

Provocation engine

# Table of Contents

1. Introduction .....	1
2. Specification .....	2
2.1. Public-facing home page .....	2
2.1.1. Conceptual design .....	2
2.2. Management interface .....	3
2.2.1. Login page .....	3
2.2.2. Messages page .....	4
2.2.2.1. Conceptual design .....	4
2.2.3. Feedback page .....	6
2.2.3.1. Conceptual design .....	6
2.3. Components .....	8
2.3.1. In-page feedback .....	8
2.3.1.1. Conceptual design .....	8
2.3.1.2. Behaviour .....	9
2.3.2. Main menu .....	9
2.3.2.1. Conceptual design .....	9
2.3.3. Table .....	9
2.3.3.1. Conceptual design .....	9
2.3.4. Modal pop-up window .....	10
2.3.4.1. Conceptual design .....	10
3. Requirements .....	11
3.1. Database .....	11
3.2. API .....	11
3.2.1. GetMessages .....	11
3.2.2. GetMessage .....	12
3.2.3. CreateMessage .....	12
3.2.4. UpdateMessage .....	13

# Chapter 1. Introduction

The Provocation Engine is a simple ideation tool that shows provocative arguments to help users think out of the box. The application was prototyped and tested during the Namahn 2018 Summer Camp.

The Provocation Engine has a public-facing home page and a management interface.

# Chapter 2. Specification

## 2.1. Public-facing home page

### 2.1.1. Conceptual design

The public-facing home page is available to all users. No login is necessary.

