

Client Side Scripting

URLs

GitHub Repository	https://github.com/mandyfarrugia/mf_swd62b_css
Remote Website	https://mf-swd62b-css.onrender.com/

Screenshots

KU2.3 - Running the provided backend

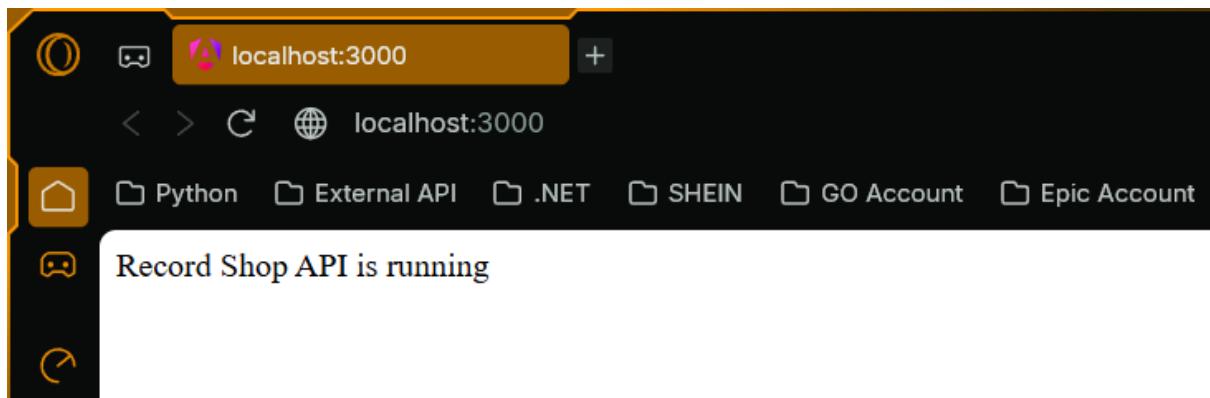
```
Mandy@DESKTOP-S43VD18 MINGW64 ~/Documents/mf_swd62b_css/backend (main)
● $ npm install
  added 67 packages, and audited 68 packages in 3s
  20 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
⚡
Mandy@DESKTOP-S43VD18 MINGW64 ~/Documents/mf_swd62b_css/backend (main)
○ $ npm start
> record_store_backend@1.0.0 start
> node server.js
Record Shop API listening on http://localhost:3000
█
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

● mandyfarrugia@Mandys-Mac-mini mf_swd62b_css % cd backend
● mandyfarrugia@Mandys-Mac-mini backend % npm install
  added 67 packages, and audited 68 packages in 1s
  20 packages are looking for funding
    run `npm fund` for details
  found 0 vulnerabilities
○ mandyfarrugia@Mandys-Mac-mini backend % npm start
> record_store_backend@1.0.0 start
> node server.js
Record Shop API listening on http://localhost:3000
█
```

KU2.3 - Accessing the API's documentation in the browser

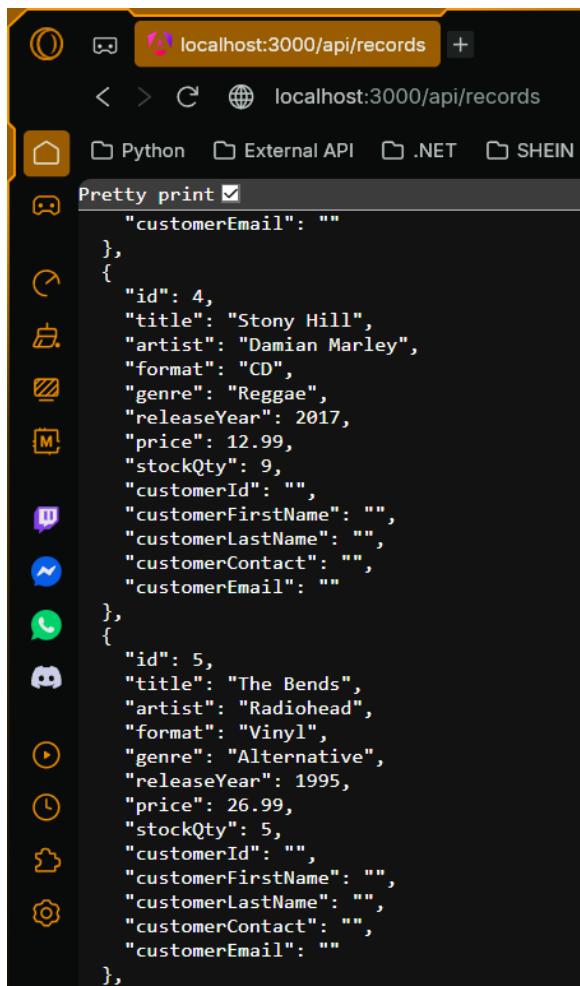
Health Check



Records

```
[{"id": 1, "title": "Californication", "artist": "Red Hot Chili Peppers", "format": "Vinyl", "genre": "Rock", "releaseYear": 1999, "price": 29.99, "stockQty": 8, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}, {"id": 2, "title": "Black Summer", "artist": "Red Hot Chili Peppers", "format": "CD", "genre": "Rock", "releaseYear": 2022, "price": 14.99, "stockQty": 12, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}, {"id": 3, "title": "Audioslave", "artist": "Audioslave", "format": "Vinyl", "genre": "Rock", "releaseYear": 2002, "price": 27.99, "stockQty": 6, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": ""}, ...]
```

The screenshot shows a web browser window with the URL 'localhost:3000/api/records' in the address bar. The page displays a JSON response with three records. Each record object contains fields such as id, title, artist, format, genre, release year, price, stock quantity, and customer information. The browser interface includes a navigation bar with icons for back, forward, and search, and a menu bar with links to Python, External API, .NET, and SHEIN. A 'Pretty print' checkbox is checked in the top right corner of the JSON response area.



