Errors

Collapse

✕ Swagger Error

`A deterministic version of a JSON Schema object.`

Jump to line **1056**

Details

▶ `Object`

⚠ Swagger Warning

`Extra JSON Reference properties will be ignored: description`

Jump to line **841**

Details

▶ `Object`

RottenMods API

RottenMods API

Version 1.0.0

License

Paths

`/bookmark`

GET /bookmark

bookmark

Summary

gets all bookmarks for a user

Description

Returns a list of appropriate bookmarks

Parameters

Name	Located in	Description	Required	Schema
userId	query	OID of the user	Yes	⇄ `undefined`

Responses

Code	Description	Schema
200	Success	⇄ ▾**BookmarkArray** [▸**Bookmark { }**]
500	Error	

Try this operation

POST /bookmark

bookmark

Summary

creates a bookmark

Parameters

Name	Located in	Description	Required	Schema
body	body	Bookmark object	Yes	⇄ ▾**BookmarkCreate** { userId: string modId: string }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

/bookmark/{bookmarkId}

```
DELETE /bookmark/{bookmarkId}
```

bookmark

Summary

deletes a bookmark by OID

Parameters

Name	Located in	Description	Required	Schema
bookmarkId	path	OID of the bookmark	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/course`

GET /course

course

Summary

searches for courses

Description

Returns a list of appropriate courses

Parameters

Name	Located in	Description	Required	Schema
s	query	Search term - case insensititve course name	No	⇄ `string`
page	query	Page number	No	⇄ `integer`
limit	query	Maximum number of results returned	No	⇄ `Integer`

Responses

Code	Description

Try this operation

POST /course

course

Summary

creates a course

Parameters

Name	Located in	Description	Required	Schema
body	body	Course object	Yes	⇄ ▾**CourseCreate** { name: string }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

/course/{courseId}

```
GET /course/{courseId}
```

course

Summary

finds a course by OID

Description

Returns a course

Parameters

Name	Located in	Description	Required	Schema
courseId	path	OID of the course	Yes	⇄ undefined

Responses

Code	Description	Schema

200	Success	⇄ ▾**Course** {
		`_id:       string`
		`name:      string`
		`rating:    ▸RatingObj { }`
		`event:     ▸EventObj { }`
		`createdAt: string`
		`updatedAt: string`
		`_v:        integer`
		}

500	Error	

Try this operation

PUT /course/{courseId}

course

Summary

updates a course

Parameters

Name	Located in	Description	Required	Schema
courseId	path	OID of the course	Yes	⇄ `undefined`
body	body	Course Object	Yes	⇄ ▾**CourseCreate** { name: string }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`DELETE /course/{courseId}`

course

Summary

deletes a course by OID

Parameters

Name	Located in	Description	Required	Schema
courseId	path	OID of the course	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/event`

POST /event

event

Summary

creates an event

Parameters

Name	Located in	Description	Required	Schema
body	body	Event object	Yes	⇄ ▾**EventCreate** { userId: string sub: string subId: string type: string }
sub	body	Type of subscriber to the event	Yes	⇄ string **Enum**: ▶ Array [4]
subId	body	OID of subscriber object	Yes	⇄ string
type	body	Type of event	Yes	⇄ string **Enum**: ▶ Array [2]

Responses

Code	Description
200	Success
500	Error

Try this operation

DELETE /event

event

Summary

deletes an event

Parameters

Name	Located in	Description	Required	Schema
body	body	Event object	Yes	⇄ ▾**EventCreate {** userId: string sub: string subId: string type: string }
sub	body	Type of subscriber to the event	Yes	⇄ string **Enum:** ▶ Array [4]
subId	body	OID of subscriber object	Yes	⇄ string
type	body	Type of event	Yes	⇄ string **Enum:** ▶ Array [2]

Responses

Code	Description
200	Success
500	Error

Try this operation

/event/find

GET /event/find

event

Summary

checks for an event

Description

Checks if an event exists for a user, an event type and a sub type

Parameters

Name	Located in	Description	Required	Schema
sub	query	Type of subscriber to the event	Yes	⇄ string **Enum:** ▶ Array [4]
subId	query	OID of the subscriber	Yes	⇄ string
userId	query	OID of subscriber object	Yes	⇄ string
type	query	Type of event	Yes	⇄ string **Enum:** ▶ Array [2]

