TutoringTrain Webservice

REST-DOKU

V1.3.1

Table of Contents

[General 2](#_Toc495784628)

[Methods 3](#_Toc495784629)

[[1] Security & Authentication 3](#_Toc495784630)

[1. Obtaining a session token 3](#_Toc495784631)

[2. Using the session token 3](#_Toc495784632)

[3. Validating the session token 4](#_Toc495784633)

[4. User roles 4](#_Toc495784634)

[[2] Error Handling and Localization 5](#_Toc495784635)

[[3] User Resource 6](#_Toc495784636)

[1. Register User 6](#_Toc495784637)

[2. Get all Users 6](#_Toc495784638)

[3. Get own User 7](#_Toc495784639)

[4. Update any User 8](#_Toc495784640)

[5. Update own User 8](#_Toc495784641)

[6. Block User 9](#_Toc495784642)

[7. Unblock User 9](#_Toc495784643)

[8. Get all Genders 10](#_Toc495784644)

[[4] Offer Resource 11](#_Toc495784645)

[1. Create Offer 11](#_Toc495784646)

[2. Update Offer 11](#_Toc495784647)

[3. Get newest Offers 12](#_Toc495784648)

[4. Get newest Offers of User 13](#_Toc495784649)

[[5] Subject Resource 14](#_Toc495784650)

[1. Create Subject 14](#_Toc495784651)

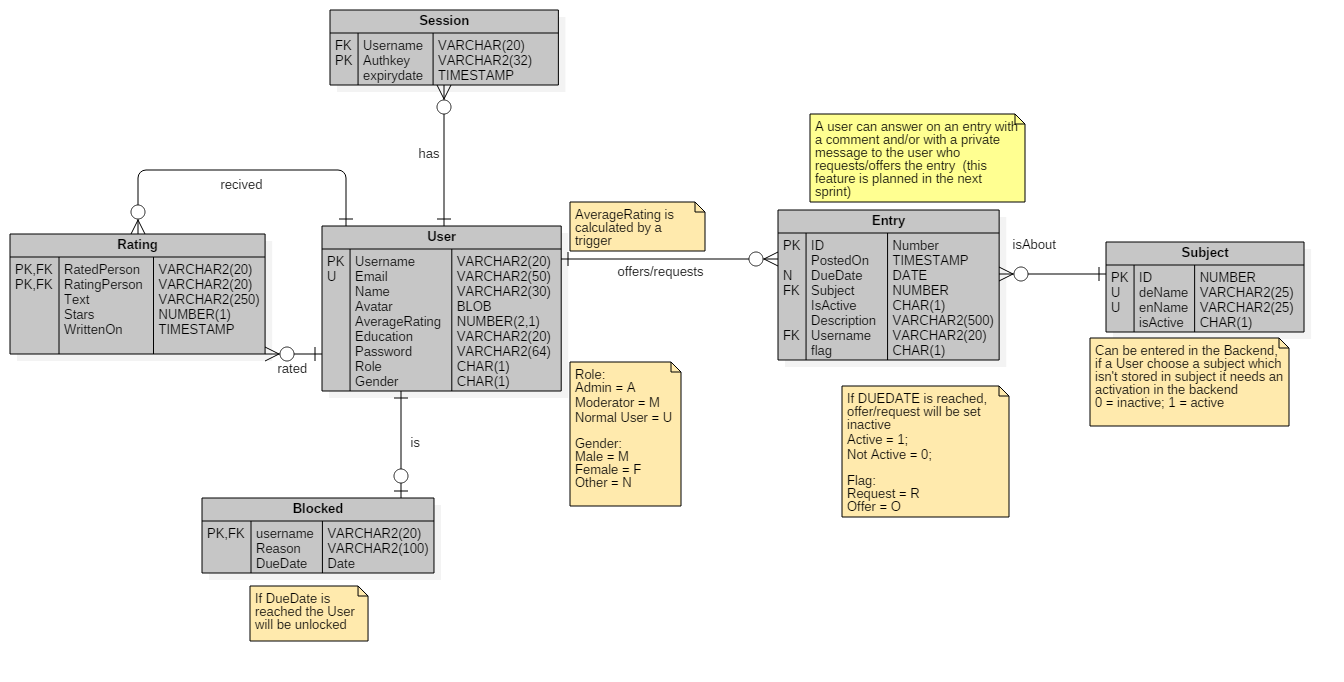
[2. Update Subject 14](#_Toc495784652)

[3. Delete Subject 14](#_Toc495784653)

[4. Get all Subjects 15](#_Toc495784654)

General

Data Model



**Only JSON format is supported.**

All JSON attributes marked in **THIS COLOR** are optional.

Dates format is: “**yyyy-MM-dd'T'hh:mm:ssZ**”  
Example: **2017-09-03T09:45:12+0200**

Methods

1. Security & Authentication

Authentication is managed via session tokens. This is a randomly generated 32-character string which is used to identify the logged in user.

