Introduction

Welcome to the Spouser API!

Down here you will fond some endpoints, examples, etc. to get you going.

TOC

Pre-face

All the `POST`, `PUT` and `PATCH` endpoints, should be able to accept `json`. But of-course the good ol' form parameters will always work.

> The default headers, which I bet you could guess would look something like this.

```
headers: {
  // 'Authorization': 'Bearer <GENERATED_ACCESS_TOKEN>',
  'Content-Type': 'application/x-www-form-urlencoded',
```

```
    'Accept-Language': 'application/json',
    'X-Requested-With': 'XMLHttpRequest'
  }
```

Authentication / Authorization

Example

```
// You can just pass the correct header with each request
var options = {
  url: 'http://localhost/api/v1/<authorized_endpoint>',
  headers: {
    'Authorization': 'Bearer <GENERATED_ACCESS_TOKEN>'
  }
};

request(options, function (error, response) {
  if (error) throw new Error(error);

  console.log(body);
});
```

> Make sure to replace `<GENERATED_ACCESS_TOKEN>` with your API key.

For most routes you MUST be authenticated to the the API.

Using the headers in your preferred method, add the following:

`Authorization: Bearer <GENERATED_ACCESS_TOKEN>`

Note You must replace <GENERATED_ACCESS_TOKEN> with the responded token from the server.

Note If an endpoint requires authentication and it has no valid user, a `401 Unauthorized` will be thrown.

Note If an endpoint is not available for a given user, a `403 Forbidden` will be thrown.

Authenticate a user

This endpoint sign a user in.

HTTP Request

`POST http://localhost/api/v1/auth/login`

Form Parameters

All these parameters are required.

Parameter	Description

Parameter	Description
email	The user's email address.
password	The prefered password.

Example

```
var request = require("request");

var options = {
  method: 'POST',
  url: 'http://localhost/api/v1/auth/login',
  body: '{
    email: "user@example.com",
    password: "P@ssw0rd."
  }'
};

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

> The above command returns JSON structured like this:

```
{
    "message": null,
    "data": {
        "auth": {
            "access_token": "<GENERATED_ACCESS_TOKEN>",
            "token_type": "bearer",
            "expires_in": 3600
        }
    },
    "errors": []
}
```

Refresh token

This should get you a whole new bearer token.

HTTP Request

`POST http://localhost/api/v1/auth/refresh`

Example

```
var options = {
  method: 'POST',
  url: 'http://localhost/api/v1/auth/refresh',
  headers: {
    'Authorization': 'Bearer <GENERATED_AUTH_TOKEN>
  }',
  body: '{"email": "user@example.com", "password": "SuperSecret"}
};

request(options, function (error, response) {
  if (error) throw new Error(error);

  console.log(body);
});
```

> The above command returns JSON structured like this:

```
{
    "message": null,
    "data": {
        "auth": {
            "access_token": "<GENERATED_ACCESS_TOKEN>",
            "token_type": "bearer",
            "expires_in": 3600
        }
    },
    "errors": []
}
```

Logout

Logging out the logged in user. Although this shouldn't happen very often, we still need implement it.

HTTP Request

`POST http://localhost/api/v1/auth/logout`

Example

```
var request = require("request");

var options = {
  method: 'POST',
  url: 'http://localhost/api/v1/auth/logout',
  headers: {
    'Authorization': 'Bearer <GENERATED_AUTH_TOKEN>
  }'
};
```

```
request(options, function (error, response) {
  if (error) throw new Error(error);

  console.log(body);
});
```

> The above command returns JSON structured like this:

```
{
  "message": "Successfully logged out",
  "data": [],
  "errors": []
}
```

Get the authenticated user

Get the current signed in user.

HTTP Request

```
GET http://localhost/api/v1/auth/me
```

Example

```
var request = require("request");

var options = {
  method: 'GET',
  url: 'http://localhost/api/v1/auth/me',
  headers: {
    'cache-control': 'no-cache',
    'Content-Type': 'application/json',
    'X-Requested-With': 'XMLHttpRequest',
    'Authorization': 'Bearer <GENERATED_AUTH_TOKEN>
  }'
};

request(options, function (error, response) {
  if (error) throw new Error(error);

  console.log(body);
});
```

> The above command returns JSON structured like this:

```
{
  "message": null,
  "data": {
    "id": 1001,
    "me": true,
    "name": "User",
    "email": "user@example.com",
    "verified": false,
    "role": "User",
    "profile": {
      "age": 38,
      "date_of_birth": "1980-12-31"
    },
    "links": {
      "profile": "http://localhost/api/v1/auth/profile/1001"
    },
    "auth": {
      "access_token": "<GENERATED_AUTH_TOKEN>",
      "token_type": "bearer",
      "expires_in": 3600
    }
  },
  "errors": []
}
```


Users

Register a user

This endpoint registers a user. If there are no errors thrown, the user will automatically be singed in.

HTTP Request

`POST http://localhost/api/v1/register`

Form Parameters

All these parameters are required.

Parameter	Description	Available options
name	The username.	
email	The user's email address.	
password	The prefered password.	
password_confirmation	The user's needs to validate the given password.	

Parameter	Description	Available options
gender_id	The prefered gender.	1) male, 2) female, 3) other/both
feels_gender_id	Which gender the user feels like.	1) male, 2) female, 3) other/both
search_gender_id	The initial gender to search on.	1) male, 2) female, 3) other/both
date_of_birth	The date of birth.	Formatted like `yyyy-mm-dd`
terms_and_conditions	User must accepted these.	
privacy_statement	User must accept this.	

Example

```
var request = require("request");

var options = {
  method: 'POST',
  url: 'http://localhost/api/v1/register',
  headers: {
    'cache-control': 'no-cache',
    // 'Content-Type': 'application/json',
    'Content-Type': 'application/x-www-form-urlencoded',
    'X-Requested-With': 'XMLHttpRequest'
  },
  body: '{
    name: "user",
    email: "user@example.com",
    password: "P@ssw0rd.",
    password_confirmation: "P@ssword.",
    gender_id: "1",
    feels_gender_id: "1",
    search_gender_id: "1",
    date_of_birth: "1980-12-31",
    privacy_statement: "1",
    terms_and_conditions: "1"
  }'
};

request(options, function (error, response, body) {
  if (error) throw new Error(error);

  console.log(body);
});
```

> The above command returns JSON structured like this:

```
{
  "message": null,
  "data": {
    "id": 1001,
    "me": true,
    "name": "User",
    "email": "user@example.com",
    "verified": false,
    "role": "User",
    "profile": {
      "age": 38,
      "date_of_birth": "1980-12-31"
    },
    "links": {
      "profile": "http://localhost/api/v1/auth/profile/1001"
    },
    "auth": {
      "access_token": "<GENERATED_AUTH_TOKEN>",
      "token_type": "bearer",
      "expires_in": 3600
    }
  },
  "errors": []
}
```

Validation Rules

Example response

The response header for request validations will be the `422 Unprocessable Entity` with the following kind of response body:

