Bike tour api

How to use

## Purpose:

This document provides a comprehensive overview of the Bike tour API exposed features.

## Scope:

The scope of this document is to detail out the request & response pattern of 2 API’s

1. Login
2. Upload .GPXs files.

## **Login API**:

**URL** – <http://btapi.dexteriity.co.in/API/Login/AuthenticateUser>

### **Request**:

It accepts a data type “LoginRequestMessage” which is having below properties:

1. **LoginName**: it accepts “Alphanumeric” characters and it’s Mandatory Field.
2. **Password**: it accepts “Alphanumeric” characters and it’s Mandatory Field.

**Response**:

Response is returned as “LoginResponseMessage” data type which has below properties:

1. **SecurityToken** – Its value is “G” after successful login otherwise it’s empty.
2. **Success** – It has “true” after successful login otherwise its ‘false’.
3. **Error** – It contains list of errors if occurs. Below are errors which are handled:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **When** | **Error Code** | **Error Message** |
| 1. | Request is empty or null | 002 | Request message should not be empty. |
| 2. | LogiName is empty. | 003 | Login name is mandatory. |
| 3. | Password is empty. | 004 | Password is mandatory. |
| 4. | User does not exist/Active | 000 | Unable to find user. |
| 5. | System error | 999 | Showing as is error. |

## .Net Example:

public void AuthenticateUser()

{

string apiUrl = "http://btapi.dexterity.co.in/api/Login/AuthenticateUser";

var client = new HttpClient();

var values = new Dictionary<string, string>()

{

{"LoginName", "6"},

{"Password", "Skis"},

};

var content = new FormUrlEncodedContent(values);

var response = client.PostAsync(apiUrl, content).Result;

var success = response.EnsureSuccessStatusCode();

if (success.IsSuccessStatusCode)

{

string jsonMessage;

using (Stream responseStream = response.Content.ReadAsStreamAsync().Result)

{

jsonMessage = new StreamReader(responseStream).ReadToEnd();

}

}

}

## **Upload File API:**

**URL** – [http://btapi.dexteriity.co.in/ API/](http://btapi.dexteriity.co.in/BiketourAPI/API/)Upload/UploadFile

### **Request**:

It accepts a data type “UploadRequestMessage” which is having below properties/Fields:

1. **LoginName**: it accepts “Alphanumeric” characters and it’s Mandatory Field.
2. **Password**: it accepts “Alphanumeric” characters and it’s Mandatory Field.
3. **FileList**: its collection/array of data type “UploadFile”. We can upload as many as files we want. “UploadFile” data type has below properties/Fields:
   1. **FileName**: Its string type field and its mandatory field. This name will be used to create file at server.
   2. **FileData**: Its binary/byte array and its mandatory. This is used as file data.

**Response**:

Response is returned as “UploadResponseMessage” data type which has below properties.

1. **Error**: This contains generic errors like 002,003,004,000,999,013,014
2. **UploadFileStatus** – its collection/array of file status and errors (if any). It has following properties:
   1. **FileName** – file name supplied sent in request message.
   2. **Status** – true in case file uploaded successfully otherwise false.
   3. **Error** – It contains list of errors if occurs. Below are errors which are handled:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **When** | **Error Code** | **Error Message** |
| 1. | Request is empty or null | 002 | Request message should not be empty. |
| 2. | LogiName is empty. | 003 | Login name is mandatory. |
| 3. | Password is empty. | 004 | Password is mandatory. |
| 4. | User does not exist/Active | 000 | Unable to find user. |
| 5. | System error | 999 | Showing as is error. |
| 6. | Upload File list is empty | 005 | File List should not be empty. |
| 7. | File name is empty | 006 | File name is mandatory. |
| 8. | File data is empty | 007 | File data should not be empty. |
| 9. | File is already uploaded | 008 | File already uploaded! |
| 10. | File does not have time information | 009 | Uploaded File does not have Time Information, Please check the file and re-upload. |
| 11. | File Created before batch start date | 010 | File cannot be uploaded prior to Batch Start Date. |
| 12. | Average speed is 0 | 011 | Invalid File!. |
| 13. | File Uploaded Successfully | 012 | File Uploaded Successfully. |
| 14. | UploaderIsNotAStudent | 013 | Only Student can upload file. |
| 17. | SpeedLimitCrossed | 016 | Speed Limit Crossed! |

## .Net Example:

public void UploadValidFile()

{

string apiUrl = "http://localhost:49916//api//Upload//UploadFile";

var client = new HttpClient();

string filePath = @"D:\Projects\Nikhil\BikeTourWebAPI\GPXFile\";

var filedata = "'FileName':'leonie fahrrad.GPX',”

+

"'FileData':'" +

File.ReadAllBytes(filePath + "leonie fahrrad.GPX").ToString() +"'";

var values = new Dictionary<string, string>()

{

{"LoginName", "6"},

{"Password", "Skis"},

{"FileList", filedata},

};

var content = new FormUrlEncodedContent(values);

var response = client.PostAsync(apiUrl, content).Result;

var success = response.EnsureSuccessStatusCode();

if (success.IsSuccessStatusCode)

{

//Perform your operations

string jsonMessage;

using (Stream responseStream = response.Content.ReadAsStreamAsync().Result)

{

jsonMessage = new StreamReader(responseStream).ReadToEnd();

}

}

}