# Messages – SPA Application – JS Apps Exam

You are assigned to implement a **Messaging Web front-end application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Kinvey). The app keeps **users** and **messages**. Users can **register**, **login**, view their **received messages** and **sent messages**, **delete** message (from their sent archive), **send a message** and **logout**.

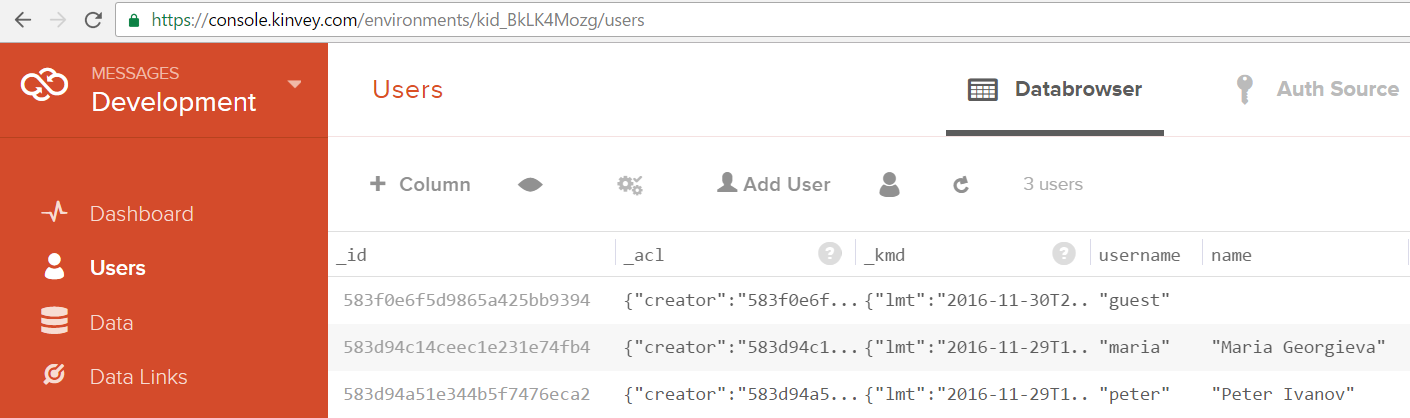
Using libraries like jQuery and React is allowed but is not obligatory.

## Create a Kinvey REST Service to Hold Users and Messages

Register at **Kinvey.com** and create an application to keep your data in the cloud.

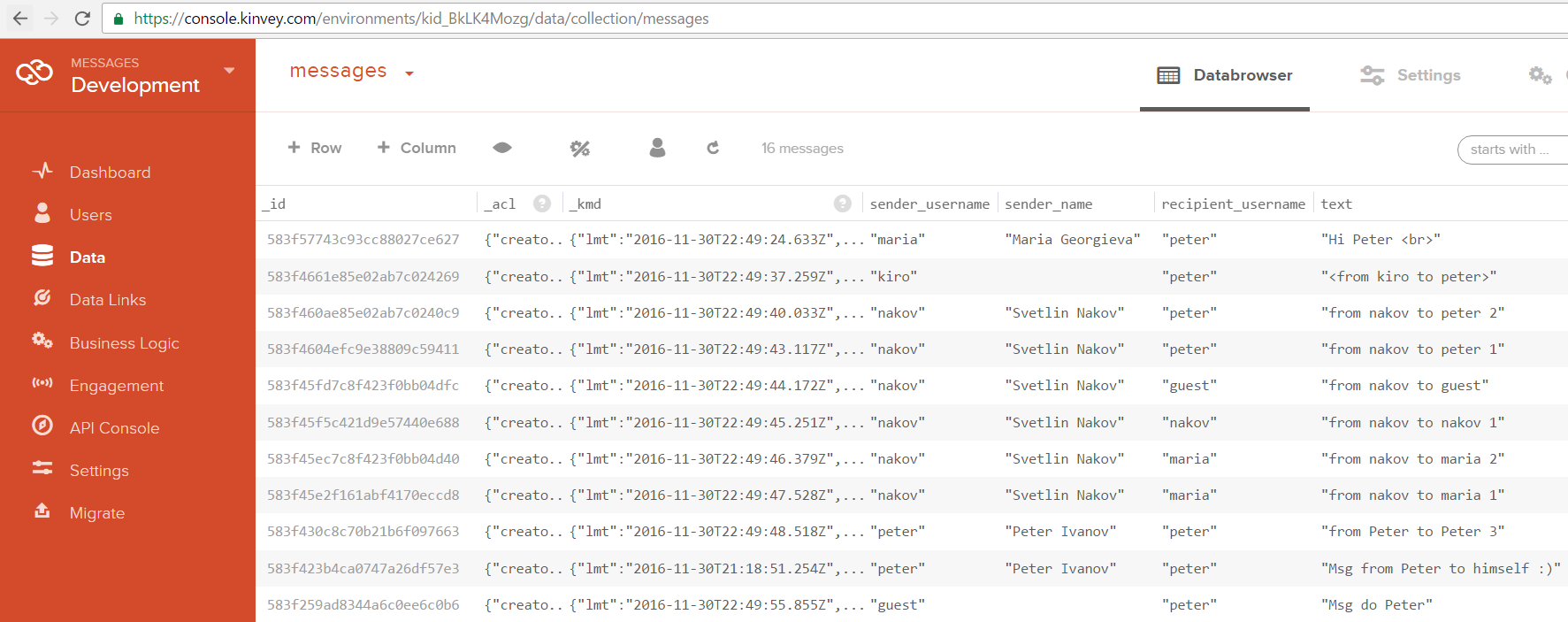
In the **Users** collection add a column “**name**”. Users will hold username, password and name. Register a few users, like shown below:

* User “guest”, password “guest”
* User “maria”, password “maria123”, name “Maria Georgieva”
* User “peter”, password “peter123”, name “Peter Ivanov”



Create a collection **messages(sender\_username, sender\_name, recipient\_username, text)** to hold the messages. All the fields will hold text values. The “**\_kmd.lmt**” field is automatically created by Kinvey and will hold a date and time in the traditional for JavaScript dates ISO8601 format returned by [Date.toJSON()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/toJSON).

Put some sample data in the collection like shown below:



Kinvey will automatically create **REST services** to access your data.

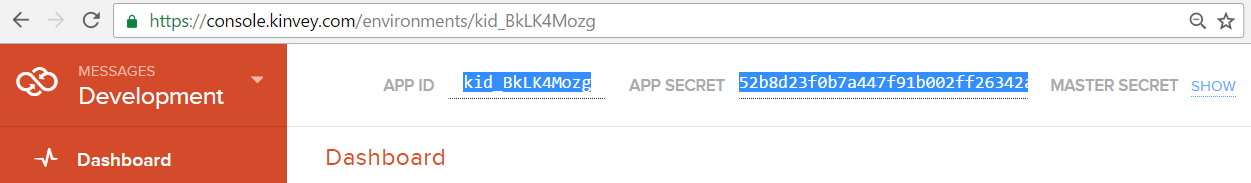
## Test the Kinvey REST Services

Using **Postman** or other HTTP client tool, test the REST service endpoints:

### User Registration (Sign Up)

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_id***/](https://baas.kinvey.com/user/app_id/) | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "new\_user",  "password": "pass123",  "name": "New User"  } |
| Response  201 Created | {  "\_id": "583f53bde004a9a90983f1b7",  "username": "new\_user",  "password": "pass123",  "name": "New User",  …  } |
| Error response  409 Conflict | { "error": "UserAlreadyExists", "description": "This username is already taken. Please retry your request with a different username", "debug": "" } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

The request needs “**Basic**” authentication. Use the Kinvey **app\_id** and Kinvey **app\_secret** as credentials.



### User Login

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_id***/login](https://baas.kinvey.com/user/app_id/login) | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "new\_user",  "password": "pass123"  } |
| Response  200 OK | {  "\_id": "583f53bde004a9a90983f1b7",  "username": "new\_user",  "name": "New User",  "\_kmd": {  "**authtoken**": "**8e6471bc-3712-4cfb-b92e-50e62a0c80….Duj5fHdM /7XHIe6KdY=**"  …  },  …  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

Successful login returns an “**authtoken**” which is later used to authenticate the CRUD operations.

### User Logout

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_id***/\_logout](https://baas.kinvey.com/user/app_id/_logout) | |
| Request headers | Authorization: Kinvey **authtoken** |
| Response  204 No Content |  |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

To logout, you need to provide the “**authtoken**” given by login / register as “**Kinvey**” authorization header.