1. Obtaining a session token

**Note:** A new session token is generated every time this method is called with valid credentials.

A token expires 24 hours after its creation.

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| POST | url/authentication | None |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "username": "testuser",  "password": "098f6bcd4621d373cade4e832627b4f6",  "requiredRole": "A"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | 654d4e57c9a868dd87759268b225ae44 |
| **401** | Password and/or username are invalid. |
| **403** | User cannot authenticate himself in current context. (eg. normal user tries to log in with admin tool 🡪 set requiredRole attribute for this to work) |
| **450** | User is currently blocked. (Reason is given in response body) |

1. Using the session token

Once a session token has been acquired it can be used to call all other methods. An exception is registering a User as this method does not require any authentication.

For every request the token has to be placed in the “Authorization”-Header in the following format:

“Bearer <token>”

**Example:**



A method called without a session token returns **401** HTTP status.

If **401** is returned when a session token was provided it can be assumed that the token has expired or is invalid. In this case you can try to acquire a new token.

If **403** is returned the user associated with the session key has insufficient privileges to perform that method.

If **450** is returned the user associated with the session key has been blocked.

If **451** is returned, the provided JSON-text is invalid. Further details might be given in the returned Error.

If **422** is returned, a provided value violated a constraint. Further details might be given in the returned Error.

If **500** is returned an unknown error happened. Further details might be given in the returned Error.

These response codes apply to all methods described below but are not mentioned there. Only method-specific response codes are listed.

1. Validating the session token

A session token can be validated using the method described below. This is useful since clients can use this feature to check whether a previously obtained, cached token is still valid.   
(Set token in Authorization header for this check)

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/authentication/check | User |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | The token is still valid |
| **200** | The token is not valid (see returned Error for further information) |

1. User roles

This System currently supports three roles:

User (U), Moderator (M) and Admin (A)

These roles extend the privileges of their inferior roles, this means a Moderator can perform all actions a User can, and an Admin can perform all actions a Moderator can.

1. Error Handling and Localization

In the event of an error / exception, an Error object of the following structure is returned:

{

"code": <error code>,

"message": <error message>

}

The following error codes exist:

|  |  |
| --- | --- |
| **Code** | **Descrption** |
| 1 | Unknown error – further details might be given in message |
| 2 | Admin only – happens when user without Admin role tries to authenticate when “requiredRole” is set to ‘A’ |
| 3 | Token invalid – the authentication token does not exist |
| 4 | Token expired – the authentication token has expired |
| 5 | Insufficient privileges – the user does not have to required role to perform this action |
| 6 | Username is null – No username was given |
| 7 | Name is null – No name was given (e.g. Create Subject) |
| 8 | User is null – No user was given |
| 9 | Offer is null – No offer was given |
| 10 | Gender is null – No gender was given |
| 11 | Subject is null – No subject was given |
| 12 | User not found – The user was not found |
| 13 | Subject not found – The subject was not found |
| 14 | Offer not found – The offer was not found |
| 15 | Gender not found – The gender was not found |
| 16 | Invalid email – The given email is of an invalid format |
| 17 | Query parameters missing – “start” and “pageSize” query parameters were not provided |
| 18 | Subject name conflict – The subject name already exists |
| 19 | Username conflict – The username is not available (already in use) |
| 20 | Email conflict – The email is not available (already in use) |
| 21 | Cannot block own user |
| 22 | Cannot unblock own user |
| 23 | Cannot block other admins |
| 24 | Authentication failed – Invalid credentials were provided |
| 25  26  27  28 | Blocked – The user is blocked from using the application |
| 29 | Invalid JSON – The provided JSON-Request is malformed |

While the “code” identifies the error, the “message” attribute provides a more detailed description of the error.

To receive a localized error message, set the “Accept-Language”-HTTP Header to the desired language. Currently only English (en) and German (de) are supported.   
If no “Accept-Language” – Header is provided, English is used by default.

1. User Resource
2. Register User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| POST | url/user/register | None |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "name": "Kevin Paul",  "username": "kevin",  "email": "kevin.paul@gmx.at",  "password": "098f6bcd4621d373cade4e832627b4f6",  "education": "HTL",  "gender": "F"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | {  "username": "kevin",  "role": "U",  "email": "kevin.paul@gmx.at",  "name": "Kevin Paul",  "education": "HTL",  "gender": "F"  } |
| **409** | Username already exists. |

1. Get all Users

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/user/all | Admin |

**Parameters**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Query | start | Start index of users to get |
| Query | pageSize | Maximum number of offers |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | [  {  "username": "kevin",  "role": "U",  "email": "kevin.paul@gmx.at",  "name": "Kevin Paul",  "education": "HTL",  "gender": "F"  },  {  "username": "test",  "role": "U",  "email": "santnere@edu.htl-villach.at",  "name": "wacco",  "education": "HTL",  "gender": "M",  "block": {  "reason": "test"  }  },  {  "username": "wacco",  "role": "A",  "email": "santnere@edu.htl-villach.at",  "name": "wacco",  "education": "HTL",  "gender": "N"  }  ] |

