**Step IV: Design Engineering**

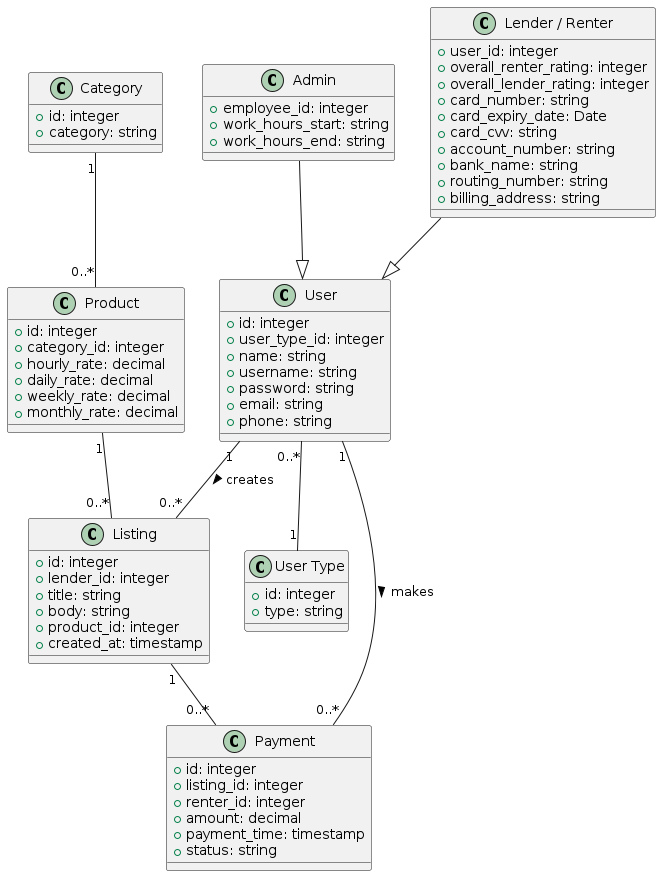
**1) Class Identification and Relationship Analysis:**

**Class descriptions:**

* + UserType: A class to define user types. Has a single attribute of enum type called “type.” This is an aggregation relationship as many users fall under a certain type for proper role identification on the platform.
    - Methods: assignUserType(), listUsersByType(),
  + User: Someone who signs up to our platform looking to perform activities on our platform. Has attributes of name, username, password, email address, and phone number.
    - Methods: hasType(), createProfile(), updateProfile(), viewProfile()
  + Renter / Lender: A renter or lender inherits User and is a type of user who is the bulk of users on our platform. They store additional information such as payment info and renter / lender overall rating.
    - Methods: sendReceipt(), sendConfirmation(), createListing() [calls on method in ListingController], viewListingsForUser(),
  + Admin : An admin inherits User is a privileged type of user who can manage the above category of users. Has additional fields such as employee ID, working hours, and other employment information that is not public but is kept for employee purposes.
    - Methods: updateListingStatus(), pauseAccount(), unpauseAccount(), forceCreateUser() [calls on createProfile in UserController with some override params]
  + Listing : A class representing a platform listing. Has attributes of title, body, user\_id, product\_id, status, and created\_at time. It is a higher level entity containing references to both the user that created it and the product contained under it.
    - Methods: createListing(), setStatus(), updateListing(), deleteListing(), viewListing(), viewListings(), createPayment() [calls on method in PaymentController], viewProduct() [calls on viewProduct in ProductController], viewCreatorUser() [calls on viewProfile of UserController].
  + Product : A Listing must have a Product (Composition relationship). As there is nothing to list for rent otherwise. Product has attributes of category\_id, hourly\_rate, daily\_rate, weekly\_rate, and monthly\_rate.
    - Methods: createProduct(), setRates(), viewProduct(), assignCategory(),
  + Category: A Product falls under a Category (Aggregation relationship) but can exist independently. Categories are used to organise rental items for easier searching for users and to assist the platform in getting metrics based on category. Its only field is the “category” field is an enum type. Products will reference the primary key of the appropriate category record.
    - Methods: listProducts()
  + Payment: A payment is the entity that stores the record of a transaction that happened relating to the rental. It refers to a listing (Association relationship) and also a user (the renter’s id). This way it keeps track of both the user creating the listing (the lender, through the user\_id field of the listing) and the renter themself. All fields under the class are: listing\_id, amount, payment\_time, renter\_id. Payment class is used for two types of payments from renter to lender: security deposit and rental payment.
    - Methods: createPayment(), processPayment(), refundPayment(), getSenderBankInfo(), getReceiverBankInfo(),