A screenshot of a web browser window titled "localhost:3000/api/records". The page displays a JSON array of records. The "Pretty print" checkbox is checked, which formats the JSON with indentation and line breaks. The JSON data includes fields such as id, title, artist, format, genre, releaseYear, price, stockQty, customerId, customerFirstName, customerLastName, customerContact, and customerEmail. The records are numbered 4 and 5.

```
Pretty print
  "customerEmail": "",
},
{
  "id": 4,
  "title": "Stony Hill",
  "artist": "Damian Marley",
  "format": "CD",
  "genre": "Reggae",
  "releaseYear": 2017,
  "price": 12.99,
  "stockQty": 9,
  "customerId": "",
  "customerFirstName": "",
  "customerLastName": "",
  "customerContact": "",
  "customerEmail": ""
},
{
  "id": 5,
  "title": "The Bends",
  "artist": "Radiohead",
  "format": "Vinyl",
  "genre": "Alternative",
  "releaseYear": 1995,
  "price": 26.99,
  "stockQty": 5,
  "customerId": "",
  "customerFirstName": "",
  "customerLastName": "",
  "customerContact": "",
  "customerEmail": ""
},
```

```
{
  "id": 6,
  "title": "OK Computer",
  "artist": "Radiohead",
  "format": "Vinyl",
  "genre": "Alternative",
  "releaseYear": 1997,
  "price": 28.99,
  "stockQty": 4,
  "customerId": "",
  "customerFirstName": "",
  "customerLastName": "",
  "customerContact": "",
  "customerEmail": ""
}
```



A screenshot of a Brave browser window displaying a JSON API response. The URL in the address bar is `localhost:3000/api/records`. The response is a pretty-printed JSON array containing four records, each representing a music album.

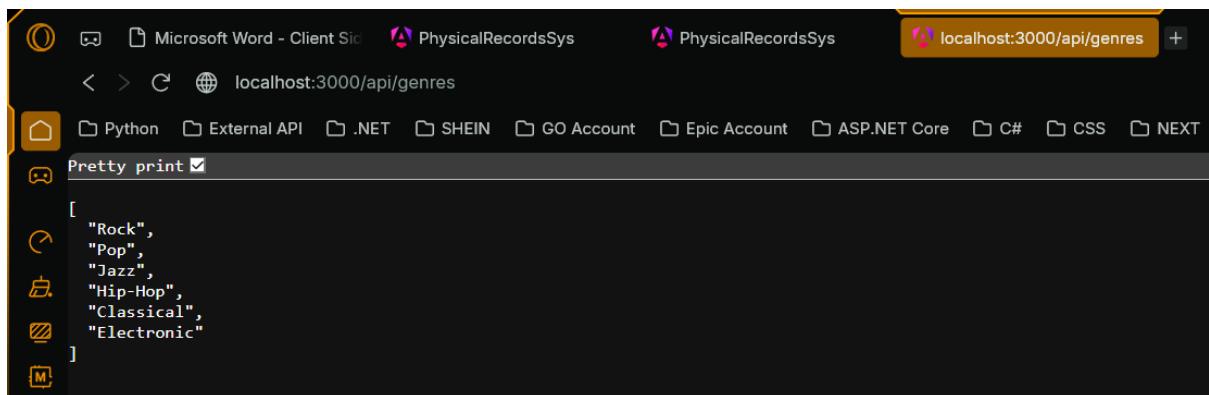
```
[  
  {  
    "id": 1,  
    "title": "Californication",  
    "artist": "Red Hot Chili Peppers",  
    "format": "Vinyl",  
    "genre": "Rock",  
    "releaseYear": 1999,  
    "price": 29.99,  
    "stockQty": 8,  
    "customerId": "",  
    "customerFirstName": "",  
    "customerLastName": "",  
    "customerContact": "",  
    "customerEmail": ""  
  },  
  {  
    "id": 2,  
    "title": "Black Summer",  
    "artist": "Red Hot Chili Peppers",  
    "format": "CD",  
    "genre": "Rock",  
    "releaseYear": 2022,  
    "price": 14.99,  
    "stockQty": 12,  
    "customerId": "",  
    "customerFirstName": "",  
    "customerLastName": "",  
    "customerContact": "",  
    "customerEmail": ""  
  },  
  {  
    "id": 3,  
    "title": "Audioslave",  
    "artist": "Audioslave",  
    "format": "Vinyl",  
    "genre": "Rock",  
    "releaseYear": 2002,  
    "price": 27.99,  
    "stockQty": 6,  
    "customerId": "",  
    "customerFirstName": "",  
    "customerLastName": "",  
    "customerContact": "",  
    "customerEmail": ""  
  },  
  {  
    "id": 4,  
    "title": "Stony Hill",  
    "artist": "Damian Marley",  
    "format": "CD",  
    "genre": "Reggae",  
    "releaseYear": 2017,  
    "price": 12.99,  
    "stockQty": 9,  
    "customerId": "",  
    "customerFirstName": ""  
  }]
```



A screenshot of a web browser window titled "localhost:3000/api/records". The page displays a JSON array of records. Each record object contains fields such as "format", "genre", "releaseYear", "price", "stockQty", "customerId", "customerFirstName", "customerLastName", "customerContact", and "customerEmail". The records listed are:

```
[{"id": 1, "title": "The White Stripes", "artist": "The White Stripes", "format": "Vinyl", "genre": "Rock", "releaseYear": 2002, "price": 27.99, "stockQty": 6, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}, {"id": 2, "title": "Stony Hill", "artist": "Damian Marley", "format": "CD", "genre": "Reggae", "releaseYear": 2017, "price": 12.99, "stockQty": 9, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}, {"id": 3, "title": "The Bends", "artist": "Radiohead", "format": "Vinyl", "genre": "Alternative", "releaseYear": 1995, "price": 26.99, "stockQty": 5, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}, {"id": 4, "title": "OK Computer", "artist": "Radiohead", "format": "Vinyl", "genre": "Alternative", "releaseYear": 1997, "price": 28.99, "stockQty": 4, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}]
```

