Индивидуальное задание по дисциплине
«Разработка программных приложений».

Вариант 1: Разработка учебного портала для преподавателей и студентов.

Описание задачи:

- Модели:
  - ✓ Пользователи: Создать модель пользователей с ролями "преподаватель" и "студент".
  - ✓ Курсы: Преподаватели могут создавать курсы. Каждый курс имеет название, описание, и преподавателя, который его ведет.
  - ✓ Занятия: В каждом курсе есть несколько занятий. Занятия содержат текстовое описание и дату создания.
  - ✓ Задания: Преподаватели могут создавать задания для студентов по каждому занятию. У каждого задания есть срок сдачи и описание.
  - ✓ Ответы студентов: Студенты могут отправлять свои ответы на задания. Ответы должны сохраняться в базе данных.
  - ✓ Оценки: Учителя могут выставлять оценки за выполненные задания.

- Функционал:
  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для преподавателей: возможность добавлять курсы, занятия и задания.
  - ✓ Для студентов: возможность просматривать доступные курсы и занятия, отправлять ответы на задания.
  - ✓ Личный кабинет: преподаватели могут просматривать список студентов, их ответы и выставлять оценки.
  - ✓ Личный кабинет студента: просмотр курсов, занятий и оценок.

- Дополнительно:
  - ✓ Использовать Django Admin для управления данными (пользователи, курсы, задания).
  - ✓ Реализовать фильтрацию и сортировку курсов по различным параметрам.
  - ✓ Настроить email-уведомления для отправки напоминаний о сроках сдачи заданий.
  - ✓ Использовать Django Forms:
    - o Форма регистрации пользователя: Создать форму для регистрации нового пользователя. Добавить

дополнительные поля, такие как "роль" (преподаватель или студент).

- Форма добавления комментария к занятию: Для каждого занятия сделать возможность добавлять комментарии. Студенты могут оставлять вопросы, а преподаватели — комментарии к материалу. Для этого нужна простая форма с полем "текст комментария".

- Форма загрузки ответа на задание: Реализовать форму, где студент может загрузить свой ответ на задание. Это может быть текстовое поле для ссылки на документ или поле для загрузки файла.

- Форма редактирования профиля: Сделать возможность для пользователей редактировать свои профили: менять имя, email.

- Форма обратной связи: Добавить страницу с формой, где пользователи могут отправлять свои отзывы или предложения по улучшению сайта.

✓ Реализовать тестирование с использованием Django Testing Framework.

Вариант 2: Система управления библиотекой

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "библиотекарь" и "читатель".
  - ✓ Книги: Библиотекари могут добавлять книги. Каждая книга имеет название, автора, описание, категорию и статус (доступна или на руках).
  - ✓ Категории книг: Каждая книга принадлежит к определенной категории (например, научная литература, художественная литература и т.д.).
  - ✓ Выдача книг: Читатели могут брать книги на время. Каждая запись о выдаче включает книгу, читателя, дату выдачи и дату возврата.
  - ✓ Возвраты: Библиотекари могут фиксировать возвраты книг и изменять их статус обратно на "доступна".
  - ✓ Штрафы: Если книга не возвращена в срок, система рассчитывает и фиксирует штраф.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для библиотекарей: возможность добавлять новые книги, назначать и учитывать их выдачу, просматривать текущие выдачи и возвращенные книги.
  - ✓ Для читателей: возможность просматривать доступные книги, брать их в аренду и отслеживать свои текущие и прошедшие выдачи.
  - ✓ Личный кабинет библиотекаря: управление книгами и учет выдач и возвратов, просмотр задолженностей и штрафов.
  - ✓ Личный кабинет читателя: просмотр взятых книг, истории выдач, уведомления о сроках возврата и задолженностях.

- Дополнительно:

  - ✓ Использовать Django Admin для управления данными (пользователи, книги, категории, выдачи).
  - ✓ Реализовать фильтрацию и сортировку книг по названию, автору, категории и статусу.
  - ✓ Настроить email-уведомления для напоминания о сроках возврата книг и начисленных штрафах.
  - ✓ Django Forms:

- Форма регистрации пользователя: Создать форму для регистрации нового пользователя с выбором роли (библиотекарь или читатель).
- Форма добавления книги: Реализовать форму для добавления книги с выбором категории и статуса.
- Форма выдачи книги: Читатели могут через форму выбрать книгу и запросить ее выдачу.
- Форма возврата книги: Библиотекари через форму фиксируют возврат книги.
- Форма редактирования профиля: Возможность редактировать профиль пользователя (имя, email).
- Форма обратной связи: Страница с формой для отправки отзывов или жалоб.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала регистрации, выдачи и возврата книг, начисления штрафов.