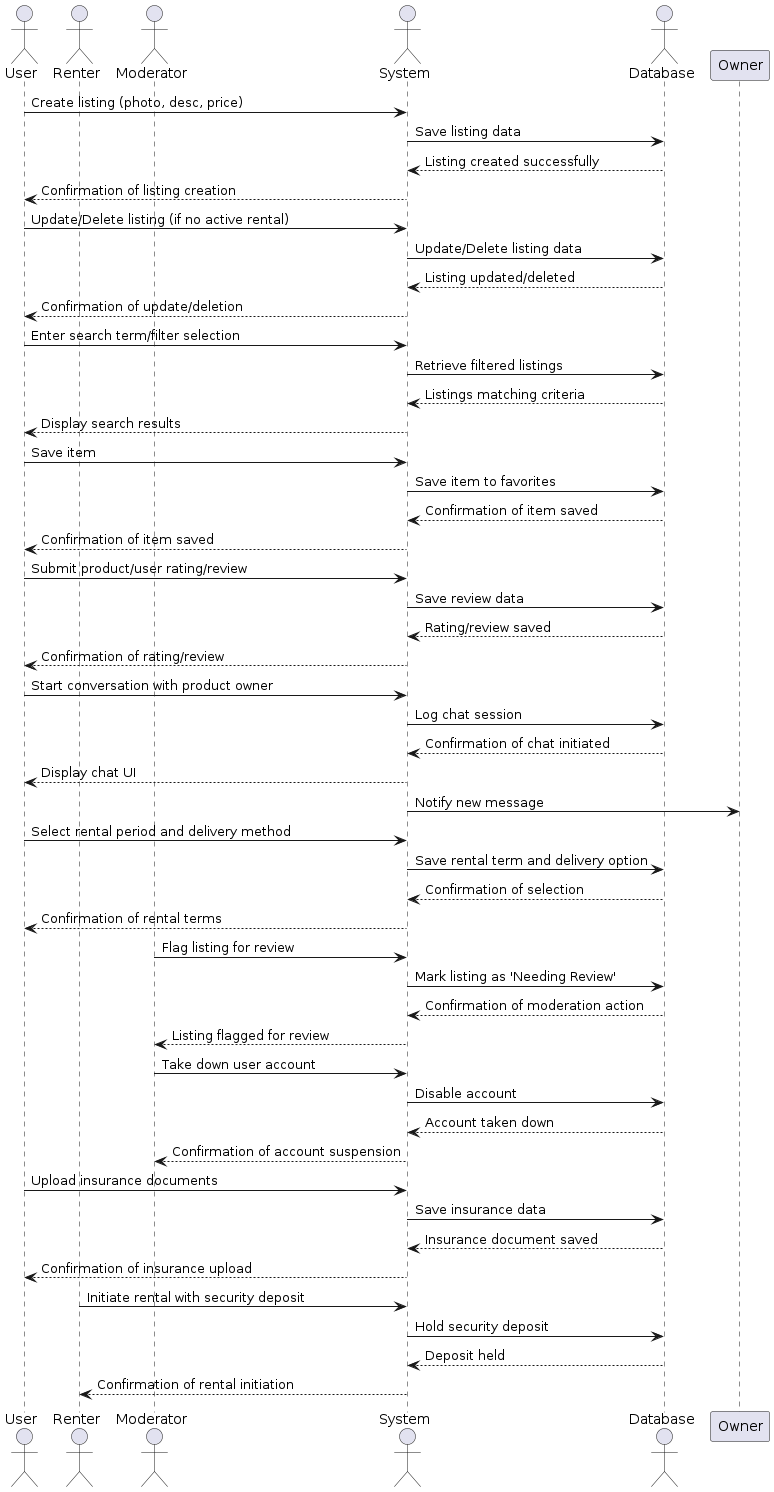
**2) UML Design Class Diagram:**

Create a **UML Design Class Diagram** that depicts the structure and relationships of the classes identified in Task 1.