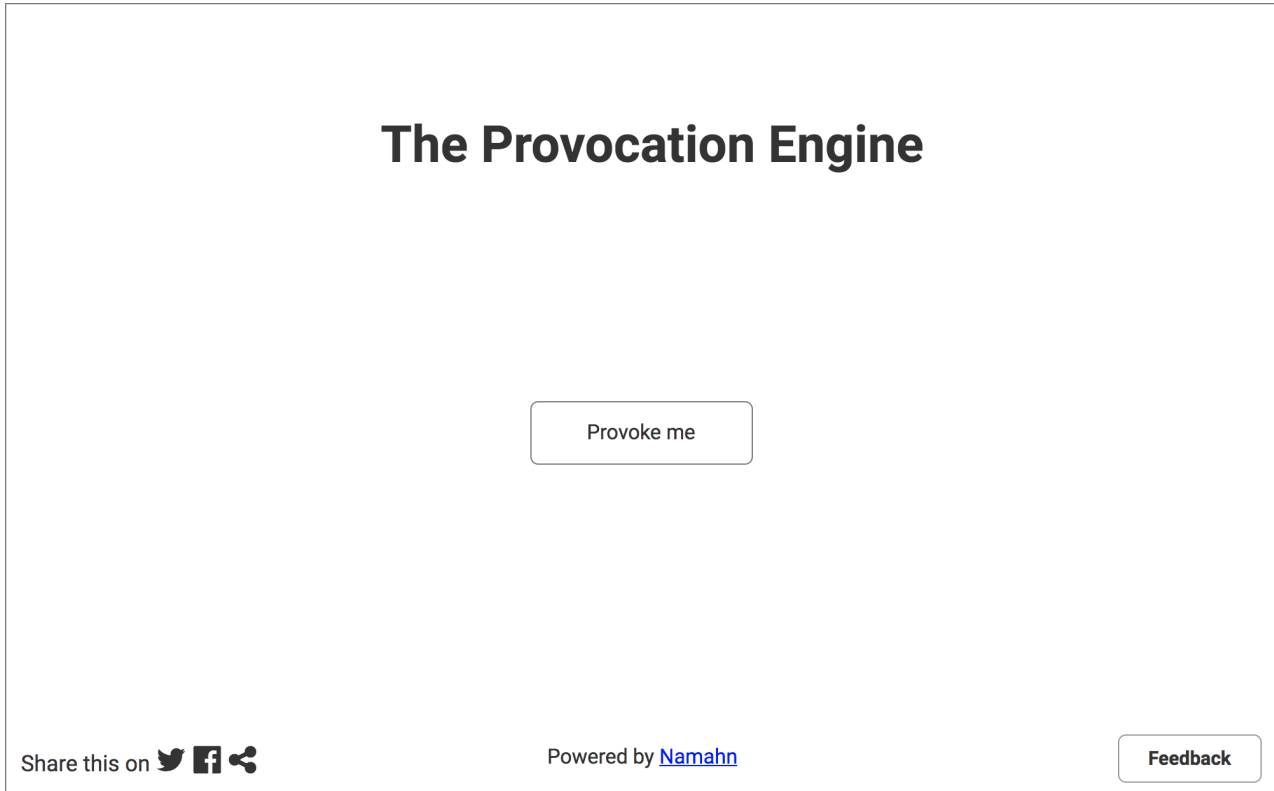


Figure 1. Public-facing home page of the Provocation Engine

The public-facing home page contains the following elements:

- **Page title:** tells users they are visiting the Provocation Engine. The page title could be text and/or a logo.
- **Provocation message:** a two-line message. When the user first visits the public-facing home page, no provocation message is shown.
- **Action button:** a button with the label *Provoke me*. Clicking the action button shows a random provocation message.
- **Namahn link:** indicates that the Provocation Engine is "Powered by Namahn".
- **Feedback:** an [in-page feedback](#) component.
- **Share buttons:** a number of buttons to share the Provocation Engine public-facing home page on social media.



need to find out what channels to share to — Twitter, Facebook, LinkedIn?, other?

## 2.2. Management interface

The Provocation Engine has a password-protected management interface.

The management interface consists of three pages: the login page, the messages page and the feedback page.

### 2.2.1. Login page

When non-authenticated users attempts to visit any management interface page, they are shown the login page.

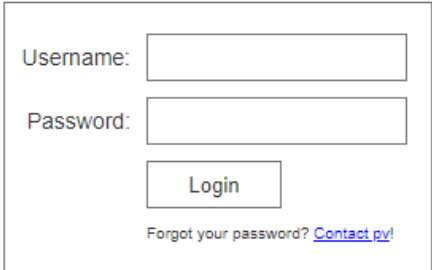
The image shows a login form centered on a white background. The form is enclosed in a thin black border. It contains two text input fields: the first is labeled 'Username:' and the second is labeled 'Password:'. Below these fields is a button labeled 'Login'. At the bottom of the form, there is a link that says 'Forgot your password? [Contact.py!](#)'.

Figure 2. Provocation Engine management interface: login page for unauthenticated users

The login page contains a form with the following elements:

- **Username:** a plain text field to enter a username
- **Password:** a password field to enter a password