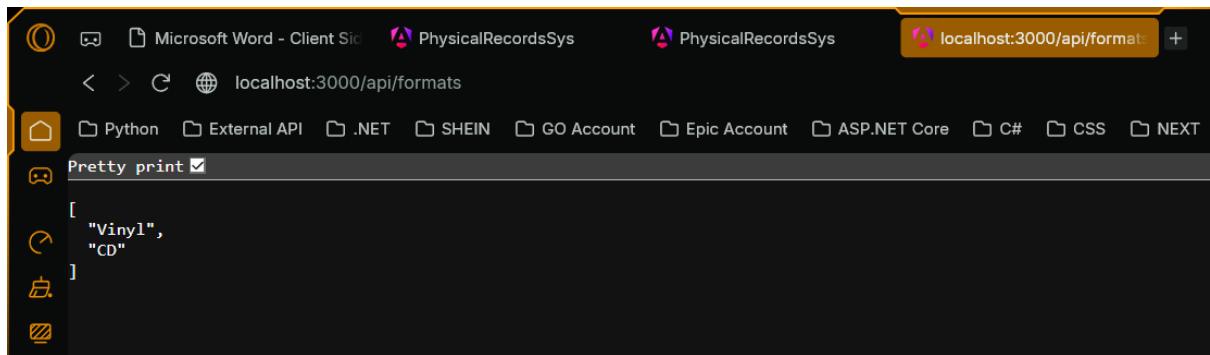
Genres



A screenshot of a web browser window titled "localhost:3000/api/genres". The page displays a JSON array of genre names. The genres listed are:

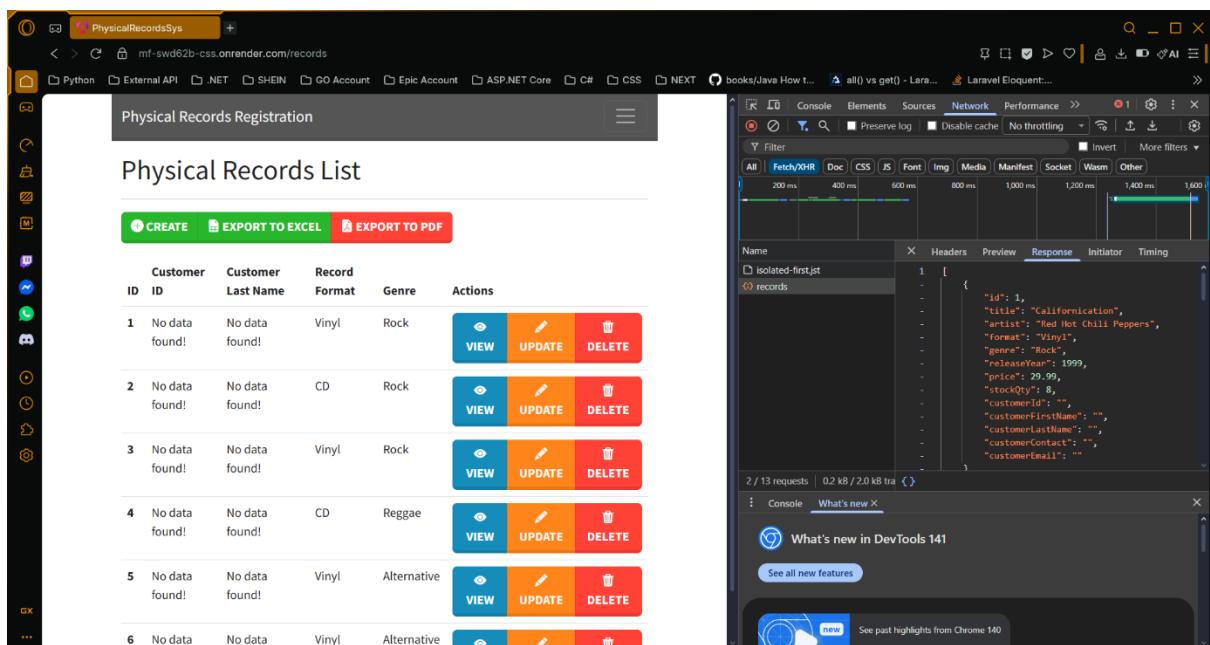
```
[{"name": "Rock"}, {"name": "Pop"}, {"name": "Jazz"}, {"name": "Hip-Hop"}, {"name": "Classical"}, {"name": "Electronic"}]
```