Responses

Code	Description
200	Event found
404	Event not found
500	Error

Try this operation

`/mod`

`GET /mod`

mod

Summary

searches for modules

Description

Returns a list of appropriate modules

Parameters

Name	Located in	Description	Required	Schema
s	query	Search term - case insensititve module title or code	No	⇄ `string`
page	query	Page number	No	⇄ `integer`
limit	query	Maximum number of results returned	No	⇄ `Integer`

Responses

Code	Description	Schema
200	Success	⇄ ▾**ModArray**[▸**Mod** { }]
500	Error	

Try this operation

`POST /mod`

mod

Summary

creates a module

Parameters

Name	Located in	Description	Required	Schema
body	body	Module object	Yes	⇄ ▾**ModCreate** { code: string title: string schoolId: string acadYear: string semester: integer description: string credit: integer workload: integer }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/mod/{modId}`

GET /mod/{modId}

mod

Summary

finds a module by OID

Description

Returns a module

Parameters

Name	Located in	Description	Required	Schema
modId	path	OID of the module	Yes	⇄ string

Responses

Code	Description	Schema

200	Success	⇄ ▼**Mod {**
		`_id:          string`
		`code:         string`
		`title:        string`
		`schoolId:     string`
		`acadYear:     string`
		`semester:     integer`
		`description:  string`
		`credit:       integer`
		`workload:     integer`
		`prereqs:      ►[]`
		`rating:       ►RatingObj { }`
		`event:        ►EventObj { }`
		`reaction:     ►ReactionObj { }`
		`createdAt:    string`
		`updatedAt:    string`

```
    _v:          integer
}
```

500	Error

Try this operation

```
PUT /mod/{modId}
```

mod

Summary

updates a module

Parameters

Name	Located in	Description	Required	Schema
modId	path	OID of the module	Yes	⇄ `string`
body	body	Module object	Yes	⇄ see ModCreate below

```
▾ModCreate {
    code:        string
    title:       string
    schoolId:    string
    acadYear:    string
    semester:    integer
    description: string
    credit:      integer
    workload:    integer
}
```

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
DELETE /mod/{modId}
```

mod

Summary

delete a module by OID

Parameters

Name	Located in	Description	Required	Schema
modId	path	OID of the module	Yes	⇄ `string`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/planned-mod`

GET /planned-mod

planned mod

Summary

finds a all planned mods by userId

Description

Returns a list of planned mods

Parameters

Name	Located in	Description	Required	Schema
userId	query	OID of the user	Yes	⇄ `undefined`

Responses

Code	Description	Schema
200	Success	⇄ ▾ **PlannedModArray** [▸ **PlannedMod** { }]
500	Error	

Try this operation

POST /planned-mod

planned mod

Summary

adds a module to a user's study plan

Parameters

Name	Located in	Description	Required	Schema
body	body	PlannedMod Object	Yes	⇄ ▾ **PlannedModCreate** { modId: string userId: string semester: integer }

Responses

Code	Description

Try this operation

RESPONSES /planned-mod

Responses

Code	Description

Try this operation

/planned-mod/{plannedModId}

PUT /planned-mod/{plannedModId}

planned mod

Summary

updates a planned module

Parameters

Name	Located in	Description	Required	Schema
plannedModId	path	OID of the planned module	Yes	⇄ `string`
body	body	Planned Module object	Yes	⇄ `▾ PlannedModCreate { modId: string userId: string semester: integer }`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
DELETE /planned-mod/{plannedModId}
```

planned mod

Summary

delete a planned mod by OID

Parameters

Name	Located in	Description	Required	Schema
plannedModId	path	OID of the planned module	Yes	⇄ `string`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/logout`

```
GET /logout
```

logout

Summary

logs an existing user out

Responses

Code	Description
200	Success

Try this operation

```
/login
```

POST /login

login

Summary

logs an existing user in

Parameters

Name	Located in	Description	Required	Schema
body	body	Login Object	Yes	⇄ ▾**Login** { email: string password: string }

```
▾Login {
   email:    string
   password: string
}
```

Responses

Code	Description	Schema
200	Success	⇄ ▾**User** { _id: string name: string email: string password: string schoolStartDate: string schoolId: string courseId: string currentYear: integer createdAt: string updatedAt: string _v: integer }
403	Not authorized	
500	Error	

```
▾User {
   _id:             string
   name:            string
   email:           string
   password:        string
   schoolStartDate: string
   schoolId:        string
   courseId:        string
   currentYear:     integer
   createdAt:       string
   updatedAt:       string
   _v:              integer
}
```

