Super assessor card API

Gabriel Marques, Martin Heggholmen og Vegard Madsen

Contents

[API Specification introduction 2](#_Toc167781299)

[Title 2](#_Toc167781300)

[Authors 2](#_Toc167781301)

[Problem 2](#_Toc167781302)

[Solution 3](#_Toc167781303)

[Implementation 3](#_Toc167781304)

[Authentication 3](#_Toc167781305)

[Other things we considered 3](#_Toc167781306)

[Spec 4](#_Toc167781307)

[Errors 4](#_Toc167781308)

[Endpoints/web services (for Rest or RPC) 4](#_Toc167781309)

[Collection 1: Users 4](#_Toc167781310)

[Collection 2: Cards 5](#_Toc167781311)

[Collection 3: Schemas 6](#_Toc167781312)

[Collection 4: Ratings 6](#_Toc167781313)

[Collection 5: Bookmarks 7](#_Toc167781314)

[Collection 6: Search 7](#_Toc167781315)

# API Specification introduction

## Title

Super assessor digital card game API

## Authors

* Gabriel Marques
* Martin Heggholmen
* Vegard Madsen

## Problem

The department of Design in Trondheim has developed a card game for creating unique assessment methods, which they are digitalizing. For this the game needs to be able to access a database of relevant cards that can be added to, removed from, and updated per the users needs. There is currently no backend API which allows for this, and it has been done entirely using a static list of cards.

## Solution

The solution is a lightweight API that allows users to access the cards from the super assessor card game. Through this API, users will be able to retrieve and manipulate the cards by changing details about them, as well as add new cards to the database and delete them. The cards also will have the possibility of changing the icons used on them, something which will be individual to the users. Additionally, the API will allow new users to be created, with different details such as position, which university they belong to as well as which department and details such as name and email address. Lastly the API will allow users to search for cards that match the given parameters (such as looking for specific card categories) and choose to get a random selection

## Implementation

Discussion about technology: paradigm picked and list of reasons

* Since this project deals with resources (such as users and cards) REST seems like an easy way to represent the resources as they can be mapped to different endpoints (such as /cards and /users) which makes it very easy to do operations on the specific resource I want.

## Authentication

As of right now there is not system for authentication to use the API. It is open to anyone.

## Other things we considered

Describe the technology architecture you’ve selected and list the pros and cons of paradigms you’ve considered and mark the one you have picked.

|  |  |  |  |
| --- | --- | --- | --- |
| Pattern, paradigm or protocol considered | Pros | Cons | Selected? |
| REST | Common and familiar. Simple CRUD operations. Easy to understand HTTP responses. | Can be less flexible than other solutions, stateless can require extra data. | Yes |
| GraphQL | Allows you to easily query exactly what you need. | Harder to set up and learn. Can get quite complex | No |
| RPC | Works bidirectionally, (good for real time uses) | No uniform standard, server side changes can break client interactions | No |

# Spec

You can structure this section in different ways. Organised by resources (if rest), by events, etc. You can add use the following tables.

## Errors

|  |  |  |  |
| --- | --- | --- | --- |
| HTTP status code | Error code | Verbose error | Description |
| 404 |  | Could not update. Resource not found | The user tried to update a user or card not found in the database |
| 500 |  | Internal server error | All other errors thrown by the backend. Something went wrong on the server side. |
|  |  |  |  |
|  |  |  |  |

## Endpoints/web services (for Rest or RPC)

If many “resources”, it can be structured by “resource”. If not, all web services in one table

### Collection 1: Users

|  |  |  |
| --- | --- | --- |
| **URI** | **Inputs** | **Outputs** |
| GET /api/users | N/A | Gets all users |
| POST /api/users | Name, surname, email, department, university, position | Creates new user |
| GET /api/users/ | ID | Get user with |
| PATCH /api/users/ | ID | Updates user with |
| DELETE /api/users/ | ID | Deletes user with |
| GET /api/users/name/ | ID | Get single user name with |
| POST /api/users/login | email, password | Return JWT-token |
| GET /api/users/logout | N/A | Logs out user |
| GET /api/users/status | N/A | Gets user status |
| GET /api/users/total | N/A | Gets total teachers |
| GET /api/users/token | N/A | Refreshes token |
| GET /api/users/account | N/A | Gets own ID |

### Collection 2: Cards

|  |  |  |
| --- | --- | --- |
| **URI** | **Inputs** | **Outputs** |
| GET /api/cards | N/A | Gets all cards |
| POST /api/cards | Name, type, description | Creates new card |
| GET /api/cards/total | N/A | Gets total cards |
| GET /api/cards/types | N/A | Gets total card types |
| GET /api/cards/:id | ID | Get card with |
| PATCH /api /cards/:id | ID | Updates card with |
| DELETE /api /cards/:id | ID | Deletes card with |
| POST /api/cards/bulk | cards (in request body) | Bulk upload cards |

### Collection 3: Schemas

|  |  |  |
| --- | --- | --- |
| **URI** | **Inputs** | **Outputs** |
| GET /api/assscheme | N/A | Gets all schemes |
| POST /api/assscheme | Name, type, description | Creates new scheme |
| GET /api/assscheme/:id | ID | Get scheme with |
| PATCH /api/assscheme/:id | ID | Updates scheme with |
| DELETE /api/assscheme/:id | ID | Deletes scheme with |

### Collection 4: Ratings

|  |  |  |
| --- | --- | --- |
| **URI** | **Inputs** | **Outputs** |
| GET /api/ratings | N/A | Gets all ratings |
| POST /api/ratings | Rating, comment, user\_id, card\_id | Creates new rating |
| GET /api/ratings/:id | ID | Get rating with |
| PATCH /api/ratings/:id | ID | Updates rating with |
| DELETE /api/ratings/:id | ID | Deletes rating with |
| GET /api/ratings/avg/:id | ID | Gets ratings average with |

### Collection 5: Bookmarks

|  |  |  |
| --- | --- | --- |
| **URI** | **Inputs** | **Outputs** |
| POST /api/bookmarks/ | userID, schemeID | Create bookmark |
| GET /api/bookmarks/ | N/A | Getall user bookmarks |
| GET /api/bookmarks/:id | ID | Getsingle bookmark |
| GET /api/bookmarks/bookmarked/:id | ID | Get single bookmark (is bookmarked) |
| DELETE /api/bookmarks/:id | ID | Delete bookmark |

### Collection 6: Search

|  |  |  |
| --- | --- | --- |
| URI | Inputs | Outputs |
| GET /search | **Queries:**  card-type, card-category, random, exclude | Gets selected cards, use random to get a random amount of cards that fit, use exclude to exclude specific cards |