THE TWIS REST API

INTRODUCTION:

The Twis Rest API is a project that was built to provide a Restful API to the future TWIS Mobile and Desktop applications.

The system uses JSON as the serialization and deserialization technique. In this document, we will look at the steps and requirements needed to use to the API properly.

The rest API is hosted on URL: **http:// 188.166.226.100**. In this documentation, we will refer to "**domain**" mentioning **http:// 188.166.226.100.**

TABLE OF CONTENTS

THE STRUCTURE AND THE MAIN STEPSs:

This rest API is subdivided into two main parts:

1. The Mobile account part, which contains all the endpoints that will help developers to create functions and forms for mobile user to interact with the API.
   a. A mobile user should first create an account.
   b. A mobile user then will be able to access the library.
   c. A mobile user can the make a test on a given book and get their average.
   d. Mobile user can get their reports by providing a date interval.
2. The School account part, which contains all the endpoints that will help developers to create functions and forms for a school account.
   a. A school manager/librarian should first create an account.
   b. A school manager/librarian gets a code to activate desktop subscriptions.
   c. Students are to be recorded.
   d. Students can access the library using the school desktops that were authenticated.
   e. A student can the make a test on a given book and get their average.
   f. Students can get their reports by providing a date interval.

Each endpoint is documented as the following:

1. The title of the Endpoint.
2. Pre request/ request that needs to be made before the actual request.
3. A description of the endpoint.
4. The URL to the endpoint.
5. An example of a request to the endpoint.

Endpoints:

Common endpoints:

1. POST/ Create User's Account:

   This is generally the endpoint that both mobile accounts and schools managers will be using for authentication.

   URL: domain/api/auth/create-user/

   Example:

   POST domain/api/auth/create-user/

   Headers: Nothing in the headers

   Body/JSON : `{"email":"mugisha@gmail.com", "password":"810Iradu"}`

   Response status 200 (when everything is ok):

   Response :

```
{

"user": {
       "email": "mugisha@gmail.com",
       "last_login": null,
       "date_joined": "2020-12-03T08:06:54.139945Z",
       "is_active": true
       },
"token": "c0ec7c1f2888094fd516fc471462cdd2e34913a6"
}
```

2. POST/ User Login endpoint
   This is an endpoint that is used when a user already have an account, and wants to login with a different device.

   URL: domain/api/auth/login-user/

   Example:

   POST/ domain/api/auth/login-user/

   Headers: There is nothing in the headers.

   Body/JSON: `{"email":"iradupat@yahoo.fr", "password":"810Iradu"}`

   Response status 200 (when everything is ok):

Response:

```
{
    "user": {
        "email": "iradupat@yahoo.fr",
        "last_login": "2020-12-03T10:16:04.328498Z",
        "date_joined": "2020-11-30T12:01:48.317290Z",
        "is_active": true
    },
    "token": "e3a197de27c1736bd22642f51a64538cf009ec34"
}
```

Mobile user's endpoints:

The mobile user's endpoints, were built for mobile accounts only, and they are the following:

1. GET/ List available Mobile plans

   This is an endpoint that lists out all the mobile plans in the system. It is an open endpoint, no authentication is required for this endpoint.

   This is the endpoint you can use to select a plan for a user.

   URL: domain/api/mobile/plan/

   Example:

   GET/domain/api/mobile/plan/

   Header: Nothing in the headers.

   Body/JSON: Nothing in the body

   Response status 200 :

   Response:

```
{
"plans": [
           {
               "id": 1,
               "plan_duration": "1 Month",
               "price": "1000.00"
           },
           {
               "id": 2,
               "plan_duration": "3 Months",
               "price": "1300.00"
          }

       ]
```

```
                }
```

2. POST/ Create Mobile account

   This is an endpoint that is there to create a mobile account for the created user.

   You cannot make a mobile account before you register for a user account.

   PREREQUEST: logged in or created account.

   URL: domain/api/mobile/register/

   Example:

   POST/ domain/api/mobile/register/

   Header: “Authorization”: “Token ${token_value}”

   “ token_value is the token gotten from the login or registration endpoint”

   Body/JSON:

```
        {
                "plan_id":1,
                "first_name": "first_name",
                "last_name": "last name",
                "class_level":"3"
        }
```

   “ plan_id from the mobile plans list api, class_level ranges from 1 to 6”

   Response status 200 (When everything is ok):

   Response:

```
        {
        "message": "The account was created successfully. \n The plan will be active aft
er payment"
        }
```

3. PUT/ Validate mobile plan(To be updated)

   This endpoint might change depending on the vendor’s response.