### List Messages by Recipient (My Messages)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/***app\_id***/messages?query={"recipient\_username":"***username***"}](https://baas.kinvey.com/appdata/%7bapp_id%7d/messages?query=%7b%22recipient_username%22:%22%7busername%7d%22%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  { "\_id": "583eb40a8c70b21b6ffdb1d7",  "sender\_username": "nakov",  "sender\_name": "Svetlin Nakov",  "recipient\_username": "maria",  "text": "Hi, Maria", ...  "\_kmd": { "lmt": "2016-11-30T11:12:10.603Z", ... }  },  { "\_id": "583eb430e004a9a909778b27",  "sender\_username": "peter",  "sender\_name": "Peter Ivanov",  "recipient\_username": "maria",  "text": "Hi, Maria, I am Peter", ...  "\_kmd": { "lmt": "2016-11-30T11:13:29.506Z", ... }  }, ...  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### List Messages by Sender (Sent Archive)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/***app\_id***/messages?query={"sender\_username":"***username***"}](https://baas.kinvey.com/appdata/app_id/messages?query=%7b%22sender_username%22:%22username%22%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  { "\_id": "583eb40a8c70b21b6ffdb1d7",  "sender\_username": "nakov",  "sender\_name": "Svetlin Nakov",  "recipient\_username": "maria",  "text": "Hi, Maria",  "\_kmd": { "lmt": "2016-11-30T22:49:49.889Z", ... }  },  { "\_id": "583f45e2f161abf4170eccd8",  "sender\_username": "nakov",  "sender\_name": "Svetlin Nakov",  "recipient\_username": "peter",  "text": "Msg from Nakov to Peter",  "\_kmd": { "lmt": "2016-11-30T22:49:47.528Z", ... }  }, ...  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### List All Users

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/user/***app\_id***/](https://baas.kinvey.com/user/app_id/) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  { "\_id": "583d94a51e34b5f747…", "username": "peter", "name": "Peter Ivanov", … },  { "\_id": "583d94c14ceec3…", "username": "maria", "name": "Maria Georgieva", … },  { "\_id": "583f0e6f5d9865a425bb9394", "username": "guest", ... }, ...  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

### Send Message

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/appdata/***app\_id***/messages](https://baas.kinvey.com/appdata/%7bapp_id%7d/messages) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "sender\_username":"peter",  "sender\_name":"Peter Ivanov",  "recipient\_username":"maria",  "text":"Hi, Maria. I am Peter"  } |
| Response  201 Created | {  "sender\_username": "peter",  "sender\_name": "Peter Ivanov",  "recipient\_username": "maria",  "text": "Hi, Maria. I am Peter",  "\_kmd": { "lmt": "2016-11-30T23:06:11.405Z", … },  "\_id": "583f5b635d9865a425be9549"  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

Note: we know that this service endpoint allows sending **fake messages**. Client apps can send a message on behalf of someone else or on behalf of non-existing user. Recipient also might be non-existing. Don’t care about this: build your app to send messages using the currently logged-in sender and existing recipients.

All messages created by the above service get automatically assigned by Kinvey **date** (in ISO8601 format) in the field “**\_kmd.lmt**” (last modify date).

### Delete Message

|  |  |
| --- | --- |
| **DELETE** [https://baas.kinvey.com/appdata/***app\_id***/messages/***msg\_id***](https://baas.kinvey.com/appdata/app_id/messages/msg_id) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | {  "count": 1  } |
| Error response  404 Not Found | { "error": "EntityNotFound", "description": "This entity not found in the collection", "debug": "" } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

Note that the currently logged-in user is allowed to **delete only messages** **created by himself**.

## Messages – HTML and CSS

You аre given the Web design of the Messages application as **HTML** + **CSS** files.

* Initially all views and forms are shown by the HTML. Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.
* You may render the views / forms / components with React, jQuery, Mustache or another UI library.

**Important**: don’t change the elements’ **class name** and **id**. Don’t rename form fields / link names / ids.

## Messages Client-Side Web Application

Design and implement a client-side front-end app (SPA) for managing the messages. Implement the functionality described below.