Try this operation

/rating

POST /rating

rating

Summary

creates a rating

Parameters

Name	Located in	Description	Required	Schema
body	body	Rating Object	Yes	⇄ ▾**RatingCreate {** userId: string sub: string subId: string type: string value: integer }
sub	body	Type of subscriber to the rating	Yes	⇄ string **Enum:** ▶ Array [3]
subId	body	OID of subscriber object	Yes	⇄ string
type	body	Type of rating	Yes	⇄ string **Enum:** ▶ Array [2]
value	body	Rating number	Yes	⇄ integer

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

GET /rating

rating

Summary

gets a rating

Description

Gets a rating

Parameters

Name	Located in	Description	Required	Schema
sub	query	Type of subscriber to the rating	Yes	⇄ string **Enum:** ▶ Array [3]
subId	query	OID of the subscriber	Yes	⇄ string
userId	query	OID of subscriber object	Yes	⇄ string
type	query	Type of rating	Yes	⇄ string **Enum:** ▶ Array [2]

Responses

Code	Description	Schema
200	Rating found	⇄ ▾**Rating {** _id: string, userId: string, sub: string, subId: string, type: string, value: integer, createdAt: string, updatedAt: string

```
▾Rating {
   _id:       string
   userId:    string
   sub:       string
   subId:     string
   type:      string
   value:     integer
   createdAt: string
   updatedAt: string
```

```
                                                  _v:         integer
                                                }
```

404	Rating not found
500	Error

Try this operation

/rating/{ratingId}

`PUT /rating/{ratingId}`

rating

Summary

updates a rating

Parameters

Name	Located in	Description	Required	Schema
ratingId	path	OID of the rating	Yes	⇄ `string`
body	body	Rating Object	Yes	⇄ ▾**RatingCreate** { userId: string, sub: string, subId: string, type: string, value: integer }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/reaction`

```
POST /reaction
```

reaction

Summary

creates a reaction

Parameters

Name	Located in	Description	Required	Schema
body	body	Reaction Object	Yes	⇄ ▾**ReactionCreate** [object Object]
sub	body	Type of subscriber to the reaction	Yes	⇄ string **Enum:** ▸ Array [3]
subId	body	OID of subscriber object	Yes	⇄ string
type	body	Type of reaction	Yes	⇄ string **Enum:** ▸ Array [3]

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

GET /reaction

reaction

Summary

gets a reaction

Description

Gets a reaction

Parameters

Name	Located in	Description	Required	Schema
sub	query	Type of subscriber to the reaction	Yes	⇄ string **Enum:** ▶ Array [3]
subId	query	OID of the subscriber	Yes	⇄ string
userId	query	OID of subscriber object	Yes	⇄ string
type	query	Type of reaction	Yes	⇄ string **Enum:** ▶ Array [3]

Responses

Code	Description	Schema
200	Reaction found	⇄ ▾**Reaction {** _id: string, userId: string, sub: string, subId: string, type: string, createdAt: string, updatedAt: string, _v: integer

}

404	Reaction not found
500	Error

Try this operation

/reaction/{reactionId}

DELETE /reaction/{reactionId}

reaction

Summary

deletes a reaction by OID

Parameters

Name	Located in	Description	Required	Schema
reactionId	path	OID of the reaction	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
500	Error

Try this operation

/reply

`POST /reply`

reply

Summary

creates a reply

Parameters

Name	Located in	Description	Required	Schema
body	body	Reply object	Yes	⇄ ▾**ReplyCreate** { userId: string, text: string, reviewId: string, replyId: string }

```
▾ReplyCreate {
    userId:   string
    text:     string
    reviewId: string
    replyId:  string
}
```

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/reply/{replyId}`

GET /reply/{replyId}

reply

Summary

finds a reply by OID

Description

Returns a reply

Parameters

Name	Located in	Description	Required	Schema
replyId	path	OID of the reply	Yes	⇄ undefined

Responses

Code	Description	Schema
200	Success	⇄ **Reply** { _id: string, userId: string, text: string, reviewId: string, replyId: string, createdAt: string, updatedAt: string, _v: integer }
500	Error	

```
▾Reply {
   _id:       string
   userId:    string
   text:      string
   reviewId:  string
   replyId:   string
   createdAt: string
   updatedAt: string
   _v:        integer
}
```