Formats

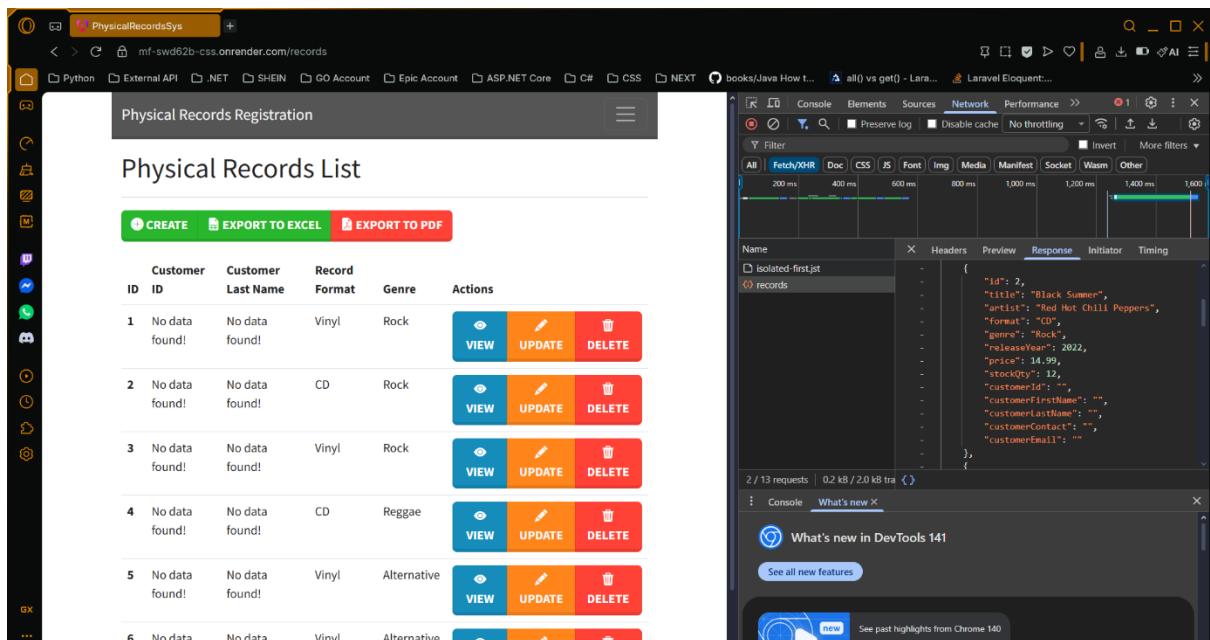


```
[ "Vinyl", "CD" ]
```

KU2.3 - Execute the GET request to retrieve all records from within the API documentation using the version of the API that enforces authentication



```
[
  {
    "id": 1,
    "title": "Californication",
    "artist": "Red Hot Chilli Peppers",
    "format": "Vinyl",
    "genre": "Rock",
    "releaseYear": 1999,
    "price": 20.99,
    "stockQty": 8,
    "customerId": "",
    "customerFirstName": "",
    "customerLastName": "",
    "customerContact": "",
    "customerEmail": ""
  },
  {
    "id": 2,
    "title": "Black Summer",
    "artist": "Red Hot Chilli Peppers",
    "format": "CD",
    "genre": "Rock",
    "releaseYear": 2002,
    "price": 14.99,
    "stockQty": 12,
    "customerId": "",
    "customerFirstName": "",
    "customerLastName": "",
    "customerContact": "",
    "customerEmail": ""
  }
]
```



```
[
  {
    "id": 1,
    "title": "Californication",
    "artist": "Red Hot Chilli Peppers",
    "format": "Vinyl",
    "genre": "Rock",
    "releaseYear": 1999,
    "price": 20.99,
    "stockQty": 8,
    "customerId": "",
    "customerFirstName": "",
    "customerLastName": "",
    "customerContact": "",
    "customerEmail": ""
  },
  {
    "id": 2,
    "title": "Black Summer",
    "artist": "Red Hot Chilli Peppers",
    "format": "CD",
    "genre": "Rock",
    "releaseYear": 2002,
    "price": 14.99,
    "stockQty": 12,
    "customerId": "",
    "customerFirstName": "",
    "customerLastName": "",
    "customerContact": "",
    "customerEmail": ""
  }
]
```

The screenshot shows a web browser window with the URL mf-swd62b-css.onrender.com/records. The main content area displays a table titled "Physical Records List" with the following data:

ID	Customer ID	Customer Last Name	Record Format	Genre	Actions
1	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
2	No data found!	No data found!	CD	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
3	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
4	No data found!	No data found!	CD	Reggae	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
5	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
6	No data	No data	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>

Below the table are three buttons: "CREATE", "EXPORT TO EXCEL", and "EXPORT TO PDF".

The right side of the screenshot shows the Network tab of the developer tools. A request labeled "isolated-firstlist" is selected, showing its response. The response body contains JSON data for record ID 3:

```
{
  "id": 3,
  "title": "Audioslave",
  "artist": "Audioslave",
  "format": "Vinyl",
  "genre": "Rock",
  "releaseYear": 2002,
  "price": 27.99,
  "stockQty": 6,
  "customerId": "",
  "customerFirstName": "",
  "customerLastName": "",
  "customerContact": "",
  "customerEmail": ""
}
```

The developer tools also show 2 / 13 requests and 0.2 kB / 2.0 kB tra.

This screenshot is identical to the one above, showing the "Physical Records List" page and the developer tools Network tab. The table data and JSON response for record ID 3 remain the same.

The screenshot shows a web browser window with the URL mf-swd62b-css.onrender.com/records. The main content area displays a table titled "Physical Records List" with the following data:

ID	Customer ID	Customer Last Name	Record Format	Genre	Actions
1	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
2	No data found!	No data found!	CD	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
3	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
4	No data found!	No data found!	CD	Reggae	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
5	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
6	No data	No data	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>

Below the table are three buttons: "CREATE", "EXPORT TO EXCEL", and "EXPORT TO PDF".

The right side of the screenshot shows the "Network" tab of the developer tools. A request labeled "isolated-firstlist" is selected, showing a response payload:

```
[{"id": 5, "title": "The Bends", "artist": "Radiohead", "format": "Vinyl", "genre": "Alternative", "releaseYear": 1995, "price": 26.99, "stockQty": 5, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}, {"id": 6, "title": "OK Computer", "artist": "Radiohead", "format": "Vinyl", "genre": "Alternative", "releaseYear": 1997, "price": 28.99, "stockQty": 4, "customerId": "", "customerFirstName": "", "customerLastName": "", "customerContact": "", "customerEmail": ""}]
```

This screenshot is identical to the one above, showing the "Physical Records List" page and the developer tools Network tab. The table data and response payload are the same.

SE2.2 – Display error message when login fails

The screenshot shows a browser window with a "Physical Records Registration" login form. The "Email" field contains "mandy.farrugia.c10352@mcast.edu.mt" and the "Password" field contains "....". A red error message "Invalid email or password." is displayed below the fields. To the right, the browser's developer tools Network tab shows a request for "login" with a response status of 200 and a JSON payload containing the error message: {"message": "Invalid email or password."}.

SE1.1 - Develop a web application front-end using common concepts such as data binding, events, and methods

Viewing individual record

The screenshot shows a browser window displaying a record detail page for "ok computer" by Radiohead. The page includes fields for Year Released (1997), Format (Vinyl), Genre (Hip-Hop), and Stock Quantity (Low in stock). Below this, there is a "Customer Information" section with fields for Customer ID (0473501), First Name (Mandy), Last Name (Farrugia), Contact (99819288), and Email (mandy.farrugia.c10352@mcast.edu.mt). At the bottom are "UPDATE" and "DELETE" buttons. To the right, the browser's developer tools Network tab shows a request for "6" with a response status of 200, displaying the JSON data for the record: {"id": 6, "title": "OK Computer", "artist": "Radiohead", "format": "Vinyl", "genre": "Hip-Hop", "releaseYear": 1997, "price": 28.99, "stockQty": 4, "customerId": "0473501", "customerFirstName": "Mandy", "customerLastName": "Farrugia", "customerContact": "99819288", "customerEmail": "mandy.farrugia.c10352@mcast.edu.mt"}.

