Tests de charge

Table des matières

Introduction

Présentation de JMeter et des tests de performance

L'outil **Apache JMeter** a été retenu pour mener à bien les tests de performance du système MedHead. JMeter est une solution open-source de référence permettant de simuler des groupes d'utilisateurs virtuels envoyant des requêtes simultanées vers nos APIs (Spring Boot / MariaDB).

Le but de ces tests est d'identifier les goulots d'étranglement (bottlenecks) et de s'assurer que l'application répond aux exigences de disponibilité et de rapidité essentielles à un système de gestion d'urgence. Nous mesurons principalement :

- **La Latence (Temps de réponse) :** Temps mis pour afficher les unités de soins les plus proches.
- **Le Débit (Throughput) :** Nombre de réservations de lits que le système peut traiter par seconde.
- **La Robustesse :** Capacité de la base de données à gérer les accès concurrents (verrous/locks) sans erreur.

Objectifs des tests

Le but est de tester le chemin critique d'un utilisateur dans l'application. Le chemin nominal d'un utilisateur est :

- **Connexion au site (Auth) :** Vérifier la robustesse du gestionnaire de sessions.
- **Sélection d'une spécialité (Read/Search) :** Tester la performance de la base de données et du cache.
- **Recherche d'unité de soins / trajet (Read/Search) :** Tester la performance de la base de données et du cache.
- **Réservation d'un lit (Write) :** Tester la gestion des verrous (locks) en base de données lors de pics d'appels.

Détail des scénarios JMeter créés

creation_utilisateurs.jmx

Création d'utilisateur avec le fichier « data_set_utilisateur.csv », qui appel l'Endpoint « /user/creation », cet appel a pour but de créer les utilisateurs avec un mdp haché pour alimenter le data_set des autres tests.

test_flux.jmx

Réplique du test de charge (processus nominal), en ne mettant pas un grand nombre d'utilisateur, le but est de paramétrer le test de charge sans faire trop d'appels.

test_de_charge.jmx

Teste nominal avec 1000 utilisateurs, c'est le test à exécuter pour générer les rapports.

Procédure de test

Après avoir installé Jmeter.

1°/ Démarrer le projet

Voir Initialisation du projet.

2°/ Importer les données

Se connecter à la bdd

```
http://localhost:8081/?server=db&username=root&db=poc_db&select=specialisation
```

Aller sur **Requête SQL**, puis exécuter les requetés du « data_set.sql »

3°/ Se placer dans le répertoire de l'exécutable

```
cd "C:\Program Files\apache-jmeter-5.6.3\bin"
```

4°/Lancer Jmeter

(Interface graphique)

Pour le paramétrage

```
.\jmeter
```

(Tests de charge)

```
.\jmeter -n -t "G:\Mon Drive\Formation\Open Classroom\Dossier Open Classroom\P11\jmeter\data_set\test_de_charge.jmx" -l resultats.jtl -e -o dossier_rapport
```

Les informations sont accessibles sur

```
~/dossier_rapport/index.html
```

5°/ Refaire le test

Pour refaire le test de charge supprimer :

```
~/dossier_rapport

resultats.jtl
```

Refaire « 2°/ Importer les données ».

Analyse des données :

Résultat

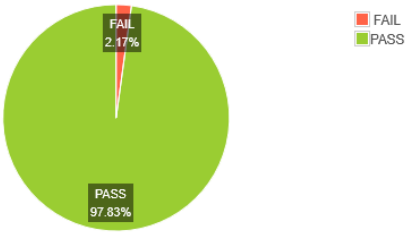
Test and Report information

Source file	"resultats.jtl"
Start Time	"12/30/25, 11:26 AM"
End Time	"12/30/25, 11:30 AM"
Filter for display	""

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.062	**500 ms**	**1 sec 500 ms**	**Total**
0.030	500 ms	1 sec 500 ms	/api/unitesoins/trajets
0.040	500 ms	1 sec 500 ms	api/reservation/lit
0.086	500 ms	1 sec 500 ms	/api/user/connection
0.092	500 ms	1 sec 500 ms	/api/specilites