Вариант 3: Система бронирования номеров в отеле

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "администратор" и "гость".
  - ✓ Номера: Администраторы могут добавлять и редактировать номера в отеле. Каждый номер имеет описание, количество мест, стоимость за ночь и статус (доступен/занят).
  - ✓ Бронирования: Гости могут бронировать номера. Каждое бронирование включает гостя, номер, дату заезда, дату выезда, количество человек и статус (подтверждено/отменено).
  - ✓ Услуги отеля: Дополнительные услуги, которые гости могут заказать (например, завтрак, трансфер).
  - ✓ Заказы услуг: Гости могут заказывать дополнительные услуги, которые будут добавлены к их бронированию.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для администраторов: возможность добавлять новые номера, просматривать текущие бронирования, подтверждать или отменять бронирования, редактировать информацию о номерах и услугах.
  - ✓ Для гостей: возможность просматривать доступные номера, бронировать их на определенные даты и заказывать дополнительные услуги.
  - ✓ Личный кабинет администратора: управление номерами, бронированиями и заказами услуг.
  - ✓ Личный кабинет гостя: просмотр текущих и прошедших бронирований, возможность отмены бронирования, управление дополнительными услугами.

- Дополнительно:

  - ✓ Использовать Django Admin для управления данными (пользователи, номера, бронирования, услуги).
  - ✓ Реализовать фильтрацию номеров по количеству мест, цене и статусу.
  - ✓ Настроить email-уведомления для отправки подтверждений бронирования и напоминаний перед датой заезда.
  - ✓ Django Forms:

- Форма регистрации пользователя: с возможностью выбрать роль (гость или администратор).
- Форма бронирования номера: реализовать форму, где гость может выбрать номер, дату заезда и выезда.
- Форма заказа услуги: гости могут заказывать дополнительные услуги через форму, привязанную к их бронированию.
- Форма редактирования профиля: возможность редактировать профиль пользователя (имя, email).
- Форма обратной связи: страница с формой для отправки отзывов о номере или сервисе.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала бронирования номеров, заказа услуг и отправки уведомлений.

Вариант 4: Платформа для управления спортивными мероприятиями

Описание задачи:

- Модели:
  - ✓ Пользователи: Создать модель пользователей с ролями "организатор" и "участник".
  - ✓ Мероприятия: Организаторы могут создавать спортивные мероприятия (например, марафоны, соревнования), указывая название, описание, дату и место проведения.
  - ✓ Регистрация на мероприятие: Участники могут регистрироваться на мероприятия. Каждая регистрация включает информацию о пользователе, мероприятии и статусе (подтверждено/отменено).
  - ✓ Результаты мероприятий: Организаторы могут добавлять результаты участников (время, место и баллы).

- Функционал:
  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для организаторов: возможность добавлять мероприятия, управлять списками зарегистрированных участников и публиковать результаты.
  - ✓ Для участников: возможность просматривать доступные мероприятия, регистрироваться на них и просматривать свои результаты.
  - ✓ Личный кабинет организатора: управление мероприятиями, регистрациями участников и результатами.
  - ✓ Личный кабинет участника: просмотр текущих и прошедших мероприятий, просмотр своих регистраций и результатов.

- Дополнительно:
  - ✓ Использовать Django Admin для управления данными (пользователи, мероприятия, результаты).
  - ✓ Реализовать фильтрацию мероприятий по дате, месту проведения и типу мероприятия.
  - ✓ Настроить email-уведомления для подтверждения регистрации и отправки результатов.
  - ✓ Django Forms:
    - o Форма создания мероприятия: форма для добавления нового мероприятия.

- Форма регистрации на мероприятие: участники могут регистрироваться на мероприятия через форму.
- Форма редактирования профиля: возможность редактировать профиль (имя, email).
- Форма обратной связи: страница с формой для отправки отзывов о мероприятиях.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала регистрации на мероприятия, публикации результатов и отправки уведомлений.

