Errors

X Swagger Error

A deterministic version of a JSON Schema object.

Jump to line 1056

Details

▶ Object

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

Error

Parameters

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

Responses

Code	Description	Schema	
200	Success	▼BookmarkArray[

Try this operation

500

POST /bookmark bookmark Summary creates a bookmark **Parameters** Required Located in Description Schema Name ▼BookmarkCreate { userId: string body Bookmark object body Yes 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** Located in Description Required Name Schema bookmarkId path OID of the bookmark Yes Responses Code Description 200 Success 403 Not authorized

Try this operation

Error

/course

500

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

POST /course course Summary creates a course **Parameters** Located in Description Required Schema Name ▼ CourseCreate { body Course object Yes body name: string Responses Description Code 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
courseld	path	OID of the course	Yes	\rightleftarrows undefined

Responses

```
Description
Code
                            Schema
                               ▼Course {
                                 id:
                                          string
                                 name:
                                          string
                                 200
         Success
                                          ►EventObj { }

    event:

                                 createdAt: string
                                 updatedAt: string
                                          integer
                                 _v:
500
         Error
```

PUT /course/{courseId}

course

Summary

updates a course

Parameters

Name	Located in	Description	Required	Schema
courseld	path	OID of the course	Yes	undefined undefin
body	body	Course Object	Yes	<pre>▼CourseCreate {</pre>

Responses

500	Error
403	Not authorized
200	Success
Code	Description

DELETE /course/{courseId} course Summary deletes a course by OID **Parameters** Located in Description Required Schema Name OID of the course courseld path Yes Responses Code Description 200 Success Not authorized 403 500 Error

/event

POST /event

event

Summary

creates an event

Parameters

Name	Located in	Description	Required	Schema
body	body	Event object	Yes	<pre> vEventCreate { userId: string sub: string subId: string type: string } </pre>
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

DELETE /event

event

Summary

deletes an event

Parameters

Name	Located in	Description	Required	Schema
body	body	Event object	Yes	<pre> vEventCreate { userId: string sub: string subId: string type: string } </pre>
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

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

200	Event found	
404	Event not found	
500	Error	

GET /mod

mod

Summary

searches for modules

Description

Returns a list of appropriate modules

Parameters

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

Responses

Code	Description	Schema	
200	Success	<pre>▼ ModArray[</pre>	
500	Error		

POST /mc

mod

Summary

creates a module

Parameters

Name	Located in	Description	Required	Schema
				▼ModCreate {
				code: string
				title: string
				schoolId: string
body	body	Module object	Yes	ightarrow acadYear: string
body	Soay	Woddie object	103	semester: integer
				description: string
				credit: integer
				workload: integer
				}

Responses

500	Error
403	Not authorized
200	Success
Code	Description

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

Description

Code

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

Schema

		_v:	integer
500	Error		
Try this ope	eration		

PUT /mod/{modId}

mod

Summary

updates a module

Parameters

Name	Located in	Description	Required	Schen	na	
modId	path	OID of the module	Yes	\rightleftarrows	string	
					▼ ModCreate {	
					code:	string
					title:	string
					schoolId:	string
body	body	Module object	Yes	\rightleftarrows	acadYear:	string
body	body	Woddie Object	163	←	semester:	integer
					description	: string
					credit:	integer
					workload:	integer
					}	

Responses

500	Error
403	Not authorized
200	Success
Code	Description

DELETE /mod/{modId} mod Summary delete a module by OID **Parameters** Located in Description Required Schema Name modId OID of the module path Yes ⇒ string Responses Code Description 200 Success Not authorized 403 500 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[

Try this operation

Error

500

POST /planned-mod

planned mod

Summary

adds a module to a user's study plan

Parameters

Name	Located in	Description	Required	Schem	a	
				•	PlannedMod	Create {
body	body	PlannedMod Object	Yes	⇄	<pre>modId: userId: semester:</pre>	string string 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	<pre>▼PlannedModCreate { modId: string userId: string semester: integer }</pre>

Responses

500	Error
403	Not authorized
200	Success
Code	Description

DELETE /planned-mod/{plannedModId} planned mod Summary delete a planned mod by OID **Parameters** Located in Description Required Schema Name plannedModId OID of the planned module path Yes string Responses Code Description 200 Success Not authorized 403 500 Error

/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
				▼Login {
body	body	Login Object	Yes	<pre> email: string password: string </pre>
				}

Responses

500

Error

Code	Description	Schem	a		
		•	User {		
			_id:	string	
			name:	string	
			email:	string	
			password:	string	
			schoolStartDate	: string	
200	Success	\rightleftarrows	schoolId:	string	
			courseId:	string	
			currentYear:	integer	
			createdAt:	string	
			updatedAt:	string	
			_v:	integer	
		}			
403	Not authorized				