### Navigation System

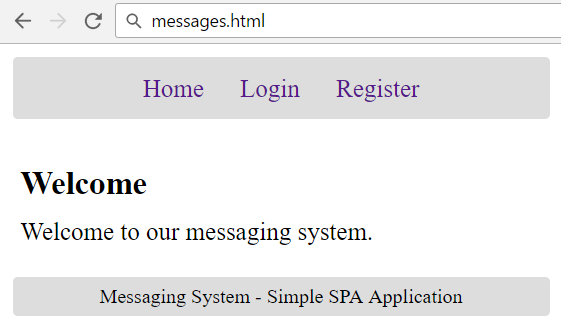
Implement a **navigation system** for the app: navigation links should correctly change the current screen (view).

* Clicking on the links in the **top navigation bar** should display the view behind the link (views are sections in the HTML code).
* Your application may **hide** by CSS (display: none) or **delete** from the DOM all unneeded elements or just display the views it needs to display.

5 score

### Home Screen

When no user is logged in, the app should display the "Home" screen holding a welcome message + three links: **[Home]**, **[Login]** and **[Register]**.

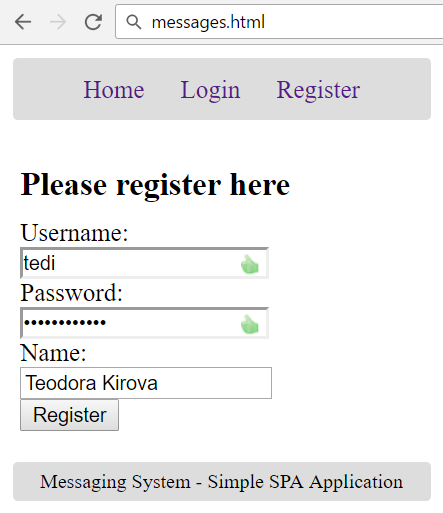


5 score

### Register User Screen

By given **username** + **password** + **name** the app should register a new user in the system.

* After a **successful registration**, a notification message “User registration successful.” should be displayed and the user home screen should be displayed.
* In case of **error**, an appropriate error message should be displayed and the user should be able to try to register again.
* **Form validation** is already implemented in the HTML, so you don’t need to add it.
* Keep the user session data in the browser’s **session storage**.

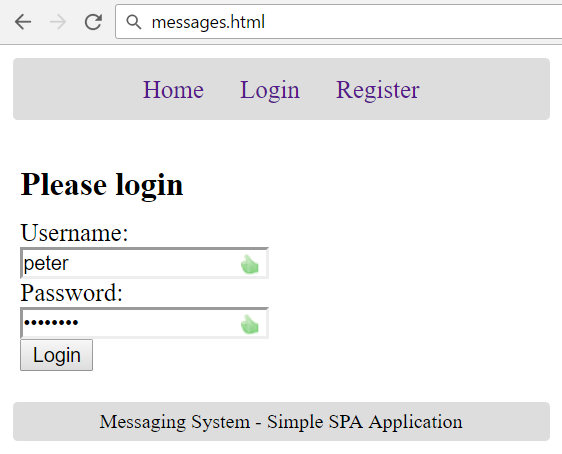


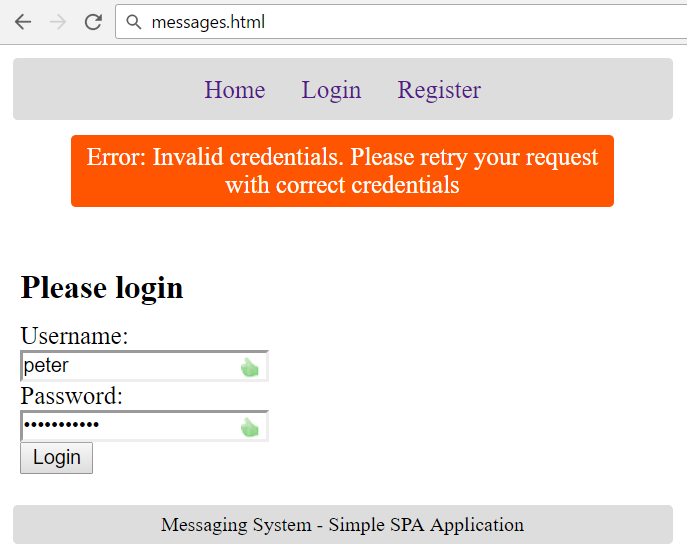
10 score

### Login User Screen

By given **username** and **password** the app should be able to login an existing user.

