**Задача**

Есть 3 файла описывающие основу фронта стандартной авторизации некоторого сервиса: html, css, js (файлы прикреплены).

1. Start.html – html описание страницы авторизации, содержащая 3 группы полей для ввода:
   1. поля регистрации (передаются серверу через асинхронный запрос – ajax; получение значений соответствующих полей во Flask: для получения ФИО - request.get\_json().get(‘f’), логина – ‘l’ вместо ‘f’, пароля – ‘p’)
   2. поля ввода логина-пароля для проверки (передаются серверу через асинхронный запрос – ajax; получение значений соответствующих полей во Flask: для получения логина - request.get\_json().get(‘l’), пароля – ‘p’ вместо ‘l’).
   3. скрытая форма ввода логина и пароля для входа. Данная форма отправляется автоматически (через JS), если пара логин-пароль соответствует значениям в базе (в соответствии с правилами хранения и использования логинов и паролей).
2. Main.css – стили (неважно)
3. Main.js – скрипты, отправляющие ajax запросы (декораторы могут быть извлечены из части ajax запроса “url:/...”), http-форму, получающие ответы на ajax запросы. Здесь интерес представляют функции Register() и Check\_auth():
   1. Register() отправляет ajax запрос, обеспечивая csrf защиту, передающий серверу значение полей регистрации. Сервер отвечает на ajax запрос данными в формате json (response = dict.fromkeys(['message', 'status']) – пример создания ответа в Python 3). Статус (status) = 1, если произошла ошибка, иначе – если регистрация успешна. Помимо статуса передается сообщение (например, «Регистрация успешна» или «В логине используются запрещенные символы» - формируется сервером)
   2. Check\_auth() также отправляет ajax запрос, передавая значения полей ввода логина и пароля. В случае если статус ответа = 0 (ошибки нет), то в данной же функции осуществляется submit http-формы (http – запрос).

Нужно реализовать (желательно на Flask, но можно и на Django):

1. База данных (SQL) для хранения необходимых данных авторизации (в соответствии с существующей практикой хранения и использования данных авторизации). Обращения к базе данных должны осуществляться без использования встроенных в Django инструментов работы с БД);
2. Серверная часть веб-приложения (авторизации), обеспечивающая: обработку каждой из групп запросов, логичное реагирование на такие запросы (проверку вводимых данных на запрещенные символы, длину и т.п.; запись в базу; проверку доступа (соответствия пары логин-пароль) пользователя);
3. В случае успешной авторизации, в соответствии с существующими практиками авторизации, пользователю должна передаваться Cookie – если пользователь повторно заходит на страницу авторизации с валидным Cookie, должна осуществляться переадресация на yandex.ru (также в случае успешной авторизации должна осуществляться соответствующая переадресация);
4. Серверная часть должна обеспечивать CSRF защиту