TECHNICAL SPECIFICATION | 2022

Mentors Nexus Application

WEB APPLICATION FOR CONTENT DELIVERY

Lubomír Rýgl
Mentors s.r.o.
Address

Contents

Figures

Tables

Listings

1 Version Control

Version	Date	User	Change Note
ver1.0.0-3cbac7	14.03.2022	Lubomír Rýgl	Initial version

Table 1: Document version history

2 Application Overview

Mentors Nexus is a software web application used by Mentors.cz to provide its users access to educational materials online. The application back-end is running on Java 18 with SpringBoot Framework and the front-end will be a Angular application. For more detail about the availabilty and architacture - please see the documentation Architecture Overview. The following sections describe the actions that can be taken by users interfacing with the application.

2.1 Application

User Course Lesson Path

USER ACTION

- Register Account - Allow user to register a new account with an email address
- Log-In to Account
- Update User
- Delete User
- Subscribe to Course
- Purchase Course
- Watch Lesson
- Finish Lesson
- Finish Course

COURSE ACTION

- Create Course
- Edit Course
- Delete Course

LESSON ACTION

- Create Lesson
- Edit Lesson
- Delete Lesson
- Assign Lesson to Course

LEARNING PATH ACTION

- Create Learingn PATH
- Edit Learingn PATH
- Delete Learning Path
- Assign Course to Learning Path

3 Architecture Overview

3.1 General Architecture

asa

3.1.1 Customer

Basic user with its information and priviledges

3.1.2 Course

Course consists of lessons

3.1.3 Lesson

Actual lesson with its content

3.1.4 Course-Category

Each course has a category assigned

3.2 AWS Architecture

Application shall be build as a three tier application leveraging AWS Routing and Auto-Scalig groups. Whole solution is designed as HA.

3.3 Endpoint Overview

List of endpoints exposed to the frontend application available for request processing

```
1       https://mentors.cz/api/v01/
```

Listing 1: Backend Application base path

Category	Allowed Method	Type	Endpoint	Description
User	GET	PUBLIC	/public-status	Shows application status
User	POST	PUBLIC	/user/resetPassword	Reset user password
User	PUT	PROTECTED	/user/update	Register a new user
User	DELETE	PROTECTED	/user/delete	Register a new user
Category	GET	PUBLIC	/public-status	Shows application status
Category	GPOSTET	PUBLIC	/user/resetPassword	Reset user password
Category	PUT	PROTECTED	/user/update	Register a new user
Category	DELETE	PROTECTED	/user/delete	Register a new user
Course	GET	PUBLIC	/public-status	Shows application status
Course	POST	PUBLIC	/user/resetPassword	Reset user password
Course	PUT	PROTECTED	/user/update	Register a new user
Course	DELETE	PROTECTED	/user/delete	Register a new user
Lesson	GET	PUBLIC	/public-status	Shows application status
Lesson	POST	PUBLIC	/user/resetPassword	Reset user password
Lesson	PUT	PROTECTED	/user/update	Register a new user
Lesson	DELETE	PROTECTED	/user/delete	Register a new user

Table 2: List of available endpoints

4 Endpoint Definitions

4.1 User

get **user/findUserById/{id}**

Get user record by user Id

Parameter

id id of storage

Response application/json

200 ok

```
{
    "id": 867654678,
}

```

404 error: storage not found

```
{
    "message": "storage with id '11' not found!"
}

```

get **user/list**

Get all users user record by user Id

Parameter

id id of storage

Response application/json

200 ok

```
{
    "id": 867654678,
}

```

404 error: storage not found

```
{
    "message": "storage with id '11' not found!"
}

```

post **/user/login**

Create a new user

Parameter

no parameter

Body application/json

```
{
    "name" : "Apfelmus",
    "count" : 25
}
```

Response application/json

200 OK

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

400 BAD REQUEST

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

post **/user/register**

Register a new user to the application and verify if the user can be created

Parameter

no parameter

Body application/json

```
{
    "name" : "Apfelmus",
    "count" : 25
}
```

Response application/json

200 OK

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

post **/user/passwordReset**

Register a new user to the application and verify if the user can be created

Parameter

no parameter

Body application/json

```
{
    "name" : "Apfelmus",
    "count" : 25
}
```

Response application/json

200 OK

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

post **/user/add**

Create a new user

Parameter

no parameter

Body application/json

```
{
    "name" : "Apfelmus",
    "count" : 25
}
```

Response application/json

200 OK

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

400 BAD REQUEST

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

put **/user/update/**

Update existing user

Parameter

no parameter

Body application/json

```
{
    "name" : "Apfelmus",
    "count" : 25
}
```

Response application/json

200 ok

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

delete **/user/storage/**

Delete existing user from the application

Parameter

no parameter

Body application/json

```
{
    "name" : "Apfelmus",
    "count" : 25
}
```

Response application/json

200 ok

```
{
    "id": 867654678,
    "name" : "Apfelmus",
    "count" : 25
}
```

5 Database Design

```sql
SELECT * FROM application_users as;
```

Listing 2: Format XML for MAC calculation example