   URL: domain/api/books/mobile/register/

   Example:

PUT/ domain/api/books/mobile/register/

Headers: No headers

Body/JSON: `{"token": "skskahsh87aha8hd"}`

Response status (200) // sent to the vendors api

4. GET/ Access the library on your phone
   This endpoint displays a list of available books after checking the authenticity of the account logged in and checking if the plan is valid (User has paid and the expiration date is not reached).

   PREREQUEST: logged in or created account, validated the plan.

   URL: domain/api/books/mobile-user/

   Example:

   GET/ domain/api/books/mobile-user/

   Headers: "Authorization": "Token ${valid_token}"

   "valid token from login or create account."

   Response status 200 (When everything is ok):

   Response:

```
{
    "books": [
        {
            "id": 2,
            "title": "Book2",
            "description": "Book2",
            "file_url": "/books/Le_petit_Nicolas_et_les_copains__PDFDrive.com_.p
df",
            "class_level": "3",
            "author": "Patrick"
        },
       {
            "id": 3,
            "title": "Book3",
            "description": "Book3",
            "file_url": "/books/Le_petit_Nicolas_et_les_copains__PDFDrive.com_.p
df",
            "class_level": "4",
            "author": "Patrick"
        },

    ]
}
```

5. GET/ get a test for a mobile user on a given book.

   This is the endpoint that will show to the mobile account user, the test on the book that he/she finished reading. The book id is required to make a test.

   PREREQUEST: logged in or created account.

   URL: domain/api/book/mobile/test/${book_id}

   Example:

   GET/ domain/api/book/mobile/test/${book_id}

   Headers: "Authorization": "Token ${valid_token}"

   "valid token from login or create account."

   Response status 200 (When everything is ok):

   Response:

```
{
    "test": {
        "id": 17,
        "book": {
            "title": "Book2",
            "question_set": [
                {
                    "description": "What is an API",
                    "answer_set": [
                        {
                            "id": 6,
                            "answer": "It is a programing interaface",
                            "is_true": true,
                            "question": 3
                        },
                        {
                            "id": 5,
                            "answer": "It is a language",
                            "is_true": false,
                            "question": 3
                        }
                    ]
                },
                {
                    "description": "What is Django",
                    "answer_set": [
                        {
                            "id": 4,
                            "answer": "A programing language",
                            "is_true": false,
                            "question": 2
                        },
                        {
                            "id": 3,
                            "answer": "A webframework",
                            "is_true": true,
                            "question": 2
                        }
                    ]
                }
            ]
        },
        "date_answered": "2020-12-05T15:00:39.941941Z",
        "avg": 0,
        "passed": false,
        "student": null,
        "mobile_user": 7
    }
}
```

“ remark that the key passed is False and the average is 0”

6. POST/ Mobile user passes the test

The mobile user, responds to the test questions and get their average on a give book.

PREREQUEST: The get a test for a mobile user request.

URL: domain/api/test/answer/${test_id}
“ test_id obtained from the get test for a mobile user request”

Headers: “Authorization”: ”Token ${valid_token}”

“valid token from login or create account.”

Example:

POST/ domain/api/test/answer/${test_id}

Body/JSON:

```
[

        {
         "answer_id":2
        },
        {
         "answer_id":6
        }
]
```

“ send an array of selected answer ids ”

Response status 200 (When everything is ok):

Response:

```
{
    "test": {
        "id": 17,
        "book": {
            "title": "Book2",
            "question_set": [
                {
                    "description": "What is an API",
                    "answer_set": [
                        {
                            "id": 6,
                            "answer": "It is a programing interaface",
                            "is_true": true,
                            "question": 3
                        },
                        {
                            "id": 5,
                            "answer": "It is a language",
                            "is_true": false,
                            "question": 3
                        }
                    ]
                },
                {
                    "description": "What is Django",
                    "answer_set": [
                        {
                            "id": 4,
                            "answer": "A programing language",
                            "is_true": false,
                            "question": 2
                        },
                        {
                            "id": 3,
                            "answer": "A webframework",
                            "is_true": true,
                            "question": 2
                        }
                    ]
                }
            ]
        },
        "date_answered": "2020-12-05T15:00:39.941941Z",
        "avg": 2,
        "passed": true,
        "student": null,
        "mobile_user": 7
    }
}
```

“ remark that the avg is now 2/2 and passed is true”

7. POST/Update mobile plan

   This is the endpoint that mobile users will have to use in order to update their plan.