- **Action button:** a button with the label *Login* to submit the form. The button is *disabled* as long as either the *username* or *password* fields are empty.

When the users enters an incorrect name and/or password, the login page advises the user she has done so with an **error message** between the *password* field and the *action button*. The name that was entered remains in the field; the password field is emptied.

Username:

Password:

The username or password is incorrect.  
Try again or [contact pv](#) for help.

Figure 3. Login: error message

## 2.2.2. Messages page

### 2.2.2.1. Conceptual design

The messages page allows an authenticated user to create new messages, and edit and delete existing messages. A message consist of two lines of plain text. The first line is no longer than 40 characters; the second line is no longer than 100 characters.

**The Provocation Engine**    Messages    Feedback    [Log out](#)

**Messages**

<input type="checkbox"/>	Line 1	Line 2
<input type="checkbox"/>	Back to the future	The year is now 2106, but the issue remains. How do you intervene?
<input type="checkbox"/>	Blast from the past	Do it like it's 1921
<input type="checkbox"/>	Brussels, Sahara desert	Brussels is a city in the middle of the Sahara desert
<input type="checkbox"/>	Blindness	A terrible disease has caused everyone to go blind
<input type="checkbox"/>	Elders only	90% of Brussels citizens are over 70 years old.
<input type="checkbox"/>	Neverland	No adults here.
<input type="checkbox"/>	Orwellian dystopia	Commuting is banned and considered illegal.
<input type="checkbox"/>	No electricity	It was never invented.
<input type="checkbox"/>	No private property	No owners. Only users.
<input type="checkbox"/>	Exchange economy	Money was never invented
<input type="checkbox"/>	Life on the move	No one stands still. Ever. Commuting is living