Requests Summary



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	46133	1000	2.17%	4453.81	2	8386	4420.00	5814.00	5994.95	6371.00	166.93	295.41	78.60
/api/specilites	11715	0	0.00%	4004.32	2	6996	4763.00	5642.00	5767.20	5991.00	42.54	81.25	21.50
/api/unitesoins/trajets	11715	727	6.21%	4893.42	6	8386	5692.00	6596.40	6729.20	6990.52	42.57	172.13	24.24
/api/user/connection	11715	0	0.00%	4066.63	58	6741	4816.00	5705.00	5840.00	6065.00	42.39	32.19	11.04
api/reservation/lit	10988	273	2.48%	4877.17	9	7797	5604.00	6463.10	6590.00	6839.22	40.07	10.93	22.17

Errors

Type of error	Number of errors	% in errors	% in all samples
404	727	72.70%	1.58%
409	267	26.70%	0.58%
500	6	0.60%	0.01%

Top 5 Errors by sampler

Sample	#Samples	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors	Error	#Errors
Total	46133	1000	404	727	409	267	500	6				
/api/unitesoins/trajets	11715	727	404	727								
api/reservation/lit	10988	273	409	267	500	6						

Résumé global de l'éxécution

Le test a été réalisé sur un volume total de 46 133 échantillons (samples).

- **Taux de succès :** 97,83 % (45 133 requêtes réussies).
- **Taux d'échec :** 2,17 % (1 000 requêtes en échec).
- **Débit (Throughput) :** Le système a traité en moyenne **166,93 transactions par seconde**.

Expérience Utilisateur et Performance (APDEX)

L'indice APDEX (Application Performance Index) global est extrêmement faible, s'élevant à 0,062 sur une échelle de 0 à 1.

- **Seuils définis** : Le seuil de satisfaction (T) était fixé à 500 ms et le seuil de frustration (F) à 1,5 s.
- **Constat :** Avec un temps de réponse moyen global de 4 453,81 ms (soit environ 4,5 secondes), la quasi-totalité des utilisateurs se situe dans la zone de frustration.
- **Percentile 95** : 95 % des requêtes ont un temps de réponse inférieur ou égal à 5 994,95 ms , ce qui confirme une lenteur généralisée.

Analyse par Endpoints

Les performances varient selon les services sollicités :

- **/api/specilites & /api/user/connection :** Ces deux services présentent un taux d'erreur de **0,00 %**. Cependant, leurs temps de réponse moyens (respectivement 4 004 ms et 4 066 ms) restent très élevés.

- **/api/unitesoins/trajets :** C'est le point le plus critique avec un taux d'erreur de **6,21 %** et le temps de réponse moyen le plus long (**4 893,42 ms**).

- **api/reservation/lit :** Ce service enregistre **2,48 % d'erreurs** pour un temps moyen de **4 877,17 ms**.

Analyse des erreurs

Sur les 1 000 erreurs enregistrées, trois types prédominent :

- **Erreur 404 (Not Found) :** Elle représente **72,70 % des erreurs** (727 cas). Ces erreurs sont concentrées sur l'endpoint /api/unitesoins/trajets, comme les données ont été créées aléatoirement, il y a surement des cas KO

- **Erreur 409 (Conflict) :** Elle représente **26,70 % des erreurs** (267 cas). Ces erreurs apparaissent exclusivement lors de la réservation de lit (api/reservation/lit), suggérant des problèmes de gestion des accès concurrents (verrous/locks) lors des pics d'appels. Ceci est lié au fait qu'il ne reste qu'un lit et au moins 2 tentatives de réservations.

- **Erreur 500 (Internal Server Error) :** Très marginale, elle ne représente que **0,60 %** des erreurs (6 cas) sur le service de réservation.

Conclusion :

Pour conclure l'analyse de ce PoC et tracer une feuille de route pour l'évolution de **MedHead**, il est essentiel de remettre les résultats obtenus en perspective avec la réalité du lancement du service. En effet avoir 1000 utilisateurs en simultané n'est pas une situation probable au vu de l'utilité de l'application. La gestion de la montée en charge devra se faire au fil de l'eau en fonction de la demande. On peut cependant déjà envisager des axes d'améliorations :

Scalabilité horizontale (Load Balancer)

Pour absorber les pics de trafic sans saturer un serveur unique, l'ajout d'un Load Balancer est enviable. Cela permettrait de distribuer les 166,93 transactions par seconde sur plusieurs instances de l'API.

