Final Project - Systems admin

Systems Admin

November 23rd, 2023

Hipolito Bautista

Mr. Umana

University of Belize

Table Of Contents

Overview

The Following document outlines usage, constraints, and endpoints of the API. Usage and contains include accepted Content-Type, http methods, endpoints, rate limit, and API Key usage. The API's sole purpose is to facilitate the submission of student loans via a Loan portal.

API Rate Limiting
This API provides the user with a limit of 40 requests every 60 seconds. Exceeding this limit will result in the request being denied. Details about the application's rate limit can be found within the header, specifically in the following fields.

X-Rate-Limit-Limit - Field which displays the maximum number of requests.
X-Rate-Limit-Remaining - Remaining the number of requests before meeting the limit.
X-Rate-Limit-Used - Number of requests Sent
X-Rate-Limit-Reset - Amount of time until rate limit resets.

Endpoints Overview:
GET /applicants (Singleton)
GET /applicants/{id} (Singleton)
POST /applicants (Singleton)
PUT /applicants/{id} (Singleton)
DELETE /applicants/{id} (Singleton)

GET /applications (Singleton)
GET /applications/{id} (Singleton)
POST /applications (Singleton)
PUT /applications/{id} (Singleton)
DELETE /applications/{id} (Singleton)

GET /applicantverification/{id} (Singleton)
POST /applicantverification (Singleton)
PUT /applicantverification/{id} (Singleton)
DELETE /applicantverification/{id} (Singleton)

Endpoints Overview - Continued
GET /contactinfo/{id} (Singleton)
POST /contactinfo (Singleton)
PUT /contactinfo/{id} (Singleton)
DELETE /contactinfo/{id} (Singleton)

GET /financialprofile/{id} (Singleton)
POST /financialprofile(Singleton)
PUT /financialprofile/{id} (Singleton)
DELETE /financialprofile/{id} (Singleton)

GET /futureacademicdetails/{id} (Singleton)
POST /futureacademicdetails(Singleton)
PUT /futureacademicdetails/{id} (Singleton)
DELETE /futureacademicdetails/{id} (Singleton)

GET /institution/{id} (Singleton)
POST /institution(Singleton)
PUT /institution/{id} (Singleton)
DELETE /institution/{id} (Singleton)

GET /pastacademicdetails/{id} (Singleton)
POST /pastacademicdetails(Singleton)
PUT /pastacademicdetails/{id} (Singleton)
DELETE /pastacademicdetails/{id} (Singleton)

GET /reference/{id} (Singleton)
POST /reference(Singleton)
PUT /reference/{id} (Singleton)
DELETE /reference/{id} (Singleton)

GET /studycost/{id} (Singleton)
POST /studycost (Singleton)
PUT /studycost/{id} (Singleton)
DELETE /studycost/{id} (Singleton)

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

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_ab598017be4550411284958a4c2014ec56000457792e7b92307137b824a24d38_6ee865bfab165006ae13495fdcd6617c1c300c0b98aaa0b078e0ff52091a96ac

Request methods: The only supported request methods are

- GET
- POST
- PUT
- DELETE

Payload and response: Some endpoints can accept bodies, which must be raw and in a json fomat. Responses are returned to the user in a json format.

User Account - For Testing

This application uses sessions to avoid unauthorized access to any endpoints or pages. Due to this, a test account for a user has been made available for testing.

```
"username": "johndoeee123",
"password": "strongpassword123"
```

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"
```

GET /applicantverification/{id} (Singleton)

- Accepts a FormId and applicant verification section of that form.
- Use case: I want to access applicant verification info directly.

```
//Sample Request
"http://127.0.0.1:8036/applicantverification/1"

//Sample Body
none

//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieved academic details associated with
the provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "Proof_Of_Address": "",
            "ID_Type": "Passport",
            "ID_Number": 92839,
            "ID_Pic": null
        }
    ]
}
```

POST /applicantverification (Singleton)

- Adds a new applicant verification section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, Proof_Of_Address, ID_Type, ID_Number, ID_Pic
- Use case: Adding a new applicant verification section to the system.

