# JS Apps Exam – Chirper Single Page Application

You are assigned to implement a **Web application** (SPA) using HTML5, JavaScript, AJAX, REST and JSON with cloud-based backend (Kinvey). The app keeps **users and chirps (tweets)**. Users can **register**, **login,** **logout**, view **feed** page (all chirps from followers sorted by time in descending), **create** a chirp, **delete** a chirp, view **own** chirps, **discover** new people, get **all chirps** by specific user (sorted by time in descending), **follow** or **unfollow** a user.

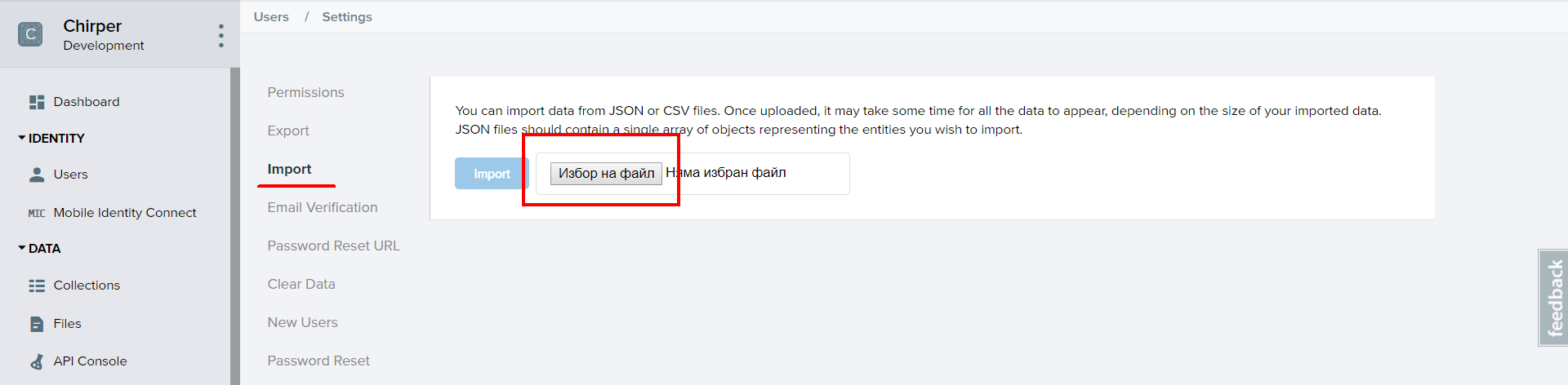
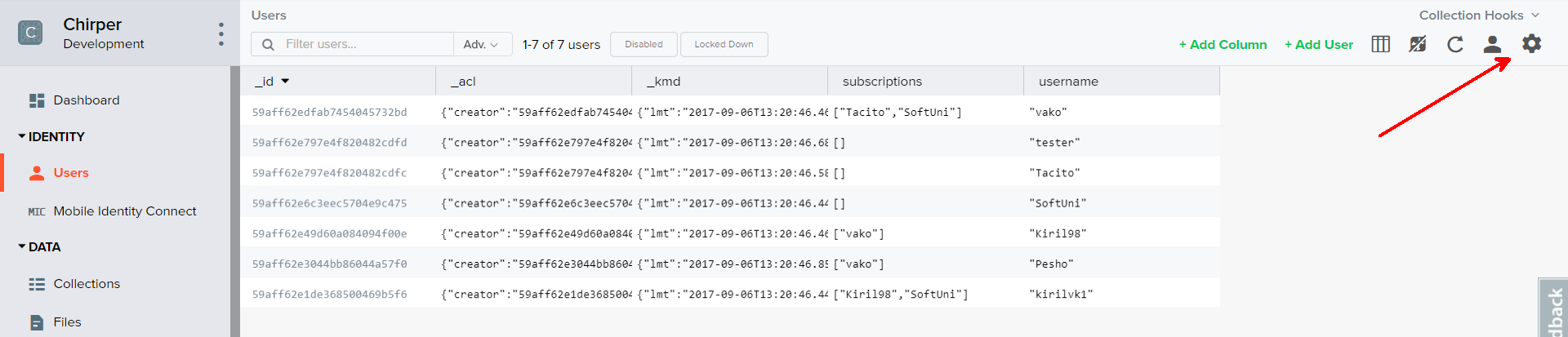
Using libraries like **jQuery**, **Handlebars** and **Sammy** is allowed but is not obligatory.

## Create a Kinvey REST Service

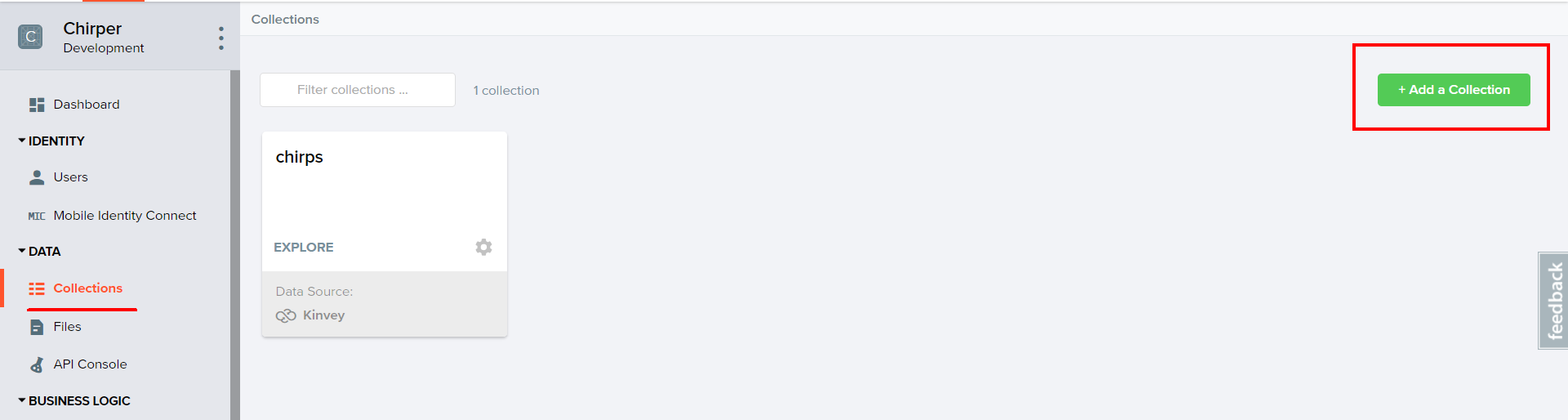
Register at **Kinvey.com** and create an application to keep your data in the cloud.

Each user has a **username**, **password** and **subscriptions (array of usernames)**.

In the **Users** collection, import the provided JSON file with sample users to get started with template data. In the **Kinvey Console**, select **Users** from the navigation of the left, click **Settings** in the upper right then scroll down to the **Import** section:



Create a collection **chirps**. Each chirp has a **text** (string with max length 150 and cannot be empty) and an **author** (username of the creator).



You are also given **one** more json file to **import** your **chirps**.