* After a **successful login**, a notification message “Login successful.” should be displayed and the user home screen should be displayed.
* In case of **error**, an appropriate error message should be displayed and the user should be able to fill the login form again.
* **Form validation** is already implemented in the HTML, so you don’t need to add it.
* Keep the user session data in the browser’s **session storage**.





5 score

### Logout

Successfully logged in user should be able to **logout** from the app.

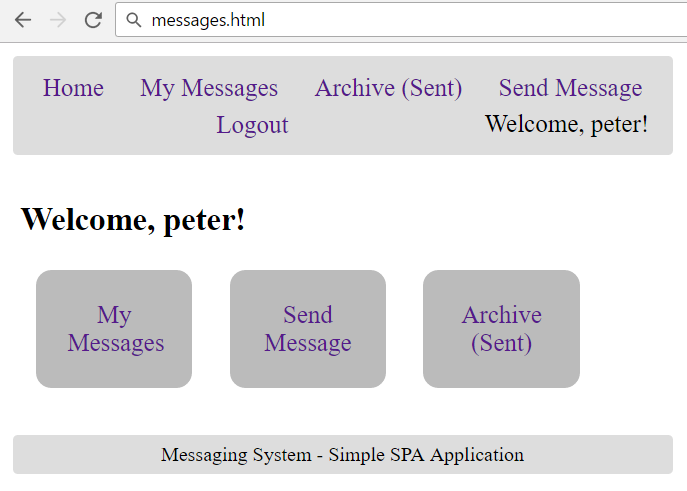
* After a **successful** logout, a **notification** message “Logout successful” should be displayed and the login screen should be shown.
* After successful logout, the **Home screen** should be shown.
* The **“logout” REST service** at the back-end should be obligatory called at logout.
* All local information in the browser (**user session data**) about the current user should be deleted.

5 score

### User Home Screen

After successful login, the app should display the **user's home screen**.

* It should hold a message **“Welcome, ” + the username** of the current user.
* At the top navigation bar display the navigation links **[Home]**, **[My Messages]**, **[Archive (Sent)]**, **[Send Message]** and **[Logout]** + “**Welcome, {username}**”.
* At the main view area display navigation boxes **[My Messages]**, **[Archive (Sent)]** and **[Send Message]**.



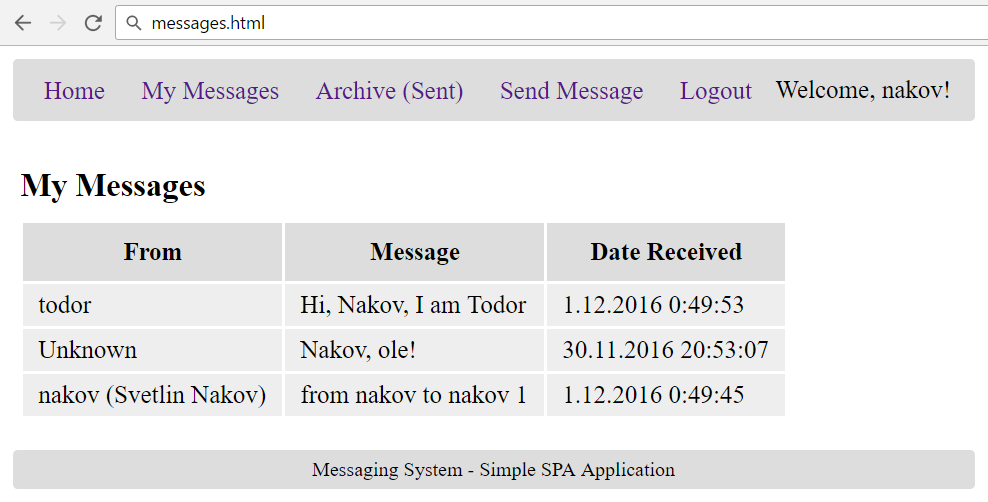
* Ensure you handle properly all HTML **special characters**, e.g. the username could be "*<peter><br>*".

5 score

### Received Messages Screen

Successfully logged users after clicking the **[My Messages]** link should be able to view all messages, sent to current user, i.e. **recipient\_username** == ***{current\_username}***.

* The messages should be listed in the **format** as shown in the Web design (see the screenshot below).
* In case of **error** (e.g. Internet connection lost), an error message should be displayed.
* In case of **no messages**, display an empty table (header row only).