```
//Sample Request
"http://127.0.0.1:8036/contactinfo"
//Sample Body
{
    "Form_ID": 2,
    "Proof_Of_Address": "Base64EncodedBlobData1",
    "ID_Type": "Passport",
    "ID_Number": 456789,
    "ID_Pic": "Base64EncodedBlobData2"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inserted Verification Data"
}
```

PUT /applicantverification/{id} (Singleton)

- Updates an existing applicant verification section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Proof_Of_Address, ID_Type, ID_Number, ID_Pic
- Use case: Updating details of an existing application verification section in the system.

```
//Sample Request
"http://127.0.0.1:8036/pastacademicdetails/1"
//Sample Body
{
    "Proof_Of_Address": "Base64EncodedBlobData1",
    "ID_Type": "Social Security",
    "ID_Number": 456789,
    "ID_Pic": "Base64EncodedBlobData2"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inserted Verification Data"
```

DELETE /applicantverification/{id} (Singleton)

- Deletes an applicant verification entry who's form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing applicant verification section from the system.

```
//Sample Request
"http://127.0.0.1:8036/applicants/123"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted applicant verification with ID: 1"
}
```

GET /contactinfo/{id} (Singleton)

- Accepts a FormId and returns the contact info section of that form.
- Use case: I want to access contact info directly.

```
//Sample Request
"http://127.0.0.1:8036/contactinfo/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieves contact info associated with the
provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "HomePhone": "555-123-4567",
            "WorkPhone": "555-987-6543",
            "CellPhone": "555-555-5555",
            "Other": "Some other contact information",
            "Email": "example@example.com"
        }
    ]
}
```

POST /contactinfo (Singleton)

- Adds a new contact info section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, HomePhone, WorkPhone, CellPhone, Other, Email
- Use case: Adding a new contact info section to the system.

```
//Sample Request
"http://127.0.0.1:8036/contactinfo"
//Sample Body
{
    "Form_ID": 1,
    "HomePhone": "555-123-4567",
    "WorkPhone": "555-987-6543",
    "CellPhone": "555-555-5555",
    "Other": "Some other contact information",
    "Email": "example@example.com"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inputted data into contact table"
}
```

PUT /contactinfo/{id} (Singleton)

- Updates an existing contact info section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: HomePhone, WorkPhone, CellPhone, Other, Email
- Use case: Updating details of an existing contact info section in the system.

```
//Sample Request
"http://127.0.0.1:8036/contactinfo/2"
//Sample Body
{
    "HomePhone": "555-123-4567",
    "WorkPhone": "555-987-6543",
    "CellPhone": "555-555-5555",
    "Other": "test",
    "Email": "example@example.com"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully updated data into contact table"
}
```

DELETE /contactinfo/{id} (Singleton)

- Deletes a contact info entry who's form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing contact info section from the system.

```
//Sample Request
"http://127.0.0.1:8036/contactinfo/2"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted contact info with ID: 2"
}
```

GET /financialprofile/{id} (Singleton)

- Accepts a FormId and returns the financial profile section of that form.
- Use case: I want to access the financial profile info directly.

```
//Sample Request
"http://127.0.0.1:8036/financialprofile/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieves financial profile associated with
the provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "living_status": "Own",
            "at_currently_address_yrs": 3,
            "at_currently_address_months": 6,
            "residents_in_home": 3,
            "dependents": 2,
            "employment_status": "Employed",
            "occupation": "Software Developer",
            "source_of_income": "Salary",
            "income_amount": 60000,
            "monthly_income": 5000,
            "total_household_income_annually": 75000
        }
    ]
}
```

POST /financialprofile (Singleton)

- Adds a new financial profile section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, living_status, at_currently_address_yrs, at_currently_address_months, residents_in_home, dependents, employment_status, occupation, source_of_income, income_amount, monthly_income, total_household_income_annually
- Use case: Adding a new financial profile section to the system.