Optimisations techniques (Cache)

Stocker les résultats des recherches fréquentes (spécialités, unités de soins) pour éviter des accès répétitifs à la base de données (Avec par exemple Redis ou la mise en cache coté client des spécialités).

Optimisations techniques (Indexation)

Optimiser les requêtes sur les coordonnées géographiques et les identifiants pour accélérer les recherches de trajets.

On peut nettement créer des zones géographiques de recherche par utilisateurs (En effet une personne qui travaille dans la région parisienne n'aura jamais besoin de connaitre les disponibilisées des hôpitaux de Marseille)

Architecture microservices et isolation des données

En passant à une architecture microservices où chaque service possède sa propre base de données.

Soumission du reporting

La mise a jour de la table des réservations pourrait être traité par un broker pour ne pas attendre le retour de cette table.

Création de la donnée :

Utilisateurs / Compte

Des utilisateurs ont été créer en masse sous Excel pour simuler 1000 connections, puis le fichier a été enregistré au format csv (pour une utilisation dans Jmeter).

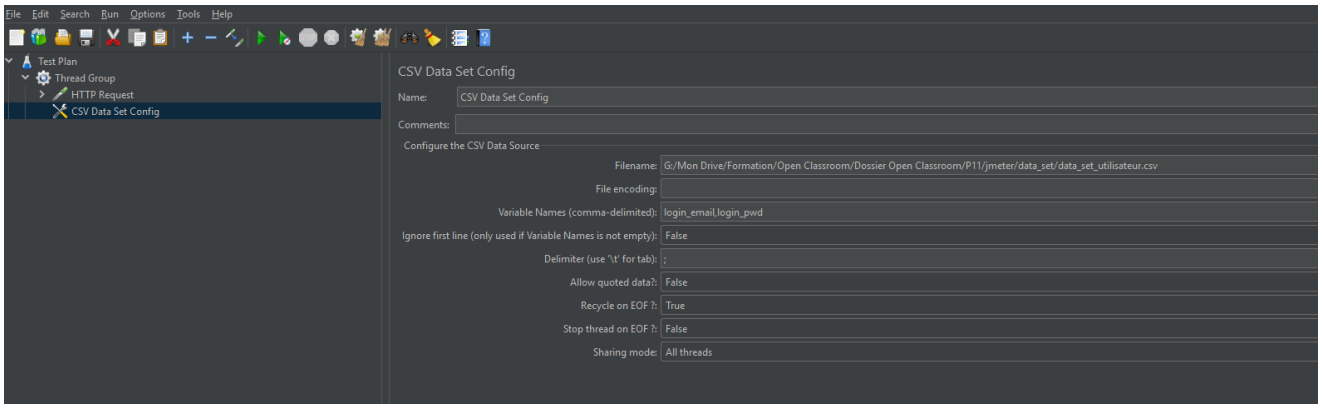
- **Formule** :

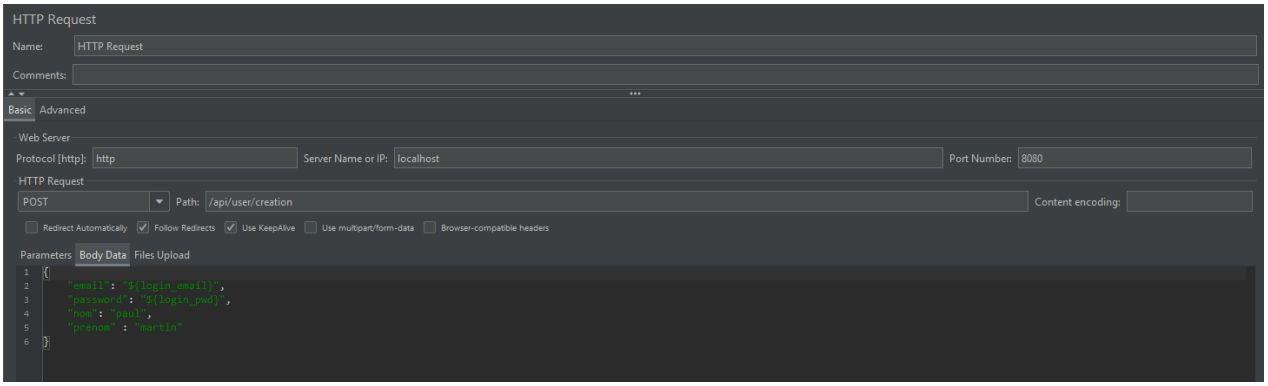
```
="utilisateur" & LIGNE() & "@compte.com"

="MotDePasseSecret&" & LIGNE()
```