Editing existing record

The screenshot shows a browser window for 'PhysicalRecordsSys' at the URL <https://mf-swd62b-css.onrender.com/records/edit/6>. The user is logged in as 'Alex Admin (admin@recordshop.com)'. The main content area displays a form titled 'Edit existing physical record' with fields for Title ('OK Computer'), Artist ('Radiohead'), Format ('Vinyl'), Genre ('Classical'), Release Year ('1997'), and Price ('28.99'). To the right of the browser window is the Network tab of the developer tools, which shows a request to 'records' with an ID of 6. The response body contains the JSON object:

```
1 {  
- "id": 6,  
- "title": "OK Computer",  
- "artist": "Radiohead",  
- "format": "Vinyl",  
- "genre": "Alternative",  
- "releaseYear": 1997,  
- "price": 28.99,  
- "stockQty": 4,  
- "customerId": "",  
- "customerFirstName": "",  
- "customerLastName": "",  
- "customerContact": "",  
- "customerEmail": ""  
}
```

This screenshot is identical to the one above, showing the same browser window for 'PhysicalRecordsSys' at the URL <https://mf-swd62b-css.onrender.com/records/edit/6>. The user is still logged in as 'Alex Admin (admin@recordshop.com)'. The form fields are the same: Title ('OK Computer'), Artist ('Radiohead'), Format ('Vinyl'), Genre ('Classical'), Release Year ('1997'), and Price ('28.99'). The Network tab in the developer tools also shows a request to 'records' with an ID of 6, but the response body is now an array: `["Vinyl", "CD"]`.

Physical Records Registration

Edit existing physical record

Title: OK Computer

Artist: Radiohead

Format: Vinyl

Genre: Classical

Release Year: 1997

Price: 28.99

Network tab in developer tools showing a response for 'records' containing the following data:

```
[{"name": "records", "value": "[\"Rock\", \"Pop\", \"Jazz\", \"Hip-Hop\", \"Classical\", \"Electronic\"]"}]
```

SE2.2 – Redirect user to list of records if login succeeds

Payload

Physical Records Registration

Logout

Records

Add records

Logged in as Alex Admin (admin@recordshop.com)

Physical Records List

ID	Customer ID	Customer Last Name	Record Format	Genre	Actions
1	No data found!	No data found!	Vinyl	Rock	VIEW UPDATE DELETE
2	No data found!	No data found!	CD	Rock	VIEW UPDATE DELETE
3	No data found!	No data found!	Vinyl	Rock	VIEW UPDATE DELETE
4	No data found!	No data found!	CD	Reggae	VIEW UPDATE DELETE

Network tab in developer tools showing a POST request to 'login' with the following payload:

```
{
  "email": "admin@recordshop.com",
  "password": "password"
}
```

Response

The screenshot shows a browser window with the title "PhysicalRecordsSys". The main content area displays a "Physical Records List" page with a table showing four records. The columns are Customer ID, Last Name, Record Format, Genre, and Actions. Each row has a "VIEW" button and an "UPDATE" button. The table header includes "Customer" and "Record". The bottom right of the table has "CREATE", "EXPORT TO EXCEL", and "EXPORT TO PDF" buttons. The developer tools Network tab is open, showing a request for "/records" with a status of 200 OK. The response payload is JSON: [{"id":3,"name":"Alex Admin","email":"admin@recordshop.com","role":"admin"}]. Other requests listed are "records" (status 200) and "login" (status 200).

KU2.4 - Implement the necessary functionality to ensure that a user is only able to go to a particular route if he/she has the appropriate user role

```
frontend > physical_records_sys > src > guards > TS role-guard.ts > roleGuard
You, 1 hour ago | 2 authors (You and one other)
1 import { inject } from '@angular/core';
2 import { CanActivateFn, Router } from '@angular/router';
3 import { AuthenticationService } from '../services/authentication-service';
4
5 export const roleGuard: CanActivateFn = (route, state) => {
6   const authenticationService = inject(AuthenticationService);
7   const router = inject(Router);
8
9   const allowedRoles = route.data['roles'] as string[]; //Fetch the roles metadata specified within data in the routing configuration.
10
11   if (!authenticationService.isAuthenticated()) {
12     router.navigate(['/login']); //Override default navigation by redirecting user to the login page instead if they are not yet authenticated.
13     return false; //Block navigation to the path specified in routerLink.
14   }
15
16   //If no roles are specified (hence accessible by either anonymous or any logged in user) or any role can access this route, then allow access.
17   if (!allowedRoles || allowedRoles.includes(authenticationService.user!.role)) {
18     return true;
19   }
20
21   router.navigate(['/records']); You, last week • Guarding seems to be working. ...
22   return false; //Block navigation to the path specified in routerLink.
23 }
```

KU2.4 - Ensure that buttons are only enabled if the user has the required user role associated with the functionality of the button

Clerk

The screenshot shows a web browser window titled "PhysicalRecordsSys" with the URL "https://mf-swd62b-css.onrender.com/records". The page header includes "Physical Records Registration", "Logout", "Records", "Add records", and a "Logged in as Chris Clerk (clerk@recordshop.com)" message. The main content is titled "Physical Records List" and features a table with the following columns: ID, Customer ID, Customer Last Name, Record Format, Genre, and Actions. The table contains 6 rows, each with a unique ID (1-6) and placeholder data ("No data found!"). The "Actions" column for each row contains three buttons: "VIEW" (blue), "UPDATE" (orange), and "DELETE" (red). All three buttons are visible and functional for every row.

ID	Customer ID	Customer Last Name	Record Format	Genre	Actions
1	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
2	No data found!	No data found!	CD	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
3	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
4	No data found!	No data found!	CD	Reggae	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
5	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
6	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>

Manager

The screenshot shows a web browser window titled "PhysicalRecordsSys" with the URL "https://mf-swd62b-css.onrender.com/records". The page header includes "Physical Records Registration", "Logout", "Records", "Add records", and a "Logged in as Mandy Manager (manager@recordshop.com)" message. The main content is titled "Physical Records List" and features a table with the same structure as the Clerk's view. The table contains 6 rows with placeholder data. The "Actions" column for each row contains three buttons: "VIEW" (blue), "UPDATE" (orange), and "DELETE" (red). In this view, the "UPDATE" and "DELETE" buttons are disabled (grayed out) for all rows, while the "VIEW" button remains active and functional.