Figure 4. Message page

The messages page consists of:

- a **main menu**
- a **page title**, "Messages"
- a **table** showing all messages, with two **action buttons**, "Add message" and "Delete"

#### 2.2.2.1.1. Messages table

The table contains the following columns:

- **Selection column:** contains a check box for every row, allowing the user to select or deselect that row. The selection column header also contains a checkbox to select or deselect all rows in the table at once.
- **Line 1:** contains the first line of the message.
- **Line 2:** contains the second line of the message.

#### 2.2.2.1.2. Creating, editing and deleting messages

To **create a new message**, the user clicks the **Create message** primary action button. A modal pop-up opens, which contains the following:

- a **window header**, containing:
  - a **window title**, "New message"
  - a **Cancel** action link. Clicking the *Cancel* action link closes the modal window without warning.
  - a **Save** primary action button. The *Save* primary action button is disabled by default. The *Save* primary action button becomes enabled when both the *Line 1* and *Line 2* fields contain at least one character. Clicking the *Save* primary action button saves the message and closes the modal window.
- two plain text **form fields**:
  - **Line 1:** the first line of the new message, at most 40 characters long.
  - **Line 2:** the second line of the new message, at most 100 characters long.

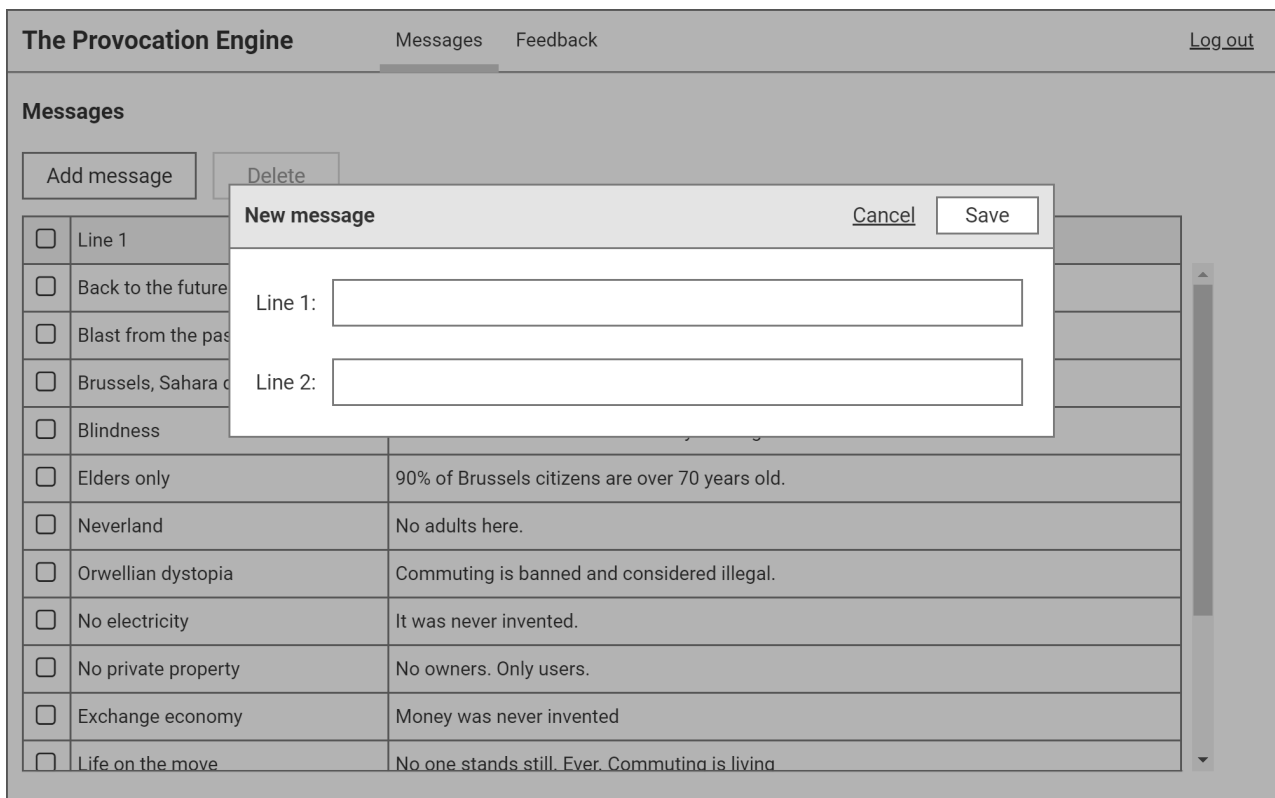


Figure 5. Adding a new message.

To **edit a message**, the user clicks the text in a row (either Line 1 or Line 2). The row is then outlined and *cancel* and *Save* links are added to the row. Both Line 1 and Line 2 for that row can be edited in place. To cancel the edit and revert to what the row was before editing, the user clicks the *cancel* link. To save the edit, the user can either click the *Save* link, or click anywhere outside Line 1 and Line 2.

<input type="checkbox"/>	Blast from the past	Do it like it's 1921	
<input type="checkbox"/>	Brussels, Sahara desert	Brussels is a city in the middle of the Sahara desert	
<input type="checkbox"/>	Blindness	A terrible disease has caused everyone to go blind	

<input type="checkbox"/>	Blast from the past	Do it like it's 1921	
<input type="checkbox"/>	Gent, Sahara desert	Gent is a city in the middle of the Sahara desert	cancel Save
<input type="checkbox"/>	Blindness	A terrible disease has caused everyone to go blind	

Figure 6. In-place editing of a message. Top: table row before editing. Bottom: table row while editing.

To **delete one or more messages**, the user selects one or more rows in the messages table and clicks the **Delete** action button. The *Delete* action button is only enabled if at least one row is selected. When clicking the *Delete* action button, the user gets a **warning** to confirm the deletion: "Are you sure you want to delete [number of selected messages] messages?"

## 2.2.3. Feedback page

### 2.2.3.1. Conceptual design

The feedback page allows an authenticated user to view collected feedback, and export and delete feedback items. A feedback item consists of a timestamp, feedback (positive, meh or negative), and



optional additional plain text feedback.