```
//Sample Request
"http://127.0.0.1:8036/financialprofile"
//Sample Body
{
  "Form_ID": 2,
  "living_status": "Rent",
  "at_currently_address_yrs": 2,
  "at_currently_address_months": 8,
  "residents_in_home": 4,
  "dependents": 3,
  "employment_status": "Self-Employed",
  "occupation": "Teacher",
  "source_of_income": "Business",
  "income_amount": 75000,
  "monthly_income": 6000,
  "total_household_income_annually": 90000
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inputted financial data into
applicantfinancialprofile"
}
```

PUT /financialprofile/{id} (Singleton)

- Updates an existing financial profile section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: living_status, at_currently_address_yrs, at_currently_address_months, residents_in_home, dependents, employment_status, occupation, source_of_income, income_amount, monthly_income, total_household_income_annually
- Use case: Updating details of an existing financial profile section in the system.

```
//Sample Request
"http://127.0.0.1:8036/financialprofile/2"
//Sample Body
{
  "living_status": "Rent",
  "at_currently_address_yrs": 2,
  "at_currently_address_months": 8,
  "residents_in_home": 4,
  "dependents": 3,
  "employment_status": "Self-Employed",
  "occupation": "Tutor",
  "source_of_income": "Business",
  "income_amount": 75000,
  "monthly_income": 6000,
  "total_household_income_annually": 90000
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully updated financial data into
applicantfinancialprofile"
}
```

DELETE /financialprofile/{id} (Singleton)

- Deletes a financial profile entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing financial profile section from the system.

```
//Sample Request -
"http://127.0.0.1:8036/financialprofile/2"
//Sample Body -
none
//Sample Response -
{
    "rc": "1",
    "log": "Successfully deleted financial profile with ID: 2"
}
```

GET /futureacademicdetails/{id} (Singleton)

- Accepts a FormId and returns the Future academic details section of that form.
- Use case: I want to access Future academic details directly.

```
//Sample Request
"http://127.0.0.1:8036/futureacademicdetails/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieved academic details associated with
the provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "Degree_Type": "Bachelor of Science",
            "Major": "Computer Science",
            "Major_Duration": "4 years",
            "Financing_Start_Date": "2020-09-01",
            "Financing_End_Date": "2024-05-30",
            "Institution_Name": "University of Dummyville",
            "Location": "Dummyville, USA"
        }
    ]
}
```

POST /futureacademicdetails (Singleton)

- Adds a new Future academic details section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, Degree_Type, Major, Major_Duration, Financing_Start_Date, Financing_End_Date, Institution_Name, Location
- Use case: Adding a new Future academic details section to the system.

```
//Sample Request
"http://127.0.0.1:8036/futureacademicdetails"
//Sample Body
{
  "Form_ID": 1,
  "Degree_Type": "Bachelor of Science",
  "Major": "Computer Science",
  "Major_Duration": "4 years",
  "Financing_Start_Date": "2020-09-01",
  "Financing_End_Date": "2024-05-30",
  "Institution_Name": "University of Dummyville",
  "Location": "Dummyville, USA"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inserted Future Academic Data into Future
Academic Data table"
}
```

PUT /futureacademicdetails/{id} (Singleton)

- Updates an existing Future academic details section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Degree_Type, Major, Major_Duration, Financing_Start_Date, Financing_End_Date, Institution_Name, Location
- Use case: Updating details of an existing Future academic details section in the system.

```
//Sample Request
"http://127.0.0.1:8036/futureacademicdetails/2"
//Sample Body
{
  "Degree_Type": "Bachelor of Science",
  "Major": "Computer Science",
  "Major_Duration": "4 years",
  "Financing_Start_Date": "2020-09-01",
  "Financing_End_Date": "2024-05-30",
  "Institution_Name": "University of Dummyville",
  "Location": "Dummyville, USA"
}
//Sample Response
{
    "rc": "1",
    "log": "Successfully Updated Future Academic Data into Future
Academic Data table"
}
```

DELETE /futureacademicdetails/{id} (Singleton)

- Deletes a Future academic details entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing Future Academic Details section from the system.

```
//Sample Request
"http://127.0.0.1:8036/futureacademicdetails/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted future academicdata with ID: 1"
}
```

GET /institution /{id} (Singleton)

- Accepts a FormId and returns the institution section of that form.
- Use case: I want to access institution details directly.

