Task #5 (3 points*): Explore and document Generative AI capabilities in modelling class model

For the task of generating the class diagram, we selected Gemini as our **Generative AI** tool for the specific purpose of generating the class diagram code. This process of generating and improving the model is more dynamic and efficient than using a traditional GUI tool, as the chat-based interface allows us to quickly refine the diagram using simple commands. Furthermore, as Gemini is developed by Google, it has the advantage of being trained on a vast and diverse dataset of text and code, including UML diagrams and concepts. This helps in understanding the complex requirements and ensures a high degree of accuracy in the generated output.

The prompts used for generating the class diagram are given below.

Prompts Used :

1. ***Prompt -*** *Our Team is designing a Class diagram for a system for the Systems Modelling project. The description of this system is given below. Generate a Class diagram based on this description. Do not include any other source of data, just the description provided to you -* ***(Description from Task #1)***

Gemini - *Answered with the code.*

2. ***Prompt -*** *How do I make use of this code? I need to generate the diagram from this.*

Gemini - *Answered with the correct instructions.*

3. ***Prompt -*** *Done, but some of the text is overlapping. e.g. "submits" link is on top of another text for another link. Can you fix that?*

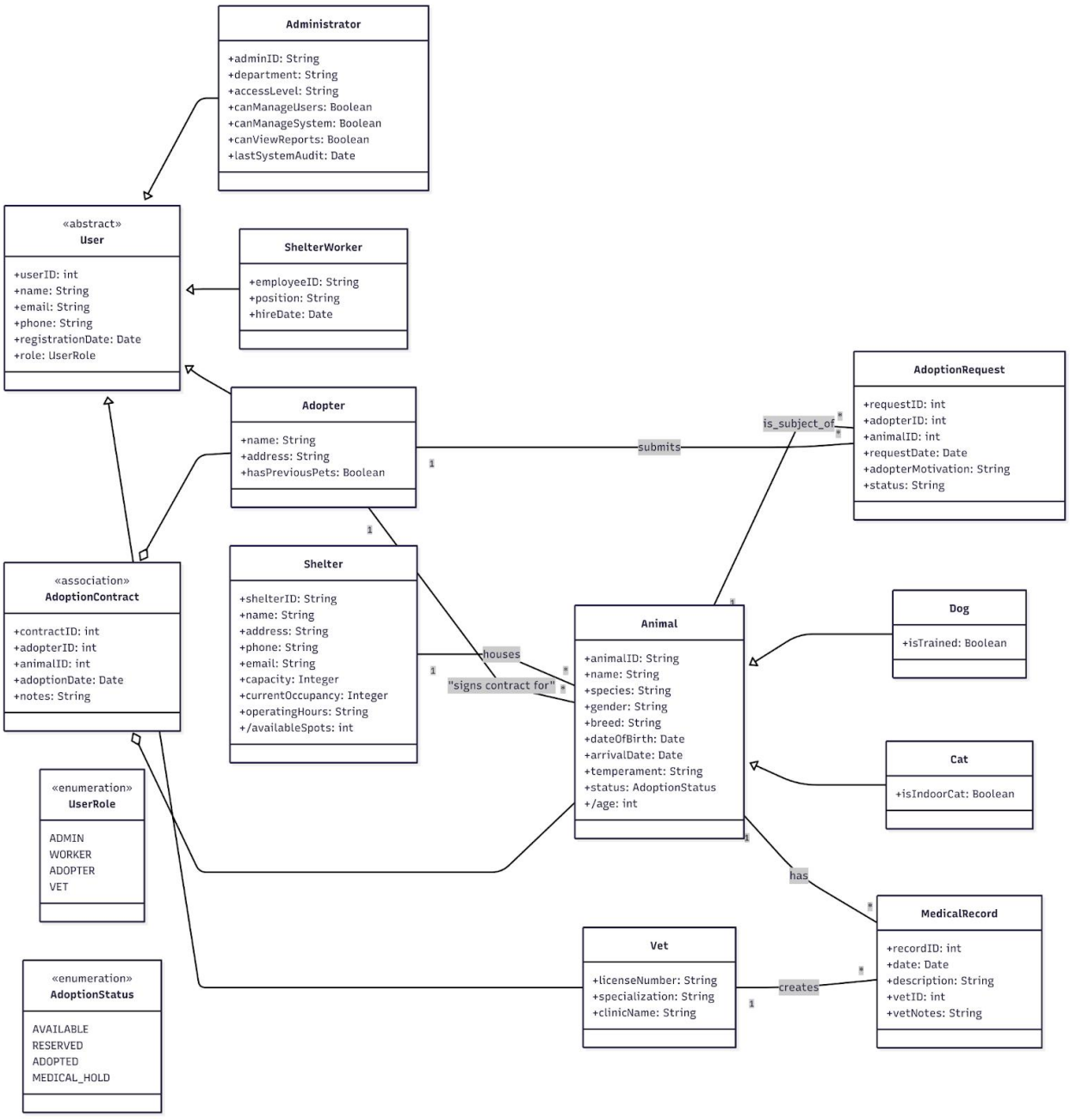
Gemini *- Answered with correct fix.*

4. ***Prompt -*** *Match all attributes and features of the classes against every single requirement given in the class description and list out any possible outliers or missing data.*