1. Get own User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/user | User |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | {  "username": "wacco",  "role": "A",  "email": "santnere@edu.htl-villach.at",  "name": "wacco",  "education": "HTL",  "gender": "M"  } |

1. Update any User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| PUT | url/user/update | Admin |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "username": "kevin",  "name": "Paul Kevin",  "email": "coolerkev@gmx.at",  "password": "9d5e3ecdeb4cdb7acfd63075ae046672",  "education": "HTL",  "gender": "N"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |
| **453** | User not found |

1. Update own User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| PUT | url/user/update/own | User |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "name": "Elias Santner",  "email": "santnere@edu.htl-villach.at",  "password": "098f6bcd4621d373cade4e832627b4f6",  "education": "HTL",  "gender": "N"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |

1. Block User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| POST | url/user/block | Admin |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "username": "test",  "reason": "test",  "duedate": "2017-09-03T09:45:12+0200"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |
| **456** | Error blocking user. (Further details may be provided via body) |

1. Unblock User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/user/unblock/{username} | Admin |

**Parameters**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Path | Username | Username of user to unblock |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |

1. Get all Genders

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/user/gender | User |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | [  {  "code": "F",  "name": "Female"  },  {  "code": "M",  "name": "Male"  },  {  "code": "N"  "name": "Unspecified"  }  ] |

1. Offer Resource
2. Create Offer

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| POST | url/offer | User |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "description": "This is just a test offer",  "subject": {  "id": 3  }  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | {  "id": 1,  "postedon": "2017-10-04T07:18:39+0200",  "isactive": "1",  "description": "This is just a test offer",  "subject": {  "id": 1,  "dename": "Englisch",  "enname": "English"  },  "user": {  "username": "wacco"  }  } |
| **452** | Subject not found |

1. Update Offer

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| PUT | url/offer | User |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "id": 1,  "isactive": "1",  "description": "This is just a test offer",  "subject": {  "id": 1  }  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |
| **452** | Subject not found |
| **454** | Offer not found |

1. Get newest Offers

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/offer/new | User |

**Parameters**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Query | start | Start index of offers to get |
| Query | pageSize | Maximum number of offers |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | [  {  "id": 2,  "postedon": "2017-10-04T07:27:58+0200",  "isactive": "1",  "description": "Test offer 2",  "subject": {  "id": 1,  "dename": "Englisch",  "enname": "English"  },  "user": {  "username": "wacco"  }  },  {  "id": 1,  "postedon": "2017-10-04T07:18:39+0200",  "isactive": "0",  "description": "This is just a test offer",  "subject": {  "id": 1,  "dename": "Englisch",  "enname": "English"  },  "user": {  "username": "wacco"  }  }  ] |
| **455** | Query parameters were not given |

1. Get newest Offers of User

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/offer/new/{username} | User |

**Parameters**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Path | Username | Username of user to get newest offers of |
| Query | start | Start index of offers to get |
| Query | pageSize | Maximum number of offers |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | [  {  "id": 2,  "postedon": "2017-10-04T07:27:58+0200",  "isactive": "1",  "description": "Test offer 2",  "subject": {  "id": 1,  "dename": "Englisch",  "enname": "English"  },  "user": {  "username": "wacco"  }  },  {  "id": 1,  "postedon": "2017-10-04T07:18:39+0200",  "isactive": "0",  "description": "This is just a test offer",  "subject": {  "id": 1,  "dename": "Englisch",  "enname": "English"  },  "user": {  "username": "wacco"  }  }  ] |
| **453** | User not found |
| **455** | Query parameters were not given |

1. Subject Resource
2. Create Subject

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| POST | url/subject | Admin |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "dename": "Programmieren",  "enname": "Programming"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | {  "id": 2,  "dename": "Programmieren",  "enname": "Programming"  } |
| **409** | Subject name already exists |

1. Update Subject

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| PUT | url/subject | Admin |

|  |  |
| --- | --- |
| **MediaType** | **Sample Request Body** |
| application/json | {  "id": 4,  "dename": "Mathe",  "enname": "Maths"  } |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |
| **452** | Subject not found |
| **409** | Subject name already exists |

1. Delete Subject

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| DELETE | url/subject/{id} | Admin |

**Parameters**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| Path | Id | Id of subject to delete |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** |  |
| **452** | Subject not found |

1. Get all Subjects

**Request**

|  |  |  |
| --- | --- | --- |
| **Method** | **URL** | **Role** |
| GET | url/subject | User |

**Response**

|  |  |
| --- | --- |
| **Status** | **Response** |
| **200** | [  {  "id": 2,  "dename": "Deutsch",  "enname": "German"  },  {  "id": 1,  "dename": "Englisch",  "enname": "English"  },  {  "id": 3,  "dename": "Mathe",  "enname": "Maths"  }  ] |