Try this operation

```
PUT /reply/{replyId}
```

reply

Summary

updates a reply

Parameters

Name	Located in	Description	Required	Schema
replyId	path	OID of the reply	Yes	⇄ `undefined`
body	body	Reply Object	Yes	⇄ see below

```
▾ReplyCreate {
   userId:   string
   text:     string
   reviewId: string
   replyId:  string
}
```

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`DELETE /reply/{replyId}`

reply

Summary

deletes a reply by OID

Parameters

Name	Located in	Description	Required	Schema
replyId	path	OID of the reply	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/reply/review/{reviewId}`

`GET /reply/review/{reviewId}`

reply

Summary

gets all replies for a review

Description

Returns a list of replies to a review

Parameters

Name	Located in	Description	Required	Schema
reviewId	path	OID of the review	Yes	⇄ `undefined`

Responses

Code	Description	Schema
200	Success	⇄ ▾**ReplyArray**[▸**Reply** { }]
500	Error	

Try this operation

`/review`

GET /review

review

Summary

gets all reviews

Description

Returns a list of reviews

Responses

Code	Description	Schema
200	Success	⇄ ▼**ReviewArray**[▶**Review** { }]
500	Error	

Try this operation

```
POST /review
```

review

Summary

creates a review

Parameters

Name	Located in	Description	Required	Schema
body	body	Review Object	Yes	⇄ ▾**ReviewCreate** { userId: string, text: string, modId: string, semesterTaken: number, acadYearTaken: number }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
/review/{reviewId}
```

GET /review/{reviewId}

review

Summary

finds a review by OID

Description

Returns a review

Parameters

Name	Located in	Description	Required	Schema
reviewId	path	OID of the review	Yes	⇄ undefined

Responses

Code	Description	Schema
200	Success	⇄ see below
500	Error	

```
▾Review {
  _id:          string
  userId:       string
  text:         string
  modId:        string
  semesterTaken: number
  acadYearTaken: string
  event:        ▸EventObj { }
  rating:       ▸RatingObj { }
  createdAt:    string
  updatedAt:    string
  _v:           integer
}
```

Try this operation

```
PUT /review/{reviewId}
```

review

Summary

updates a review

Parameters

Name	Located in	Description	Required	Schema
reviewId	path	OID of the review	Yes	⇄ `string`
body	body	Review Object	Yes	⇄ see below

```
▾ReviewCreate {
   userId:        string
   text:          string
   modId:         string
   semesterTaken: number
   acadYearTaken: number
}
```

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`DELETE /review/{reviewId}`

review

Summary

deletes a review by OID

Parameters

Name	Located in	Description	Required	Schema
reviewId	path	OID of the review	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

`/review/mod/{modId}`

GET /review/mod/{modId}

review

Summary

paginate reviews for single module

Description

Returns a list of reviews

Parameters

Name	Located in	Description	Required	Schema
modId	path	OID of the module	No	⇄ string
page	query	Page number	No	⇄ integer
limit	query	Maximum number of results returned	No	⇄ Integer

Responses

Code	Description	Schema
200	Success	⇄ (see below)

```
▾Review {
  _id:           string
  userId:        string
  text:          string
  modId:         string
  semesterTaken: number
  acadYearTaken: string
  event:         ▸EventObj { }
  rating:        ▸RatingObj { }
  createdAt:     string
  updatedAt:     string
  _v:            integer
}
```

500	Error

Try this operation

/review/user/{userId}

GET /review/user/{userId}

review

Summary

paginate reviews for single module

Description

Returns a list of reviews by a user

Parameters

Name	Located in	Description	Required	Schema
userId	path	OID of the user	No	⇄ string
page	query	Page number	No	⇄ integer
limit	query	Maximum number of results returned	No	⇄ Integer

Responses

Code	Description	Schema
200	Success	⇄ (see below)

```
▾Review {
   _id:           string
   userId:        string
   text:          string
   modId:         string
   semesterTaken: number
   acadYearTaken: string
   event:         ▸EventObj { }
   rating:        ▸RatingObj { }
   createdAt:     string
   updatedAt:     string
   _v:            integer
}
```

500	Error

Try this operation

/school

GET /school

school

Summary

searches for schools

Description

Returns a list of appropriate schools

Parameters

Name	Located in	Description	Required	Schema
s	query	Search term - case insensititve school name	No	⇄ string
page	query	Page number	No	⇄ integer
limit	query	Maximum number of results returned	No	⇄ Integer

Responses

Code	Description

Try this operation