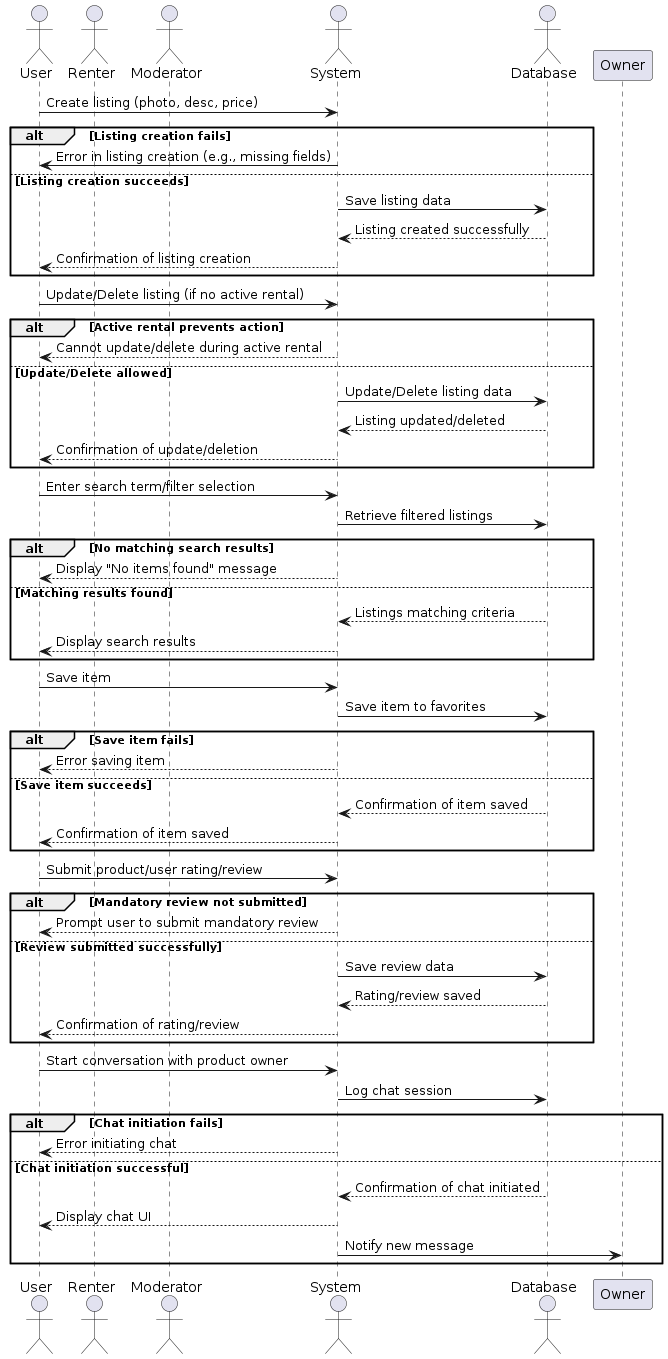


**3) UML Sequence Diagram STEP1**

**UML Sequence Diagram** with the **SUCCESSFUL FLOW**:

****

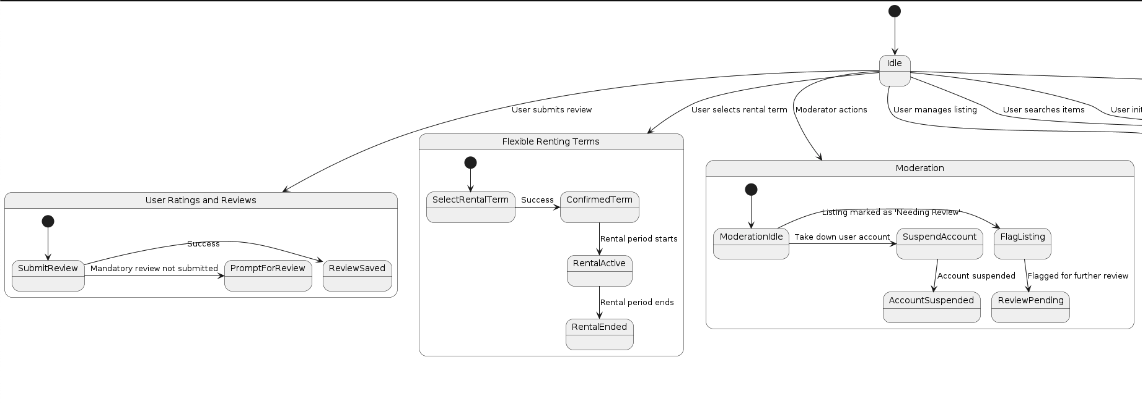
**4) UML Sequence Diagram STEP 2 with