ID	Customer ID	Customer Last Name	Record Format	Genre	Actions
1	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
2	No data found!	No data found!	CD	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
3	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
4	No data found!	No data found!	CD	Reggae	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
5	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
6	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>

Administrator

The screenshot shows a web application titled "Physical Records Registration". At the top, there are navigation links for "Logout", "Records", and "Add records". A message indicates the user is "Logged in as Alex Admin (admin@recordshop.com)". Below the header, the title "Physical Records List" is displayed. A green button bar contains three buttons: "+ CREATE", "EXPORT TO EXCEL", and "EXPORT TO PDF". The main content is a table with the following columns: ID, Customer ID, Customer Last Name, Record Format, Genre, and Actions. The table contains six rows, each representing a record with the following data:

ID	Customer ID	Customer Last Name	Record Format	Genre	Actions
1	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
2	No data found!	No data found!	CD	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
3	No data found!	No data found!	Vinyl	Rock	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
4	No data found!	No data found!	CD	Reggae	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
5	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>
6	No data found!	No data found!	Vinyl	Alternative	<button>VIEW</button> <button>UPDATE</button> <button>DELETE</button>

KU4.1 – Enabling routing

```
frontend > physical_records_sys > src > app > ts app.config.ts > ...
  "Mandy, 2 weeks ago | 2 authors (You and one other)
1 import { ApplicationConfig, provideBrowserGlobalErrorListeners } from '@angular/core';
2 import { provideRouter } from '@angular/router';
3
4 import { routes } from './app.routes';
5 import { provideHttpClient } from '@angular/common/http';
6
7 export const appConfig: ApplicationConfig = {
8   providers: [
9     provideBrowserGlobalErrorListeners(),
10    provideRouter(routes),
11    provideHttpClient()
12  ]
13};
```

```

1 import { Routes } from '@angular/router';
2 import { RecordsListComponent } from '../components/records-list-component/records-list-component';
3 import { LoginComponent } from '../components/login-component/login-component';
4 import { RecordsAddNewComponent } from '../components/records-add-new-component/records-add-new-component';
5 import { roleGuard } from '../guards/role-guard';
6 import { RecordsEditComponent } from '../components/records-edit-component/records-edit-component';
7 import { RecordIndividualViewComponent } from '../components/record-individual-view-component/record-individual-view-component';
8 import { NotFoundComponent } from '../components/not-found-component/not-found-component';
9
10 export const routes: Routes = [
11   { path: '', redirectTo: 'records', pathMatch: 'full' },
12   { path: 'login', component: LoginComponent },
13   [
14     { path: 'records', You, 23 hours ago • Only authenticated users can view records,
15      component: RecordsListComponent,
16      canActivate: [roleGuard],
17      data: { roles: ['clerk', 'manager', 'admin'] }
18    ],
19    [
20      { path: 'records/view/:id',
21        component: RecordIndividualViewComponent,
22        canActivate: [roleGuard],
23        data: { roles: ['clerk', 'manager', 'admin'] }
24      },
25      [
26        { path: 'records/add-new',
27          component: RecordsAddNewComponent,
28          canActivate: [roleGuard],
29          data: { roles: ['clerk', 'manager', 'admin'] }
30        },
31        [
32          { path: 'records/edit/:id',
33            component: RecordsEditComponent,
34            canActivate: [roleGuard],
35            data: { roles: ['manager', 'admin'] }
36          },
37          { path: 'error-404-not-found', component: NotFoundComponent },
38          { path: '**', component: NotFoundComponent, pathMatch: 'full' }
39        ]
40      ]
41    ]
42  ];

```

KU4.1 - Use routing to control which component is rendered in the content area of the main component

```

frontend > physical_records_sys > src > app > app.html > div.container.mt-3 > router-outlet
You, last week | 2 authors (You and one other) | Go to component
1 <header-component></header-component>
2 <div class="container mt-3">
3   <router-outlet></router-outlet> You, last week • Working on the view component,
4 </div>

```

AA3.2 - Built-in lowercase pipe

```
frontend > physical_records_sys > src > components > record-individual-view-component > record-individual-view-component.ts > RecordIndividualViewComponent
  You, 11 hours ago | 1 author (You)
1 import { StocksService } from '../../../../../services/stocks-service';
2 import { AuthenticationService } from '../../../../../services/authentication-service';
3 import { PhysicalRecordDto } from '../../../../../dtos/physical-records-dto';
4 import { Component, OnInit, signal, WritableSignal } from '@angular/core';
5 import { PhysicalRecordsApiService } from '../../../../../services/physical-records-api-service';
6 import { CurrencyPipe, LowerCasePipe, NgClass } from '@angular/common';
7 import { ActivatedRoute, Router, RouterLink, RouterModule } from '@angular/router';
8 import { StockStatusPipe } from "../../pipes/stock-status-pipe";
9 import { PhysicalRecordsFrontendService } from '../../../../../services/physical-records-frontend-service';
10 import { CurrentRouteService } from '../../../../../services/current-route-service';
11
12 You, 11 hours ago | 1 author (You)
13 @Component({
14   standalone: true,
15   selector: 'record-individual-view-component',
16   imports: [RouterLink, RouterModule, CurrencyPipe, LowerCasePipe, StockStatusPipe, NgClass],
17   templateUrl: './record-individual-view-component.html',
18   styleUrls: ['./record-individual-view-component.css'],
19 })
20   You, last week • Prepared component to view individual records. ...
```

```
frontend > physical_records_sys > src > components > record-individual-view-component > record-individual-view-component.html >
  You, 5 hours ago | 1 author (You) | Go to component
1 <div class="mb-3">
2   <button class="btn btn-primary" type="button" routerLink="/records">
3     <i class="bi bi-arrow-left"></i>
4     Back to all records
5   </button>
6 </div>
7 <div class="card shadow-sm">
8   <div class="card-body">
9     <div class="mb-3">
10       <h1 class="card-title mb-0">
11         {{ this.recordById()?.title | lowercase }}
12       </h1>
```