* Format the message **sender** and **date** as shown in the screenshot above. You may use the following helper functions to format dates and senders:

|  |
| --- |
| **function** *formatDate*(dateISO8601) {  **let** date = **new** Date(dateISO8601);  **if** (Number.isNaN(date.getDate()))  **return ''**;  **return** date.getDate() + **'.'** + *padZeros*(date.getMonth() + 1) +  **"."** + date.getFullYear() + **' '** + date.getHours() + **':'** +  *padZeros*(date.getMinutes()) + **':'** + *padZeros*(date.getSeconds());   **function** *padZeros*(num) {  **return** (**'0'** + num).slice(-2);  } }  **function** *formatSender*(name, username) {  **if** (!name)  **return** username;  **else if** (username)  **return** name + **' ('** + username + **')'**;  **else**  **return** name; } |

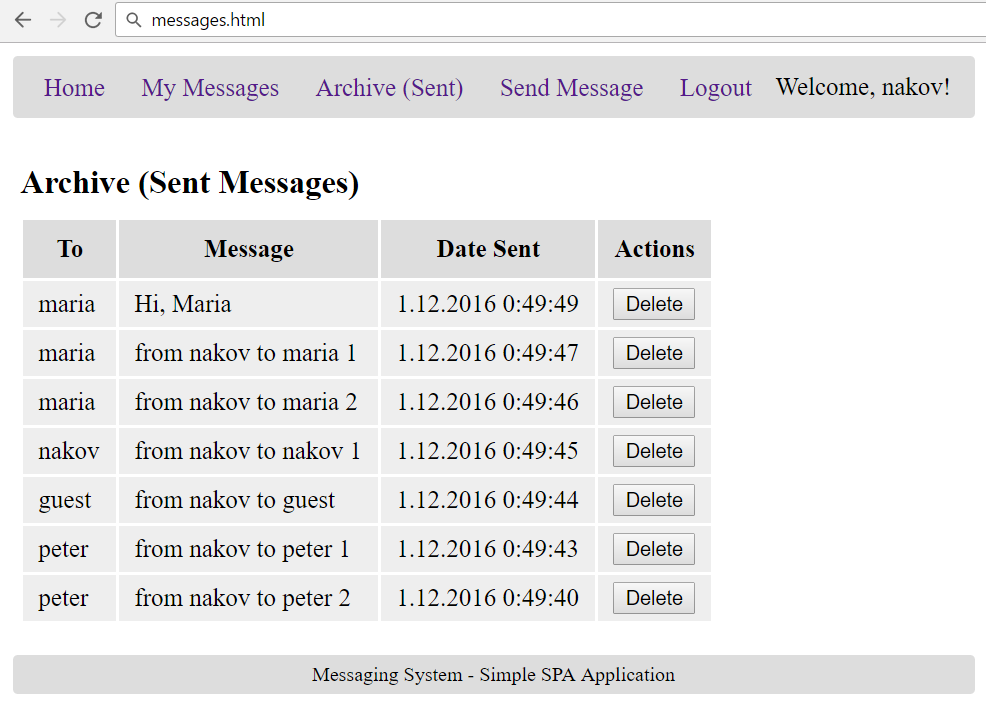
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".

20 score

### Sent Messages Screen

Successfully logged users after clicking the **[Archive (Sent)]** link should be able to view all messages, sent by the current user, i.e. **sender\_username** == ***{current\_username}***.

* The messages should be listed **as shown in the Web design** (see the screenshot below).
* In case of **error** (e.g. Internet connection lost), an error message should be displayed.
* Display **[Delete]** button for each sent message. The button will delete the message.
* In case of **no messages**, display an empty table (header row only).
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".

