# Homepage

* Displays a loading spinner until: the stores products and the user’s cart has been fetched from the database. The fetched data is stored in a client-side array *(JavaScript array)*. If the user is not logged in and does not have a cart, a new cart is created using their session ID. If the user is logged in and does not have a cart or their cart has expired, a new one is created using their user document ID.
* The fetched products are returned in descending order so that the most recently added product is at the top of the page.
* All cart documents in the carts collection that are older than 30 minutes are deleted.
* 6 products are displayed per page, and pagination is used (located at the bottom of the page) to navigate through the products. The products are displayed by iterating over the products array.
* The user can view a product or add it to their cart. When a user views a product, a modal appears. The modals content (e.g. title, description, information, etc.) is obtained from the products array.
* When a user adds a product to their cart, a loading spinner appears. The product is fetched from the database to check that the product is in stock. If it’s in stock, the product is added to the user’s cart and the product in the products JavaScript array is updated with a new quantity. The product document in the database is also updated with a new quantity. If the product is not in stock or the selected quantity is greater than the available quantity, the modal and card is updated. A green animation effect is displayed on the shopping cart button in the navigation bar to notify that a product has been added.
* Users can sort the listed products by newest and price. They can also filter by decade and a price range.
* Users can search for a product by title, actors, year and directors. The search is dynamic and case-insensitive so different products will appear as the user searches. The number of results are also displayed. Additionally, users can sort and filter the results.

# Shopping Cart

* Guest users and registered users are able to purchase products.
* The cart document has a field named ‘userID’ which is assigned the user’s session ID if they’re not logged in, or their user document ID.
* The number of products in a user’s cart is displayed next to the shopping cart icon in the navigation bar.
* Clicking the shopping cart icon will display a menu. If the cart is empty, “Your shopping cart is empty!” is displayed. Otherwise, a table containing all of the products in the user’s cart is displayed.
* The shopping cart’s table displays the products name, format, quantity, price and a delete button. It also displays a price breakdown.
* Users can delete products from their cart. The quantity for the product that was deleted from a user’s cart will be added back to the products document in the database. This also occurs when the shopping cart expires (30-minute time limit).
* A checkout button is displayed on the menu and will redirect the user to the checkout page when clicked.
* A product that is added to the cart is displayed at the bottom of the table if other products exist in the cart already.
* The shopping cart’s price breakdown is updated every time a user adds a new product to it.

# Log In & Registration

* Users can log in and register by clicking the user icon located in the navigation bar.
* The registration form cannot be submitted until all form fields are valid.
* Invalid form fields have a red outline with an error message.
* Registration and log in form fields are validated using the Validator library and regular expressions.
* E-mail addresses and phone numbers that are already in use cannot be used when registering.
* When a user submits the registration form, a user document is added to the user’s collection in the database. The user document contains all of the user input entered in the registration form except for the confirm password field. Before sending the user input to the backend, the user’s password is hashed rather leaving it in plain text.
* The user must enter a valid e-mail address and password to log in. To validate the log in credentials, the e-mail address and password (which is hashed) is sent to the backend. The backend fetches a user document containing the e-mail address submitted. If a user document with that e-mail has been found and the password matches the password in the user document, the user is logged in. Otherwise, an error message stating that the log in credentials are invalid is displayed.
* When a user is logged in, the user’s account information is stored in a PHP session variable – ‘userInformation’. The user’s document ID is also stored in another session variable – ‘userID’. The page is then reloaded so that the website can prepare itself again (*note: see the first bullet point under the homepage heading*).

# User Account

* The user account modal allows the user to edit their profile, view their order history, change their password and sign out. If the user is an admin, there is a button that will redirect them to the CMS panel when clicked.
* Upon clicking the ‘edit profile’ button in the profile section, the profile form fields are enabled. The user can edit all fields apart from their date of birth. When saving the profile changes, the form input is sent to the backend and the backend updates the user’s user document in the database. The session variable containing the user’s information is also updated. The user may also cancel their changes. When cancelling changes, a request is made to the backend for the userInformation session variable. When the userInformation session variable is returned, the profile form fields are updated with the information stored in the userInformation variable. The form fields are disabled afterwards.
* The order history section displays all of the user’s order history in descending order. If the user has not placed any orders, an error message is displayed. If the user has placed orders, a table is displayed. Each row of the table contains the purchase date, order number, grand total, order status and a view button. If the view button is clicked, the order ID is sent to the backend. The backend searches the orders collection for an order with the same ID and returns a html page which is loaded into the order history content container. The user can also return by clicking the Go back button. The go back button will make another request to the backend which returns the html for the order history table. The html is, once again, loaded into the order history content container.
* Users can change their password. For a user to successfully change their password, they must enter their current password and a new password. Both the current password and new password is hashed and validated in the backend. If the submitted current password matches the user’s password stored in the database, the user’s document is updated with the new password. Otherwise, an error message is displayed.
* When a user signs out a request is made to the backend. The backend will set session variables such as loggedIn, isAdmin, userInformation and userID to null/false and refresh the page.

