**COS 216: Netcentric Computer Systems**

**Practical Assignment 4 – API documentation**

**Name: Caleb Johnstone**

**Student number: u19030119**

**Additional parameters:**

1. **sort, filter**
2. **email, password**

**Explanation of the additional parameters and examples illustrating their use**

1. **sort, filter**

These are used when the user updates their settings for the trending page using the sidebar. The settings sidebar can be accessed by clicking the settings button in the footer. This button only shows in the footer if the user is logged in. The user’s API key is stored in local storage when they log in or sign up. A user is logged in automatically once they have signed up. Since the settings button only shows when the user is logged in (i.e there is an API key stored in local storage) then the settings sidebar can only be used by logged in users. The settings sidebar has a radio group for sorting and 2 selects – one for filtering by the genre and one for filtering by the year. Once the user clicks the save button to save their settings a POST request is sent to the API with the values from these 3 input elements.

The request parameters sent to the API follow the following format:

**key=<API key>&type=update&sort=<sort value>&filter=[genre=<genre value>, year=<year value>]&return[]=NULL**

where:

**<API key>** is the user’s API key used to identify their preferences in the user\_preference table in the database.

**<sort value>** can have one of the values: artist, title, album, NULL. The NULL value is when the user has no sort radio button selected when they have clicked the save button. If the sort value is NULL this means that any previously stored sorting value that the user chose must be deleted. If the sort value is one of the other 3 values, this value is stored in the user\_preference table either by updating the existing record or inserting a new record – depending on whether or not the user already had a sort value chosen or not respectively.

**filter=[genre=<genre value>, year=<year value>]** specifies the filtering for the genre and the year. The values **<genre value>** and **<year value>** can be the string value NULL or a value from the relevant select element. If **<genre value>** or **<year value>** is NULL then any previously stored filtering value for the user is deleted. If **<genre value>** or **<year value>** is a value from the relevant select element then this value is stored in the user\_preference table in the database.

The sorting and filtering preference values are stored in the user\_preference table in the database. An example can be seen in the image below:

A screenshot of a cell phone

Description automatically generated

Only the values for the user with an id of 39 in the User table have been shown. Each user preference value is stored in a separate record in the user\_preference table. The column names, from left to right are: userID, type and value. The type is either: “sort”, “filterGenre” or “filterYear”. The value attribute is used to store the actual value to sort or filter by.

The user could then use the settings sidebar to change these preferences. For example, let us say that the user changes their genre to “deep house”, clears the year select - to not filter by year - and sorts by artist. The following image would be the appearance of the settings sidebar:

A close up of a screen

Description automatically generated

Once the user clicks the save button in the top left corner, this will send the update request to the API which will in turn perform the necessary operations on the user\_preference table. The update request to the API would have the following parameters and their values:

**key=<API key of user 39>&type=update&sort=artist&filter=[genre=deep house, year=NULL]&return[]=NULL**

The records relating to the user with id 39 will then be changed to the following:



The “filterGenre” value was updated to “deep house”, the “filterYear” record was deleted and the “sort” value was updated to “artist”.

The selected values in the above image of the settings sidebar will then be selected on the trending page’s input elements, if the trending page is being viewed by the user behind the settings sidebar. This will cause the songs that are displayed to change as a result of these new sorting and filtering values.

1. **email, password**

These parameters are used when validating the user’s login details once the email and password inputs have been checked for having the correct characters and length etc. A request using the POST method, with the type as login is sent to the API with the values of the email and password that the user entered on the login form. The validation done once a login request has been sent is testing whether there is a user with the entered email in the User table and whether the password they entered is the correct one.

A login request follows the following format:

**type=login&email=<email value>&password=<password value>&return[]=key**

where:

**<email value>** is the email that the user entered on the form.

**<password value>** is the password that the user entered on the form.

**key** is the user’s API key. This is only returned when the user has successfully logged in. Otherwise the returned value is an error message stating that the user’s log in details are invalid.

An example that can be used when demoing is the following:

A picture containing monitor, black, clock, dark

Description automatically generated

The login request sent to the API will be the following:

**type=login&email=timj@gmail.com&password=CalebJ#12&return[]=key**

The user details are checked by the API and the API key is returned for the user with the email [**timj@gmail.com**](mailto:timj@gmail.com), since the details entered are valid login details. The user is then shown the launch page.