Вариант 5: Система управления заказами для ресторана

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "администратор" и "гость".
  - ✓ Меню: Администраторы могут добавлять и редактировать блюда в меню, включая название, описание, категорию (закуски, основные блюда, десерты) и цену.
  - ✓ Заказы: Гости могут оформлять заказы, выбирая блюда из меню. Каждый заказ включает блюда, пользователя, дату и статус (в обработке/готово/доставлено).
  - ✓ Отзывы: После доставки заказа гости могут оставлять отзывы о блюдах.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для администраторов: возможность добавлять новые блюда в меню, управлять заказами, изменять их статус и просматривать отзывы гостей.
  - ✓ Для гостей: возможность просматривать меню, выбирать блюда, оформлять заказы и оставлять отзывы.
  - ✓ Личный кабинет администратора: управление блюдами, заказами и отзывами.
  - ✓ Личный кабинет гостя: просмотр истории заказов, возможность отмены заказа до начала обработки и оставление отзывов.

- Дополнительно:

  - ✓ Использовать Django Admin для управления данными (пользователи, меню, заказы, отзывы).
  - ✓ Реализовать фильтрацию меню по категориям блюд и ценам.
  - ✓ Настроить email-уведомления для подтверждения заказа и уведомлений о его статусе.
  - ✓ Django Forms:
    - Форма создания блюда: администраторы могут добавлять блюда в меню через форму.
    - Форма оформления заказа: гости могут выбирать блюда и оформлять заказ через форму.
    - Форма редактирования профиля: возможность редактировать профиль (имя, email).

- Форма отзыва: гости могут оставлять отзывы на заказанные блюда.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала оформления заказов, изменения их статуса и отправки уведомлений.

Вариант 6: Платформа для управления недвижимостью

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "администратор" и "клиент".
  - ✓ Объявления о недвижимости: Администраторы могут добавлять объявления с описанием объекта (квартира, дом), площадью, местоположением, ценой и статусом (свободно/сдано).
  - ✓ Запросы на просмотр: Клиенты могут отправлять запросы на просмотр объектов. Каждый запрос включает объект, дату и статус (подтверждено/ожидание).
  - ✓ Договоры аренды: После подтверждения просмотра и решения, администраторы могут заключать договоры аренды с клиентами.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для администраторов: возможность добавлять объекты недвижимости, подтверждать запросы на просмотр и оформлять договоры аренды.
  - ✓ Для клиентов: возможность просматривать доступные объекты, отправлять запросы на просмотр и заключать договоры аренды.
  - ✓ Личный кабинет администратора: управление объектами, запросами и договорами аренды.
  - ✓ Личный кабинет клиента: просмотр заявок на просмотр и заключенных договоров.

- Дополнительно:

  - ✓ Использовать Django Admin для управления данными (пользователи, объекты, запросы).
  - ✓ Реализовать фильтрацию объектов по цене, местоположению и площади.
  - ✓ Настроить email-уведомления для подтверждения просмотров и оформления договоров.
  - ✓ Django Forms:
    - Форма добавления объекта недвижимости.
    - Форма отправки запроса на просмотр.
    - Форма редактирования профиля.

- Форма обратной связи: возможность отправлять отзывы о качестве обслуживания.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала управления объектами, запросов на просмотр и заключения договоров аренды.

Вариант 7: Платформа для онлайн-курсов по кулинарии

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "преподаватель" и "ученик".
  - ✓ Курсы: Преподаватели могут создавать курсы по кулинарии. Каждый курс включает название, описание, список рецептов и статус (активен/завершен).
  - ✓ Уроки: Каждый курс состоит из уроков с видеоуроками, рецептами и пошаговыми инструкциями.
  - ✓ Комментарии и отзывы: Ученики могут оставлять отзывы и комментарии на курсы.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для преподавателей: возможность добавлять курсы и уроки, управлять комментариями и отзывами.
  - ✓ Для учеников: возможность просматривать курсы, проходить уроки и оставлять отзывы.
  - ✓ Личный кабинет преподавателя: управление курсами, уроками и отзывами.
  - ✓ Личный кабинет ученика: просмотр курсов, оставленных комментариев и прогресса по урокам.

- Дополнительно:

  - ✓ Использовать Django Admin для управления курсами, уроками и отзывами.
  - ✓ Реализовать фильтрацию курсов по сложности, рейтингу и продолжительности.
  - ✓ Настроить email-уведомления для уведомлений о новых уроках и комментариях.
  - ✓ Django Forms:
    - Форма создания курса.
    - Форма добавления комментария к уроку.
    - Форма редактирования профиля.
    - Форма обратной связи: страница для отправки предложений и отзывов о курсах.