# Checkout

* If the user’s cart is empty, an error message is displayed on the page. The user will not be able to checkout until their cart contains a product.
* The first tab of the checkout page (‘Shopping Cart’) displays the user’s shopping cart and a price breakdown. When clicking the ‘Checkout’ button the tab is changed to the ‘Shipping & Billing’ tab. The user will be required to enter their shipping and billing information. After clicking ‘Continue’, they will need to enter their card details. A regular expression is used to validate the card number. Finally, the user can review their order. The ‘Review & Place Order’ tab displays their shopping cart, a price breakdown, their shipping and billing information, and delivery option. When satisfied, they can place their order by clicking the ‘Place Order’ button. Clicking the ‘Place Order’ button will send all of the information previously listed (*their shopping cart, price breakdown, etc.*) to the backend. The backend will store this in a new orders document. The orders document will also contain the user’s ID if they’re logged in. The user’s ID is added to the document so that it can be found when search for the user’s previous orders.
* After an order has been stored in the database, an order number is returned from the backend. The order confirmation page is loaded into the page’s container and displays an order successful message with the user’s order number. The user can then go back to the homepage and see the order in his/her order history. Also, the user’s cart is emptied.

# Tracking & Recommendations

* Tracked items are stored in an array in localStorage.
* The ‘tracking’ item in localStorage is an array that contains objects. Each object contains the date and time it was added to localStorage, the product’s ID and a view count.
* The view count is incremented by 1 every time the user clicks the ‘View Product’ button located on the product’s card. Initially, the view count is set to 1.
* Whenever a user clicks the ‘View Product’ button located on the product’s card, the ‘tracking’ item in localStorage is retrieved. Objects with a date and time older than 30 minutes from the current time are removed from the array. Then, the array is searched for the viewed product ID. If the ID is found, the view count is incremented by 1. Otherwise, an object (as mentioned in bullet point 2) is created and pushed into the array. Finally, the array is sorted by view count in descending order, and the ‘tracking’ item in localStorage is updated with the new and sorted array of objects.

**Side note:** Recommendations were not implemented but it may be worth mentioning that we could use the products stored in localStorage to recommend movies that have similar actors, directors, etc. on the checkout page.

# CMS Panel

* The CMS panel allows administrator accounts to add, remove, edit and modify order statuses.
* To add a product, the administrator must fill all of the fields and select an image to upload for the product’s cover. When the form has been submitted, the backend adds a product document to the database. The product document contains all of the information that was input into the form fields and the name of the file chosen as the product’s cover. The product cover is saved to disk, server-side. Storing the name of the product’s cover in the database allows the file to be found on disk.
* When clicking the ‘Remove Product’ tab, all products are fetched from the database and stored in a client-side array. Each product is displayed in a column and can be edited, deleted or searched for by title, actors and directors. Removing the product will delete the product’s document from the products collection and the product’s cover from the servers disk. Clicking ‘Edit’ will open up a modal. The modal contains a form with the product’s information set in the text fields. The administrator can change each text field and save the changes by clicking the ‘Save Changes’ button. Saving changes will update the product’s document in the products collection and the product’s cover if a new cover has been uploaded. After removing or editing a product, the products are fetched once again and stored in a client-side array with updated products/product information.
* Similarly, the orders tab fetches all of the orders from the database and stores them in a client-side array. The administrator can change the order status of an order (pending, dispatched, delivered), view an order, search for an order by order number, status and name. A modal is displayed when clicking ‘View’ on an order. The modal displays the order’s information, e.g. shipping and billing information, package, date of purchase, etc. Making a change to an order’s status will result in the order being updated in the database and a new orders array being returned and stored in a client-side array.

# Security, Privacy & Legal Issues

**Some but there may be more:**

* No terms of service
* No privacy policy
* No customer support
* No secure SSL
* No feature that temporarily locks the account when the log in credentials have been entered wrong 3 times in a row
* No alert to inform the user that data is being stored on their computer (tracking – localStorage)