Alternative Flows**



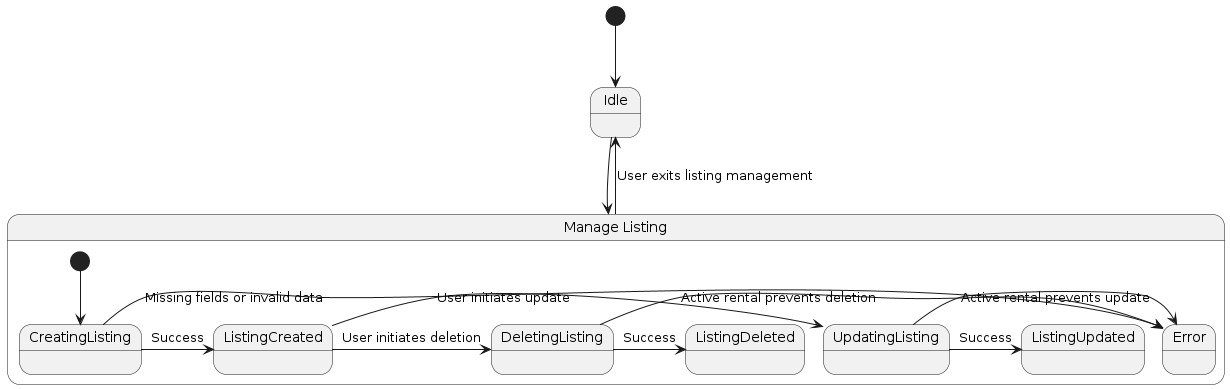
**5) UML State Diagram**

Create a **UML State Diagram** for the same use cases in Task 3, visualizing all the the successful and alternative flows of your systems