```
POST /school
```

school

Summary

creates a school

Parameters

Name	Located in	Description	Required	Schema
body	body	School Object	Yes	⇄ ▾**SchoolCreate {** name: string }

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
/school/{schoolId}
```

```
GET /school/{schoolId}
```

school

Summary

finds a school by OID

Description

Returns a school

Parameters

Name	Located in	Description	Required	Schema
schoolId	path	OID of the school	Yes	⇄ `undefined`

Responses

Code	Description	Schema
200	Success	⇄ see below
500	Error	

```
▾School {
  _id:       string
  name:      string
  rating:    ▸RatingObj { }
  event:     ▸EventObj { }
  createdAt: string
  updatedAt: string
  _v:        integer
}
```

Try this operation

```
PUT /school/{schoolId}
```

school

Summary

updates a school

Parameters

Name	Located in	Description	Required	Schema
schoolId	path	OID of the school	Yes	⇄ `string`
body	body	School Object	Yes	⇄ ▾**SchoolCreate** `{ name: string }`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
DELETE /school/{schoolId}
```

school

Summary

deletes a school by OID

Parameters

Name	Located in	Description	Required	Schema
schoolId	path	OID of the school	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
/user
```

POST /user

user

Summary

creates a user

Parameters

Name	Located in	Description	Required	Schema
body	body	User Object	Yes	⇄ ▾**UserCreate {** name: string, email: string, password: string, schoolStartDate: string, schoolId: string, courseId: string, currentYear: integer }
email	body	Email of the user (only accept .edu emails)	Yes	⇄ string

Responses

Code	Description
200	
403	Not authorized
500	Error

Try this operation

/user/{userId}

```
GET /user/{userId}
```

user

Summary

finds a user by OID

Description

Returns a user

Parameters

Name	Located in	Description	Required	Schema
userId	path	OID of the user	Yes	⇄ `undefined`

Responses

Code	Description	Schema
200	Success	⇄ see below
500	Error	

```
▾User {
   _id:             string
   name:            string
   email:           string
   password:        string
   schoolStartDate: string
   schoolId:        string
   courseId:        string
   currentYear:     integer
   createdAt:       string
   updatedAt:       string
   _v:              integer
}
```

Try this operation

```
PUT /user/{userId}
```

user

Summary

updates a user

Parameters

Name	Located in	Description	Required	Schema
userId	path	OID of the user	Yes	⇄ `string`
body	body	User Object	Yes	⇄ see below

```
▾UserCreate {
   name:            string
   email:           string
   password:        string
   schoolStartDate: string
   schoolId:        string
   courseId:        string
   currentYear:     integer
}
```

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

```
DELETE /user/{userId}
```

user

Summary

deletes a user by OID

Parameters

Name	Located in	Description	Required	Schema
userId	path	OID of the user	Yes	⇄ `undefined`

Responses

Code	Description
200	Success
403	Not authorized
500	Error

Try this operation

Models

Bookmark

```
▾Bookmark {
   _id:       string
   userId:    string
   modId:     string
```

⇄

```
    createdAt: string
    updatedAt: string
    _v:        integer
  }
```

BookmarkArray

```
  ▾BookmarkArray[
⇄   ▸Bookmark { }
  ]
```

BookmarkCreate

```
  ▾BookmarkCreate {
    userId: string
⇄
    modId:  string
  }
```

Course

```
  ▾Course {
    _id:       string
    name:      string
    rating:    ▸RatingObj { }
⇄   event:     ▸EventObj { }
    createdAt: string
    updatedAt: string
    _v:        integer
  }
```

CourseArray

```
  ▾CourseArray[
⇄   ▸Course { }
  ]
```

CourseCreate

```
⇄   ▾CourseCreate {
      name: string
    }
```

EventCreate

```
    ▾EventCreate {
      userId: string
      sub:    string
⇄     subId:  string
      type:   string
    }
```

EventObj

```
    ▾EventObj {
      like: ▸EventProp { }
⇄     view: ▸EventProp { }
    }
```

EventProp

```
    ▾EventProp {
⇄     count: integer
    }
```

Login

```
    ▾Login {
      email:    string
⇄     password: string
    }
```

ModCreate

```
⇄   ▾ModCreate {
       code:        string
       title:       string
       schoolId:    string
       acadYear:    string
       semester:    integer
       description: string
       credit:      integer
       workload:    integer
    }
```