- **Format des données** :

```
utilisateur1@compte.com;MotDePasseSecret&1

utilisateur2@compte.com;MotDePasseSecret&2

utilisateur3@compte.com;MotDePasseSecret&3

utilisateur4@compte.com;MotDePasseSecret&4

utilisateur5@compte.com;MotDePasseSecret&5
```

Ensuite l'appelle à l'API a été effectué pour créer les utilisateurs dans Jmeter (creation_utilisateurs.jmx). Les données une fois mise en base de données ont été extraite pour les mettre dans le data_set.sql.





Utilisateurs / Position aléatoire

Des positions aléatoires ont été utilisées pour simuler des cas fonctionnels (si on met toujours les mêmes positions, les spatialisations remontées sont toujours les mêmes, ce qui entraine une concurrence dans la réservation des lits qui n'est pas réaliste).

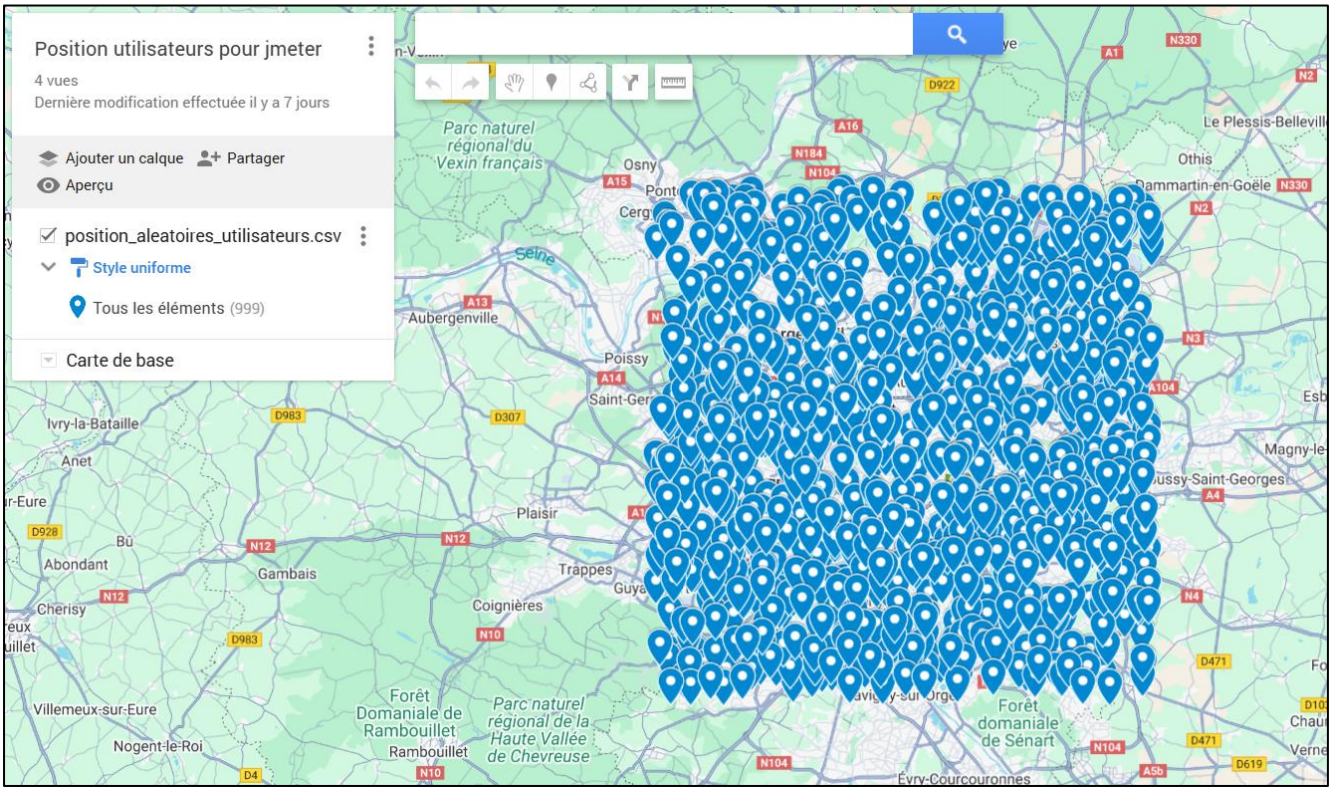
Formule :

```
=ARRONDI(48,8564826 + (ALEA()-0,5)*(40/111); 6)

=ARRONDI(2,3522219 + (ALEA()-0,5)*(40/(111*COS(RADIANS(48,8564826)))); 6)
```