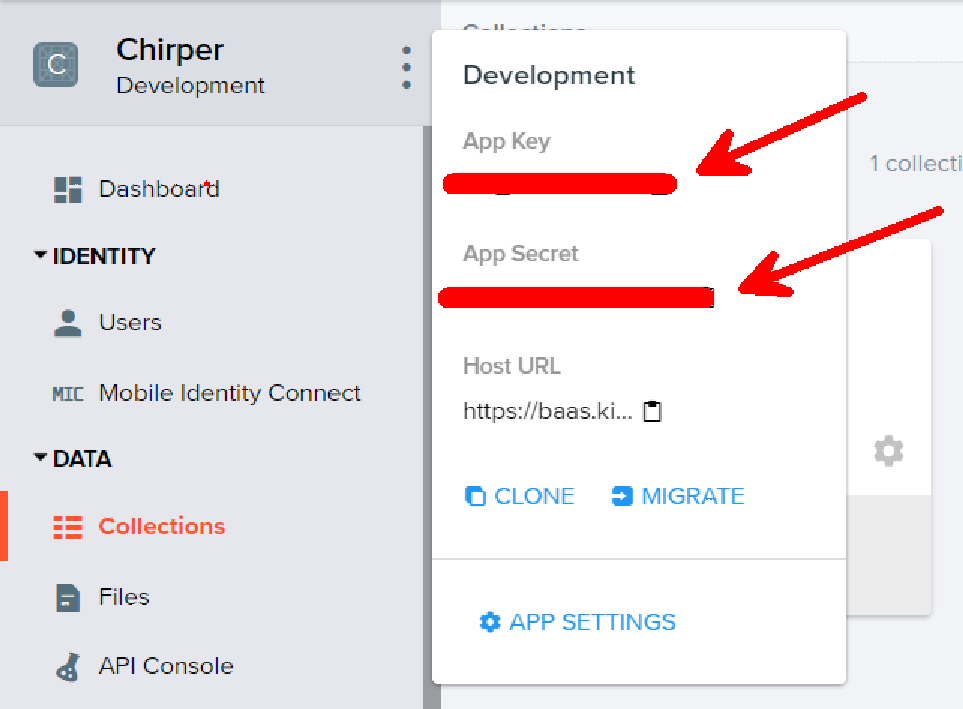
## Test the Kinvey REST Services

Using **Postman** or other HTTP client tool (you can use Kinvey’s built-in **API Console**), test the REST service endpoints:

### User Registration (Sign Up)

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_key***/](https://baas.kinvey.com/user/app_key/) | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "testuser",  "password": "testuserpass890",  "subscriptions": []  } |
| Response  201 Created | {  "\_id": "**59affdae3044bb86044a79bd**",  "username": "**testuser**",  "password": "**testuserpass890**",  "subscriptions": **[]**  } |
| 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 Key** and Kinvey **App Secret** as credentials.



### User Login

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_key***/login](https://baas.kinvey.com/user/app_key/login) | |
| Request headers | Authorization: Basic base64(app\_id:app\_secret)  Content-Type: application/json |
| Request body | {  "username": "testuser",  "password": "testuserpass890"  } |
| Response  200 OK | {  "\_id": "59930c78a743e20c7d3fca77",  "username": "testuser",  "subscriptions": []  "\_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\_key***/\_logout](https://baas.kinvey.com/user/app_key/_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 all Chirps by followers (Feed – sorted by post time, descending)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/***app\_key***/chirps?query={"author":{"$in": **[${subs}]**}}&sort={"\_kmd.ect": 1}](https://baas.kinvey.com/appdata/app_key/chirps?query=%7b%22author%22:%7b%22$in%22:%20%5b$%7bsubs%7d%5d%7d%7d&sort=%7b%22_kmd.ect%22:%201%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "59ae7752e884417c04e803a4",  "text": "yohooo",  "\_acl": {  "creator": "59ae425c14ee505004862409"  },  "author": "vako",  "\_kmd": {  "lmt": "2017-09-05T11:38:02.944Z",  "ect": "2017-09-05T10:07:14.314Z"  }  },… |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

Where subs is an array of subscriptions (by username).

### Create Chirp

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/appdata/***app\_key***/chirps](https://baas.kinvey.com/appdata/app_key/chirps) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "text": "**MyTestChirp**",  "author": "**testuser**"  } |
| Response  201 Created | {  "\_id": "59931398996ab5127d2a84d1",  "author": "Kiril",  "title": "Test Post",  "description": "Test Post Description",  "url": "https://en.wikipedia.org/wiki/Santorini",…  } |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

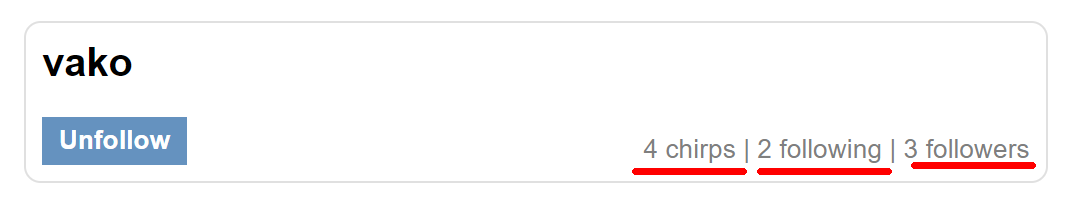
### Delete Chirp

|  |  |
| --- | --- |
| **DELETE** [https://baas.kinvey.com/appdata/***app\_key***/chirps/***chirp\_id***](https://baas.kinvey.com/appdata/app_key/chirps/chirp_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": "" } |

### User Chirps (View all chirps by user, sorted by post time, descending)

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/**app\_key**/chirps?query={"author":"**username**"}](https://baas.kinvey.com/appdata/app_key/chirps?query=%7b%22author%22:%22username%22%7d)&sort={"\_kmd.ect": 1} | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "59b004cfdfab74540457702a",  "text": "#testuserchirp",  "author": "testuser",  "\_acl": {  "creator": "59affdae3044bb86044a79bd"  },  "\_kmd": {  "lmt": "2017-09-06T14:23:11.069Z",  "ect": "2017-09-06T14:23:11.069Z"  }  },.. |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

The **next** 3 queries are needed to **display** profile **stats.**



### Count Chirps

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/appdata/***app\_key***/chirps?query={"author":"**username**"}](https://baas.kinvey.com/appdata/app_key/chirps?query=%7b%22author%22:%7b%22$in%22:%20%5b$%7bsubs%7d%5d%7d%7d&sort=%7b%22_kmd.ect%22:%20-1%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "59b004cfdfab74540457702a",  "text": "#testuserchirp",  "author": "testuser",  "\_acl": {  "creator": "59affdae3044bb86044a79bd"  },  "\_kmd": {  "lmt": "2017-09-06T14:23:11.069Z",  "ect": "2017-09-06T14:23:11.069Z"  }  },.. |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

This query will return **all** of the chirps by **given** user.

### Following

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/user/***app\_key***/?query={"username":"**username**"}](https://baas.kinvey.com/user/app_key/?query=%7b%22username%22:%22username%22%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "59afe6ad68c53f4a045534df",  "subscriptions": [  "vako"  ],  "username": "Kiril98",  "\_acl": {  "creator": "59afe6ad68c53f4a045534df"  },  "\_kmd": {  "lmt": "2017-09-06T12:37:47.242Z",  "ect": "2017-09-06T12:14:37.861Z"  }  }  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

This query will return the **needed** user. The **count** of the people that the user is **following** is the **length** of the subscriptions array.