```
//Sample Request
"http://127.0.0.1:8036/institution/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieves institution associated with the
provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "Address": "4 university heights",
            "District": "Cayo",
            "City_Town_Village": "Belmopan"
        }
    ]
}
```

POST /institution (Singleton)

- Adds a new institution section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, Address, District, City_Town_Village
- Use case: Adding a new institution section to the system.

```
//Sample Request
"http://127.0.0.1:8036/institution"
//Sample Body
{
    "Form_ID": 2,
    "Address": "123 Main Street",
    "District": "Sample District",
    "City_Town_Village": "Sample City"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inpputed institution data into institution
table"
}
```

PUT /institution /{id} (Singleton)

- Updates an existing institution section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Address, District, City_Town_Village
- Use case: Updating details of an existing institution section in the system.

```
//Sample Request
"http://127.0.0.1:8036/institution/2"
//Sample Body
{
    "Address": "123 Main Street",
    "District": "Belize",
    "City_Town_Village": "Sample City"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully updated institution data into institution
table"
}
```

DELETE /institution /{id} (Singleton)

- Deletes an institution entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing institution section from the system.

```
//Sample Request
"http://127.0.0.1:8036/institution/2"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted institution with ID: 2"
}
```

GET /pastacademicdetails/{id} (Singleton)

- Accepts a FormId and returns the Past academic details section of that form.
- Use case: I want to access Past academic details directly.

```
//Sample Request
"http://127.0.0.1:8036/pastacademicdetails/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieved academic details associated with
the provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "Recent_Education": "Master's",
            "Degree_Earned": "Master of Business Administration",
            "Qualification_Earned": "MBA in Finance"
        }
    ]
}
```

POST /pastacademicdetails (Singleton)

- Adds a new Past academic details section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, Recent_Education, Degree_Earned, Qualification_Earned
- Use case: Adding a new Past academic details section to the system.

```
//Sample Request
"http://127.0.0.1:8036/pastacademicdetails "
//Sample Body
{
    "Form_ID": 2,
    "Recent_Education": "Master's in Business Administration",
    "Degree_Earned": "Master of Business Administration",
    "Qualification_Earned": "MBA in Finance"
}
//Sample Response
{
    "rc": "1",
    "log": "Successfully inputted Past Academic Data into Past
Academic Data table"
}
```

PUT /pastacademicdetails/{id} (Singleton)

- Updates an existing Past academic details section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Recent_Education, Degree_Earned, Qualification_Earned
- Use case: Updating details of an existing Past academic details section in the system.

```
//Sample Request
"http://127.0.0.1:8036/pastacademicdetails/2"
//Sample Body
{
     "Recent_Education": "Master's",
    "Degree_Earned": "Master of Business Administration",
    "Qualification_Earned": "MBA in Finance"
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully updated Past Academic Data into Past
Academic Data table"
}
```

DELETE /pastacademicdetails/{id} (Singleton)

- Deletes a Past academic details entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing Past academic details section from the system.

```
//Sample Request
"http://127.0.0.1:8036/pastacademicdetails/1"
//Sample Body
none
//Sample Response -
{
    "rc": "1",
    "log": "Successfully deleted pastacademicdetails with ID: 1"
}
```

GET /reference/{id} (Singleton)

- Accepts a FormId and returns the reference section of that form.
- Use case: I want to access a reference section.

```
//Sample Request
"http://127.0.0.1:8036/reference/2"

//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieves reference associated with the
provided form ID",
    "applications": [
        {
            "Form_ID": 2,
            "name": "John Doe",
            "email": "johndoe@example.com",
            "phone_number": "123-456-7890",
            "relation": "Friend",
            "real_estate_collateral": 0
        }
    ]
}
```

POST /reference (Singleton)

- Adds a new references section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, name, email, phone_number, relation, real_estate_collateral
- Use case: Adding a references info section to the system.

