WildWatch RESTful API Specification

Student: *Aly Shmahell* **Matricula:** *258912* **E-mail:** *aly.shmahell@student.univaq.it*

Preamble:

This project is a part of a large multi-course project:

WildWatch: a mobile application implemented within the **Mobile Application course**; this app can be operated by authorized **users**.

WildWatch Web: a web interface implemented within the **Web Engineering course**, consists of two web interfaces:

A public web app which can be operated by anonymous **guests**.

A curator web app which can be operated by authorized wildlife table **curators**.

WildWatch RESTful API: an API implemented within the **Advanced Web Development course**, which would handle the databases; therefore will manage the data generated by the users, and provide the data back to the users, the guests and the curators.

The Service:

URL: auth

VERB: POST

Input:

{“firstname”: str, “lastname”: str, “username”: str, “password”: str}

Semantics: If **(firstname & lastname)** are **not empty** then it creates a new user account based on the provided credentials. Otherwise it establishes a session based on the provided credentials and grants an {SID}.

URL: auth/{SID}

VERB: DELETE

Semantics: closes the session and invalidates the {SID}.

URL: auth/{SID}/profile

VERB: GET

Output:

{“fullname”: str, “website”: str, “bio”: str, “photo”: image/jpeg}

Semantics: returns the user profile information associated with the {SID}. This resource is only available to users; if the {SID} belongs to a curator, the return status would be **403 Forbidden**.

URL: auth/{SID}/profile

VERB: DELETE

Semantics: deletes the user account then closes the session and invalidates the {SID}. This resource is only available to users; if the {SID} belongs to a curator, the return status would be **403 Forbidden**.

URL: auth/{SID}/profile/{category}

VERB: PUT

Input:

{ “value”: str | image/jpeg }

Semantics: updates the user info according to the {category}, which could be: fullname, website, bio, password, or photo. This resource is only available to users; if the {SID} belongs to a curator, the return status would be **403 Forbidden**.

URL: auth/{SID}/profile/{userid}

VERB: DELETE

Semantics: deletes the user account associated with {userid}; if and only if {SID} belongs to a curator and {userid} belongs to a user. Otherwise the return status would be **403 Forbidden**.

URL: auth/{SID}/wildlife

VERB: POST

Input:

{“type”: str, “species”: str, “photo”: image/jpeg, “notes”: str, “location”: tuple, “date”: int}

Semantics: submits a new wildlife entry to the wildlife table. This resource is only available to users; if the {SID} belongs to a curator, the return status would be **403 Forbidden**.

URL: auth/{SID}/wildlife?text=str&filters=dict&location=tuple&area=tuple

VERB: GET

Output:

if {SID} belongs to a user account:

[

{“wildlifeid”: int, “type”: str, “species”: str, “photo”: image/jpeg, “notes”: str, “location”: tuple, “date”: int},

]

if {SID} belongs to a curator account:

[

{

“wildlifeid”: int, “type”: str, “species”: str, “photo”: image/jpeg, “notes”: str, “location”: tuple, “date”: int, “userid”: int,

“reports”: [{“reportid”: int, “code”: int, “text”: str},]

},

]

Semantics: fetches all the wildlife entries in the wildlife table based on the user’s or curator’s **location** and the size of their map **area**, the results are filtered by first matching the **filters** part of the query to the wildlife table columns using an SQL WHERE clause, and then by matching the **text** part of the query to the notes column in the wildlife table using an off-the-shelf TFIDF algorithm. If {SID} belongs to a curator account, the filtering process is also applied on the reports table, then only wildlife entries with unresolved reports matching their ‘wildlifeid’ will be returned in the response, with a copy of the reports added to the response.

URL: guest/wildlife?text=str&filters=dict&location=tuple&area=tuple

VERB: GET

Output:

[

{“wildlifeid”: int, “type”: str, “species”: str, “photo”: image/jpeg, “notes”: str, “location”: tuple, “date”: int},

]

Semantics: fetches all the wildlife entries in the wildlife table based on the guest’s chosen **location** and the size of their map **area**, the results are filtered first by matching the **filters** part of the query to the wildlife table columns using an SQL WHERE clause, and then by matching the **text** part of the query to the notes column in the wildlife table using an off-the-shelf TFIDF algorithm. If the request ‘Accept’ header is set to ‘application/zip’, then the response’s ‘Content-Type’ will be set to ‘application/zip’ and the ‘Content-Disposition’ will be set to ‘attachment’ and the return value will be a zip archive of the json objects in the output array.

URL: guest/wildlife/{wildlifeid}

VERB: GET

Output:

{“wildlifeid”: int, “type”: str, “species”: str, “photo”: image/jpeg, “notes”: str, “location”: tuple, “date”: int}

Semantics: downloads a single wildlife entry from the wildlife table according to its {wildlifeid}. The response ‘Content-Disposition’ is set to ‘attachment’ and the output value will be a json file containing the output json object.

URL: guest/report

VERB: POST

Input:

{“wildlifeid”: int, “code”: int, “text”: str}

Semantics: submits a new report about a wildlife entry to the reports table, the report could be about animal abuse, improper fire, fake entries ...etc.

URL: auth/{SID}/report/{reportid}?cascade=bool

VERB: PUT

Semantics: submits a report resolution request to the API, which marks the report as solved, this kind of report concerns animal abuse or similar issues, if found genuine the curator would contact the authorities, then resolve the report, it will be updated as solved in the reports table but not deleted. If **cascade** is **true** then all other reports about the same wildlife entry will be resolved as well. This resource is only available to curators; if the {SID} belongs to a user, the return status would be **403 Forbidden**.

URL: auth/{SID}/report/{reportid}?cascade=bool

VERB: DELETE

Semantics: submits a report deletion request to the API, which deletes the report from the reports table. If **cascade** is **true** then the wildlife entry will be deleted from the wildlife table as well; in that case all other reports associated with the entry will be deleted as well. This resource is only available to curators; if the {SID} belongs to a user, the return status would be **403 Forbidden**.

Entity Relationship Model: