Тестовое задание на позицию Python developer.

Суть задачи - написать распределенное клиент-серверное приложение, представляющее собой простой чат для общения пользователей друг с другом.   
  
Клиентом будет являться веб-приложение, выполняющее две функции.

1. - Регистрация/авторизация нового посетителя чата.

2. - Прием и отправка сообщений другим посетителям.

Сервер - автономное приложение, которое принимает информацию от клиентов и выполняет следующее:

1. Регистрирует / авторизует нового посетителя чата в своей базе  
2. Принимает он него сообщение и транслирует всем остальным клиентам чата

Далее идут более подробные тех. описания клиента и сервера.

Клиент реализован как веб-приложение на Фласке (Flask) или Django. Реализация интерфейса произвольная. Как минимум должно быть 3 страницы: Логин, Регистрация, Чат. На странице Чат достаточно разместить контролы для просмотра и отправки сообщений.

При первом заходе , клиент должен отобразить простую веб-форму с предложением авторизоваться (никнейм, пароль). Либо зарегистрироваться (в этом случае необходимо ввести новый ник-нейм и пароль, нажать на кнопку "Регистрация"). В момент регистрации клиент передает на сервер информацию о новом посетителе. Сервер принимает запрос, и сохраняет эти данные в базе. В качестве базы использовать любую SQL подобную БД. Если в базе уже есть такой ник, то сервер отвечает ошибкой и клиент передает посетителю сообщение о том, что его никнейм не может быть использован для регистрации. Если посетитель был зарегистрирован ранее, клиент должен отправить авторизационные данные (ник и логин) на сервер, получить одобрение и сохранить в Http сессии на какое-то время (пусть это будет 30 мин). Т.е. если посетитель закроет страничку чата в браузере и потом откроет ее заново, клиент должен проверить наличие авторизационных данных в сессии, и если они есть, то не требовать повторной авторизации и сразу пустить посетителя в чат. Если ник/пароль не совпали, то сервер отвечает с ошибкой и клиент показывает сообщение "Вы не можете быть авторизованы. Попробуйте другой ник или пароль". В данной части задания требуется обратить внимание на передачу данных между клиентом и сервером. *Пароль не должен передаваться в открытом виде.*  
  
Как только посетитель авторизован, ему открывается список текущих сообщений (которые пишут другие посетители), поле для ввода своих сообщений и кнопка "Отправить". Сообщение о появлении нового пользователя в чате должно быть оттранслировано всем активным клиентам. Как только посетитель пишет сообщение и нажимает "Отправить", оно уходит на сервер, а тот транслирует это сообщение всем другим клиентам. При этом исходный посетитель должен видеть свое же сообщение отобразившееся в списке сообщений чата. Вместе с сообщением в этом списке выводится никнейм посетителя , отправившего сообщение.  
  
Если посетитель уходит (закрывает страничку), но потом возвращается, ему должны быть показаны все сообщения между остальными участниками чата, которые данный посетитель пропустил.  
  
Сервер представляет из себя HTTP Rest сервис предоставляющий соответствующие АПИ для регистрации, проверки логина и транслирования сообщений.   
  
И ВАЖНО! Сервер должен использовать очередь брокер сообщений (например RabbitMQ, Kafka) ([http://rabbitmq.com](http://rabbitmq.com/), https://kafka.apache.org/) для отправки сообщений всем клиентам. В случае использования RMQ , для каждого клиента должна быть создана очередь сообщений (QUEUE). Трансляция сообщения должна идти через EXCHANGE, к которому подключены все очереди. EXCHANGE должен дублировать сообщения во все очереди.

Вся инфраструктура (клиент, сервер, база данных, брокер сообщений) должны быть упакованы в Docker. Пояснительная записка к заданию должна содержать команды запуска всех требуемых контейнеров. Предлагается использование docker compose для поднятия ифнраструктуры.

Код должен быть отформатирован и снабжен необходимым разумным кол-вом комментариев.