Format des donées :

```
Latitude,Longitude

48.726195,2.577904

48.952718,2.259881
```

Affichage des données :



Spécialisation / Position

Les spécialisations ont été récupérées sur :

https://data.iledefrance.fr/explore/dataset/les_etablissements_hospitaliers_franciliens/download?format=csv

Ensuite, les 528 premières adresses, ont été concaténées avec des « lits_disponibles », « hopital_id » et « specialisation_id » aléatoires, pour créer le data_set.sql.

```
=ALEA.ENTRE.BORNES(10;50)
```

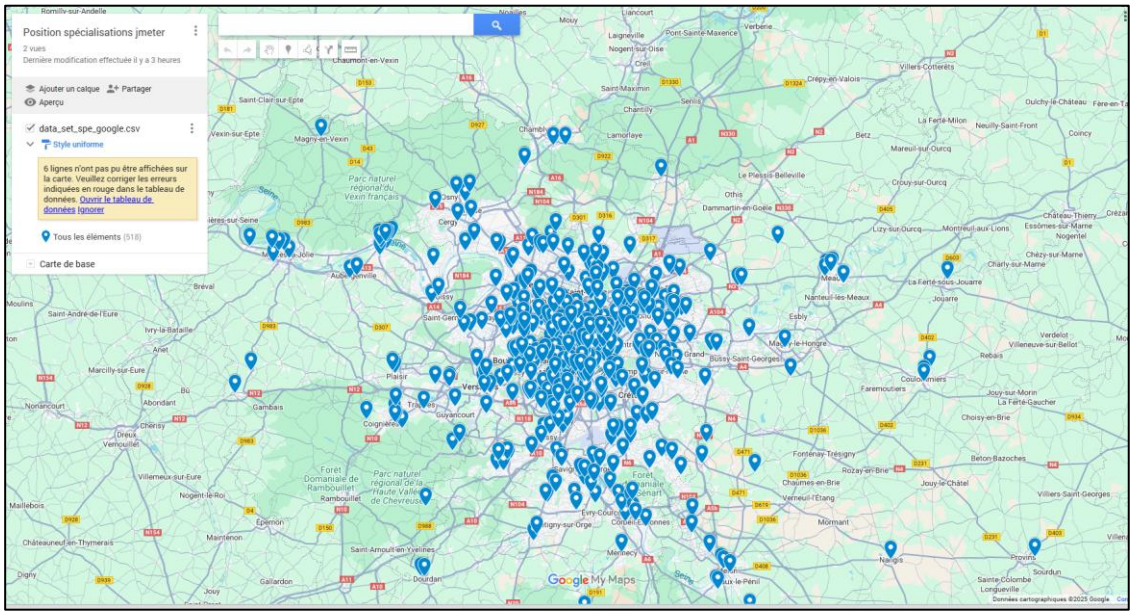
A	B	C	D	E	F	G
raison_socia	adresse_con	cp_ville	lat	lng		
HOPITAL DE J	10 R DE L'ECL	95800 CERGY	49.0512627	2.0304689		(1,"10 R DE L'ECLIPSE 95800 CERGY",49.0512627,2.030468917,35,3 , 9) ,
CH DE VERSA	1 R RICHAUD	78011 VERSA	48.8074016	2.1337584		(2,"1 R RICHAUD 78011 VERSAILLES CEDEX",48.8074016,2.13375843,31,17 , 13) ,
CMP 92I05 EN	259 AV ROGE	92370 CHAVI	48.8174288	2.1962218		(3,"259 AV ROGER SALENGRO 92370 CHAVILLE",48.8174288,2.196221825,10,20 , 15) ,
HOPITAL DE J	12 R ERNEST	92310 SEVRE	48.820483	2.2143718		(4,"12 R ERNEST RENAN 92310 SEVRES",48.820483,2.214371824,10,7 , 17) ,
CMP ADULTE	5 R RABELAIS	92600 ASNIEI	48.9131996	2.2906781		(5,"5 R RABELAIS 92600 ASNIERES SUR SEINE",48.9131996,2.290678124,32,20 , 19) ,
POLYCLINIQ	42 R P.VAILLA	92240 MALAK	48.8130707	2.2968778		(6,"42 R P.VAILLANT COUTURIER 92240 MALAKOFF",48.8130707,2.296877811,18,18 , 18) ,
APPARTEMEN	16 R DES MOI	91160 LONGJ	48.6899321	2.3015811		(7,"16 R DES MOUSSERONS 91160 LONGJUMEAU",48.6899321,2.301581110,26,18 , 19) ,
CENTRE PARI	167 R RAYMC	75014 PARIS	48.8303682	2.3112489		(8,"167 R RAYMOND LOSSERAND 75014 PARIS",48.8303682,2.311248920,26,11 , 7) ,
HOP DE JOUF	3 R RIDDER	75014 PARIS	48.8311258	2.3118698		(9,"3 R RIDDER 75014 PARIS",48.8311258,2.311869829,25,18 , 4) ,
CMP ADULTE	18 R SALNEU	75017 PARIS	48.8855503	2.3143885		(10,"18 R SALNEUVE 75017 PARIS",48.8855503,2.31438856,32,1 , 3) ,

