Establishing contracts between frontend and backend

Key terms

Frontend	Also known as the React PWA application.
Backend	This includes API and Database, and any layers in between
DB	Database

Index

Users **3**
- User object: 3
- GET /users 3
  - URL Params 3
  - Data Params 3
  - Headers 3
  - Success Response 3
- GET /user:id 4
  - URL Params 4
  - Data Params 4
  - Headers 4
  - Success Response 4
  - Error Response: 4
- GET /users/:id/achievements 5
  - URL Params 5
  - Data Params 5
  - Headers 5
  - Success Response 5
  - Error Response: 5
- POST /user/:id/xp 6
  - URL Params 6
  - Data Params 6
  - Headers 6
  - Success Response 6
  - Error Response: 6

Questions **7**
- Question object: 7
- GET /questions 7
  - URL Params 7
  - Data Params 7
  - Headers 7
  - Success Response 7
- GET /question/:level 8
  - URL Params 8
  - Data Params 8
  - Headers 8
  - Success Response 8
  - Error Response: 8
- GET /question/:id/answer 9
  - URL Params 9

Data Params....9
Headers....9
Success Response....9
Error Response:....9
GET /questions/topic....10
URL Params....10
Data Params....10
Headers....10
Success Response....10
Contents....10
GET /questions/level/topic....11
URL Params....11
Headers....11
Data Params....11
Success Response....11
POST /question/:id/answer....12
URL Params....12
Data Params....12
Headers....12
Success Response....12
Contents....12

Users

User object:

```
{
      id: integer
      name: string
      surname: string
      username: string
      email: string
      password: string
      currentLevel: integer
      joinDate: date
      xp: float
}
```

GET /users

Returns all users in the database.

URL Params

None

Data Params

None

Headers

Content-Type: application/json

Success Response

Code: 200

Contents

```
{
  users: [
           {<user_object>},
           {<user_object>},
           {<user_object>}
         ]
}
```

GET /user:id

Returns the specified user.

URL Params

Required: id=[integer]

Data Params

None

Headers

Content-Type: application/json

Authorization: Bearer <OAuth Token>

Success Response

Code: 200

Contents

```
{ <user_object> }
```

Error Response:

Code: 404

Content:

```
{ error : "User doesn't exist" }
```

OR

Code: 401

Content:

```
{ error : error : "You are unauthorized to make this request." }
```

GET /users/:id/achievements

Returns all achievements specific to a user.

URL Params

Required: id=[integer]

Data Params

None

Headers

Content-Type: application/json

Authorization: Bearer <OAuth Token>

Success Response

Code: 200

Content:

```
{
  achievements: [
            {<achievement_object>},
            {<achievement_object>},
            {<achievement_object>}
          ]
}
```

Error Response:

Code: 404

Content:

```
{ error : "User doesn't exist" }
```

OR

Code: 401

Content:

```
{ error : error : "You are unauthorized to make this request." }
```

POST /user/:id/xp

Updates the user's xp.

URL Params

Required: id=[integer]

Data Params

{
    id: integer,
    xp: float
}

Headers

Content-Type: application/json
Authorization: Bearer <OAuth Token>

Success Response

Code: 200
Content:
`{ <user_object> }`

Error Response:

Code: 404
Content:
`{ error : "User doesn't exist" }`

OR

Code: 401
Content:
`{ error : error : "You are unauthorized to make this request." }`

Questions

Question object:

```
{
      Q_id: integer
      topic: string
      difficulty: string
      level: integer
      questionText: string
      xpGain: float
}
```

GET /questions

Gets all the questions from the database.

URL Params

None

Data Params

None

Headers

Content-Type: application/json

Success Response

Code: 200

Contents

```
{
  questions: [
            {<question_object>},
            {<question_object>},
            {<question_object>}
          ]
}
```

GET /question/:level

Gets all questions based on the input level.

URL Params

Required: level=[integer]

Data Params

None

Headers

Content-Type: application/json

Authorization: Bearer <OAuth Token>

Success Response

Code: 200

Contents

```
{
  questions: [
            {<question_object>},
            {<question_object>},
            {<question_object>}
          ]
}
```

Error Response:

Code: 404

Content:

```
{ error : "Level doesn’t exist" }
```

OR

Code: 401

Content:

```
{ error : error : "You are unauthorized to make this request." }
```

GET /question/:id/answer

Gets the answer to a specific question.

URL Params

Required: Q_id=[integer]

Data Params

None

Headers

Content-Type: application/json

Authorization: Bearer <OAuth Token>

Success Response

Code: 200

Contents

```
{
  answer: [
            {<answer_object>},
          ]
}
```

Error Response:

Code: 404

Content:

```
{ error : "Question doesn't exist" }
```

OR

Code: 401

Content:

```
{ error : error : "You are unauthorized to make this request." }
```

GET /questions/topic

Returns all of the questions for that specific topic

URL Params

Required: topic="topic_tag"

Data Params

None

Headers

Content-Type: application/json

Success Response

Code: 200

Contents

```
{
  questions: [
            {<question_object>},
            {<question_object>},
            {<question_object>}
          ]
}
```

GET /questions/level/topic

Get all of the questions based on their level and their topic.

URL Params

Required: level=[integer]

Required: topic="topic_tag"

Headers

Content-Type: application/json

Data Params

```
{
  answer: string
}
```

Success Response

Code: 200

Contents

```
{
  result: boolean
}
```

POST /question/:id/answer

Send the question's answer through to be checked if correct.

URL Params

Required: Q_id=[integer]

Data Params

```
{
  question: [
            {<question_object>},
            {<question_object>},
            {<question_object>}
          ]
}
```

Headers

Content-Type: application/json

Success Response

Code: 200

Contents

```
{
  questions: [
            {<question_object>},
            {<question_object>},
            {<question_object>}
          ]
}
```