**All requests that can be made to the API**

**This section is where no extra parameters were used, but it has been included so that the marker will know what the appropriate values for the parameters are when using the API tester and have an idea of how the requests are structured that are used to populate the various pages with the information that they require.**

1. **Using the API key**

**–> Only logged in users can perform the operations that these types perform**

* 1. **Update type**

The request parameters sent to the API follow the following format:

**key=<API key>&type=update&sort=<sort value>&filter=[genre=<genre value>, year=<year value>]&return[]=NULL**

The key parameter gets the value of the user’s API key. This request is sent from the sidebar settings panel to update the users sorting and filtering as described in the “Additional parameters” section above.

* 1. **Get type**
     1. **User preferences**

The request parameters sent to the API follow the following format:

**key=<API key>&type=get&return[]=userPreferences**

The user preferences for sorting and filtering are retrieved from the user\_preference table in the database. Their API key is used to work out their id in the User table and then get the corresponding user preferences from the user\_preference table. These values are then used to populate the selects – for filtering – and radio buttons – for sorting, for the sidebar settings panel and the trending page bar at the top of the page just below the top navigation bar.

* + 1. **User rating**

**key=<API key>&type=get&title=<title of the page>&id=<song ID>&return[]=userRating**

where:

<API key> is the user’s API key

<title of the page> is the title of the page that the song, that we are getting the value of the user rating for, is on

<song ID> is the id of the song in its respective table

The title is used to figure out which table the song is in. So the Trending, New Release and Featured pages mean that the trending\_song, newRelease\_song and featured\_song tables are used respectively. The API key is used to work out the value of the user’s id in the User table. This table name, the song ID and the user id are used to find the user rating in the user\_rating table. This user rating is then used by the relevant page to display the user rating next to the rating slider and set the value attribute of the rating slider.

If the user previously has not rated the song then none is displayed next to the rating slider and the slider’s value is set to zero to show that they have not rated the song yet.

* 1. **Rate type**

The request parameters sent to the API follow the following format:

**key=<API key>&type=rate&title=<title of the page>&id=<song ID>&value=<song rating>&return[]=NULL**

where:

<API key> is the user’s API key

<title of the page> is the title of the page that the song is on

<song ID> is the id of the song in its respective table

<song rating> is the rating the user gives the song out of ten

If the user previously has not rated the song the user’s rating is inserted into the user\_rating table. If the user has rated this song in the past then their rating in the user\_rating table is updated.

1. **Not using the API key**

**These types do not require that the API key is passed as a parameter because user’s that are logged in should also be able to view the songs on the various pages and need to login and will therefore not have an API key in that case. Only logged in user’s can rate songs as well as to be able to view the sidebar settings panel and save their sorting and filtering preferences. Therefore the types relating to these operations require that the API key is passed as a parameter when making a request to the API.**

* 1. **Info type**

The request parameters sent to the API follow the following format:

**type=info&title=<title of the page>&return[]=\***

where:

**<title of the page>** is the title of the page to return all of the song data for. The title can take on one of the following values: trending, newReleases, topRated, featured, calendar. The trending and calendar pages use the same songs. The trending page shows all of the song information whereas the calendar page just shows the song’s artist and title on the release date in the calendar. As a result of this – using title=trending or title=calendar both return the same data when used in a request.

**return[]=\*** is used to state that all of the songs’ data for that page should be returned.

The relevant external API is used to source the song data and return it to the page to be used. Spotify is used for the Trending and Calendar pages while Deezer is used for the New Releases and Featured pages. When extracting the song data from the external API, if the song is not in the database table already, then the relevant data is extracted for that song and inserted into the table.

Utilising the data that is already in the database optimises the retrieval of the data needed for a page and reduces the time it takes for a response to be returned. As a result of this – load times on the client side are lower, making for a better user experience.

* 1. **Login type**

The request parameters sent to the API follow the following format:

**type=login&email=<email value>&password=<password value>&return[]=key**

The login type has been explained in the “Additional parameters” section above.