AirSports app - live tracking website interactions

# Basic flow

## Initial start-up

The user must be able to login with existing credentials, register as a new user, or reset the password of a forgotten account.

New user registration triggers email verification, and the account is locked until the user has clicked the verification link in the email. Until that point it is not possible to login, and the website will return an error message indicating that the account has not been verified.

Once the app has successfully logged in it will receive a basic authentication token which is valid for this login session. This token is the basis for all authenticated API calls. These calls are:

* Logout
* Retrieve profile
* Update profile
* Update partial profile

## Successful login

After successfully logging in and receiving the authentication token, the app should fetch the user profile and display any useful information. Of special interest is app\_tracking\_id which is the secret tracking ID the app should use for position reports to the traccar server. This should not be displayed to the user, nor should it be edited. Note that this can be changed (although it shouldn’t), so the app should always retrieve the updated value when starting up. The other special field is app\_aircraft\_registration, I feel it is expected that the usable update often though should be easily accessible outside of the main profile update view. This is a string that is meant to contain the registration of the aircraft the user is currently operating. Its only use is to label the aircraft on the global map when the user is not participating in a contest. It can thus be anything in principle, but the text related to the field should strongly suggest to use aircraft registration.

## Profile updating

The user should be able to update his/her profile. This includes changing first name and last name, picture, phone, country, and biography. The email address should not be changed, since this is linked to the user address, and the tracking ID should not be displayed at all.

# Authentication actions

Below is a list of all actions related to signup, authentication, and logout.

## Login

Endpoint: /api/accounts/login

Method: POST

Payload: {“email”: “”, “password”:””}

### Description

Email and password are both known by the user and should be entered in an appropriate form the first time the application initialises. The information should be remembered by the application to use when retrieving authentication tokens for API access.

Returns 200 if successful, 400 if there are errors, and 401 if unauthorised.

## Logout

Endpoint: /api/accounts/logout

Method: GET

### Description

Not sure if the app needs to care about this.

Requires token authentication, returns 200 if successful, 401 if failed.

## Signup

Endpoint: /api/accounts/signup

Method: POST

Payload: {“email”: “”, “password”:””, “first\_name”:””, “last\_name”:””}

### Description

The user must enter a valid email address, a strong password, a first name, and a last name. All fields are required. The app should perform some basic password health checks to ensure that a strong password is chosen.

Posting to this endpoint triggers a verification email to be sent to the entered email address. The account remains unverified (and therefore not open for login) until the link in the email has been clicked. The app should try to login every time it starts (after initial registration), and notify the user if login fails with a tip to check the verification email. If the verification email never arrives, the user has to repeat the signup process.

On successful email verification the user account is either tied to an existing profile (by matching email address), or a new profile is created and linked to the user.

The response returns an object with a single key “token”. This key is used for token authentication for API access according to Django rest framework documentation.

Returns 201 if successful (created) or 400 on error.

## Password reset

Endpoint: /api/accounts/password/reset

Method: POST

Payload: {“email”: “”}

### Description

This post will trigger resetting the password (maybe label as “forgotten password”, should only be available if the user is not logged in). This will first send an email to the user with a verification link. The remaining password reset flow will be carried out directly with the website.

# Profile actions

Below is a list of all actions related to profile management.

## Retrieve profile

Endpoint: /api/v1/userprofile/retrieve\_profile/

Method: GET

### Description

Requires token authentication. Returns a “Person” object with details about the person tied to the user account.

## Update profile

Endpoint: /api/v1/userprofile/update\_profile/

Method: PUT

### Description

Requires token authentication. Allows putting an entire ”Person” object. Unless the app will be handling all profile fields, I suggest that the partial update method is employed.

## Update partial profile

Endpoint: /api/v1/userprofile/partial\_update\_profile/

Method: PATCH

### Description

Requires token authentication. Allows patching a subset of a ”Person” object.