### Followers

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/user/***app\_key***/?query={"subscriptions":"**username**"}](https://baas.kinvey.com/user/app_key/?query=%7b%22subscriptions%22:%22username%22%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "59afe70614ee5050048cecfb",  "subscriptions": [  "Kiril98",  "SoftUni"  ],  "username": "kirilvk1",  "\_acl": {  "creator": "59afe70614ee5050048cecfb"  },  "\_kmd": {  "lmt": "2017-09-06T12:16:19.636Z",  "ect": "2017-09-06T12:16:06.222Z"  }  },  {  "\_id": "59affdae3044bb86044a79bd",  "subscriptions": [  "Kiril98",  "Tacito"  ],…  ] |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

The following query will return all **users** that are **subscribed** to given username.

### Discover Page

|  |  |
| --- | --- |
| **GET** [https://baas.kinvey.com/user/***app\_key***/](https://baas.kinvey.com/user/app_key/?query=%7b%22subscriptions%22:%22username%22%7d) | |
| Request headers | Authorization: Kinvey authtoken |
| Response  200 OK | [  {  "\_id": "59ae425c14ee505004862409",  "subscriptions": [  "Tacito",  "SoftUni"  ],  "username": "vako",  "\_acl": {  "creator": "59ae425c14ee505004862409"  },  "\_kmd": {  "lmt": "2017-09-05T17:14:56.960Z",  "ect": "2017-09-05T06:21:16.793Z"  }  },  {  "\_id": "59ae864f603d08cc6fedf3ab",  "username": "Tacito",  "\_acl": {  "creator": "59ae864f603d08cc6fedf3ab"  },  "subscriptions": [],  "\_kmd": {  "lmt": "2017-09-05T11:42:53.473Z",  "ect": "2017-09-05T11:11:11.629Z"  }  },.. |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

In the discover page you should **display** all users **except** the currently logged.

### Follow

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_key***/***user\_id***](https://baas.kinvey.com/user/app_key/user_id) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "subscriptions": [  "testuser",  "vako",  "**Tacito**"  ]  } |
| Response  200 OK | {  "subscriptions": [  "testuser",  "vako",  "**Tacito**"  ],  "\_id": "59afe6ad68c53f4a045534df",  "username": "Kiril98",… |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

The PUT query does **not** need to include the **whole** user object. Modify **only** the subscriptions array.

### Unfollow

|  |  |
| --- | --- |
| **POST** [https://baas.kinvey.com/user/***app\_key***/***user\_id***](https://baas.kinvey.com/user/app_key/user_id) | |
| Request headers | Authorization: Kinvey authtoken  Content-Type: application/json |
| Request body | {  "subscriptions": [  "testuser",  "vako"  ]  } |
| Response  200 OK | {  "subscriptions": [  "testuser",  "vako"  ],  "\_id": "59afe6ad68c53f4a045534df",  "username": "Kiril98",… |
| Error response  401 Unauthorized | { "error": "InvalidCredentials", "description": "Invalid credentials. Please retry your request with correct credentials", "debug": "" } |

The **same** query as “**Follow**”. Only this time we **remove** the wanted username from the subscriptions array.

## Chirper–HTML and CSS

You аre given the Web design of the **Chirper** 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 **jQuery** or **Mustache** **Handlebars**.

**Important**: don’t change the elements’ **class name** and **id**. Don’t rename form fields / link names / ids. You are **allowed** to add **data attributes** to any elements. You may modify **href attributes** of links and add **action/method attributes** to forms, to allow the use of a routing library.

## Chirper Client-Side Web Application

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

### 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.



### Navigation System (5 pts)

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

* Clicking on the links in the **navbar** or the links on each **individual** post 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.
* The given **navbar** should be visible **only** for logged in users. Anonymous users can **only** view the **sign in/register** section.



### Register User Screen (10 pts)

By given **username**, **password and repeat password** 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 **feed page** with all **chirps by followers** should be shown.
* You **need** to validate the **input**. A username **should** be a string with at **least** 5 characters **long**. Passwords **input** fields shouldn’t be **empty**. Both passwords **should** match.
* In case of **error** (eg. invalid username/password), an appropriate error **message** should be displayed and the user should be able to **try** to register again.
* Keep the user session data in the browser’s **session storage**.
* That **includes** the subscriptions array. Since local/session storage **only** keeps **strings** the array must be **stringified** or **kept** as a variable and **updated** on every init.
* Clear **all** input fields after **successful** register.



### Login User Screen (5 pts)

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 and the **feed page** with all **chirps by followers** should be shown.
* 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** should be the **same** as register.
* Keep the user session data in the browser’s **session storage**.
* That **includes** the subscriptions array. Since local/session storage **only** keeps **strings** the array must be **stringified** or **kept** as a variable and **updated** on every init.
* Clear **all** input fields after **successful** login.



### Logout (5 pts)

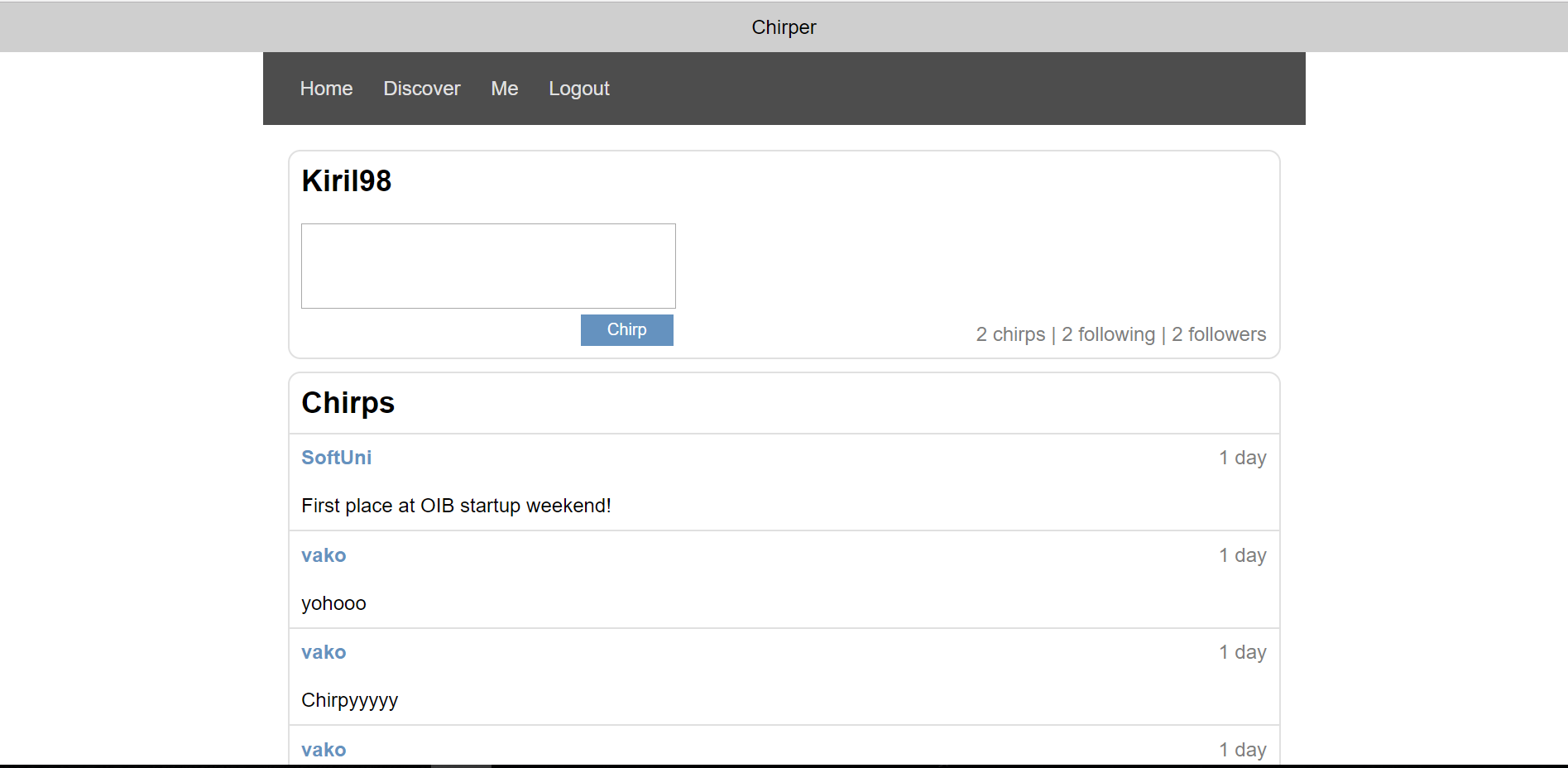
Successfully logged in user should be able to **logout** from the app.

