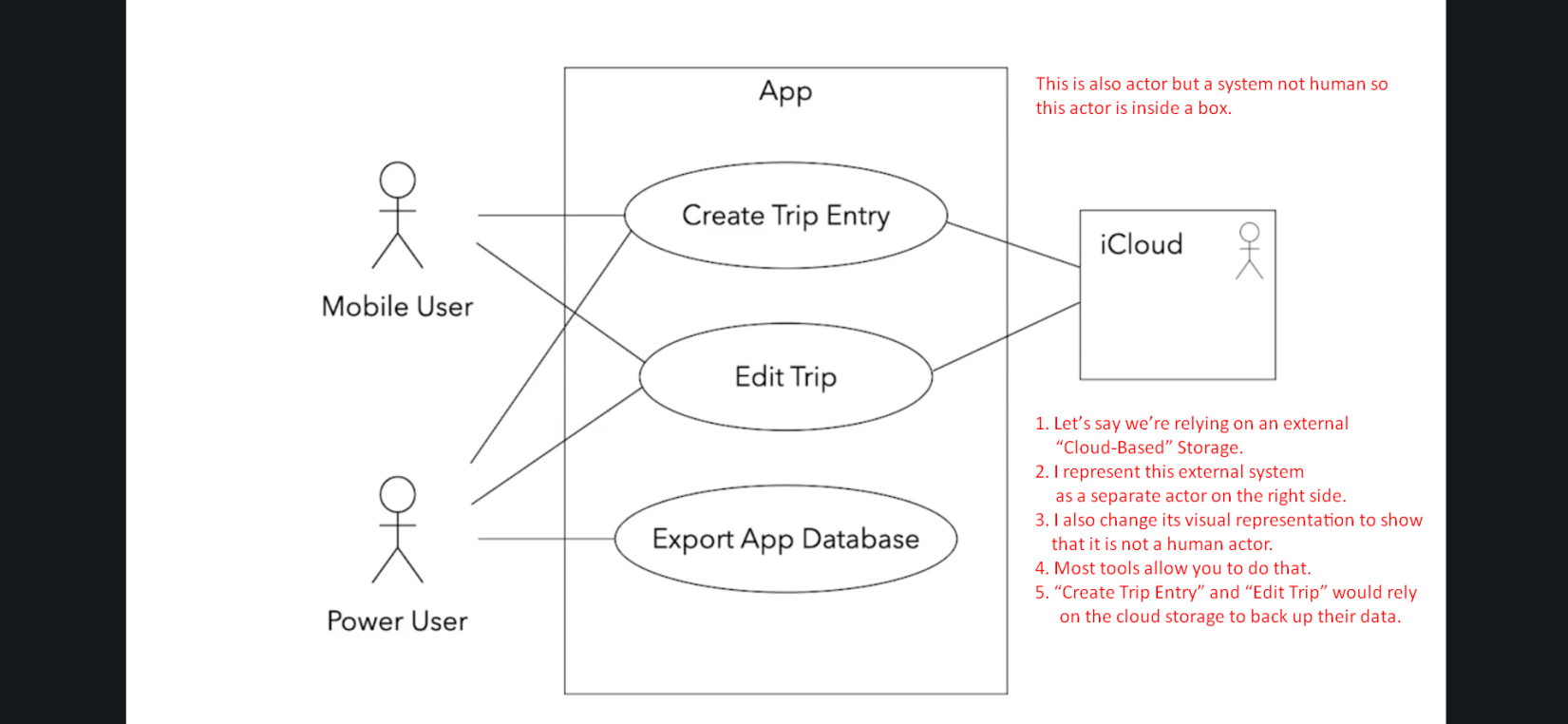
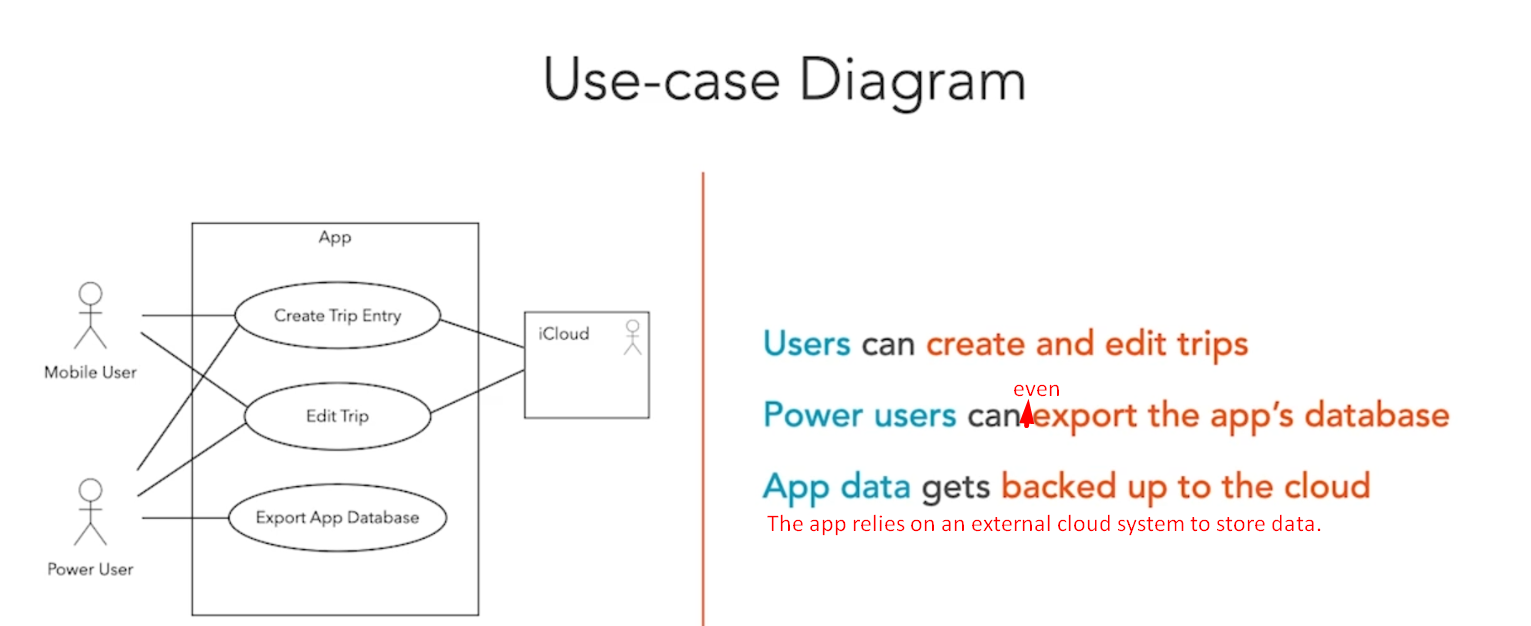
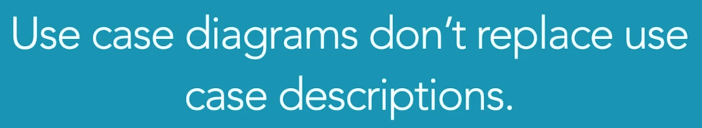
1. **Use Case Diagram**:
   1. One of the **simplest UML Diagrams**.
   2. Its purpose is to **visualize** the **functional requirements** of a system.
   3. “**Use Case Diagram**” shows groups of **related use cases**. 🡺 Use Case Diagram = Related Use Cases 🡺 Overall System.
      1. Sometimes they may include other use cases.
   4. The result is an overview of the system that may include several written use cases.
   5. We rarely create Use-Case Diagram for single Use Case Description.
2. To present a use case
   1. We draw an oval in the middle of the screen and put the title the use case in it.
   2.   
      These are the Use Cases from our Travel Expense App mentioned before.
3. **Actor**:
   1. **Stick figure** to represent an actor.
   2. Actor can be one of the following interacting with our system.
      1. A user or
      2. Other system.
   3. We draw an actor to the right or left of the diagram.
   4. Actor name goes below the stick figure.
   5. Primary Actor on the left of the Use-Case and Secondary Actor on the right.
   6. 
4. **Line**:
   1. To represent an **interaction** by actor for a particular use-case.  
      Interaction b/w actor and a use-case
   2. 
5.   
   A mobile user can create a trip but can’t export the app’s database.  
   The power use can perform all these actions.
6. **Boundary**: When a system interacts with other system.  
   