- ✓ • Реализовать тестирование с использованием Django Testing Framework для проверки функционала создания курсов, прохождения уроков и оставления комментариев.

Вариант 8: Платформа для управления медицинскими записями

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "врач" и "пациент".
  - ✓ Записи на прием: Пациенты могут записываться на прием к врачу, указывая дату, время и причину обращения.
  - ✓ Врачи: Врачи могут просматривать записи пациентов, отмечать статус посещения (ожидание/пройдено) и добавлять диагнозы.
  - ✓ Медицинские записи: Врачи могут вносить медицинские записи о пациентах, включая диагнозы, рекомендации и назначенные лекарства.
  - ✓ Рецепты: Врачи могут выписывать рецепты пациентам, которые включают лекарство, дозировку и рекомендации по применению.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для врачей: возможность просматривать записи на прием, управлять медицинскими записями пациентов и выписывать рецепты.
  - ✓ Для пациентов: возможность записываться на прием, просматривать историю посещений, рецептов и медицинских записей.
  - ✓ Личный кабинет врача: управление записями на прием, медицинскими записями и рецептами.
  - ✓ Личный кабинет пациента: просмотр истории приемов, диагнозов и выписанных рецептов.

- Дополнительно:

  - ✓ Использовать Django Admin для управления пользователями (пациенты, врачи), записями и медицинскими данными.
  - ✓ Реализовать фильтрацию записей на прием по дате, врачу и статусу.
  - ✓ Настроить email-уведомления для напоминаний о приеме и уведомлений о новых медицинских записях или рецептах.
  - ✓ Django Forms:
    - Форма записи на прием: для пациентов.
    - Форма добавления медицинской записи: для врачей.

- Форма редактирования профиля: возможность изменения данных пользователя.
- Форма обратной связи: страница для отзывов о враче или системе.

✓ • Реализовать тестирование с использованием Django Testing Framework для проверки функционала записи на прием, внесения медицинских записей и отправки уведомлений.

Вариант 9: Платформа для управления проектами и задачами

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "менеджер" и "сотрудник".
  - ✓ Проекты: Менеджеры могут создавать проекты, задавать сроки выполнения и назначать сотрудников на проекты.
  - ✓ Задачи: Каждому проекту назначаются задачи. Менеджеры могут назначать сотрудников для выполнения задач и отслеживать прогресс.
  - ✓ Отчеты: Сотрудники могут загружать отчеты о выполненных задачах, указывая прогресс, затраченное время и результат.
  - ✓ Комментарии: Менеджеры и сотрудники могут добавлять комментарии к задачам для уточнения деталей или обсуждения проблем.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для менеджеров: возможность добавлять проекты, назначать задачи сотрудникам и просматривать отчеты о выполнении.
  - ✓ Для сотрудников: возможность просматривать назначенные задачи, загружать отчеты и добавлять комментарии к задачам.
  - ✓ Личный кабинет менеджера: управление проектами, задачами, отчетами и комментариями.
  - ✓ Личный кабинет сотрудника: просмотр назначенных проектов и задач, загрузка отчетов и ведение обсуждений.

- Дополнительно:

  - ✓ Использовать Django Admin для управления данными (пользователи, проекты, задачи).
  - ✓ Реализовать фильтрацию задач по статусу выполнения, проектам и срокам.
  - ✓ Настроить email-уведомления для уведомлений о новых задачах и напоминания о сроках выполнения.
  - ✓ Django Forms:
    - Форма создания проекта: для менеджеров.
    - Форма добавления отчета: сотрудники могут загружать отчеты по выполненным задачам.
    - Форма редактирования профиля: возможность редактирования данных пользователя.

- Форма обратной связи: возможность отправлять отзывы о проекте или задачах.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала создания проектов, назначения задач и загрузки отчетов.

Вариант 10: Платформа для организации выставок и мероприятий

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "организатор" и "посетитель".
  - ✓ Выставки: Организаторы могут создавать выставки, указывая название, описание, дату и место проведения, а также список участников.
  - ✓ Участники: Каждая выставка может включать различных участников (например, художников, дизайнеров), которые представляют свои работы.
  - ✓ Билеты: Посетители могут приобретать билеты на выставки, указав дату посещения и тип билета (обычный/премиум).

