Problem-1

Task description

Your task is to implement a Web API that utilizes simple token-based authentication.

Requirements

The API should support the following requests:

- POST /api/user – add a user to the database (kept in memory).
- Request Body (JSON):
- user_id: string
- login: string
- password: string
- Response:
- HTTP 400, if the body is empty;
- HTTP 201, if the user has been created. The response body can be empty.
- POST /api/authenticate – authenticate a user.
- Request Body (JSON):
- login: string
- password: string
- Response:
- HTTP 400, if the body is empty;
- HTTP 404, if there is no user of the given login name in the database;
- HTTP 401, if a user with the given login name exists, but the password does not match that saved in the database for the corresponding user;
- HTTP 200, if a user with the given login name exists and the given password matches that saved in the database. The response body should be in the shape of:{
  "token": "<uuid>"
  }

- POST /api/logout – log out a user. Requires a token in the request's headers.
- Response:
- HTTP 401, if the token is invalid;
- HTTP 200, if the user is logged out successfully. The token that was passed on becomes invalid.
- POST /api/articles – create an article consisting of title, content and the level of its visibility. Only a user with a valid session can create articles.
- Request Body (JSON):
- article_id: string
- title: string
- content: string
- visibility: 'public' | 'private' | 'logged_in':

- public – the article is available publicly;
- private – the article is only accessible to the creator;
- logged_in – only users with valid a session can access the article.
- Response:
- HTTP 400, if the body is empty;
- HTTP 401, if the provided token is invalid;
- HTTP 201, if an article has been created. The response body can be empty.
- GET /api/articles – return a list of articles. The result depends on the token.
- Response: HTTP 200:
- If a valid token is given in the request's headers, return:
- all public articles;
- all articles with visibility: 'logged_in';
- the sender's articles;
- Otherwise, return only public articles;
- An article object is consisted of the following fields: article_id, title, content and user_id which all are strings, and the visibility field which equals one of these values: public, private, logged_in.
- The articles might appear in any order.
- Added content should be kept in memory; no database/storage back end is available.
- The token should be added as an authentication-header header to a request, wherever applicable.
- A token is associated with a user. It is considered to be invalid if the token was used to log the user out, or if it has never been created as a result of logging the user in.
- It is entirely possible that a user has multiple valid tokens. Sending two consecutive login requests can be completed successfully, and the token returned by the first request does not become invalid as a result of the second request.
- The body of a response with status codes 400–499 can be empty.
- HTTP 5xx error codes are considered errors and must not be returned.
- The default export should be an http.Server object that is returned by app.listen().

Assumptions

- users have a unique id and unique login – the server will never receive two POST /api/user requests with the same id or login; articles, likewise, also have a unique id;
- the content-type header will be set to application/json in every such POST request;
- all strings passed on in request bodies are non-empty;
- id strings (along with other strings) are any string from 1 to 100 characters.
- Use console.log and console.error for debugging purposes.

Available packages/libraries

- Node.js 18.9.0
- Express 4.18.0
- Lodash 4.17.21
- uuid 7.0.0

Examples

1. Example 1

If we add a user such as:

```
{ "user_id": "42", "login": "frank", "password": "p4ssw0rd" }
```

and then call POST /api/authenticate with the same login and password, an example response can be:

```
{ "token": "1b9d6bcd-bbfd-4b2d-9b5d-ab8dfbbd4bed" }
```

2. Example 2

If the user from Example 1 creates the following articles:

```
{ "articles_id": "art1", "title": "tit1", "content": "c1", "visibility": "public" }
{ "articles_id": "art2", "title": "tit2", "content": "c2", "visibility": "private" }
{ "articles_id": "art3", "title": "tit3", "content": "c3", "visibility": "logged_in" }
```

then calling GET /api/articles (with no token passed on as a request's header) gives only one article object:

```
[
  {
    "articles_id": "art1",
    "title": "tit1",
    "content": "c1",
    "visibility": "public",
    "user_id": "42"
  }
]
```

but when the user's token is passed on as the authentication-header header, all three article objects are returned.

Then, if we perform this sequence of actions:

- create a new user;
- log the user in;
- call the GET /api/articles request again with a new token;

we should expect the following result:

```
[
  {
    "articles_id": "art1",
    "title": "tit1",
    "content": "c1",
    "visibility": "public",
    "user_id": "42"
  },
  {
    "articles_id": "art3",
    "title": "tit3",
    "content": "c3",
    "visibility": "logged_in",
    "user_id": "42"
  }
]
```

3. Example 3

An example of the headers that are included into a POST request sent to the /api/articles endpoint:

```
{
  "content-type": "application/json",
  "authentication-header": "bde6708a-0d14-48d7-9e30-ffc71d1d7667"
}
```