   1. Let’s say we’re relying on an external “**Cloud-Based**” Storage.
   2. I represent this external system as a separate actor on the right side.
      1. I also change its visual representation to show that it is not a human actor.
      2. Most tools allow you to do that.
   3. The use cases “**Create Trip Entry**” and “**Edit Trip**” would rely on the cloud storage to back up their data.

  
The frame makes it obvious where our app boundaries end.

1. Use case diagrams provide a clear way to communicate the high-level features and the scope of the system.
2. You can quickly tell what our system does just by looking at this Use Case Diagrams.
3.   
   Such a simple diagram makes it clear what the system does and doesn’t do.   
   A customer or a user can easily see if needed features are missing.  
   The absence of use cases shows what the system doesn’t do.
4. The UML Use Case Diagram includes other artifacts and relationships b/w use cases.  
   We’re going to ignore them as they tend to overcomplicate our design and the benefits are questionable.
5. You can’t go wrong if you focus on the actors, the use cases, and their interactions.
6. You will be able to easily create your own Use-Case Diagrams and communicate your ideas in a clear and concise way.
7. Use case diagrams provide an easy-to-understand overview of the features of our system.
8.   
   Keep in mind that Use-Case diagrams are not a replacement for written Use-Case Descriptions.   
   Use-Case Descriptions include more information to ensure that we don’t miss any important details or requirements.