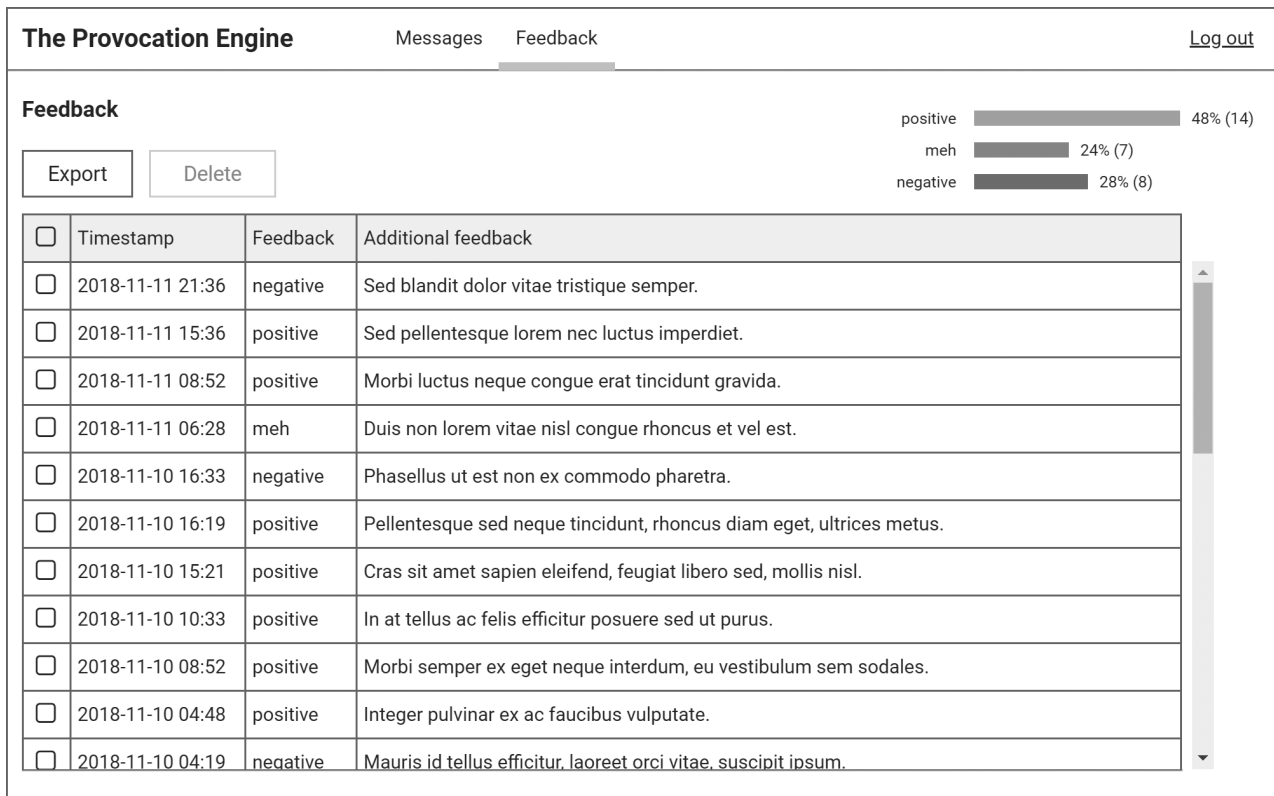


Figure 7. Feedback page

The feedback page consists of:

- a **main menu**
- a **page title**, "Feedback"
- a **bar chart** showing the total feedback at a glance
- a **table** showing all feedback, with two **action buttons**, "Export" and "Delete"

#### 2.2.3.1.1. Bar chart

The bar chart shows the absolute totals and percentage values for positive, meh and negative feedback.

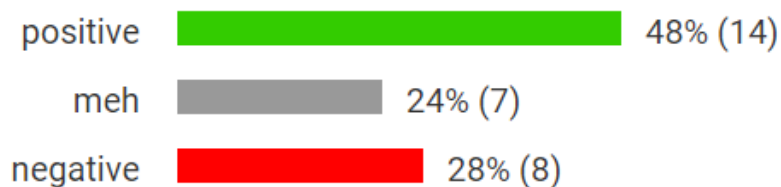


Figure 8. Bar chart

#### 2.2.3.1.2. Feedback table

The table contains the following columns:

- **Selection column:** contains a check box for every row, allowing the user to select or deselect that row. The selection column header also contains a checkbox to select or deselect all rows in the table at once.

- **Timestamp:** contains the timestamp (date and time) of the feedback.
- **Feedback:** contains the feedback (positive, meh or negative)
- **Additional feedback:** contains optional additional feedback.

#### 2.2.3.1.3. Exporting and deleting feedback

To **export feedback**, the user clicks selects zero, one or more rows in the table, and clicks the the **Export** primary action button. The user is then prompted to save an Excel version of the selected rows; if no rows are selected, all rows are exported.

To **delete one or more rows**, the user selects one or more rows in the feedback table and clicks the **Delete** action button. The *Delete* action button is only enabled if at least one row is selected. When clicking the *Delete* action button, the user gets a **warning** to confirm the deletion: "Are you sure you want to delete [number of selected feedback items] feedback items?"

## 2.3. Components

### 2.3.1. In-page feedback

The in-page feedback form allows users to give feedback about the page they are currently on. The feedback can be simple (positive / meh / negative), and optionally more extensive.

