Тестовое задание

SPA-приложение: Комментарии.

Пользователь может оставлять комментарии, как на картинке;
Старался сделать похожий дизайн. У меня картинка в комменте – не аватар юзера, а информационное изображение.

Все введенные комментарии пользователем сохраняются в реляционной базе данных (БД), включая данные о пользователе (данные которые помогут идентифицировать клиента).
Так и сделано.
Для авторизованных пользователей я сохраняю только FK на юзера и для удобства сохраняю username в поле nickname.
Для гостей сохраняются:
- nickname
- email
- homepage
- IP

Данные геолокации подтягиваются по IP отдельной manage коммандой.
Чтобы не зависеть от скорости внешнего сервиса.

Обязательно использовать:
- OOP
Django ORM
- SQL
MySQL

Форма записи должна иметь поля:
UserName (цифры и латинские буквы), обяз.
nickname
E-mail (формат email), обяз.
email
Home page (формат url), необяз
homepage
CAPTCHA
Text (запрещены теги кроме разрешенных)
body

Реализованы требуемые ограничения для полей.
Для авторизованного пользователя вывожу краткую форму.

Главная страница должна соответствовать следующим требованиям:
1. На каждую запись можно написать сколько угодно записей (каскадное отображение)
Я реализовал схему с двумя уровнями. Ограничил каскадность.
Модель позволяет хранить и извлекать комментарии любой вложенности.
А по смыслу моего проекта и по дизайну я самоограничился.

2. Заглавные комментарии должны выводиться в виде таблицы с возможностью сортировки по Username, Email и Datetime, в прямом и обратном порядке.
Реализовал.

3. Сообщения разбиваются на страницы по 25 комментов.
Реализовал. Параметр в настройках проекта.
4. Защита от XSS и SQL инъекций.
За счёт встроенных возможностей Django

5. Приветствуется создание простейшего дизайна с помощью CSS
Использовал Bootstrap 3 как основу.

JavaScript и работа с файлами

1. К комменту пользователь может добавить картинку или текстовый файл.
Реализовал. В модели и в форме.

2. Изображение должно быть не больше 320x240 , уменьшать если больше. Допустимые форматы.
Реализовал.

3. Текстовый файл должен быть не более 100КБ чистый txt
Реализовал

4. Визуальные эффекты для изображений.
Реализовал.

Регулярные выражения

1. Пользователь может использовать только разрешённые теги.
Реализовал. В js и python

2. Проверка на валидность xhtml
Я не делал это регулярками.
Парсил с помощью lxml и смотрел на ошибки.

JavaScript и Ajax
1. Валидация данных на стороне сервера и клиента.
Все формы валидируются на сервере.
На клиенте форма для коммента первого уровня валидируется, поля проверяются, но запрета на отсылку нет.
А вот к форме ответа на коммент я валидацию не прикручивал. Там всё аналогично за исключением того, что эта форма грузится через ajax и нужно каждый раз дёргать функцию валидации из основной страницы. Я решил сэкономить.

2. функция предпросмотр страницы.
Не сделал.
Планировал прикрутить TinyMCE, настроив его, закрыв лишнее. Но не доделал.

3. Панель с кнопками для добавления тегов.
Не делал. Смотри предыдущий пункт.
Чистый фронтенд. Изобретать велосипед не хочется.

4. Добавление визуальных эффектов приветствуется.
Только zoom для изображений.

Не очень понял про ajax в этих пунктах.
Аякса много. За счёт использования htmx.
Отличная библиотека.

Требования к результату.
Обязательные инструменты
Django
yes
Django ORM
yes
Frontend (Vue, React or Angular)
htmx, alpine
Git
yes
Docker
no, virtualenv only
WebSocket
no