   URL: domain/api/mobile/plan/update/

   Headers: “Authorization”: ”Token ${valid_token}”

   “valid token from login or create account.”

   Example:

   POST/ domain/api/mobile/plan/update/

   Body/JSON:

   ```
   {"plan_id":1}
   ```
   “ The id of the plan to register to”

   Response status 200 (when everything is ok):

   ```
   {
       "message": "The plan was updated successfully. The plan will be active after paymen
   t"
   }
   ```

   “After this, user should wait for the vendor to send the token on the validate plan endpoint. ”

School Endpoints:

This are endpoints that developers will use to make request on school desktop applications.

1. Register a school for the first time
   This the endpoint to create a school manager’s account and add a plan.

   PREREQUEST: login or create account requests

   URL: domain/api/register/school/

   Headers: “Authorization”: ”Token ${valid_token}” *"valid token from login or create account."*

Example:

POST/ domain/api/register/school/

Body/JSON:

```
{
   "plan_id":1,
   "school_name": "school name here",
   "devices":50
}
```

"school name must be unique, devices is the number of computers they will use, plan id"

Response status 200 (when everything is ok):

Response:

```
{
       message": "School created. The plan will be activated after payment",
       "school_id": 11,
       "code": "PwzjB4s4zS",
       "devices": 50,
       "unity_price": 1000.0,
       "total_amount": 50000.0
}
```

"The code value will be used to activate a computer and have access to a plan."

2. POST/ Validate school plan

This is a callback endpoint and it might change depending on the vendor's response.

URL: domain/api/school/plan/validate/${school_id}

Example:

PUT/ domain/api/school/plan/validate/${school_id}

Headers: No headers

Body/JSON: `{"token": "skskahsh87aha8hd"}`

Response status (200) // sent to the vendors api.

3. POST/ Connect a PC using the school code.

   This will connect a desktop by getting the authentication token of the school manager.

   PREREQUEST: login or register

   URL: domain/api/ school/add/device/${code}
   “code is the school code obtained in the registration process”

   Headers: “Authorization”: ”Token ${valid_token}”

   “valid token from login or create account.”

   Example:

   POST/ domain/api/school/add/device/${code}

   Response status 200 (When everything is ok)

   Response:

```
{
    "message": " Computer added to the list, remaining computers 49",
    "token": "aa593ce58b230b5c9e14f3328bad7371162fd4e3",
    "remaining_devices": 49
}
```
   “ remark that the number of remaining devices reduces as you make a request.”

4. Create a student account

   URL: domain/api/school/record-student/

   Headers: “Authorization”: ”Token ${valid_token}”

   “valid token from login or create account.”

   Example:

   POST/ domain/api/school/record-student/

   Body/JSON:

```
{
    "first_name":"{{first_name}}",
    "last_name": "{{last_name}}",
    "class_level":3
}
```

Response status 200 (when everything is ok)

Response :

```
{
    "students": [
        {
            "id": 11,
            "first_name": "Patrick 1607183588931",
            "last_name": "Iradu 1607183588931",
            "class_level": "3",
            "school": 11
        }
    ]
}
```

“ returns a list of students in the same class level”

5. POST/ Record students using an XL file

   This endpoint sends an EXCEL file where the first column is a list of first names, the second column is a list of second names and the third column contains the student’s class level ranging between 1 and 6.

   URL: domain/api/school/record-students/

   Headers: “Authorization”: “Token ${valid_token}”

   “valid token from login or create account.”

   Example:

   POST/ domain/api/school/record-students/

   Response status 200 (When everything is ok)

   Response:

6. GET/Get students from a level

   This is the endpoint you could use to display students from primary 2 or 3 etc..

   URL: domain/api/school/record-student/

   Headers: “Authorization”: ”Token ${valid_token}”

   “valid token from login or create account.”

   Example:

   GET/ domain/api/school/record-student/

   Body/JSON:

   ```
   {"level": 3}
   ```

   Response status 200 (When everything is ok)

   Response:

   ```
   {
       "students": [
           {
               "id": 11,
               "first_name": "Patrick 1607183588931",
               "last_name": "Iradu 1607183588931",
               "class_level": "3",
               "school": 11
           }
       ]
   }
   ```

7. GET/ student access the library

   The endpoint that a student should use to display books in the computer.