#### 2.3.1.1. Conceptual design

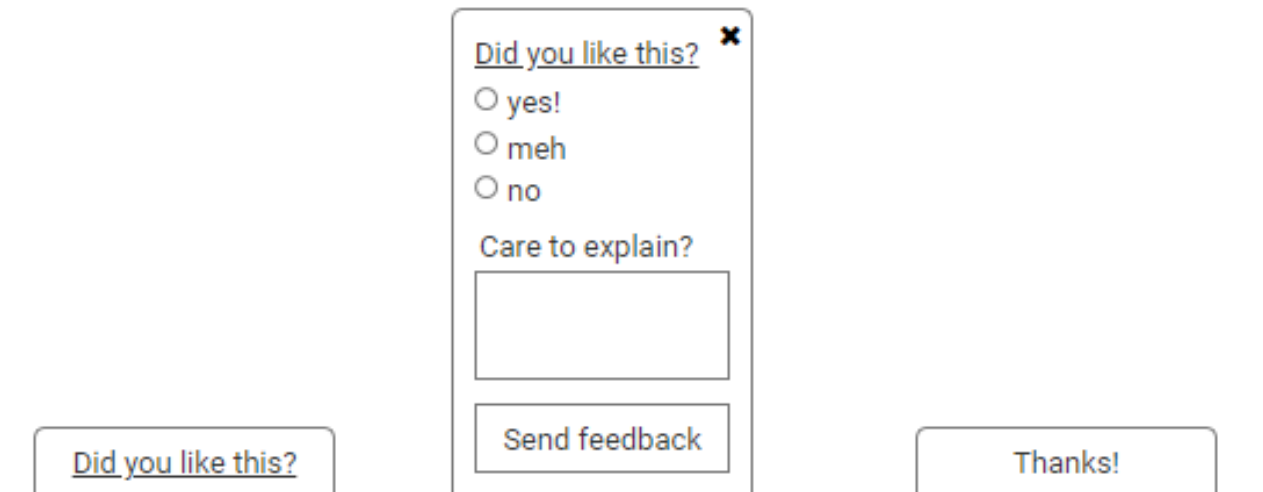


Figure 9. In-page feedback component. Left: default closed state. Middle: open state. Right: final state, after having sent feedback.

The in-page feedback component has three states:

- **Closed state:** an action button, which when selected shows the *open state* of the in-page feedback component.
- **Open state:** a feedback form, containing the following elements:
  - **Feedback options:** a radio button group with three options (positive / meh / negative).
  - **Detailed feedback:** a multi-line text field where users can enter more detailed feedback.

- **Action button:** a button to submit the feedback form. When clicked, the *final state* of the in-page feedback component is shown.
- **Close button:** a button to close the feedback form, possibly without sending feedback. When clicked, the *closed state* of the in-page feedback component is shown.
- **Final state:** a label, thanking the user for their feedback.

#### 2.3.1.2. Behaviour

Only the last selected *feedback option* in any give session is permanently stored in the application, i.e. when a user selects the "positive" feedback option, then changes her mind and selects the "meh" feedback option and even later the "negative" feedback option in the same session, only the "negative" option is recorded.

When the user selects one of the three *feedback options* and then selects the *close button*, a subsequent selection of the action button on the *closed state* will show the *open state* with the previously selected *feedback option* selected.

### 2.3.2. Main menu

The main menu appears once a user is authenticated. It contains navigation items and a way to log out.

#### 2.3.2.1. Conceptual design

The Provocation Engine	Messages	Feedback	<a href="#">Log out</a>
------------------------	----------	----------	-------------------------

Figure 10. Main menu

It's pretty straightforward, yo.

### 2.3.3. Table

A table is used to show tabular data.

#### 2.3.3.1. Conceptual design

<input type="checkbox"/>	Line 1	Line 2
<input type="checkbox"/>	Back to the future	The year is now 2106, but the issue remains. How do you intervene?
<input type="checkbox"/>	Blast from the past	Do it like it's 1921
<input type="checkbox"/>	Brussels, Sahara desert	Brussels is a city in the middle of the Sahara desert
<input type="checkbox"/>	Blindness	A terrible disease has caused everyone to go blind
<input type="checkbox"/>	Elders only	90% of Brussels citizens are over 70 years old.
<input type="checkbox"/>	Neverland	No adults here.
<input type="checkbox"/>	Orwellian dystopia	Commuting is banned and considered illegal.
<input type="checkbox"/>	No electricity	It was never invented.
<input type="checkbox"/>	No private property	No owners. Only users.
<input type="checkbox"/>	Exchange economy	Money was never invented

Figure 11. Table

A table has rows and columns. There may be a header row too.

