User Guide

Installation

Prerequisites:

- Docker + Docker Compose
- Flutter SDK (for the frontend)
- Git

Steps:

1. Clone the repo

```
git clone https://github.com/DB-GL-Group/Community-Sports-League-Scheduler-Stats.git
cd Community-Sports-League-Scheduler-Stats
```

2. Environment variables

```
cp .env.example .env
```

3. Start the database + apply migrations

```
make db-start
make db-migrate
```

4. Start backend + worker (FastAPI + RQ)

```
make backend-start
```

5. Start frontend (web)

```
make flutter-setup
cd frontend
flutter run -d chrome
```

How to run

Local URLs:

- Backend API: http://localhost:8000
- Frontend: Flutter web run output (usually http://localhost:PORT)

Hosted URL:

- Not provided in this repo. Use local setup.

Default credentials / roles

Roles are seeded in `db/migrations/V1__init.sql`:

- FAN, MANAGER, REFEREE, ADMIN

Default admin user is seeded in `db/migrations/V2__seed_admin.sql`:

- Email: admin@example.com
- Password: admin
- Note: replace the password hash if you want to change the admin password.

Data seeding

Seed helpers:

- Create fake players:

```
python backend/helper/players.py
```

- Create debug matches:

```
python backend/helper/debug_matches.py --division 1 --count 5 --status in_progress
```

These scripts use `DATABASE_URL` from `.env`.

Step-by-step demo script

1. Log in as admin

```
curl.exe -X POST http://localhost:8000/auth/login -H "Content-Type: application/json" -d '{"email":"admin@example.com","password":"admin"}'
```

Save the returned `access_token` as `ADMIN_TOKEN`.

2. Generate role keys (manager, referee)

```
curl.exe -X POST http://localhost:8000/user/admin/role-keys -H "Authorization: Bearer <ADMIN_TOKEN>" -H "Content-Type: application/json" -d '{"role":"MANAGER"}'
```

Repeat with role `REFEREE`.

3. Sign up a manager and a referee using the keys

```
curl.exe -X POST http://localhost:8000/auth/signup -H "Content-Type: application/json" -d '{"first_name":"Mia","last_name":"Manager","email":"manager@example.com","password":"test123","roles":["FAN","MANAGER"],"role_keys":{"MANAGER":"<MANAGER_KEY>"}}'
```

```
curl.exe -X POST http://localhost:8000/auth/signup -H "Content-Type: application/json" -d '{"first_name":"Rene","last_name":"Ref","email":"ref@example.com","password":"test123","roles":["FAN","REFEREE"],"role_keys":{"REFEREE":"<REFEREE_KEY>"}}'
```

4. Manager creates a team and adds players

```
# Log in as manager
curl.exe -X POST http://localhost:8000/auth/login -H "Content-Type: application/json" -d '{"email":"manager@example.com","password":"test123"}'
```

Use the returned token to:

```
curl.exe -X POST http://localhost:8000/user/manager/team -H "Authorization: Bearer <MANAGER_TOKEN>" -H "Content-Type: application/json" -d '{"division":1,"name":"Suisse","short_name":"SUI","color_primary":"#D32F2F","color_secondary":"#FFFFFF"}'
```

5. Admin runs scheduler

```
curl.exe -X POST http://localhost:8000/user/admin/scheduler/run -H "Authorization: Bearer <ADMIN_TOKEN>"
```

6. Referee sets availability

```
curl.exe -X PUT http://localhost:8000/user/referee/availability -H "Authorization: Bearer <REF_TOKEN>" -H "Content-Type: application/json" -d '{"slot_ids":[1,2,3]}'
```

7. Admin uses console to add goal/card/substitution and finalize a match Open the Admin Console UI and use the panels to add events, then finalize.

8. Fan views matches, rankings, and stats Open the frontend and visit Matches / Rankings / Stats pages.