* After a **successful** logout, a **notification** message “Logout successful.” should be displayed.
* After successful logout, the **Sign In 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.

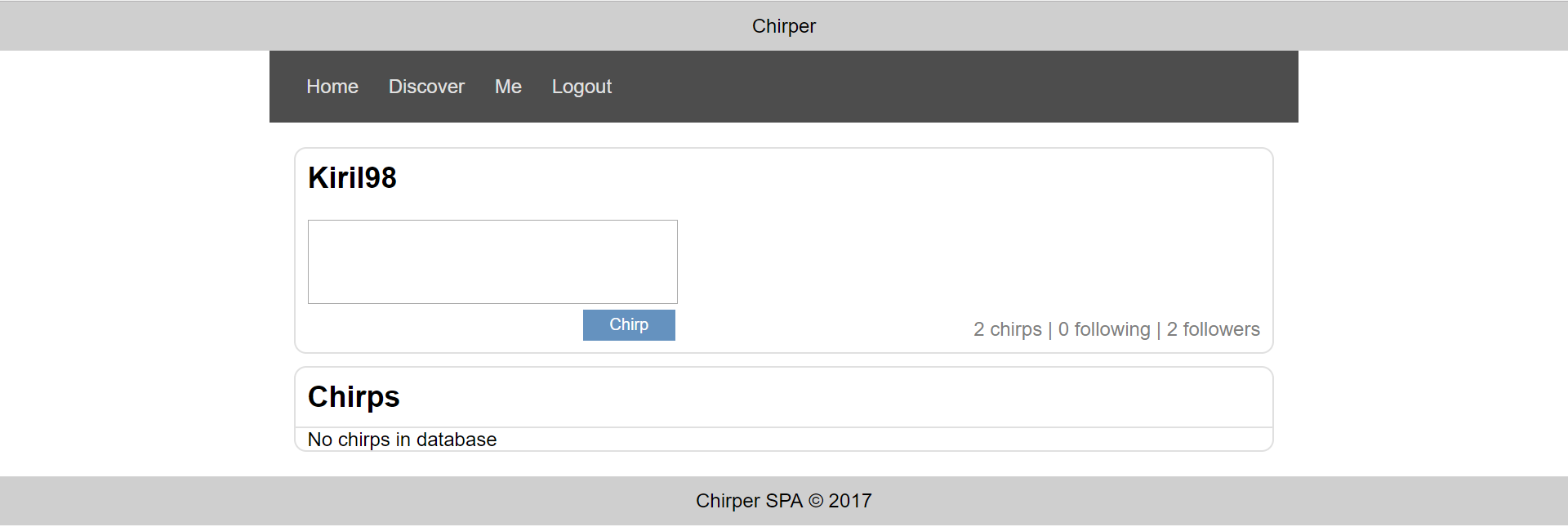
### Feed/Home Screen (20 pts)

Successfully logged users should be welcomed by the **feed** screen which can be accessed **after** clicking the **[Home]** link in the **navbar**. The feed **screen** contains all chirps from **subscriptions or people that the user is following** (sorted by time posted in descending). It also contains the **create** a chirp **form**.

* The **chirps** should be listed in the **format** as shown in the Web design (see the screenshot below).
* Each **chirp** has **text**, **author** and information about **when** it was **created**. For the **time created** information you will use a helper function given **below**. It requires a **parameter** which is the entity creation date of the post (or **\_kmd.ect** property of the post object).
* When the **follower username** link is clicked, your app should **redirect** to the individual feed page of the user.
* The feed screen should also contain **information** about the **count** all of all chirps that the user has **posted**, number of people that the user is **following**, and number of **followers** that the user has.
* In case of **error** (e.g. Internet connection lost), an error message should be displayed.
* In case of **no chirps**, display ‘No chirps in database’.



If the user hasn’t follown anyone display the following:



* Format the entity creation **date** of a **chirp** as shown in the screenshot above. You may **use** the following helper function:

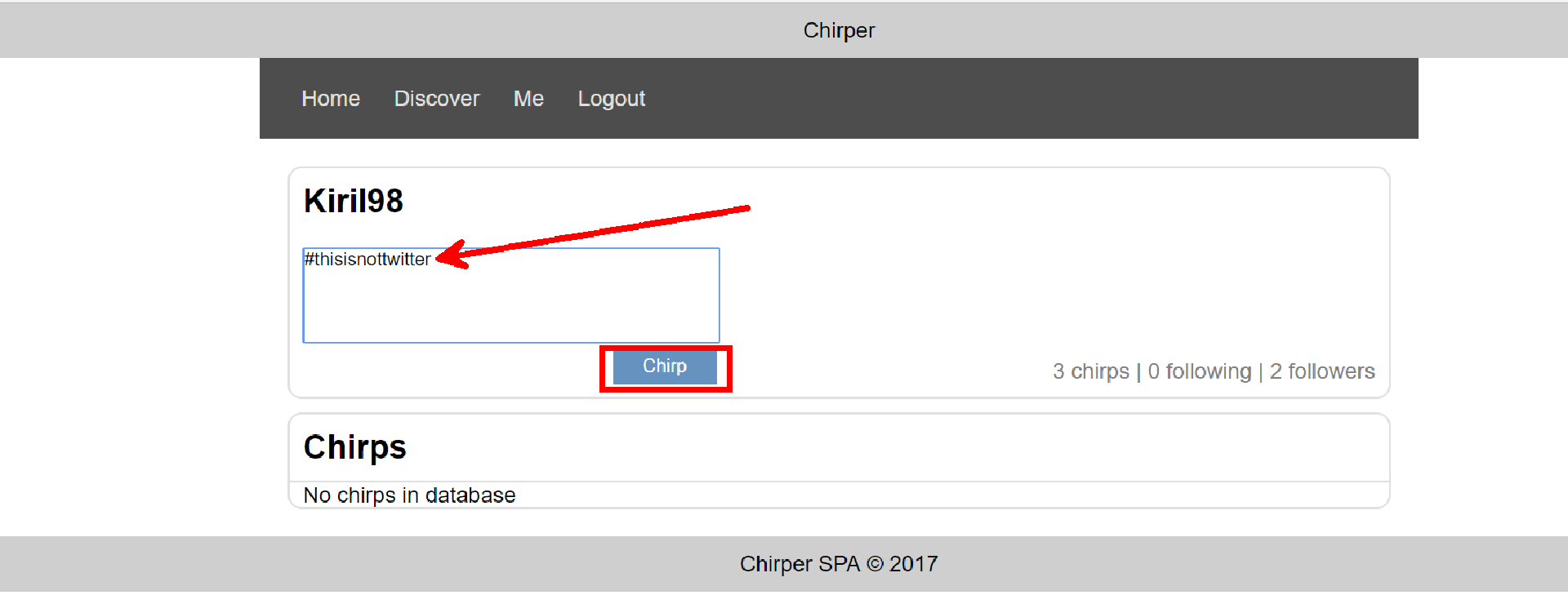
|  |
| --- |
| **function** *calcTime*(dateIsoFormat) {  **let** diff = **new** Date - (**new** Date(dateIsoFormat));  diff = ***Math***.floor(diff / 60000);  **if** (diff < 1) **return 'less than a minute'**;  **if** (diff < 60) **return** diff + **' minute'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 60);  **if** (diff < 24) **return** diff + **' hour'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 24);  **if** (diff < 30) **return** diff + **' day'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 30);  **if** (diff < 12) **return** diff + **' month'** + *pluralize*(diff);  diff = ***Math***.floor(diff / 12);  **return** diff + **' year'** + *pluralize*(diff);  **function** *pluralize*(value) {  **if** (value !== 1) **return 's'**;  **else return ''**;  } } |

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

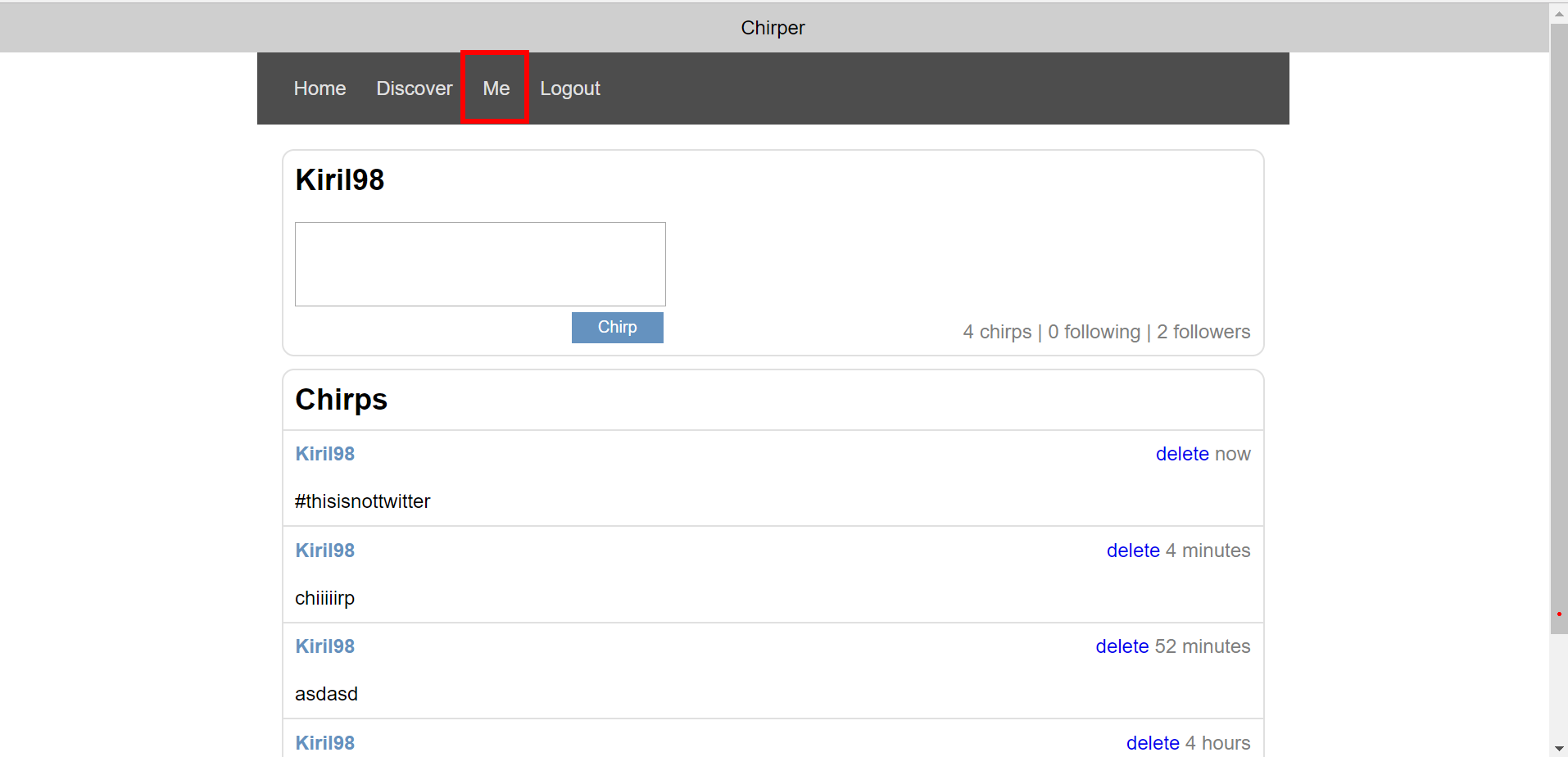
### Post Chirp (10 pts)

Logged in users can **post** chirps. Clicking the **[Chirp]** button should directly **add** a chirp into the **kinvey** database. A chirp has **text** and **author**.

* After a **successful** chirp creation, a notification message “Chirp published.” should be displayed and the **user feed** should be shown (the screen with all chirps posted by the **currently logged** user).
* A chirp text shouldn’t be **empty** and shouldn’t contain more than **150 symbols**.
* Clear the **create** chirp input **field** after **successful** creation.
* Ensure you handle properly all HTML **special characters**, e.g. the message text could hold "*Hi, <peter>*".