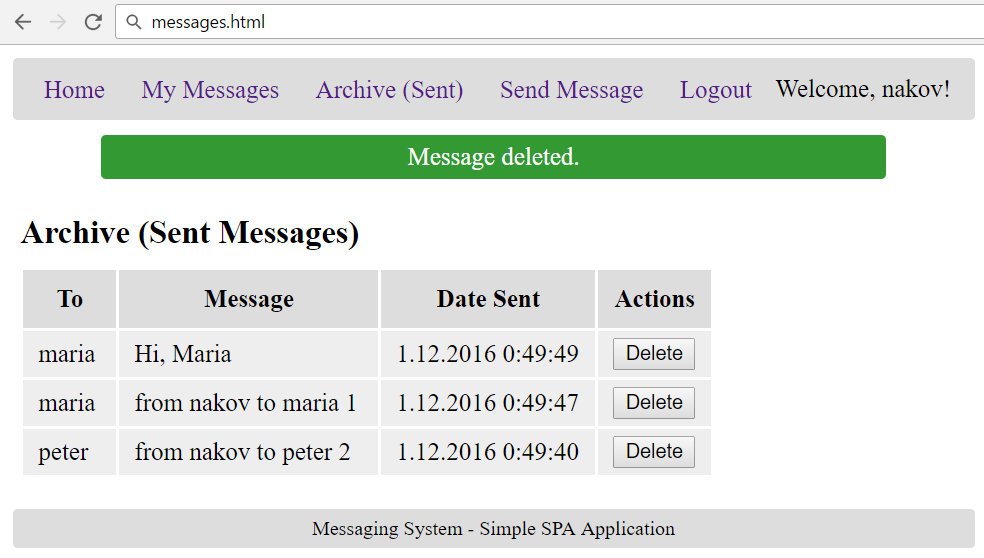


15 score

### Delete Message Screen

Successfully logged in users should be able to **delete their sent (archive) messages** by clicking on the **[Delete]** button in the table above.

* After **successful** message delete a notification message “Message deleted.” should be shown.
* In case of **error** (e.g. Internet connection lost / unauthorized request / missing message), an error message should be displayed.
* Deleting works immediately, with **no confirmation**.

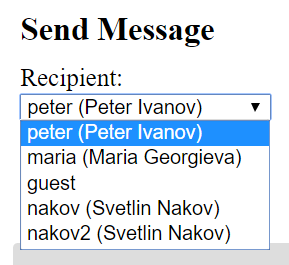
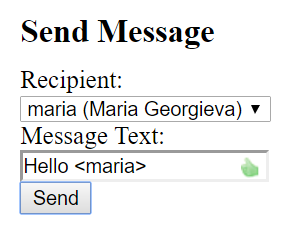


10 score

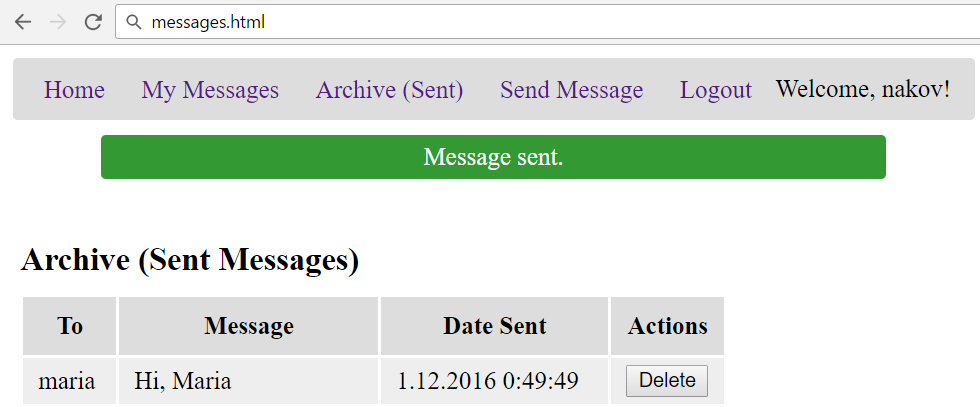
### Send Message Screen

Successfully logged in users should be able to **send new messages** by choosing an **existing recipient**, entering a **text** and clicking the **[Send]** button.

* Before the “send message” form is shown, it should first **load all users** from the back-end and display them into the drop-down list of recipients.

* After a **successful** message sending, a notification message “Message sent.” should be displayed and all sent messages (**archive**) should be shown.
* In case of **error**, an appropriate error message should be displayed and the user should be able to fill the “send messages” form and try again.
* **Form validation** is already implemented in the HTML, so you don’t need to add it.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".



20 score

### Notifications

The application should notify the users about the result of their actions.

* In case of successful action an **informational (green) notification message** should be shown, which disappears automatically after 3 seconds or manually when the user clicks it.



* In case of **error**, an **error notification message** (red) should be shown which disappears on user click.



* During the AJAX calls a **loading notification message (blue)** should be shown. It should disappear automatically as soon as the AJAX call is completed.



Good luck!