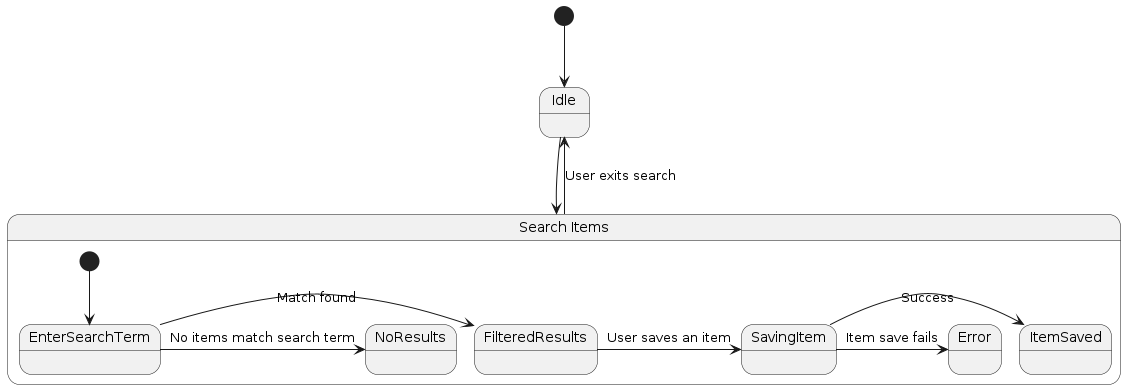
**State Diagram for User Ratings / Reviews, Renting Terms, Moderation:**

* 

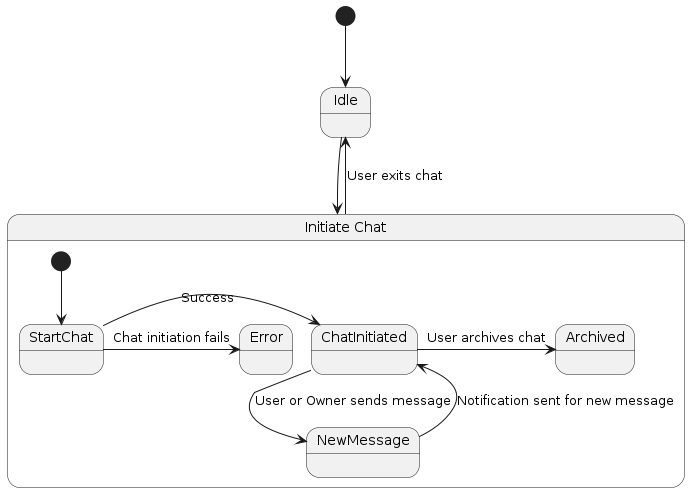
**State Diagram for User Manages Listing:**

****

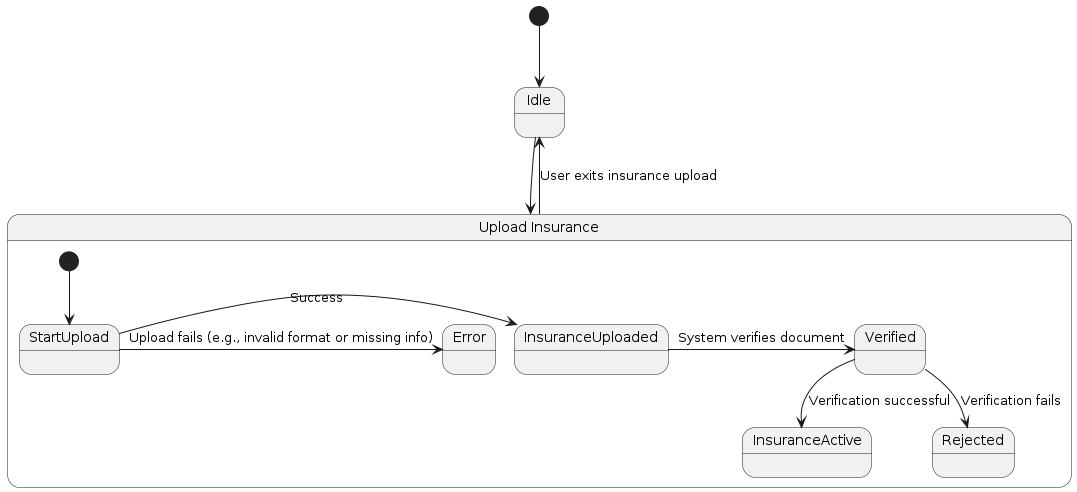
**State Diagram for User Searches Items:**

****

**State Diagram for User Initiates Chat:**

****

**State Diagram for User Uploads insurance:**

****