```
//Sample Request
"http://127.0.0.1:8036/reference"
//Sample Body
{
    "Form_ID": 2,
    "name": "John Doe",
    "email": "johndoe@example.com",
    "phone_number": "123-456-7890",
    "relation": "Friend",
    "real_estate_collateral": 1
}
//Sample Response
{
    "rc": "1",
    "log": "Successfully references data into references table"
}
```

PUT /reference/{id} (Singleton)

- Updates an existing references section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, name, email, phone_number, relation, real_estate_collateral
- Use case: Updating details of an existing references section in the system.

```
//Sample Request
"http://127.0.0.1:8036/reference/2"
//Sample Body
{
    "name": "John Doe",
    "email": "johndoe@example.com",
    "phone_number": "123-456-7890",
    "relation": "Friend",
    "real_estate_collateral": 0
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully updated reference in references table"
}
```

DELETE /reference/{id} (Singleton)

- Deletes a reference entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing references section from the system.

```
//Sample Request
"http://127.0.0.1:8036/reference/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted reference with ID: 1"
}
```

GET /studycost/{id} (Singleton)

- Accepts a FormId and returns the study cost section of that form.
- Use case: I want to access study costs directly.

```
//Sample Request
"http://127.0.0.1:8036/studycost/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieves studycost associated with the
provided form ID",
    "applications": [
        {
            "Form_ID": 1,
            "P_Tuition_SchoolFee": 200,
            "P_Books_Supplies": 220,
            "P_Boarding_Lodging": 240,
            "P_Traveling": 260,
            "P_Expenses": 290,
            "P_total": 2600,
            "Tuition_SchoolFee": 100,
            "Books_Supplies": 120,
            "Boarding_Lodging": 140,
            "Traveling": 180,
            "Expenses": 1000,
            "total": 2500
        }
    ]
}
```

POST /studycost (Singleton)

- Adds a new study cost section to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: Form_ID, P_Tuition_SchoolFee, P_Books_Supplies, P_Boarding_Lodging, P_Traveling, P_Expenses, P_total, Tuition_SchoolFee, Books_Supplies, Boarding_Lodging, Traveling, Expenses, total
- Use case: Adding a new study cost section to the system.

```
//Sample Request
"http://127.0.0.1:8036/studycost"
//Sample Body
{
  "Form_ID": 2,
  "P_Tuition_SchoolFee": 5000,
  "P_Books_Supplies": 250,
  "P_Boarding_Lodging": 800,
  "P_Traveling": 100,
  "P_Expenses": 400,
  "P_total": 6950,
  "Tuition_SchoolFee": 4500,
  "Books_Supplies": 200,
  "Boarding_Lodging": 750,
  "Traveling": 80,
  "Expenses": 350,
  "total": 5900
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully inputted study costs "
}
```

PUT /studycost/{id} (Singleton)

- Updates an existing study cost section of the form in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: P_Tuition_SchoolFee, P_Books_Supplies, P_Boarding_Lodging, P_Traveling, P_Expenses, P_total, Tuition_SchoolFee, Books_Supplies, Boarding_Lodging, Traveling, Expenses, total
- Use case: Updating details of an existing study cost section in the system.

```
//Sample Request
"http://127.0.0.1:8036/studycost/2"
//Sample Body
{
  "P_Tuition_SchoolFee": 10000,
  "P_Books_Supplies": 2500,
  "P_Boarding_Lodging": 800,
  "P_Traveling": 100,
  "P_Expenses": 400,
  "P_total": 6950,
  "Tuition_SchoolFee": 4500,
  "Books_Supplies": 200,
  "Boarding_Lodging": 750,
  "Traveling": 80,
  "Expenses": 350,
  "total": 5900
}

//Sample Response
{
    "rc": "1",
    "log": "Successfully updated study costs"
}
```

DELETE /studycost/{id} (Singleton)

- Deletes a study cost entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing study cost section from the system.