Try this operation

/rating

POST /rating

rating

Summary

creates a rating

Parameters

Name	Located in	Description	Required	Schema
body	body	Rating Object	Yes	<pre>vRatingCreate { userId: string sub: string zubId: string type: string value: integer }</pre>
sub	body	Type of subscriber to the rating	Yes	string ⇄ Enum: ▶ Array [3]
subId	body	OID of subscriber object	Yes	
type	body	Type of rating	Yes	string Շ Enum: ▶ Array [2]
value	body	Rating number	Yes	⇄ integer

Responses

500	Error
403	Not authorized
200	Success
Code	Description

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				
		▼Rating {				
		_id: string				
		userId: string				
		sub: string				
		subId: string				
200	Rating found	ightharpoonup type: string				
	3	value: integer				
		<pre>createdAt: string</pre>				
		updatedAt: string				

```
404 Rating not found

500 Error

Try this operation
```

/rating/{ratingId}

PUT /rating/{ratingId} rating Summary updates a rating **Parameters** Located in Description Required Schema Name ratingId OID of the rating path Yes ▼RatingCreate { userId: string sub: string body body Rating Object Yes subId: string type: string value: integer Responses Code Description 200 Success Not authorized 403 500 Error Try this operation

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	
type	body	Type of reaction	Yes	string Enum: ▶ Array [3]

Responses

Code	Description
200	Success
403	Not authorized
500	Error

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	Schem	Schema		
		•	Reaction {		
			_id:	string	
			userId:	string	
			sub:	string	
			subId:	string	
200	Reaction found	\rightleftarrows	type:	string	
			<pre>createdAt:</pre>	string	
			updatedAt:	string	
			_v:	integer	

404 Reaction not found

500 Error

Try this operation

/reaction/{reactionId}

					reactio
Summary					
deletes a reaction by OID					
Parameters					
Name	Located in	Description	Required	Schema	
reactionId	path	OID of the reaction	Yes	\rightleftarrows undefined	
Responses					
Code		Description			
200		Success			
500		Error			

POST /reply reply Summary creates a reply **Parameters** Required Located in Description Schema Name ▼ ReplyCreate { userId: string string text: body body Reply object Yes 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

		▼]	Reply { _id: userId:	string
				-
			userId:	at min a
				string
200 Success			text:	string
	Success	⇄	reviewId:	string
200	Success	←	replyId:	string
			<pre>createdAt:</pre>	string
			updatedAt:	string
			_v:	integer
		}	_	-

Try this operation

Error

500

PUT /reply/{replyId}

reply

Summary

updates a reply

Parameters

Name	Located in	Description	Required	Schema
replyId	path	OID of the reply	Yes	undefined undefin
body	body	Reply Object	Yes	<pre> vReplyCreate { userId: string text: string reviewId: string replyId: string } </pre>

Responses

500	Error
403	Not authorized
200	Success
Code	Description

DELETE /reply/{replyId} reply Summary deletes a reply by OID **Parameters** Located in Description Required Schema Name replyId path OID of the reply Yes Responses Code Description 200 Success Not authorized 403 500 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 OID of the review path Yes □ undefined Responses Description Schema Code ▼ReplyArray[200 Success ►Reply { } **500** Error

/review

GET /review

review

Summary

gets all reviews

Description

Returns a list of reviews

Responses

Code Description Schema

Z00 Success

ReviewArray[

Review { }

500 Error

POST /review review Summary creates a review **Parameters** Required Schema Located in Description Name ▼ ReviewCreate { userId: string string text: body body **Review Object** Yes modId: string semesterTaken: number acadYearTaken: number Responses Description Code 200 Success

Not authorized

Error

Try this operation

403

500

/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
                                  ▼Review {
                                     id:
                                                   string
                                     userId:
                                                   string
                                     text:
                                                   string
                                                   string
                                     modId:
                                     semesterTaken: number
200
          Success
                                    acadYearTaken: string
                                                   ►EventObj { }
                                     event:
                                     rating:
                                                   ▶RatingObj { }
                                     createdAt:
                                                   string
                                     updatedAt:
                                                   string
                                                   integer
```

Try this operation

Error

500

PUT /review/{reviewId}

review

Summary

updates a review

Parameters

Name	Located in	Description	Required	Schen	na		
reviewId	path	OID of the review	Yes	\rightleftarrows	string		
					▼ ReviewCreate	· {	
					userId:	string	
					text:	string	
body	body	Review Object	Yes	\rightleftharpoons	modId:	string	
					semesterTak	en: number	
					acadYearTak	en: number	
					}		

Responses

Code	Description
200	Success
403	Not authorized
500	Error

DELETE /review/{reviewId} review Summary deletes a review by OID **Parameters** Located in Description Required Schema Name reviewId OID of the review path Yes Responses Code Description 200 Success Not authorized 403 500 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
                                   ▼Review {
                                     id:
                                                    string
                                     userId:
                                                    string
                                     text:
                                                    string
                                     modId:
                                                    string
                                     semesterTaken: number
200
          Success
                                     acadYearTaken: string
                                     event:
                                                    ►EventObj { }
                                     rating:
                                                   ▶RatingObj { }
                                     createdAt:
                                                    string
                                     updatedAt:
                                                    string
                                                    integer
                                     _v:
```