Gemini - *Answered with missing details.*

5. ***Prompt -*** *Apply all these fixes, add the missing data, and regenerate the code. Once regenerated, go through the requirements again and match against the final output and list out all discrepancies.*

Gemini - *Answered with the following diagram without any errors.*



Compare the model produced by Generative AI tools with the one designed by you - assess both critically. Was it possible to achieve the same result that you obtained in Task #2? If not, what were the differences? What are strengths and weaknesses of GenAI tools in addressing this task?

Overall, the Generative AI diagram looks technically superior and more aligned with formal UML standards, while the manual model is far easier to read and understand visually.

For example - The GenAI model correctly applied formal UML notations from our course materials. For instance, it used **<<abstract>>** for the User class to enforce that only specific roles can be created, and modeled the **AdoptionContract** as an **<<association>>** class to show it's dependent on the adopter-animal relationship.

It also correctly identifies both UserRole and AdoptionStatus as attributes with a fixed set of values and models both as separate **<<enumeration>>** classes.

On the other hand, the diagram made by us is significantly cleaner. We were able to arrange classes properly to minimize crossing lines, making it much easier to read. The GenAI's auto-layout was messy, even after multiple tries, which undermines a diagram's primary purpose as a clear communication tool.

GenAI Strengths vs. Weaknesses

The GenAI's main strengths were its speed in creating a baseline. It suggested and implemented certain improvements as well.

However, its primary weakness was the poor automatic layout. This created a cluttered diagram that required manual cleanup. So, while GenAI is certainly a useful tool for the baseline diagram, it can not be completely relied upon.

Had the use of GenAI tools allowed you to improve either the description (Task#1) or the model (Task#2)?

Yes, the use of the GenAI tool significantly improved our class diagram model.

The output of the first prompt was a direct translation of our initial description, but the interactive process allowed us to quickly improve its quality. The tool suggested and implemented several key improvements we had not yet modeled, such as:

- Remodeling the **AdoptionContract** as a more precise **association class.**
- Changing the age attribute to derived data for better data integrity.
- Identifying enumeration classes.

<<< BONUS ASSIGNMENT >>>

Generating the diagram with another LLM - Claude by Anthropic.

1. ***Prompt -*** *Our Team is designing a Class diagram for a system. The description of this system is given below. Generate a Class diagram based on this description*
   - **(*Description from Task #1)***

