Привет, кандидаты!

При решении задачи просим проявить все свои знания в области IT.
Мы ждем от вас решения, которые покажут ваши знания, опыт и понимание того, как решаются проблемы и создается хороший программный код. Можно прислать только часть решения, обязательно указав, что именно у вас получилось, а что нет.

Требуется спроектировать и реализовать сервис, позволяющий пользователям отправлять сообщения друг другу (аналог мессенджера).

Приложение необходимо разработать в виде **RESTful API** на языке **Java** с использованием фреймворка **Spring Boot**. Данные, получаемые и отправляемые приложением, должны быть в формате **JSON**.

Приложение должно реализовывать **упрощенный** функционал мессенджера.

Основные требования:

- Регистрация и авторизация:
  - Пользователь должен иметь возможность зарегистрироваться в системе. При регистрации необходимо указать email, пароль, никнейм, имя и фамилию. Обязательно нужно осуществлять проверку на уникальность логина и email - иначе не давать зарегистрироваться.
    Критерии:
    - *есть API для регистрации, сохраняющий пользователя в хранилище (**обязательно**);*
    - *есть хэширование паролей;*
    - *есть подтверждение почты через ссылку в письме, отправленном на указанную почту.*
  - Пользователь должен иметь возможность войти в систему. Система в свою очередь должна запомнить, что пользователь авторизован (хранение данных о состоянии сессии в хранилище, JWT токен и т.д.).
    Критерии:
    - *есть API, позволяющий залогиниться в системе и сохраняющее информацию о сессии любым способом (**обязательно**);*
    - *есть поддержка Spring Security;*
    - *информация о сессии хранится в JWT токенах и передается в HTTP хэдерах или куках.*

- Пользователь должен иметь возможность выйти из системы (разлогиниться), при этом его предыдущая сессия должна стать неактивной. *Критерии:*
  - *есть API, позволяющий завершить текущую сессию и разлогиниться (**обязательно**);*
  - *есть механизмы защиты от обхода разлогина (например, при использовании JWT токенов можно инвалидировать их).*

- Профиль пользователя:
  - Пользователь должен иметь возможность обновить данные своего профиля, такие как никнейм, имя, фамилию, email, (дополнительно можно добавить статус, аватар, био и так далее). *Критерии:*
    - *есть API, позволяющий изменять базовую информацию профиля (**обязательно**);*
    - *при изменении email есть подтверждение изменения ссылкой на указанный новый email.*
  - Пользователь должен иметь возможность обновить пароль своей учетной записи (отдельный запрос). *Критерии:*
    - *есть API, позволяющий обновить пароль (**обязательно**).*
  - Пользователь должен иметь возможно удалить свой аккаунт. *Критерии:*
    - *есть API, позволяющий удалить аккаунт пользователя (**обязательно**);*
    - *реализован перевод профиля в статус “Не активен” с дальнейшей возможностью восстановить профиль в течение некоторого времени.*

- Социальная часть:
  - Пользователь должен иметь возможность отправлять сообщения другому пользователю по его никнейму. Если другого пользователя не существует (неправильно указан никнейм), отправить сообщение невозможно. *Критерии:*
    - *есть API, позволяющий отправить другому пользователю сообщение, реализована проверка на существование пользователя (**обязательно**).*
  - Пользователь должен иметь возможность просматривать историю общения с другим пользователем. *Критерии:*
    - *есть API, позволяющий просматривать историю сообщений с конкретным пользователем (**обязательно**);*
    - *обмен и просмотр сообщений реализован с помощью веб-сокетов.*

    - ДОПОЛНИТЕЛЬНО: возможность добавлять других пользователей в друзья, а также просматривать список своих друзей. *Критерии:*
        - *есть API, позволяющий просматривать друзей, а также добавлять в друзья другого пользователя;*
        - *есть возможность ограничивать получение сообщений только своим кругом друзей;*
        - *есть возможность просматривать друзей другого пользователя, и, соответственно, возможность скрывать свой список друзей.*

Будет плюсом:

- использование базы данных (PostgreSQL или MongoDB) для хранения данных о пользователях и переписках между ними;
- документирование запросов через Swagger;
- написание тестов.

При выполнении важно обратить внимание на следующее:

- Ваша задача - сделать рабочий вариант приложения. От кода который не запускается пользы мало.
- Код может быть сколько угодно сложным, но не забывайте — лучше более простой, но рабочий вариант.
- Старайтесь разрабатывать приложение в соответствии с правилами чистого кода, это не олимпиадная задача и читаемость тоже оценивается.

При отправке выполненной работы требуется:

- снять короткий видеоролик (до 60 сек) с демонстрацией работы приложения (запись экрана) и загрузить его на диск-хранилище;
- прислать ссылку на git репозиторий с выполненным заданием;
- добавить в корень репозитория README (в формате md или txt) с примерами REST-запросов и ответов и кратким описанием работы приложения (обязательно нужно указать, если были выполнены какие-то доп.задания).

Удачи вам!