Assignment #5 Validation and Verification

Advanced Web Development

October 11th, 2023

Hipolito Bautista

Group M/W

Mr. Lester Pech

University of Belize

Endpoints Overview:

- GET /applicants (Collection)
- GET /applicants/{id} (Singleton)
- POST /applicants (Singleton)
- PUT /applicants/{id} (Singleton)
- DELETE /applicants/{id} (Singleton)

These endpoints have been structured in a hierarchical manner. Access a group of applicants with the /applicants endpoint. To manipulate or access a specific applicant, use the /{id} suffix.

Prerequisites

Header Content: All API Requests are required to container a header with

- Content-Type set to application/json
- API_KEY - Prefix_Key_Checksum

Sample API_KEY:
awt_ab598017be4550411284958a4c2014ec56000457792e7b92307137b824a24d3
8_6ee865bfab165006ae13495fdcd6617c1c300c0b98aaa0b078e0ff52091a96ac

Request methods: The only supported request methods are

- GET
- POST
- PUT
- DELETE

Database Tables: For these endpoints to function, a database with a table named applicants is necessary. The table should have the following fields:

```
CREATE TABLE applicant (
  `id` INT PRIMARY KEY,
  `first_name` VARCHAR(255),
  `last_name` VARCHAR(255),
  `username` VARCHAR(255),
  `password` VARCHAR(255),
  `country` VARCHAR(255)
);
```

For key implementation, we also require four tables

```
CREATE TABLE applicant_key (
  `id` INT PRIMARY KEY,
  `owner_id` INT,
  `api_key` VARCHAR(255),
  `status` INT,
  FOREIGN KEY (`owner_id`) REFERENCES `applicant`(`id`)
);
CREATE TABLE permission (
  `id` INT PRIMARY KEY,
  `parent` VARCHAR(255),
  `resource` VARCHAR(255),
  `status` INT
);
CREATE TABLE applicant_key_permission (
  `id` INT PRIMARY KEY,
  `key_id` INT,
  `permission_id` INT,
  `method` VARCHAR(255),
  `status` INT,
  FOREIGN KEY (`key_id`) REFERENCES `applicant_key`(`id`),
  FOREIGN KEY (`permission_id`) REFERENCES `permission`(`id`)
);
```

Keys are made with the sha256 hash function. It hashes the userid, time, and secret to generate the key. The key checksum is made with the secret key “toast.”

Endpoint Details:

GET /applicants (Collection)

- Returns a collection of all registered applicants.
- Options:
- Order - “column” : “asc || desc”
- Paging - “start:”, “end:”
- Use case: Retrieve all the applicants in the system.

Sample Request - http://127.0.0.1:8036/applicants

Sample Body -

```
"order":{"first_name":"asc", "country":"desc"},
"paging": {"start":0, "end": 2}
```

Sample Response

```
   "rc": "1",
   "log": "Success",
   "applicants": [{
         "id": 7,
         "first_name": "Antonio",
         "last_name": "Vezey",
         "username": "avezey6",
            "password":
"$2a$04$1N69yd68UOJgVmzvgXAMHuJ2IdJUkjk.CjxMZjwRrP6lkFYFww3bK",
            "country": "Philippines"},
        {
            "id": 3,
            "first_name": "Betsey",
            "last_name": "Davie",
            "username": "bdavie2",
            "password":
"$2a$04$qXz5TbzXkiJY2kqe2MMMbO4Q6md87N5FMN7ZJ7oPqu3hIHAnofNEW
",
            "country": "China"}
    ]
```

GET /applicants/{id} (Singleton)

- Accepts an ID and returns details of the requested applicant.
- Use case: Fetching details of a specific applicant in the system.

Sample Request - http://127.0.0.1:8036/applicants/1

Sample Body - None

Sample Response -

```
    "rc": "1",
    "log": "Successfully retrieves user",
    "applicants": [
        {
            "id": 1,
            "first_name": "John",
            "last_name": "Doe",
            "username": "johndoe322",
            "password":
"$2y$10$BfBLUQ1b9t6DMejTcJ3IM.3pXHDGlWIcWV4zvnU6IMB9ObJijgWs6",
            "country": "USA"
        }
    ]
```

POST /applicants (Singleton)

- Adds a new applicant to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: id, first_name, last_name, country, username, password.
- Use case: Adding a new applicant to the system.

Sample Request - http://127.0.0.1:8036/applicants

Sample Body -

```
{
  "id": 123,
  "first_name": "John",
  "last_name": "Doe",
  "country": "USA",
  "username": "johndoeee123",
  "password": "strongpassword123"
}
```

Sample Response -

```
"rc": "1",
"log": "Successfully inputted user"
```

PUT /applicants/{id} (Singleton)

- Updates an existing applicant in the table using the ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: first_name, last_name, country, username, password.
- Use case: Updating details of an existing applicant in the system.

Sample Request - http://127.0.0.1:8036/applicants/123

Sample Body -

```
{
   "first_name": "John",
   "last_name": "Doe",
   "country": "USA",
   "username": "UpdatingThisUsername",
   "password": "strongpassword123"
}
```

Sample Response -

```
"rc": "1",
"log": "Successfully updated user"
```

DELETE /applicants/{id} (Singleton)

- Deletes an applicant from the collection using the specified ID.
- Use case: Removing an existing applicant from the system.

Sample Request - http://127.0.0.1:8036/applicants/123

Sample Body - None

Sample Response -

```
"rc": "1",
"log": "Successfully deleted applicant with ID: 123"
```

Error Codes

Database Connection Error:
Error message: "Error connecting to MySQL Error: [error message]"
Description: This happens when there is an error connecting to the MySQLDB.

Invalid API Key:
Error message: "api key not present in request"
Description: This happens when the API key is not provided in the request headers. As outlined in the Prerequisites, every request must has an API key in it's header.

Invalid API Key Format:
Error message: "invalid api key format"
Description: The format of the API key is incorrect. Keys should follow the format of Prefix_Key_Checksum.

Invalid API Key Checksum:
Error message: "invalid api key checksum."
Description: The calculated checksum doesn't match the provided checksum in the API key.

API Key Not Found:
Error message: "api key not found in db."
Description: The API key in the header is not found in the database.

API Key Has No Permissions:
Error message: "api key has no permissions."
Description: The API key is valid but has no permissions assigned to it.

Redis Connection Error:
Error message: "unable to connect to redis"
Description: Error connecting to the Redis server.

Redis Authentication Failed:
Error message: "Redis auth failed"

Description: Authentication to Redis server fails, more than likely because of wrong credentials.

Redis Object Creation Failed:
Error message: "redis object creation failed."
Description: Indicates an error during the creation of a Redis object.

Invalid Student ID:
Error message: "Invalid Student ID, Must be positive"
Description: Occurs when an invalid negative or non-numeric student ID is provided.

Invalid Resource Request:
Error message: "unknown resource request"
Description: Occurs when the requested resource is not recognized.

Permission Denied:
Error message: "Permission denied to resource"
Description: The api key does not have permission to access the requested resource.

Method Not Allowed:
Error message: "Method not allowed"
Description: The HTTP method used is not allowed.

Invalid User Data:
Error message: "Invalid [field] data"
Description: Indicates that the user data provided is invalid.

No Data Provided:
Error message: "No data provided."
Description: Occurs when no data is provided in the request.
Unable to Input:

Error message: "Unable to input"
Description: There's an issue with inserting data into the database.

Unable to Update User:
Error message: "Unable to update user"
Description: There was an error updating user data in the database.
Unable to Delete User:
Error message: "Unable to delete user"
Description: Occurs when there's an issue with deleting a user from the db.