Mod

```
⇄   ▾Mod {
       _id:         string
       code:        string
       title:       string
       schoolId:    string
       acadYear:    string
       semester:    integer
       description: string
       credit:      integer
       workload:    integer
       prereqs:     ▸[]
       rating:      ▸RatingObj { }
       event:       ▸EventObj { }
       reaction:    ▸ReactionObj { }
       createdAt:   string
       updatedAt:   string
       _v:          integer
}
```

ModArray

```
▾ModArray[
⇄   ▸Mod { }
]
```

PlannedMod

```
▾PlannedMod {
   _id:               string
   modId:             string
   userId:            string
   semester:          integer
   mod:               ▸Mod { }
⇄  missingPrereqs:    ▸[]
   allPrereqsPresent: boolean
   createdAt:         string
   updatedAt:         string
   _v:                integer
}
```

PlannedModCreate

```
▾PlannedModCreate {
   modId:    string
⇄  userId:   string
   semester: integer
}
```

PlannedModArray

```
▾PlannedModArray[
⇄   ▸PlannedMod { }
]
```

Rating

```
  ▾Rating {
    _id:       string
    userId:    string
    sub:       string
    subId:     string
⇄   type:      string
    value:     integer
    createdAt: string
    updatedAt: string
    _v:        integer
  }
```

RatingCreate

```
  ▾RatingCreate {
    userId: string
    sub:    string
⇄   subId:  string
    type:   string
    value:  integer
  }
```

RatingObj

```
  ▾RatingObj {
    difficulty: ▸RatingProp { }
⇄   star:       ▸RatingProp { }
  }
```

RatingProp

```
⇄ ▾RatingProp {
     count: integer
     value: number
   }
```

Reaction

```
⇄ ▾Reaction {
     _id:       string
     userId:    string
     sub:       string
     subId:     string
     type:      string
     createdAt: string
     updatedAt: string
     _v:        integer
   }
```

ReactionCreate

```
⇄ ▾ReactionCreate [object Object]
```

ReactionObj

```
⇄ ▾ReactionObj {
     difficulty: ▸ReactionProp { }
     star:       ▸ReactionProp { }
   }
```

ReactionProp

```
⇄ ▾ReactionProp {
     count: integer
   }
```

Reply

```
  ▾Reply {
     _id:       string
     userId:    string
     text:      string
     reviewId:  string
⇄    replyId:   string
     createdAt: string
     updatedAt: string
     _v:        integer
  }
```

ReplyArray

```
  ▾ReplyArray[
⇄    ▸Reply { }
  ]
```

ReplyCreate

```
  ▾ReplyCreate {
     userId:   string
     text:     string
⇄    reviewId: string
     replyId:  string
  }
```

Review

```
  ▾Review {
     _id:           string
     userId:        string
     text:          string
```

```
⇄     modId:          string
      semesterTaken: number
      acadYearTaken: string
      event:          ▸EventObj { }
      rating:         ▸RatingObj { }
      createdAt:      string
      updatedAt:      string
      _v:             integer
    }
```

ReviewArray

```
    ▾ReviewArray[
⇄     ▸Review { }
    ]
```

ReviewCreate

```
    ▾ReviewCreate {
      userId:         string
      text:           string
⇄     modId:          string
      semesterTaken: number
      acadYearTaken: number
    }
```

School

```
    ▾School {
      _id:       string
      name:      string
      rating:    ▸RatingObj { }
⇄     event:     ▸EventObj { }
```

```
    createdAt: string
    updatedAt: string
    _v:        integer
}
```

SchoolArray

```
▾ SchoolArray[
⇄   ▸ School { }
]
```

SchoolCreate

```
▾ SchoolCreate {
⇄   name: string
}
```

User

```
▾ User {
    _id:             string
    name:            string
    email:           string
    password:        string
    schoolStartDate: string
⇄   schoolId:        string
    courseId:        string
    currentYear:     integer
    createdAt:       string
    updatedAt:       string
    _v:              integer
}
```

UserCreate

```
▾UserCreate {
   name:           string
   email:          string
   password:       string
⇄  schoolStartDate: string
   schoolId:       string
   courseId:       string
   currentYear:    integer
}
```

Try a sample here

API Design First Workflow

Easily create API specifications

Write API specs in YAML...

...Preview documentation in Swagger