After **successful** creation the **user feed** should be displayed. It can also be accessed by clicking the [**Me**] link in the **navbar**:

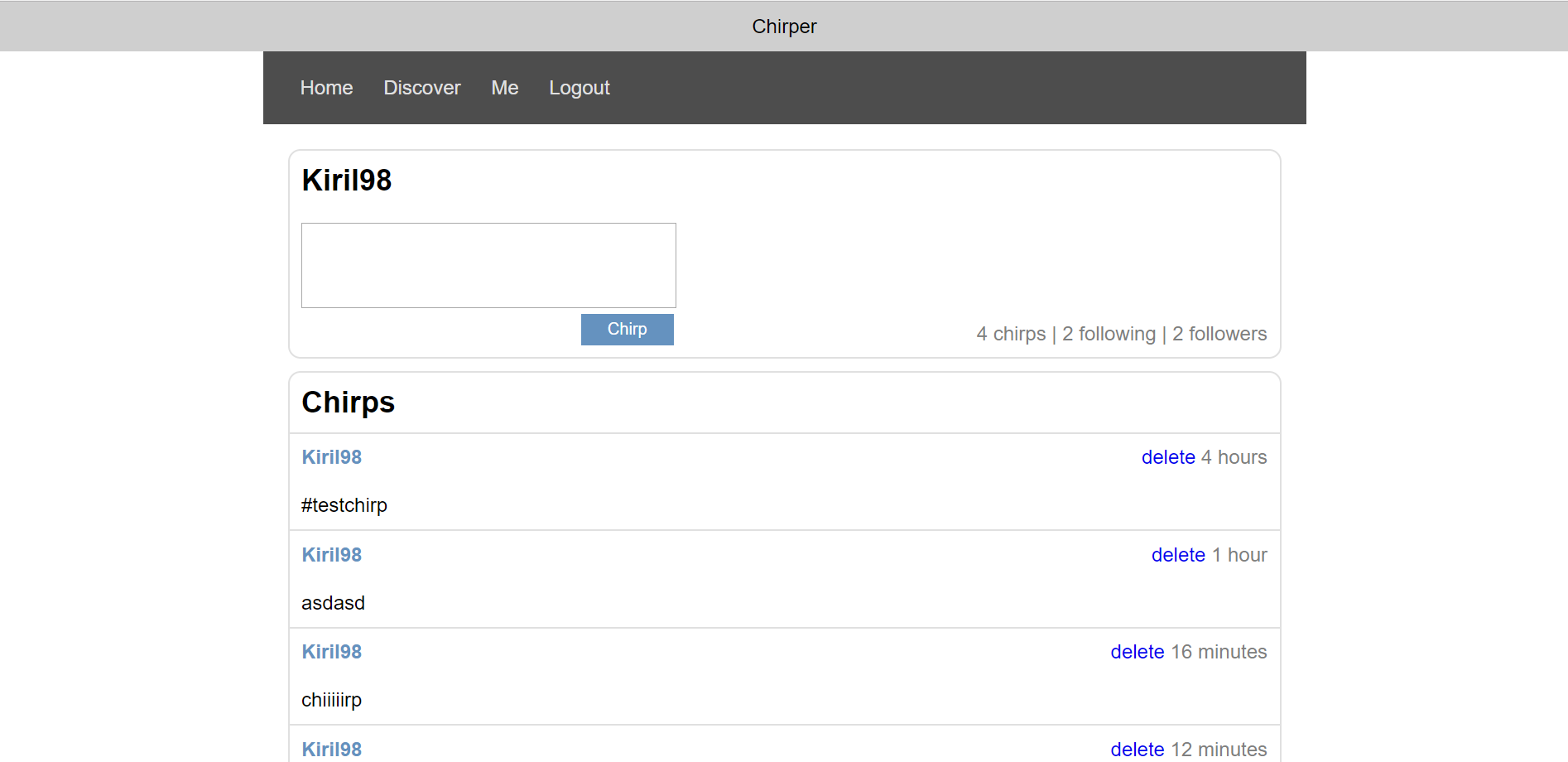


### User Feed (15 pts)

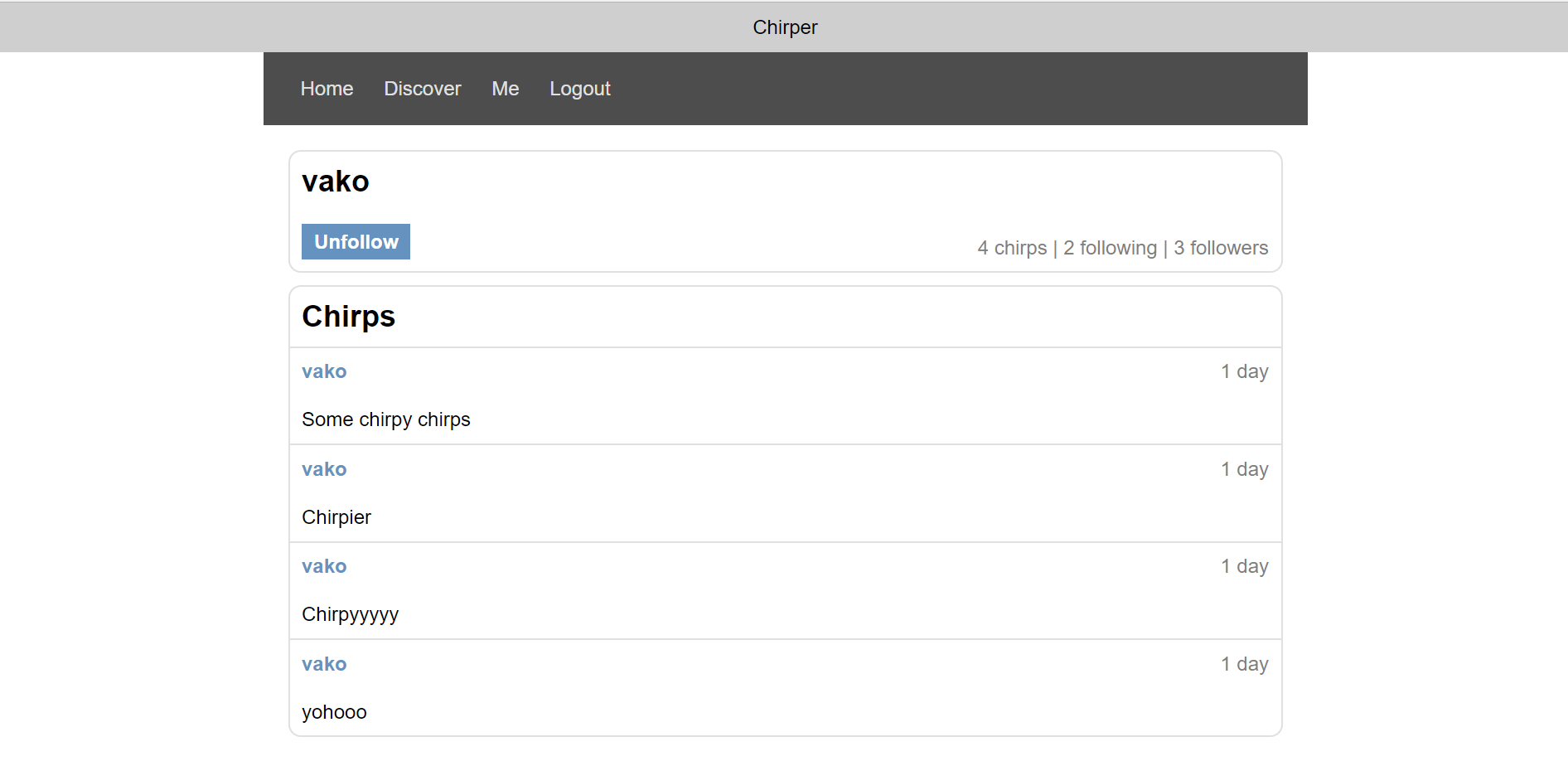
Logged in users can view **individual** feeds (profile page). The **currently logged** user has an individual feed page where he can view his own chirps, **create** a chirp and delete his own **chirps**. Each logged in user can **access** their own feed by clicking the [**Me**] link.