   URL: domain/api/books/student/${student_id}

   Headers: “Authorization”: ”Token ${valid_token}”

   “valid token from login or create account.”

Example :

GET/ domain/api/books/student/${student_id}

Response status 200 (When everything is ok)

Response:

```
{
    "books": [
        {
            "id": 2,
            "title": "Book2",
            "description": "Book2",
            "file_url": "/books/Le_petit_Nicolas_et_les_copains__PDFDrive.com_.pdf"
,
            "class_level": "3",
            "author": "Patrick"
        }
    ]
}
```

"Returns a list of books for the student's class level."

8. GET/Student gets a test

After reading a book, students should get their test on this endpoint.

PREREQUEST: display book endpoint and device must have been added to the plan.

URL: domin/api/book/review/test/${book_id}/${student_id}

Headers: "Authorization": "Token ${valid_token}"

"valid token from login or create account."

Example:

GET/ domain/api/book/review/test/${book_id}/${student_id}

Response status 200 (when everything is ok)

Response:

```
{
    "test": {
        "id": 18,
        "book": {
            "title": "Book2",
            "question_set": [
                {
                    "description": "What is an API",
                    "answer_set": [
                        {
                            "id": 6,
                            "answer": "It is a programing interaface",
                            "is_true": true,
                            "question": 3
                        },
                        {
                            "id": 5,
                            "answer": "It is a language",
                            "is_true": false,
                            "question": 3
                        }
                    ]
                },
                {
                    "description": "What is Django",
                    "answer_set": [
                        {
                            "id": 4,
                            "answer": "A programing language",
                            "is_true": false,
                            "question": 2
                        },
                        {
                            "id": 3,
                            "answer": "A webframework",
                            "is_true": true,
                            "question": 2
                        }
                    ]
                }
            ]
        },
        "date_answered": "2020-12-05T16:17:33.375739Z",
        "avg": 0,
        "passed": false,
        "student": 11,
        "mobile_user": null
    }
}
```

" remark that the average is 0 and passed is false."

9. Send student's answers

After responding to the questions, send back response.

PREREQUEST: Student should get the test first.

URL: domain/api/test/answer/${test_id}

"test_id from the previous request"

Headers: "Authorization": "Token ${valid_token}"

"valid token from login or create account."

Example:

POST/ domain/api/test/answer/${test_id}

Body/JSON:

```
[
        {
                "answer_id":2
        },
        {
                "answer_id":4
        }
]
```

"an array of answer ids"

Response status 200 (When everything is ok):

Response:

```
{
"test": {
        "id": 18,
        "book": {
        "title": "Book2",
        "question_set": [
                {
                "description": "What is an API",
                    "answer_set": [
                            {
                            "id": 6,
                            "answer": "It is a programing interaface",
                            "is_true": true,
                            "question": 3
                            },
                            {
                            "id": 5,
                            "answer": "It is a language",
                            "is_true": false,
                            "question": 3
                            }
                    ]
                    },
                    {
                    "description": "What is Django",
                    "answer_set": [
                            {
                            "id": 4,
                            "answer": "A programing language",
                            "is_true": false,
                            "question": 2
                            },
                            {
                            "id": 3,
                            "answer": "A webframework",
                            "is_true": true,
                            "question": 2
                            }
                    ]
                    }
            ]
            },
            "date_answered": "2020-12-05T16:17:33.375739Z",
            "avg": 1,
            "passed": true,
            "student": 11,
            "mobile_user": null
    }
    }
```

“The avg is ½ now and passed is true”

10. GET/ Get student’s repots

This endpoint receive a starting date and an ending date and generate a report of test passed by the student (the same thing for the mobile user).
(Date format: YYYY-MM-DD Ex: 2020-12-02).

URL: domain/api/report/get/

Headers: “Authorization”: ”Token ${valid_token}”

“valid token from login or create account.”

Example:

GET/ domain/api/report/get/

Body/JSON:

```
{"user_type":"STUDENT", "user_id":{{student_id}},
"start_date":"2020-10-11", "stop_date":"2020-12-12"}
```

“user_type can be STUDENT or MOBILE for mobile users”

