**Petclinic Sample Application Requirements**

The users of the application are employees of the clinic who in the course of their work need to view and manage information regarding the veterinarians, the clients, and their pets.

1. The sample application supports the following Use Cases :
2. View a list of veterinarians and their specialties
3. View information pertaining to a pet owner
4. Update the information of a pet owner
5. Add a new pet owner to the system
6. View information pertaining to a pet
7. Update the information of a pet
8. Add a new pet to the system
9. View information pertaining to a pet's visitation history
10. Add information pertaining to a visit to the pet's visitation history
11. The following is an overview of the database schema used in Petclinic.
12. TABLE: owners

PRIMARY KEY id

1. TABLE: types

PRIMARY KEY id

1. TABLE: pets

PRIMARY KEY id

FOREIGN KEY type\_id references the types table id field

FOREIGN KEY owner\_id references the owners table id field

1. TABLE: vets

PRIMARY KEY id, int

1. TABLE: specialties

PRIMARY KEY id, int

1. TABLE: vet\_specialties - a link table for vets and their specialties

FOREIGN KEY vet\_id references the vets table id field

FOREIGN KEY specialty\_id references the specialties table id field

1. TABLE: visits

PRIMARY KEY id

FOREIGN KEY pet\_id references the pets table id field

1. Logging Manager: - This is responsible to log all the CRUD operations. Please use AOP to enable logging in the different methods.
2. Validation Manager: - This is responsible to validate the veterinarians, the clients, and their pets.
3. Veterinarian Pool : Holds the pool of veterinarians whose appointment has not yet been booked.
4. Use concurrency to book a veterinarian for some fixed time(say 5000 ms) whenever an appointment is made. Use the concurrency concepts like CountDownLatch, Threadpool, Executor etc. to implement this feature.
5. We need to make sure that not more than 5 clients should be assigned to a veterinarian at same time. A BlockingQueue should be used to implement this feature. If 6th client is trying to book then that client should wait until one of the existing client is serviced by the veterinarian.
6. Use JUnit to write test cases for Order Service and Validation Manager.
7. Please use AOP to find out timing for each function.
8. Do proper exception handling whenever required and use custom exceptions.
9. Use Hibernate to interact with DB. Use HQL to query the DB.
10. Demonstrate various hibernate mappings-one to one, one to many etc. between Order & Client DAO. Use lazy fetching and foreign key in these mappings.
11. Use transactions for DB interactions.
12. Demonstrate second level cache to store frequently accessed clients.
13. Use advanced Hibernate concepts like Filter, Formula, Any etc. in the DAO classes.