Claude *- Answer with code and how to turn it into a diagram.*

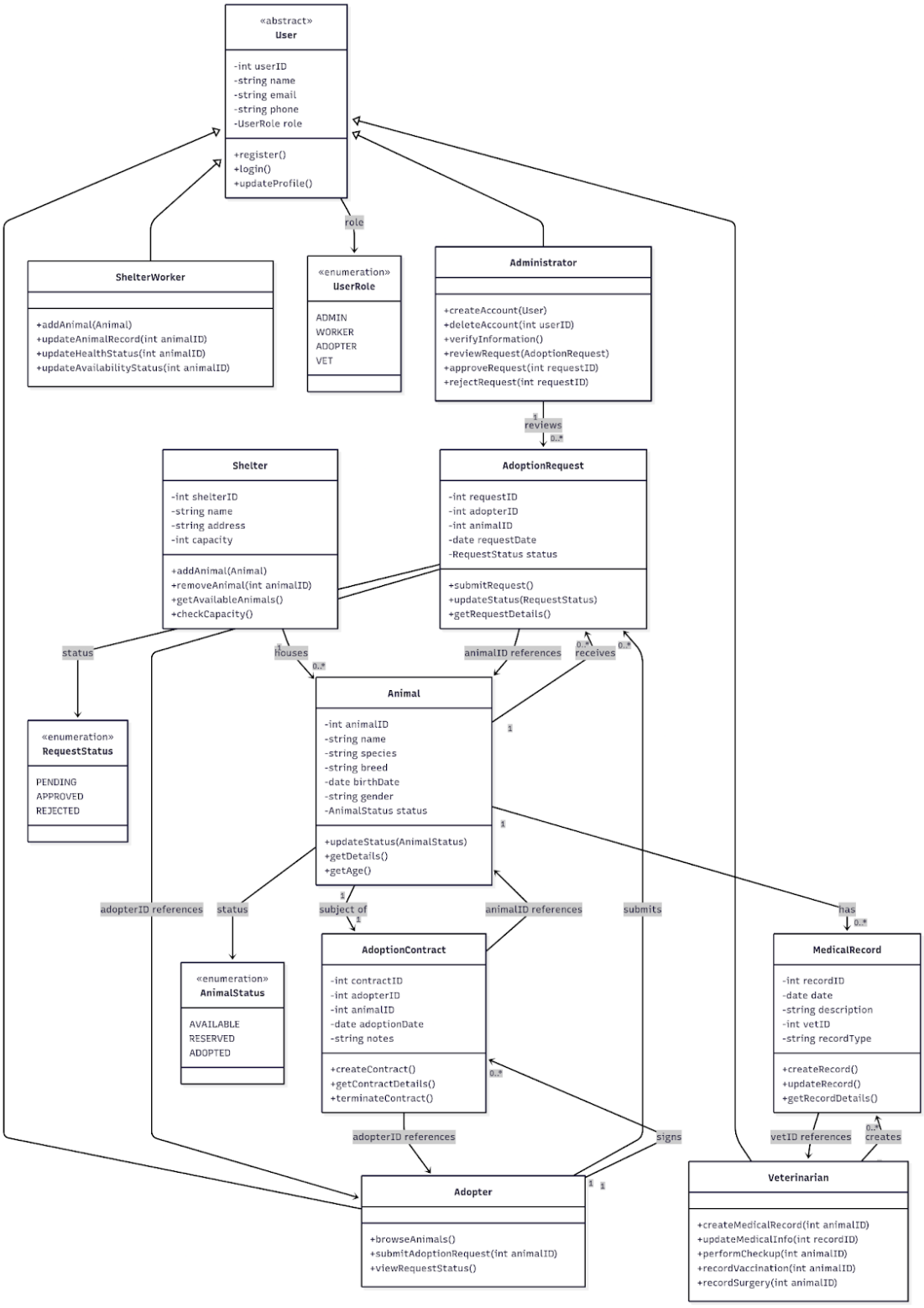
2. ***Prompt -*** *okay, is there any way you can improve this diagram based on my original description of the project?*

Claude *- Answer with the code.*

3. ***Prompt -*** *Match all attributes and features of the classes against every single requirement given in the class description and list out any possible outliers or missing data.*

Claude - *Answered with some observations but no major change.*

Final Diagram generated by Claude -


«abstract»
User
-int userID
-string name
-string email
-string phone
-UserRole role
+register()
+login()
+updateProfile()
role
ShelterWorker
+addAnimal(Animal)
+updateAnimalRecord(int animalID)
+updateHealthStatus(int animalID)
+updateAvailabilityStatus(int animalID)
«enumeration»
UserRole
ADMIN
WORKER
ADOPTER
VET
Administrator
+createAccount(User)
+deleteAccount(int userID)
+verifyInformation()
+reviewRequest(AdoptionRequest)
+approveRequest(int requestID)
+rejectRequest(int requestID)
1
reviews
0..*
Shelter
-int shelterID
-string name
-string address
-int capacity
+addAnimal(Animal)
+removeAnimal(int animalID)
+getAvailableAnimals()
+checkCapacity()
AdoptionRequest
-int requestID
-int adopterID
-int animalID
-date requestDate
-RequestStatus status
+submitRequest()
+updateStatus(RequestStatus)
+getRequestDetails()
status
1
houses
0..*
animalID references
0..*
receives
0..*
«enumeration»
RequestStatus
PENDING
APPROVED
REJECTED
Animal
-int animalID
-string name
-string species
-string breed
-date birthDate
-string gender
-AnimalStatus status
+updateStatus(AnimalStatus)
+getDetails()
+getAge()
1
1
1
adopterID references
status
1
subject of
1
animalID references
submits
has
0..*
«enumeration»
AnimalStatus
AVAILABLE
RESERVED
ADOPTED
AdoptionContract
-int contractID
-int adopterID
-int animalID
-date adoptionDate
-string notes
+createContract()
+getContractDetails()
+terminateContract()
0..*
MedicalRecord
-int recordID
-date date
-string description
-int vetID
-string recordType
+createRecord()
+updateRecord()
+getRecordDetails()
adopterID references
signs
vetID references
0..*
creates
Adopter
+browseAnimals()
+submitAdoptionRequest(int animalID)
+viewRequestStatus()
1
1
Veterinarian
+createMedicalRecord(int animalID)
+updateMedicalInfo(int recordID)
+performCheckup(int animalID)
+recordVaccination(int animalID)
+recordSurgery(int animalID)