Response status 200 (When everything is ok)
Response:

```
{
    "test": [
        {
            "id": 18,
            "book": {
                "title": "Book2",
                "question_set": [
                    {
                        "description": "What is an API",
                        "answer_set": [
                            {
                                "id": 6,
                                "answer": "It is a programing interaface",
                                "is_true": true,
                                "question": 3
                            },
                            {
                                "id": 5,
                                "answer": "It is a language",
                                "is_true": false,
                                "question": 3
                            }
                        ]
                    },
                    {
                        "description": "What is Django",
                        "answer_set": [
                            {
                                "id": 4,
                                "answer": "A programing language",
                                "is_true": false,
                                "question": 2
                            },
                            {
                                "id": 3,
                                "answer": "A webframework",
                                "is_true": true,
                                "question": 2
                            }
                        ]
                    }
                ]
            },
            "date_answered": "2020-12-05T16:17:33.375739Z",
            "avg": 1,
            "passed": true,
            "student": 11,
            "mobile_user": null
        }
    ],
    "from": "2020-10-11",
    "to": "2020-12-12"
}
```
“ returns a list of tests passed and the date range”

Other Endpoints:

1. POST/ Pay plan using MOMO
   This is the endpoint that will be used to pay the amount specified in the plan, for both Mobile users and School managers willing to use MOMO while paying for the activation of their plan.

   PREREQUEST: updating a plan or registering to the system are the prerequisites for this action.

   URL for Mobile users: domain/api/mobile/pay/
   URL for Schools: domain/api/school/pay/

   Headers: "Authorization": "Token ${valid_token}"

   "valid token from login or create account."

   Example:

   Mobile user

   POST/ domain/api/mobile/pay/

   Body /JSON:

```
{"phone": "0789421906"}
```

   School manager

   POST/ domain/api/school/pay/

   Body /JSON:

```
{"phone": "0789421906"}
```

2. POST/ Record students using an EXCEL File.

   This endpoint receives a file and record students from the file.

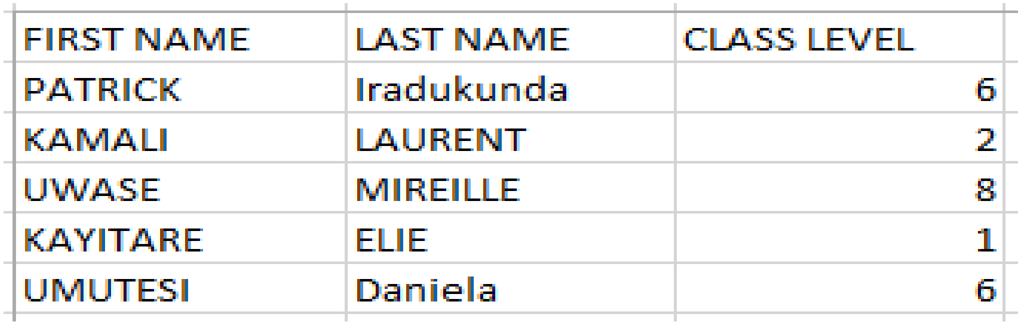
   PREREQUEST: Manager should have logged in or created an account for the school.

   URL: domain/api/school/record-students/

   Headers: “Authorization”: ”Token ${valid_token}”

   “valid token from login or create account.”

   The file format:



FIRST NAME	LAST NAME	CLASS LEVEL
PATRICK	Iradukunda	6
KAMALI	LAURENT	2
UWASE	MIREILLE	8
KAYITARE	ELIE	1
UMUTESI	Daniela	6

   The file is made of three columns and at the beginning of each column, there is a title “Firstname, lastname etc..”. NB: The system will by default skip the first row, do not record any student’s records on the first row.

   Example:

   POST/ domain/api/school/record-students/

   Body/ FormData:
   Key: file
   Value: students_list.xlsx

3. PUT / Update students' details

   The endpoint to update a student's information manually.

   PREREQUESITS: School manager should be logged in first.

   URL: domain/api/school/update-student-info/${student_id}

   Headers: "Authorization": "Token ${valid_token}"

   "valid token from login or create account."

   Example:

   Body/ JSON:

```
{"first_name": "Marita", "class_level":2, "last_name":"Mujawamariya"}
```

Status Responses:

Status 200: Everything is ok

Returns a JSON response

Status 400: Bad request in the request body

Returns a JSON object with a key "error": followed by errors to correct.

Status 404: When the resource is not found

Returns a JSON response with key "error": with a message inside.

Status 401: when trying to access resource without having the privilege for that.

Returns a JSON response with a key of "error": with a message inside.

Credentials and accounts

Opay

Email: twisomere@gmail.com

Pwd: Opay@123

Digital Ocean

ssh   yves@188.166.226.100

Pwd: 2020yves

API Super admin credentials

Email: twis@gmail.com

Pwd: 2020yves