AA3.2 – Custom pipe implementation

```
frontend > physical_records_sys > src > pipes > stock-status-pipe.ts > StockStatusPipe
  You, 6 days ago | 1 author (You)
1 import { StocksService } from '../../../../../services/stocks-service';
2 import { Pipe, PipeTransform } from '@angular/core';
3
4 You, 6 days ago | 1 author (You)
5 @Pipe({
6   name: 'stockStatus',
7 })
8 export class StockStatusPipe implements PipeTransform {
9   constructor(private stocksService: StocksService) {}
10
11   transform(stockQuantity: number | undefined): string | null {
12     if(stockQuantity === undefined) return null;
13     return this.stocksService.getStatusLabel(stockQuantity);
14 }
```

AA3.2 – Custom pipe usage in user interface

```
frontend > physical_records_sys > src > components > record-individual-view-component > record-individual-view-component.html > div.card.shadow-sm > div.card-body >
  31   <div class="col-md-3">
  32     <strong>Stock Quantity</strong>
  33     <p class="mb-0"><span [ngClass]="this.getCssClassBasedOnStock(this.recordById()?.stockQty)">{{ this.recordById()?.stockQty |
  34       stockStatus }}</span></p>

```

AA3.4 - Implement a data transfer object to model a registration

```
ts physical-records-dto.ts ×
frontend > physical_records_sys > src > dtos > ts physical-records-dto.ts > PhysicalRecordDto > customerContact
  You, last week | 2 authors (You and one other)
  1  export interface PhysicalRecordDto {
  2    id: number;
  3    title: string;
  4    artist: string;
  5    format: string;
  6    genre: string;
  7    releaseYear: number;
  8    price: number;
  9    stockQty: number;
 10    customerId: string;
 11    customerFirstName: string;
 12    customerLastName: string;
 13    customerContact: string;      You, last week • Renamed formControlName for each element accord...
 14    customerEmail: string;
 15  }

```

AA3.4 – Implement the necessary HTTP requests

```
frontend > physical_records_sys > src > services > ts physical-records-api-service.ts > PhysicalRecords ApiService > updatePhysicalRecord
  You, 23 hours ago | 3 authors ("Mandy and others")
  1  import { environment } from '../environments/environment';
  2  import { HttpClient } from '@angular/common/http';
  3  import { Injectable } from '@angular/core';
  4  import { Observable } from 'rxjs';
  5  import { PhysicalRecordDto } from '../dtos/physical-records-dto';
  6
  You, 23 hours ago | 3 authors ("Mandy and others")
  7  @Injectable({
  8    providedIn: 'root',
  9  })
 10  export class PhysicalRecords ApiService {
 11    private readonly endpoint: string = `${environment.apiUrl}/records`;
 12
 13    constructor(private http: HttpClient) {}
 14
 15    public getPhysicalRecords(): Observable<PhysicalRecordDto[]> {
 16      return this.http.get<PhysicalRecordDto[]>(this.endpoint);
 17    }
 18
 19    public getPhysicalRecordById(id: number): Observable<PhysicalRecordDto> {
 20      return this.http.get<PhysicalRecordDto>(`${this.endpoint}/${id}`);
 21    }
 22
 23    public createPhysicalRecord(record: PhysicalRecordDto): Observable<PhysicalRecordDto> {
 24      return this.http.post<PhysicalRecordDto>(this.endpoint, record);
 25    }
 26
 27    public updatePhysicalRecord(id: number, record: PhysicalRecordDto): Observable<PhysicalRecordDto> {
 28      return this.http.put<PhysicalRecordDto>(`${this.endpoint}/${id}`, record);
 29    }
 30
 31    public deletePhysicalRecord(id: number): Observable<void> {
 32      return this.http.delete<void>(`${this.endpoint}/${id}`);
 33    }
 34  }

```

AA4.2 - An ID card number shall consist of a numeric string with an alphabetical character at the end

Customer ID

ID Number (for example, 0473501L)

ID must start with 0, contain 7 digits, and end with A, G, H, M, or L.

Customer ID

473501L

ID must start with 0, contain 7 digits, and end with A, G, H, M, or L.

Customer ID

0473501

ID must start with 0, contain 7 digits, and end with A, G, H, M, or L.

Customer ID

0473501K

ID must start with 0, contain 7 digits, and end with A, G, H, M, or L.

AA4.2 - Contact number must be numeric, non-negative and at least 8 characters long

Customer Contact Number

Customer's phone number (for example, 99819288)

A valid contact number is required.

Customer Contact Number

-99819288

A valid contact number is required.

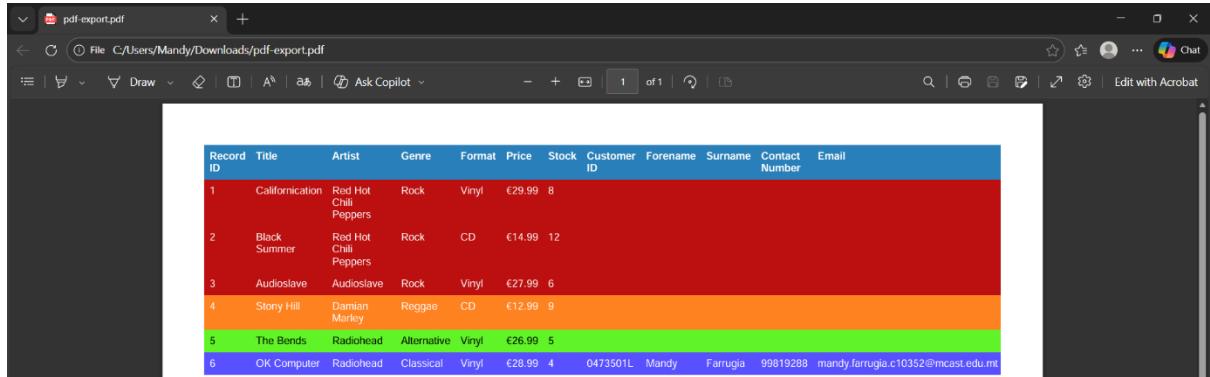
Customer Contact Number

9981928

A valid contact number is required.