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для организаторов: возможность добавлять выставки, управлять участниками и просматривать проданные билеты.
  - ✓ Для посетителей: возможность просматривать выставки, покупать билеты и оставлять отзывы.
  - ✓ Личный кабинет организатора: управление выставками, участниками и отчетами о продаже билетов.
  - ✓ Личный кабинет посетителя: просмотр купленных билетов и оставленных отзывов.

- Дополнительно:

  - ✓ Использовать Django Admin для управления выставками, пользователями и билетами.
  - ✓ Реализовать фильтрацию выставок по дате, месту и типу мероприятия.
  - ✓ Настроить email-уведомления для подтверждения покупки билета и напоминаний о выставках.
  - ✓ Django Forms:
    - Форма создания выставки: для организаторов.

- Форма покупки билета: для посетителей.
- Форма редактирования профиля: возможность редактирования данных пользователя.
- Форма обратной связи: страница для отзывов о выставках.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала создания выставок, продажи билетов и отправки уведомлений.

Вариант 11: Система управления фитнес-клубом

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "тренер" и "клиент".
  - ✓ Тренировки: Тренеры могут добавлять и редактировать тренировки, указывая название, описание, продолжительность и тип (индивидуальная/групповая).
  - ✓ Клиенты: Клиенты могут записываться на тренировки, выбирать тренера и дату занятия.
  - ✓ Абонементы: Клиенты могут приобретать абонементы, которые позволяют посещать определенное количество тренировок.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для тренеров: возможность добавлять тренировки, управлять расписанием и просматривать записки клиентов.
  - ✓ Для клиентов: возможность просматривать доступные тренировки, записываться на них и отслеживать свои посещения.
  - ✓ Личный кабинет тренера: управление тренировками, клиентами и расписанием.
  - ✓ Личный кабинет клиента: просмотр расписания тренировок и истории посещений.

- Дополнительно:

  - ✓ Использовать Django Admin для управления тренировками, абонементами и пользователями.
  - ✓ Реализовать фильтрацию тренировок по типу, дате и тренеру.
  - ✓ Настроить email-уведомления для подтверждения записи на тренировку и напоминаний о предстоящих занятиях.
  - ✓ Django Forms:
    - Форма создания тренировки: для тренеров.

- Форма записи на тренировку: для клиентов.
- Форма редактирования профиля: возможность редактирования данных пользователя.
- Форма обратной связи: возможность отправлять отзывы о тренировках и тренерах.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала записи на тренировки, управления расписанием и отправки уведомлений.

Вариант 12: Платформа для управления кулинарными рецептами

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "шеф-повар" и "любитель кулинарии".
  - ✓ Рецепты: Шеф-повара могут добавлять и редактировать рецепты, указывая название, описание, ингредиенты, время приготовления и шаги.
  - ✓ Категории: Рецепты можно классифицировать по категориям (завтраки, обеды, ужины, десерты).
  - ✓ Комментарии: Любители кулинарии могут оставлять отзывы и комментарии к рецептам.

- • Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для шеф-поваров: возможность добавлять рецепты, управлять категориями и просматривать отзывы.
  - ✓ Для любителей кулинарии: возможность просматривать рецепты, оставлять комментарии и сохранять любимые рецепты в избранное.
  - ✓ Личный кабинет шеф-повара: управление рецептами, категориями и отзывами.
  - ✓ Личный кабинет любителя кулинарии: просмотр сохраненных рецептов и оставленных комментариев.

- Дополнительно:

  - ✓ Использовать Django Admin для управления рецептами, пользователями и комментариями.
  - ✓ Реализовать фильтрацию рецептов по категориям, времени приготовления и популярности.
  - ✓ Настроить email-уведомления для подтверждения добавления рецепта и уведомлений о новых комментариях.
  - ✓ Django Forms:
    - Форма добавления рецепта: для шеф-поваров.
    - Форма комментария: для любителей кулинарии.

- Форма редактирования профиля: возможность редактирования данных пользователя.
- Форма обратной связи: страница для отправки отзывов о сайте и рецептах.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала добавления рецептов, оставления комментариев и отправки уведомлений.