```
{
  "message": "The given data was invalid.",
  "errors": {
    "name": [
        "validation.min.string"
    ],
    "email": [
        "validation.required"
    ]
}
```

TODO

TODO Return the values for the rules. Like :value, :date, etc.

item	text
validation.accepted	The :attribute must be accepted.
validation.active_url	The :attribute is not a valid URL.
validation.after	The :attribute must be a date after :date.
validation.after_or_equal	The :attribute must be a date after or equal to :date.
validation.alpha	The :attribute may only contain letters.
validation.alpha_dash	The :attribute may only contain letters, numbers, dashes and underscores.
validation.alpha_num	The :attribute may only contain letters and numbers.
validation.array	The :attribute must be an array.
validation.before	The :attribute must be a date before :date.
validation.before_or_equal	The :attribute must be a date before or equal to :date.
validation.between.array	The :attribute must have between :min and :max items.
validation.between.file	The :attribute must be between :min and :max kilobytes.
validation.between.numeric	The :attribute must be between :min and :max.
validation.between.string	The :attribute must be between :min and :max characters.
validation.boolean	The :attribute field must be true or false.
validation.confirmed	The :attribute confirmation does not match.
validation.date	The :attribute is not a valid date.
validation.date_equals	The :attribute must be a date equal to :date.
validation.date_format	The :attribute does not match the format :format.
validation.different	The :attribute and :other must be different.
validation.digits	The :attribute must be :digits digits.
validation.digits_between	The :attribute must be between :min and :max digits.
validation.dimensions	The :attribute has invalid image dimensions.
validation.distinct	The :attribute field has a duplicate value.
validation.email	The :attribute must be a valid email address.
validation.ends_with	The :attribute must end with one of the following: :values
validation.exists	The selected :attribute is invalid.
validation.file	The :attribute must be a file.
validation.filled	The :attribute field must have a value.
validation.gt.array	The :attribute must have more than :value items.

item	text
validation.gt.file	The :attribute must be greater than :value kilobytes.
validation.gt.numeric	The :attribute must be greater than :value.
validation.gt.string	The :attribute must be greater than :value characters.
validation.gte.array	The :attribute must have :value items or more.
validation.gte.file	The :attribute must be greater than or equal :value kilobytes.
validation.gte.numeric	The :attribute must be greater than or equal :value.
validation.gte.string	The :attribute must be greater than or equal :value characters.
validation.image	The :attribute must be an image.
validation.in	The selected :attribute is invalid.
validation.in_array	The :attribute field does not exist in :other.
validation.integer	The :attribute must be an integer.
validation.ip	The :attribute must be a valid IP address.
validation.ipv4	The :attribute must be a valid IPv4 address.
validation.ipv6	The :attribute must be a valid IPv6 address.
validation.json	The :attribute must be a valid JSON string.
validation.lt.array	The :attribute must have less than :value items.
validation.lt.file	The :attribute must be less than :value kilobytes.
validation.lt.numeric	The :attribute must be less than :value.
validation.lt.string	The :attribute must be less than :value characters.
validation.lte.array	The :attribute must not have more than :value items.
validation.lte.file	The :attribute must be less than or equal :value kilobytes.
validation.lte.numeric	The :attribute must be less than or equal :value.
validation.lte.string	The :attribute must be less than or equal :value characters.
validation.max.array	The :attribute may not have more than :max items.
validation.max.file	The :attribute may not be greater than :max kilobytes.
validation.max.numeric	The :attribute may not be greater than :max.
validation.max.string	The :attribute may not be greater than :max characters.
validation.mimes	The :attribute must be a file of type: :values.
validation.mimetypes	The :attribute must be a file of type: :values.
validation.min.array	The :attribute must have at least :min items.

item	text
validation.min.file	The :attribute must be at least :min kilobytes.
validation.min.numeric	The :attribute must be at least :min.
validation.min.string	The :attribute must be at least :min characters.
validation.not_in	The selected :attribute is invalid.
validation.not_regex	The :attribute format is invalid.
validation.numeric	The :attribute must be a number.
validation.present	The :attribute field must be present.
validation.regex	The :attribute format is invalid.
validation.required	The :attribute field is required.
validation.required_if	The :attribute field is required when :other is :value.
validation.required_unless	The :attribute field is required unless :other is in :values.
validation.required_with	The :attribute field is required when :values is present.
validation.required_with_all	The :attribute field is required when :values are present.
validation.required_without	The :attribute field is required when :values is not present.
validation.required_without_all	The :attribute field is required when none of :values are present.
validation.same	The :attribute and :other must match.
validation.size.array	The :attribute must contain :size items.
validation.size.file	The :attribute must be :size kilobytes.
validation.size.numeric	The :attribute must be :size.
validation.size.string	The :attribute must be :size characters.
validation.starts_with	The :attribute must start with one of the following: :values
validation.string	The :attribute must be a string.
validation.timezone	The :attribute must be a valid zone.
validation.unique	The :attribute has already been taken.
validation.uploaded	The :attribute failed to upload.
validation.url	The :attribute format is invalid.
validation.uuid	The :attribute must be a valid UUID.

Errors

The Spouser API uses the following error codes:

Error Code	Meaning
400	Bad Request -- Your request is invalid.
401	Unauthorized -- Your API key is wrong.
403	Forbidden -- The user is not authenticated to access this route.
404	Not Found -- The specified user could not be found.
405	Method not allowed -- You requested a method that doesn't like to do this.
422	Unprocessable Entity -- Form validation error.
429	Too Many Requests -- Too many requestes! Slow down!
500	Internal Server Error -- We had a problem with our server. Try again later.
503	Service Unavailable -- We're temporarily offline for maintenance. Please try again later.