### 🔹 1. База данных

**Файл:** models.py, init\_db.py, init\_full\_schedule.sql

Нужно:

* В таблицу users добавить поле role (participant, organizer, admin).
* При инициализации БД (init\_db.py, init\_full\_schedule.sql) создать нескольких тестовых пользователей с разными ролями.
* Таблица events → можно оставить название, но содержимое трактовать как **этап олимпиады**. Добавить поле stage (например, «отборочный», «заключительный», «тренировка»).

### 🔹 2. Backend (Flask)

**Файл:** app.py

* Обновить логику маршрутов:
  + /schedule — доступ для **участника** (только свои события) и для **организатора/админа** (видят общее расписание).
  + /add\_event — доступ только для **организатора** и **админа**.
  + Сделать декоратор или функцию для проверки ролей.

**Файл:** models.py

* Дополнить модель User (поле role).
* Дополнить модель Event (поле stage).

### 🔹 3. Frontend (шаблоны)

**Папка:** templates/

* **base.html** — добавить отображение меню в зависимости от роли (if current\_user.role == 'organizer' → показать «Добавить событие»).
* **schedule.html** — расширить фильтрацию: добавить фильтр по этапу олимпиады.
* **add\_event.html** — форма добавления события (название, дата, время, место, этап олимпиады).
* В будущем можно сделать отдельные страницы: «Управление пользователями» для админа.

### 🔹 4. Инициализация и тесты

**Файл:** init\_db.py

* При старте наполнить тестовыми пользователями:
  + alice (role = participant),
  + bob (role = organizer),
  + admin (role = admin).

**Файл:** init\_full\_schedule.sql

* Перезаполнить расписание примерами этапов олимпиады.

### 🔹 5. (опционально) Новый файл

* **role\_utils.py** — можно вынести туда функцию @role\_required("organizer"), чтобы разграничивать доступ.

📌 Итого, обязательно будут обновляться:

* models.py
* app.py
* init\_db.py
* init\_full\_schedule.sql
* base.html, schedule.html, add\_event.html

Добавятся:

* add\_event.html
* (опционально) role\_utils.py