Вариант 13: Платформа для управления спортивными командами

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "тренер" и "игрок".
  - ✓ Команды: Тренеры могут создавать команды, указывая название, вид спорта и список игроков.
  - ✓ Матчи: Каждая команда может участвовать в матчах, где фиксируются дата, соперник, результат и место проведения.
  - ✓ Тренировочные сессии: Тренеры могут планировать тренировки, указывая дату, время и содержание тренировки.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для тренеров: возможность добавлять команды, планировать тренировки и просматривать результаты матчей.
  - ✓ Для игроков: возможность просматривать расписание тренировок и матчей, а также получать уведомления о предстоящих событиях.
  - ✓ Личный кабинет тренера: управление командами, матчами и тренировками.
  - ✓ Личный кабинет игрока: просмотр расписания тренировок и матчей, а также результатов.

- Дополнительно:

  - ✓ Использовать Django Admin для управления командами, матчами и пользователями.
  - ✓ Реализовать фильтрацию матчей по дате, команде и результату.
  - ✓ Настроить email-уведомления для напоминаний о тренировках и матчах.
  - ✓ Django Forms:
    - Форма создания команды: для тренеров.
    - Форма добавления матча: для тренеров.

- Форма редактирования профиля: возможность редактирования данных пользователя.
- Форма обратной связи: возможность отправлять отзывы о команде или тренировках.

✓ Реализовать тестирование с использованием Django Testing Framework для проверки функционала создания команд, планирования тренировок и отправки уведомлений.

Вариант 14: Платформа для организации путешествий

Описание задачи:

- • Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "турист" и "гид".
  - ✓ Туры: Гиды могут добавлять и редактировать туры, указывая название, описание, стоимость, даты и места посещения.
  - ✓ Записи на тур: Туристы могут записываться на туры, выбирая даты и количество участников.
  - ✓ Отзывы о турах: Туристы могут оставлять отзывы о посещенных турах и гидах.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для гидов: возможность добавлять туры, управлять записями и просматривать отзывы.
  - ✓ Для туристов: возможность просматривать доступные туры, записываться на них и оставлять отзывы.
  - ✓ Личный кабинет гида: управление турами, записями и отзывами.
  - ✓ Личный кабинет туриста: просмотр записей на туры и оставленных отзывов.

- Дополнительно:

  - ✓ Использовать Django Admin для управления турами, пользователями и отзывами.
  - ✓ Реализовать фильтрацию туров по датам, стоимости и типу путешествия.
  - ✓ Настроить email-уведомления для подтверждения записи на тур и напоминаний о предстоящих поездках.
  - ✓ Django Forms:
    - Форма добавления тура: для гидов.
    - Форма записи на тур: для туристов.

- Форма редактирования профиля: возможность редактирования данных пользователя.
- Форма обратной связи: возможность отправлять отзывы о турах и гидах.

✓ • Реализовать тестирование с использованием Django Testing Framework для проверки функционала добавления туров, записи на туры и оставления отзывов.

Вариант 15: Система управления фриланс-проектами

Описание задачи:

- Модели:

  - ✓ Пользователи: Создать модель пользователей с ролями "работодатель" и "фрилансер".
  - ✓ Проекты: Работодатели могут размещать проекты, указывая название, описание, бюджет и сроки выполнения.
  - ✓ Заявки: Фрилансеры могут подавать заявки на проекты, указывая свои расценки и описание опыта.
  - ✓ Отчеты о выполнении: Фрилансеры могут загружать отчеты о выполненных проектах для оценки работодателями.

- Функционал:

  - ✓ Регистрация и авторизация пользователей.
  - ✓ Для работодателей: возможность размещать проекты, просматривать заявки и выбирать исполнителей.
  - ✓ Для фрилансеров: возможность просматривать доступные проекты, подавать заявки и загружать отчеты.
  - ✓ Личный кабинет работодателя: управление проектами, заявками и отзывами фрилансеров.
  - ✓ Личный кабинет фрилансера: просмотр поданных заявок, отчетов и полученных отзывов.

- Дополнительно:

  - ✓ Использовать Django Admin для управления проектами, пользователями и заявками.
  - ✓ Реализовать фильтрацию проектов по бюджету, срокам и статусу.
  - ✓ Настроить email-уведомления для подтверждения заявок и напоминаний о сроках выполнения.
  - ✓ Django Forms:
    - Форма создания проекта: для работодателей.
    - Форма подачи заявки: для фрилансеров.
    - Форма редактирования профиля: возможность редактирования данных пользователя.
    - Форма обратной связи: возможность отправлять отзывы о проектах.

- ✓ • Реализовать тестирование с использованием Django Testing Framework для проверки функционала размещения проектов, подачи заявок и отправки уведомлений.