Affichage des données :

data_set_spe_google.csv

Rechercher dans le table — 1–200 sur 518

	adresse_complete;cp_ville
7	10 R DE L'ECLIPSE;95800 CERGY
8	1 R RICHAUD;78011 VERSAILLES CEDEX
9	259 AV ROGER SALENGRO;92370 CHAVILLE
10	12 R ERNEST RENAN;92310 SEVRES
11	5 R RABELAIS;92600 ASNIERES SUR SEINE
12	42 R P.VAILLANT COUTURIER;92240 MALAKOFF



Autres données

Les autres données ont été mises dans le data_set.sql en suivant les spécifications.

```sql
INSERT INTO groupe_specialite (id, nom) VALUES
(1, 'Surgical Specialties'),
(2, 'Medical Specialties'),
(3, 'Mental Health'),
(4, 'Radiology & Pathology'),
(5, 'Obstetrics & Gynaecology');

-- Surgical Specialties (ID 1)
INSERT INTO specialisation (id, nom, groupe_specialite_id) VALUES
(1, 'GENERAL SURGERY', 1),
(2, 'UROLOGY', 1),
(3, 'TRAUMA & ORTHOPAEDICS', 1),
(4, 'ENT', 1),
(5, 'OPHTHALMOLOGY', 1),
(6, 'ORAL SURGERY', 1),
(7, 'NEUROSURGERY', 1),
(8, 'PLASTIC SURGERY', 1),
(9, 'CARDIOTHORACIC SURGERY', 1),
(10, 'PAEDIATRIC SURGERY', 1),
(11, 'ACCIDENT & EMERGENCY', 1);

-- Medical Specialties (ID 2)
INSERT INTO specialisation (id, nom, groupe_specialite_id) VALUES
(12, 'ANAESTHETICS', 2),
(13, 'GENERAL MEDICINE', 2),
(14, 'GASTROENTEROLOGY', 2),
(15, 'ENDOCRINOLOGY', 2),
(16, 'CLINICAL HAEMATOLOGY', 2),
(17, 'AUDIOLOGICAL MEDICINE', 2),
(18, 'CLINICAL GENETICS', 2),
(19, 'CARDIOLOGY', 2),
(20, 'DERMATOLOGY', 2),
(21, 'RESPIRATORY MEDICINE', 2),
(22, 'INFECTIOUS DISEASES', 2),
(23, 'NEPHROLOGY', 2),
(24, 'NEUROLOGY', 2),
(25, 'RHEUMATOLOGY', 2),
(26, 'PAEDIATRICS', 2),
(27, 'GERIATRIC MEDICINE', 2);
```