

# Establishing contracts between frontend and backend

## Key terms

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

# Index

<b>Users</b>	<b>3</b>
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
<b>Questions</b>	<b>7</b>
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	
<b>Practice Endpoints</b>	
<a href="#">GET/practice</a>	
URL Params	12
Data Params	12
Headers	12
Success Response	12
Contents	
<b>Answer Endpoints</b>	
POST /question/:id/answer	12
URL Params	12
Data Params	12
Headers	12
Success Response	12
Contents	
<b>XP and Multiplayer Endpoints</b>	
POST /question/:id/answer	12
URL Params	12
Data Params	12
Headers	12
Success Response	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>}
  ]
}
```

## **GET /questionsById/:id**

Fetches a specific question by its ID.

### **URL Params**

Required: id=[integer]

### **Headers**

Content-Type: application/json

### **Success Response**

**Code:** 200

**Contents:**

```
{  
  "question": { <question_object> }  
}
```

### **GET /answers/:id**

Returns all answers for a specific question.

#### **URL Params**

Required: `id=[integer]`

#### **Headers**

Content-Type: application/json

#### **Success Response**

**Code:** 200

**Contents:**

```
{  
  "answers": [  
    { <answer_object> },  
    { <answer_object> }  
  ]  
}
```

## **Practice Endpoints**

### **GET /practice**

Returns 10 practice questions and their answers.

#### **Headers**

Content-Type: application/json

**Success Response****Code:** 200**Contents:**

```
{  
  "questions": [  
    { <question_object_with_answers> },  
    { <question_object_with_answers> }  
  ]  
}
```

**GET /practice/type/:questionType**

Returns 10 questions of a specific type and their answers.

**URL Params**

Required: `questionType=[string]` (e.g., "multiple-choice" or "true-false")

**Headers**

Content-Type: application/json

**Success Response****Code:** 200**Contents:**

```
{  
  "questions": [  
    { <question_object_with_answers> },  
    { <question_object_with_answers> }  
  ]  
}
```



# Answer Endpoints

## POST /submit-answer

Checks if the selected answer is correct and awards XP.

### Headers

Content-Type: application/json

### Data Params

```
{  
  "questionId": [integer],  
  "selectedAnswer": "[string]"  
}
```

### Success Response

**Code:** 200

**Contents:**

```
{  
  "correct": true,  
  "xpAwarded": [integer]  
}
```

## POST /question/:id/submit

Validates user answers to specific questions and awards XP.

### URL Params

Required: `id=[integer]`

### Headers

Content-Type: application/json

### Data Params

```
{  
  "answers": [ "[string]", "[string]" ]  
}
```

```
}
```

### Success Response

**Code:** 200

**Contents:**

```
{  
  "results": [  
    { "correct": true, "xp": 10 },  
    { "correct": false, "xp": 0 }  
  ]  
}
```

### POST /validate-answer

Validates if math answers match.

#### Headers

Content-Type: application/json

#### Data Params

```
{  
  "expected": "[string]",  
  "submitted": "[string]"  
}
```

### Success Response

**Code:** 200

**Contents:**

```
{  
  "isValid": true  
}
```

### POST /quick-validate

Performs quick validation of math answers.

#### **Headers**

Content-Type: application/json

#### **Data Params**

```
{  
  "expected": "[string]",  
  "input": "[string]"  
}
```

#### **Success Response**

**Code:** 200

**Contents:**

```
{  
  "valid": true  
}
```

### **POST /validate-expression**

Checks if a math expression is valid.

#### **Headers**

Content-Type: application/json

#### **Data Params**

```
{  
  "expression": "[string]"  
}
```

#### **Success Response**

**Code:** 200

**Contents:**

```
{  
  "valid": true  
}
```

```
}
```

# XP and Multiplayer Endpoints

## POST /singleplayer

Handles XP calculations for a single player answering a question.

### Headers

Content-Type: application/json

### Data Params

```
{  
  "userId": [integer],  
  "questionId": [integer],  
  "correct": true  
}
```

### Success Response

**Code:** 200

**Contents:**

```
{  
  "xpAwarded": 10,  
  "newTotalXP": 150  
}
```

## POST /multiplayer

Handles XP and match processing for multiplayer mode.

### Headers

Content-Type: application/json

### Data Params

```
{
```

```
"player1": {  
  "userId": [integer],  
  "score": [integer]  
},  
"player2": {  
  "userId": [integer],  
  "score": [integer]  
},  
"questionId": [integer]  
}
```

#### **Success Response**

**Code:** 200

**Contents:**

```
{  
  "matchResult": "player1_wins",  
  "xpUpdates": {  
    "player1": 20,  
    "player2": 5  
  }  
}
```