* After a **successful** chirp creation, a notification message “Chirp published.” should be displayed and the **user feed** should be shown again.
* The **user feed** screen should also contain **information** about the **count** all of all chirps that the user has **posted**, number of people that the user is **following**, and number of **followers** that the user has.
* All chirps are listed in **descending** by time **creation**.
* In case of **no chirps**, display ‘No chirps in database’.
* In case of **error** (e.g. Internet connection lost), an error message should be displayed.

**My feed page:**



**Different user feed page:**



The **button** on the top left of the screen can be either **Follow** or **Unfollow** depending on if the user’s username is in the **subscriptions** array.

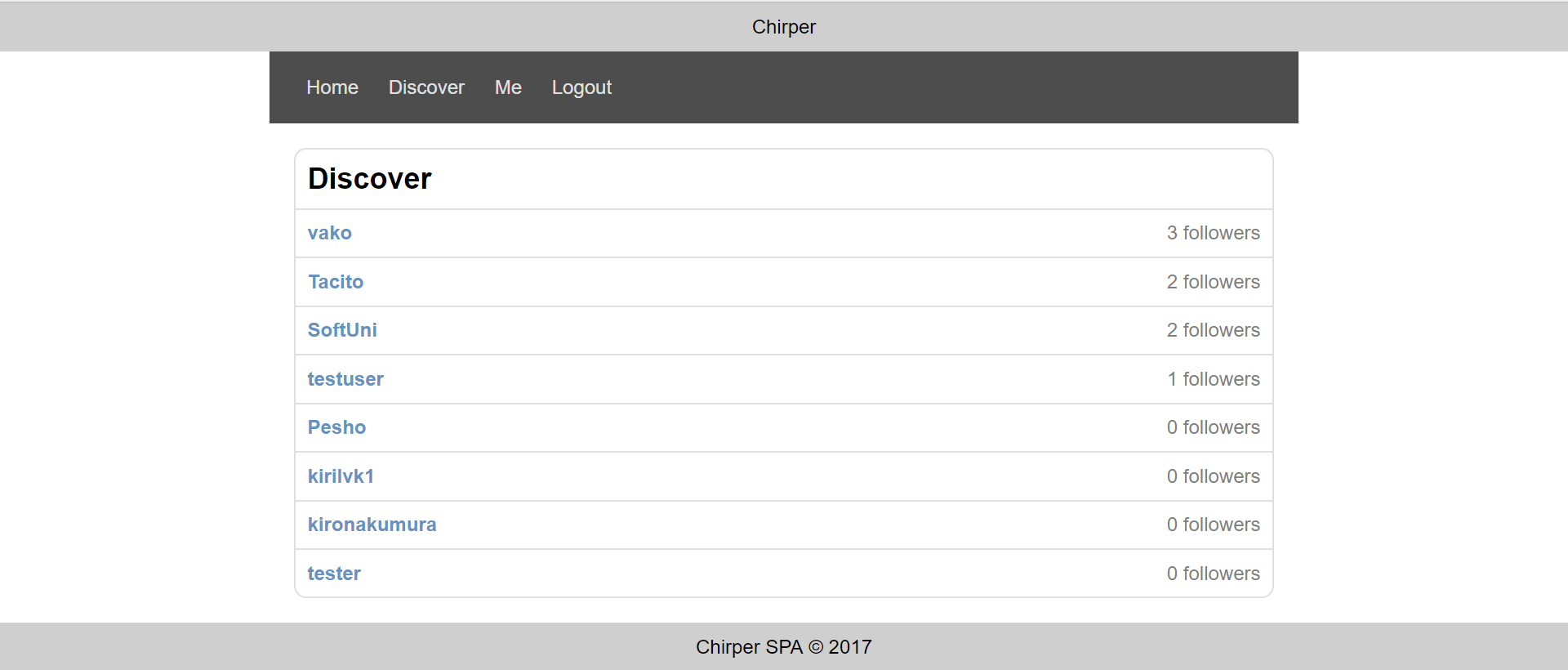
### Delete Chirp (5 pts)

Authors **should** be able to **delete** their **own** chirps by clicking the [**delete**] button on each individual chirp.

* After **successful** post delete a notification message “**Chirp deleted.**” should be displayed and the **user feed** should be shown again.
* 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**.

### Discover People Screen (15 pts)

By clicking the [**Discover**] link from the navigation the **following** screen should be displayed:

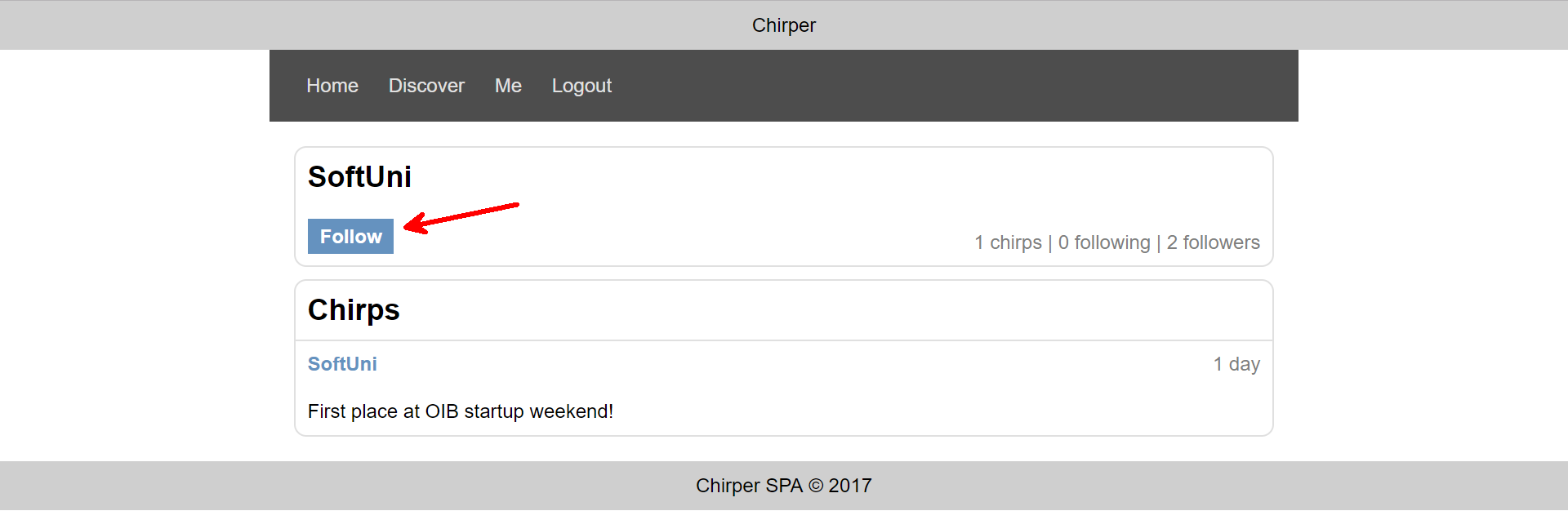


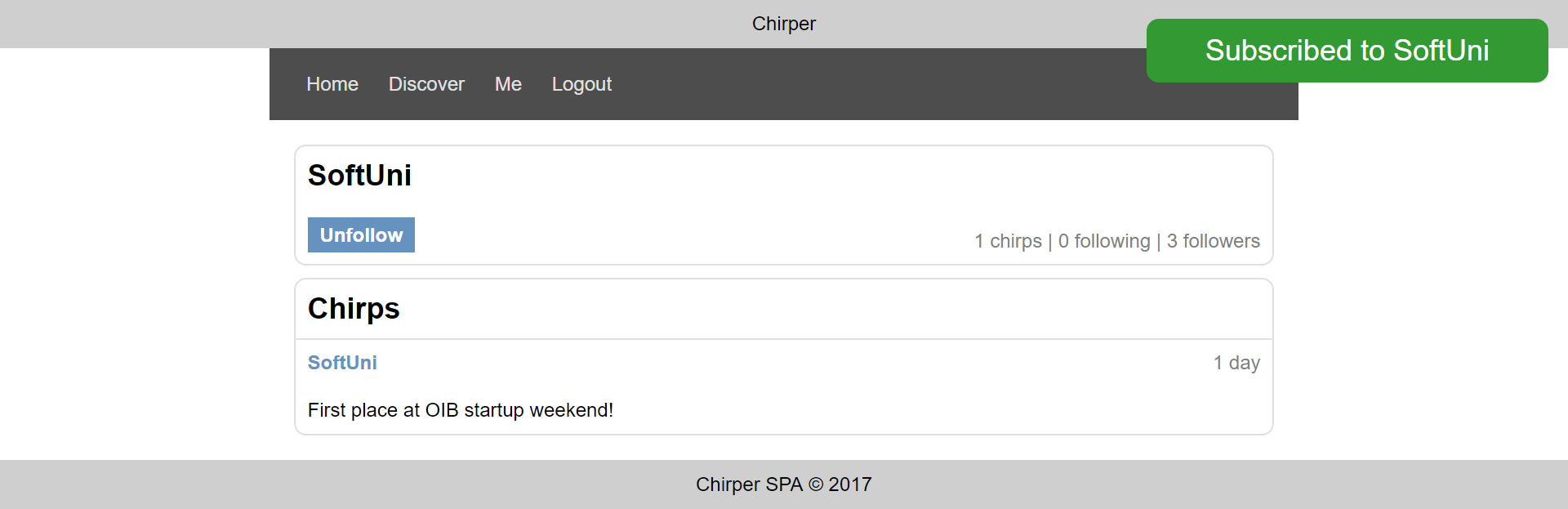
* These are all of the **users** in the database **except** the currently logged.
* Clicking on the username should redirect to the user’s feed page
* Next to **each** username is the number of **followers** that the user has.
* In case of **error** (e.g. Internet connection lost / unauthorized request / missing message), an error message should be displayed.

### Follow User (5 pts)

Logged in users can **follow** other users by clicking on the [**Follow**] link on each user **feed page**.

* After a **successful** following, a notification message “**Subscribed to {username}**” should be **displayed** and the **user feed** should be shown again with **updated** stats.
* The **username** of the **newly** followed user should be added in the **subscriptions** array of the currently logged in user.
* In case of **error** (e.g. Internet connection lost / unauthorized request / missing message), an error message should be displayed.

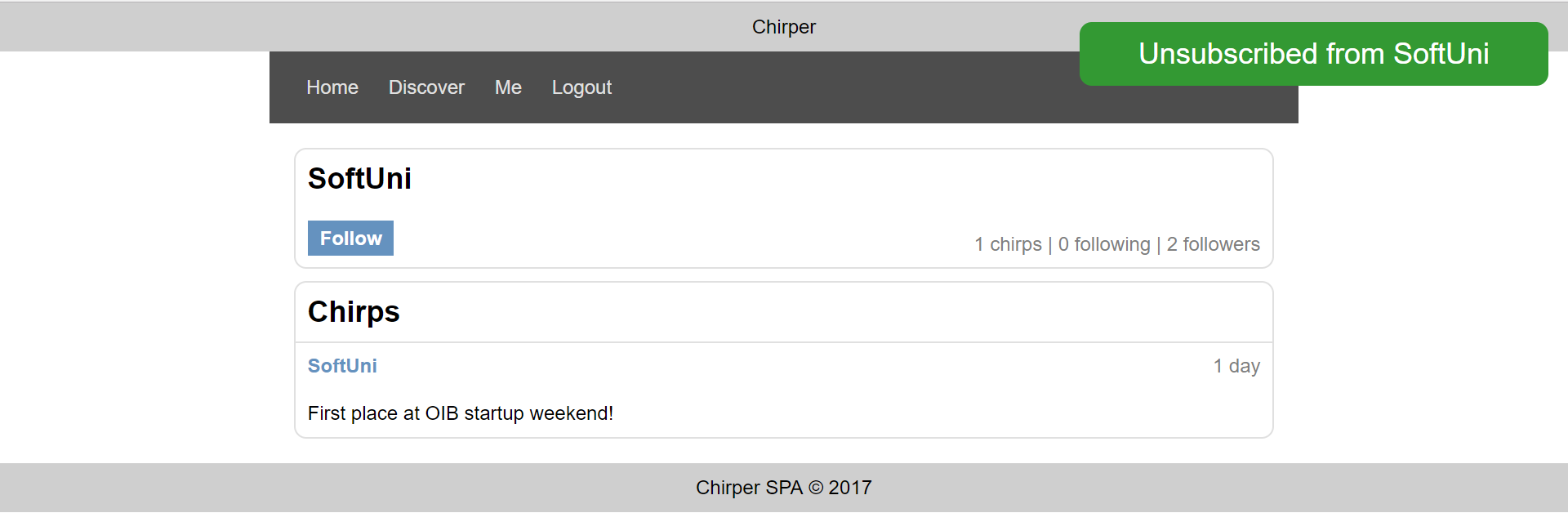




### Unfollow User (5 pts)

Logged in users can **unfollow** other users by clicking on the [**Unfollow**] link on each user **feed page**.

* After a **successful** unfollow, a notification message “**Unsubscribed to {username}**” should be **displayed** and the **user feed** should be shown again with **updated** stats.
* The **username** of the **newly** followed user should be **removed** from the **subscriptions** array of the currently logged in user.
* In case of **error** (e.g. Internet connection lost / unauthorized request / missing message), an error message should be displayed.



## Subtmitting Your Solution

Place in a ZIP file your project folder. Exclude the node\_modules folder. Upload the archive to the Judge.