Schema

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
                                   ▼Review {
                                      id:
                                                    string
                                      userId:
                                                    string
                                      text:
                                                    string
                                      modId:
                                                    string
                                      semesterTaken: number
200
          Success
                                     acadYearTaken: string
                                      event:
                                                     ►EventObj { }
                                      rating:
                                                     ▶RatingObj { }
                                      createdAt:
                                                     string
                                      updatedAt:
                                                     string
                                                     integer
                                      _v:
```

Schema

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

POST /school school Summary creates a school **Parameters** Located in Description Required Schema Name ▼ SchoolCreate { body School Object Yes body name: string Responses Description Code 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
schoolld	path	OID of the school	Yes	\rightleftarrows undefined

Responses

```
Description
Code
                           Schema
                               ▼School {
                                id:
                                         string
                                name:
                                         string
                                200
         Success
                                         ►EventObj { }
                                event:
                                createdAt: string
                                updatedAt: string
                                         integer
                                _v:
```

Try this operation

Error

500

PUT /school/{schoolId}

school

Summary

updates a school

Parameters

Name	Located in	Description	Required	Schema
schoolld	path	OID of the school	Yes	⇒ string
body	body	School Object	Yes	<pre>▼SchoolCreate {</pre>

Responses

Code	Description
200	Success
403	Not authorized
500	Error

DELETE /school/{schoolId} school Summary deletes a school by OID **Parameters** Located in Description Required Schema Name schoolId path OID of the school Yes Responses Code Description 200 Success Not authorized 403 500 Error Try this operation

/user

Summary creates a user Parameters Name Located in Description Required Schema string email: string email: string schoolId: string courseId:							user	POST /
Parameters Name Located in Description Required Schema VuserCreate { name string email string password string password string schoolIdar	user							
Parameters Name Located in Located in Description Description Required Responses Required Schema TuserCreate { name: string email: string password: string schoolIstartDate: string schoolIstartDate: string schoolIstartDate: string courseId: string currentYear: integer } string required Responses email body Email of the user (only accept .edu emails) Yes ≠ string currentYear: integer required required Responses Code Description							ary	Summ
Name Located in Description Required VUserCreate { name: string email: string password: string password: string schoolStartDate: string schoolId: string schoolId: string courseId: string schoolId: string schoolId: string courseId: string schoolId: string courseId: string schoolId: string courseId: string schoolId: string courseId: string courseId: string courseId: string schoolId: string courseId: string courseId: string courseId: string schoolId: string courseId: string courseId: string courseId: string courseId: string courseId: string schoolId: string courseId: string c							user	creates a
body body User Object Pemail body Email of the user (only accept .edu emails) Responses Code Description Puser Create { name: string email: string password: string schoolStartDate: string schoolId: string courseId: string currentYear: integer } Prosponses Pro							eters	Param
body body User Object Yes Yes Password: string password: string schoolStartDate: string schoolId: string courseId: string currentYear: integer Pemail body Email of the user (only accept .edu emails) Responses Code Description Description			Schema	Sche	Required	Description	Located in	Name
body body User Object Yes Yes Password: string password: string schoolStartDate: string schoolId: string courseId: string currentYear: integer			▼UserCreate {					
body body User Object Yes Password: string schoolStartDate: string schoolId: string courseId: string currentYear: integer Password: string schoolId: string courseId: string currentYear: integer Password: string schoolId: string courseId: string currentYear: integer Password: string schoolId: string currentYear: string currentY								
body body User Object Yes schoolStartDate: string schoolId: string courseId: string currentYear: integer } email body Email of the user (only accept .edu emails) Yes ≥ string Responses Code Description 200		=						
schoolId: string courseId: string currentYear: integer } email body Email of the user (only accept .edu emails) Yes Responses Code Description 200		=	=	\rightarrow	Voc	Lloor Object	hody	hody
email body Email of the user (only accept .edu emails) Responses Code Description 200		=		←	165	Oser Object	body	body
email body Email of the user (only accept .edu emails) Responses Code Description 200		=						
email body Email of the user (only accept .edu emails) Responses Code Description 200		=						
Responses Code Description 200								
Code Description 200			⇒ string	⇄	Yes	Email of the user (only accept .edu emails)	body	email
200							nses	Respo
						Description		Code
403 Not authorized								200
						Not authorized		403
500 Error						Error		500
Try this operation							on ovation	Tm, th:-

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	

Responses

Code	Description	Schema
		▼User {
		_id: string
		name: string
		email: string
		password: string
		schoolStartDate: string
200	Success	⇒ schoolId: string
		courseId: string
		currentYear: integer
		createdAt: string
		updatedAt: string
		_v: integer
		}
500	Error	

PUT /user/{userId}

user

Summary

updates a user

Parameters

Name	Located in	Description	Required	Schen	na	
userId	path	OID of the user	Yes	→ string		
				▼UserCreate {		
					name:	string
					email:	string
					password:	string
body	body	User Object	Yes	\rightleftarrows	schoolStartDate	: string
					schoolId:	string
					courseId:	string
					currentYear:	integer
				•	}	

Responses

500	Error
403	Not authorized
200	Success
Code	Description

DELETE /user/{userId} user Summary deletes a user by OID **Parameters** Located in Description Required Name Schema path OID of the user userId Yes Responses Code Description 200 Success Not authorized 403 500 Error

Models

ightleftarrows

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 {
                string
     _id:
                string
     name:
                ▶RatingObj { }
     rating:
                ▶EventObj { }
     event:
     createdAt: string
     updatedAt: string
                integer
     _v:
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
\rightleftarrows
     semester:
                   integer
     description: string
     credit:
                   integer
     workload:
                   integer
Mod
   ▼Mod {
     _id:
                   string
     code:
                   string
     title:
                   string
     schoolId:
                   string
     acadYear:
                   string
                   integer
     semester:
     description: string
     credit:
                   integer
     workload:
                   integer
                   ▶[]
     prereqs:
                   ▶RatingObj { }
     rating:
                   ▶EventObj { }
      event:
     reaction:
                   ▶ReactionObj { }
     createdAt:
                   string
     updatedAt:
                   string
                   integer
     _v:
```

```
ModArray
   ▼ModArray[
      ▶ Mod { }
PlannedMod
   ▼PlannedMod {
     _id:
                         string
     modId:
                         string
                         string
      userId:
      semester:
                         integer
      mod:
                         ▶ Mod { }
\rightleftarrows
     missingPrereqs:
                         ▶[]
     allPrereqsPresent: boolean
     createdAt:
                         string
     updatedAt:
                         string
                         integer
      _v:
PlannedModCreate
   ▼PlannedModCreate {
     modId:
                string
     userId:
               string
     semester: integer
PlannedModArray
   ▼ PlannedModArray[
      ▶PlannedMod { }
```

```
Rating
   ▼Rating {
     _id:
     userId:
     sub:
     subId:
     type:
     value:
     createdAt: string
     updatedAt: string
     _v:
```

