**Site for scheduling a service (massage, therapy, haircut, notary etc.)**

# Brief description of the system

The goal is to create a website that allows customers to register for a service via registered users and open calendar with service providers schedule information.

The website is equipped with information about services and specialties of multiple service providers.

# Main system features

User registration and login.

Registering for a service by choosing a provider.

Checking the schedule by service providers.

Locking calendar days for vacation or other reasons.

Option to cancel or reschedule service.

# General Guidelines

Website development using Django and Django REST Framework as an API.

Introducing the division into models, views and controllers in the application and placing the appropriate logic in each of them.

Securing access to the application and functionality by using django.contrib.auth.

# Functionalities

*Home page*

* creating the first controller and a view file
* creating files with style / script definitions (bootstrap + possible own), which will be attached to each subsequent page (include)
* the site should have the name of the website and the log in and sign up buttons on the top section

# User registration

* registration form containing:
* login (email) – checking if the e-mail is correct,
* password – must consist of at least 8 characters but not more than 30 characters,
* name to display – the field cannot be empty or contain white space only, and the maximum length is 50 characters.
* a user with a given email can register only once.
* the user should have system roles associated with it, that will cover at least two cases: the service provider and the ordinary user. Every registered person automatically gets the "standard user" role.
* the password is kept in a database in a form that prevents snooping / recovery.

# User login

* login form containing login and password.
* login using django.contrib.auth (to create the appropriate configuration).
* after successful logging in, the user should be redirected to the home page, where instead of the login / Sign up buttons the following information will be displayed: "Logged in as: email".

# Adding a new service

# Service List

* on the home page, in the central part, a list of all current events should be placed - each element of the list should contain:
* highlighted headline with the event title,
* event date from/to,
* first 50 characters of the description.
* events should be sorted from the nearest.

# Event search engine

* at the top of the home page you should add a form containing:
* text box to enter a phrase,
* optional choose box (dropdown): future, ongoing and future, all, - "search" button.
* the entered phrase is to be searched in the title.
* search results should be on a separate page, in the same layout as on the home page.
* the search results page should also contain the search form as on the home page, and its fields should be set according to the currently selected criteria.

# Detailed view of the event

* a separate page on which all event features will be visible: title, dates from / to, full description.
* on the home page and on the search results page, link the title so that when clicked, it will take you to the specific event page.

# Add comments to the event

* under the general information about the event an add comment form should be placed.
* the comment can be up to 500 characters long.
* only a logged in user can add a comment.
* below form all comments added so far should be displayed in the order from the most recent.

# Signing up for an event

* on the event page you should add the option (button) to subscribe to it, but only for logged in users.
* if the current user is already registered, instead of the button he will see the relevant information and optionally a button to unsubscribe from the event.
* next to the general information about the event, put a list of all currently registered users.

# API for other websites - listing of events

* The API should meet the REST recommendations.
* the method should return a list of all future events.
* optionally, it can additionally enable filtering of returned events to a date range.

# Application displaying events downloaded from the API

* you need to build a second application (Django) that will consume the event API and display the list in some of its views
* optional: adding in the application the possibility of filtering events to a selected date range, using the filtering on the API side

*Additional tasks and extensions:*

# Possibility to add a picture to the event

* adding the ability to add a graphic file to the add / edit event form.
* saving the uploaded file to disk or to some external cloud storage service via API or to make such a service with your own API (advanced version)
* serving files for displaying in the event details (plus possibly in other places).

# Editing an event

* additional page, that will allow you to edit the created event.
* only the owner or administrator of the event can edit the event (new role for the user).
* the editing option should appear at least on the event details page.

# My events

* section for the logged in user, where he will see all events, both those in which he participated and those that he owns.
* the latter should be able to switch to editing (mechanism in the previous task).
* the list should allow filtering:
* role:
* all
* organizer - participant
* when:
* future
* ongoing and future
* past
* all
* date (optional):
* field from
* field to

# Additional requirements

* it is necessary to ensure an aesthetic and functional way of presenting data
* data collected from users should be pre-validated