## 2.3.4. Modal pop-up window

A modal pop-up window appears in a lightbox etc.

### 2.3.4.1. Conceptual design

The screenshot shows a web application titled "The Provocation Engine" with tabs for "Messages" and "Feedback", and a "Log out" link. The "Messages" tab is active, displaying a list of messages in a table. A modal pop-up window titled "New message" is open, allowing the user to add a new message. The modal has "Cancel" and "Save" buttons. The background table contains the following data:

<input type="checkbox"/>	Line 1	Line 2
<input type="checkbox"/>	Back to the future	The year is now 2106, but the issue remains. How do you intervene?
<input type="checkbox"/>	Blast from the past	Do it like it's 1921
<input type="checkbox"/>	Brussels, Sahara desert	Brussels is a city in the middle of the Sahara desert
<input type="checkbox"/>	Blindness	A terrible disease has caused everyone to go blind
<input type="checkbox"/>	Elders only	90% of Brussels citizens are over 70 years old.
<input type="checkbox"/>	Neverland	No adults here.
<input type="checkbox"/>	Orwellian dystopia	Commuting is banned and considered illegal.
<input type="checkbox"/>	No electricity	It was never invented.
<input type="checkbox"/>	No private property	No owners. Only users.
<input type="checkbox"/>	Exchange economy	Money was never invented
<input type="checkbox"/>	Life on the move	No one stands still. Ever. Commuting is living

Figure 12. Modal pop-up window

It's modal.

# Chapter 3. Requirements

## 3.1. Database

- **tblMessages**: holds the messages
  - **id**: unique ID
  - **line1**: max. 40 alphanumeric characters
  - **line2**: max. 100 alphanumeric characters
- **tblFeedback**: holds feedback from visitor
  - **id**: unique ID
  - **timestamp**: date/time
  - **feedback**: one character, indicating positive, meh or negative feedback
  - **extended**: alphanumeric text

## 3.2. API

### 3.2.1. GetMessages

A GET request to /api/messages returns a list of all messages@.

Request:

```
GET /api/messages
```

Response:

```
Status: 200 OK
Content-Type: application/json
Body:
[
  {
    "message-id": "15",
    "line1": "Back to the future",
    "line1": "The year is now 2106, but the issue remains. How do you intervene?"
  },
  {
    "message-id": "16",
    "line1": "last from the past",
    "line1": "Do it like it's 1921"
  },
  ...
]
```

### 3.2.2. GetMessage

A GET request to `/api/messages/{message-id}` returns one message.

Request:

```
GET /api/messages/15
```

Response when succesful:

```
Status: 200 OK
Content-Type: application/json
Body:
{
  "message-id": "16",
  "line1": "Blast from the past",
  "line1": "Do it like it's 1921"
}
```

Response when the message doesn't exist:

```
Status: 404 Not Found
Content-Type: text
Body:
Unknown message
```

### 3.2.3. CreateMessage

A message can be created by POSTing json to `/api/messages`.

Required fields:

- line1
- line2

Request:

```
POST /api/messages
Content-type: application/json
Body:
{
  "line1": "Blast from the past",
  "line2": "Do it like it's 1921"
}
```

Respons bij succes:

```
Status: 201 Created
Content-Type: application/json
Body:
{
  "id": 15,
  "line1": "Blast from the past",
  "line2": "Do it like it's 1921"
}
```

### 3.2.4. UpdateMessage

A PUT request to `/api/messages/{message-id}` with json in the body of the request updates a message.

The message-id in the request must be the same as the message-id in the body.

Request:

```
PUT /api/messages/de:ca:fb:ad:27:08
Content-Type: application/json
Body:
{
  "message-id": "15",
  "line1": "Blast from the past",
  "line2": "Do it like it's 1921"
}
```

Response when succesful:

```
Status: 201 Created
Content-Type: application/json
Body:
{
  "message-id": "15",
  "line1": "Blast from the past",
  "line2": "Do it like it's 1921"
}
```

Response if the message does not exist:

```
Status: 404 Not Found
Content-Type: text
Body:
Unknown message
```