RatingCreate

RatingProp

```
▼RatingCreate {
     userId: string
     sub:
             string
     subId: string
     type:
             string
     value: integer
RatingObj
   ▼RatingObj {
     difficulty: >RatingProp { }
                 ▶RatingProp { }
     star:
```

string

string

string

string

string

integer

integer

```
▼RatingProp {
     count: integer
     value: number
Reaction
   ▼Reaction {
     id:
              string
     userId:
              string
     sub:
              string
     subId:
              string
     type:
              string
     createdAt: string
     updatedAt: string
              integer
     _v:
ReactionCreate
ReactionObj
   ▼ReactionObj {
     difficulty: >ReactionProp { }
               ▶ ReactionProp { }
     star:
ReactionProp
   ▼ReactionProp {
     count: integer
```

Reply

```
▼Reply {
     _id:
                 string
     userId:
                string
     text:
                 string
     reviewId: string
     replyId:
                string
     createdAt: string
     updatedAt: string
                 integer
      _v:
ReplyArray
   ▼ReplyArray[
      ▶Reply { }
ReplyCreate
   ▼ReplyCreate {
     userId:
               string
     text:
                string
\rightleftharpoons
     reviewId: string
     replyId: string
Review
   ▼Review {
     id:
                     string
     userId:
                     string
     text:
                     string
```

```
modId:
                     string
      semesterTaken: number
\rightleftharpoons
      acadYearTaken: string
                     ▶EventObj { }
      event:
      rating:
                     ▶RatingObj { }
      createdAt:
                     string
      updatedAt:
                     string
                     integer
      _v:
ReviewArray
   ▼ReviewArray[
      ▶Review { }
ReviewCreate
   ▼ ReviewCreate {
      userId:
                     string
      text:
                     string
     modId:
                     string
      semesterTaken: number
      acadYearTaken: number
School
   ▼School {
     _id:
                 string
                 string
      name:
                 ▶RatingObj { }
      rating:
      event:
                 ▶EventObj { }
```

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

UserCreate

```
vUserCreate {
    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