```
//Sample Request
"http://127.0.0.1:8036/studycost/2"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted studycost with ID: 2"
}
```

GET /applications (Collection)

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

```
//Sample Request
"http://127.0.0.1:8036/applications"
//Sample Body
{
    "order":{"name":"asc"},
    "paging": {"start":0, "end": 2}
}

//Sample Response
{
    "rc": "1",
    "log": "Success",
    "applications": [
        {
            "ApplicantID": 1,
            "Form_ID": 1,
            "name": "Hipolito Bautista",
            "address": "2 baboon avenue",
            "area": "Tiger Town",
            "country": "Belize",
            "district": "Cayo",
            "city_town_village": "Belmopan",
            "DateOfBirth": "2002-09-17",
            "Age": 21,
            "nationality": "Belizean",
            "PEP_status": 1,
            "start_date": "2020-09-17",
            "end_date": "2024-09-17"
        },
        {
            "ApplicantID": 2,
            "Form_ID": 2,
            "name": "John Doe",
```

```
            "address": "123 Main Street",
            "area": "Suburbia",
            "country": "USA",
            "district": "Downtown",
            "city_town_village": "Smallville",
            "DateOfBirth": "1990-01-01",
            "Age": 30,
            "nationality": "American",
            "PEP_status": 1,
            "start_date": "2023-01-01",
            "end_date": "2023-12-31"
        }
    ]
}
```

GET /applications/{id} (Singleton)

- Accepts a FormId and returns the application summary of that form.
- Use case: I want to access application summaries directly.

```
//Sample Request
"http://127.0.0.1:8036/applications/1"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully retrieves application",
    "applications": [
        {
            "ApplicantID": 1,
            "Form_ID": 1,
            "name": "Hipolito Bautista",
            "address": "2 baboon avenue",
            "area": "Tiger Town",
            "country": "Belize",
            "district": "Cayo",
            "city_town_village": "Belmopan",
            "DateOfBirth": "2002-09-17",
            "Age": 21,
            "nationality": "Belizean",
            "PEP_status": 1,
```

```
            "start_date": "2020-09-17",
            "end_date": "2024-09-17"
        }
    ]
}
```

POST /applications (Singleton)

- Adds a new application summary to the table.
- Requirements:
    - Body in postman.
    - Body format: Raw
- Data: applicant_id, form_id, name, address, area, district, country, city_town_village, date_of_birth, age, nationality, pep_status, start_date, end_date, institution_id
- Use case: Adding a new application summary to the system.

```
//Sample Request
"http://127.0.0.1:8036/applications"
//Sample Body
{
    "applicant_id": 2,
    "form_id": 2,
    "name": "John Doe",
    "address": "123 Main Street",
    "area": "Suburbia",
    "district": "Downtown",
    "country": "USA",
    "city_town_village": "Smallville",
    "date_of_birth": "1990-01-01",
    "age": 30,
    "nationality": "American",
    "pep_status": true,
    "start_date": "2023-01-01",
    "end_date": "2023-12-31",
    "institution_id": 456
}
//Sample Response
{
    "rc": "1",
    "log": "Successfully inputted application"
}
```

PUT /applications/{id} (Singleton)

- Updates an existing application summary in the table using the form ID provided in the endpoint.
    - Requirements:
    - Body in postman.
    - Body format: Raw
- Data: applicant_id, form_id, name, address, area, district, country, city_town_village, date_of_birth, age, nationality, pep_status, start_date, end_date, institution_id
- Use case: Updating details of an existing application summary in the system.

```
//Sample Request
"http://127.0.0.1:8036/applications/2"
//Sample Body
{
    "name": "ronan James",
    "address": "123 Main Street",
    "area": "Suburbia",
    "district": "Downtown",
    "country": "USA",
    "city_town_village": "Smallville",
    "date_of_birth": "1990-01-01",
    "age": 30,
    "nationality": "American",
    "pep_status": true,
    "start_date": "2023-01-01",
    "end_date": "2023-12-31",
    "institution_id": 456
}
//Sample Response
{
    "rc": "1",
    "log": "Successfully updated application"
}
```

DELETE /applications/{id} (Singleton)

- Deletes an application summary entry whose form ID matches the Form ID provided in the endpoint.
- Use case: Removing an existing application summary from the system.

```
//Sample Request
"http://127.0.0.1:8036/applications/3"
//Sample Body
none
//Sample Response
{
    "rc": "1",
    "log": "Successfully deleted application with ID: 3"
}
```

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.