SE2.1 – Exporting records

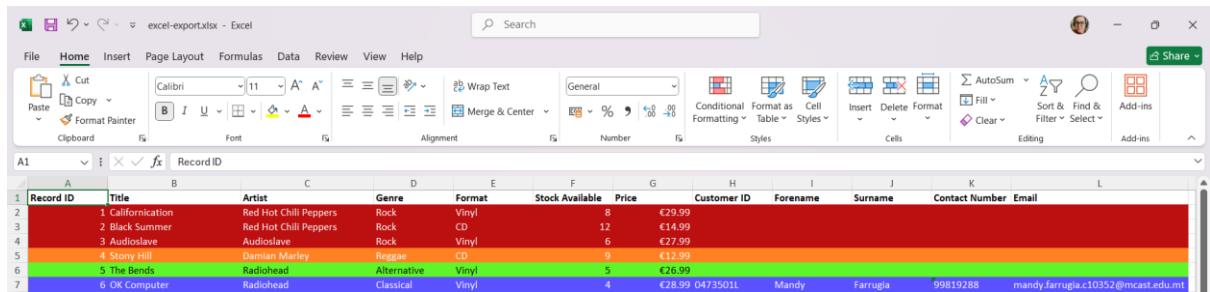
PDF



A screenshot of a PDF viewer window titled "pdf-export.pdf". The window shows a table of music records with the following data:

Record ID	Title	Artist	Genre	Format	Price	Stock	Customer ID	Forename	Surname	Contact Number	Email
1	Californication	Red Hot Chili Peppers	Rock	Vinyl	€29.99	8					
2	Black Summer	Red Hot Chili Peppers	Rock	CD	€14.99	12					
3	Audioslave	Audioslave	Rock	Vinyl	€27.99	6					
4	Story Hill	Damian Marley	Reggae	CD	€12.99	9					
5	The Bends	Radiohead	Alternative	Vinyl	€26.99	5					
6	OK Computer	Radiohead	Classical	Vinyl	€28.99	4	0473501L	Mandy	Farrugia	99819288	mandy.farrugia.c10352@mcast.edu.mt

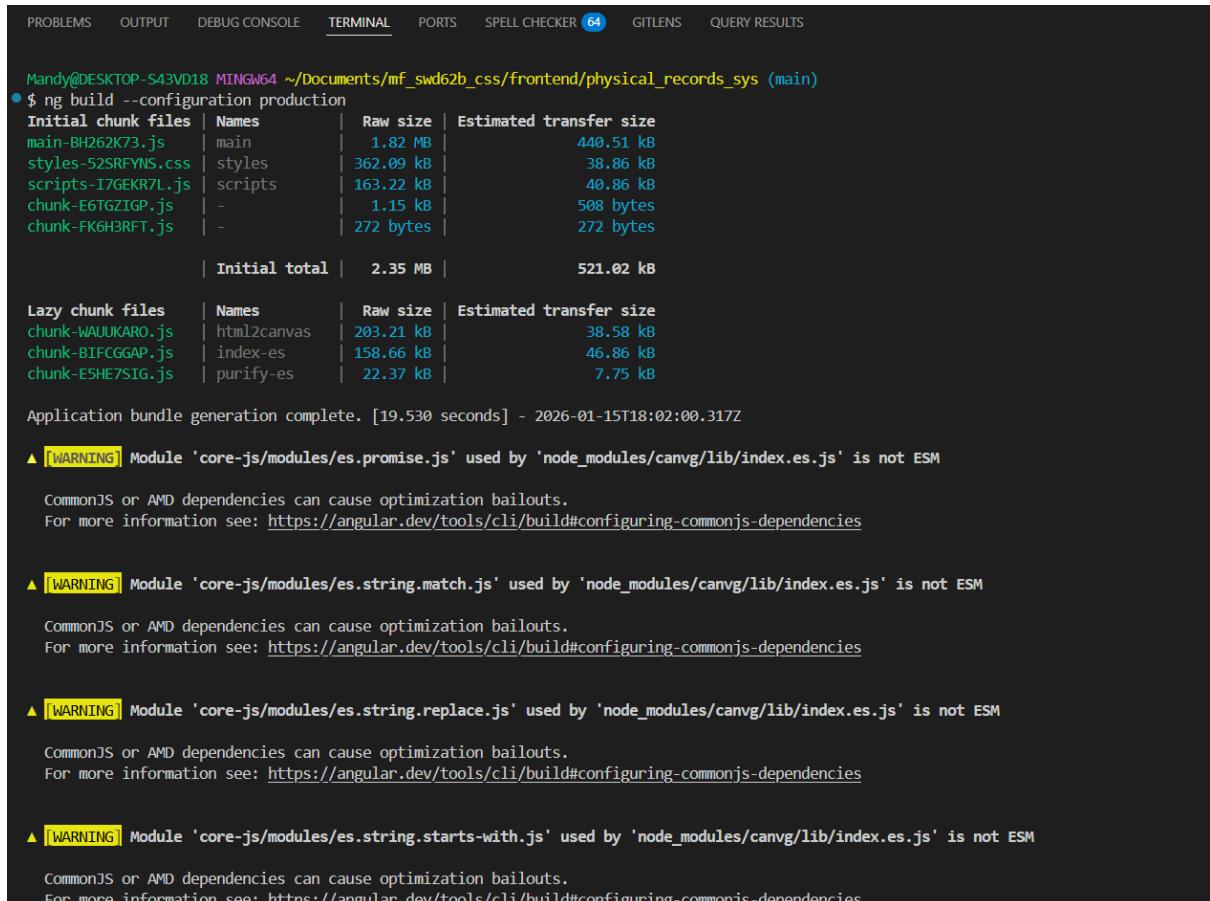
Excel



A screenshot of an Excel spreadsheet titled "excel-export.xlsx - Excel". The spreadsheet contains the same data as the PDF, with the following structure:

Record ID	Title	Artist	Genre	Format	Stock Available	Price	Customer ID	Forename	Surname	Contact Number	Email
1	Californication	Red Hot Chili Peppers	Rock	Vinyl	8	€29.99					
2	Black Summer	Red Hot Chili Peppers	Rock	CD	12	€14.99					
3	Audioslave	Audioslave	Rock	Vinyl	6	€27.99					
4	Story Hill	Damian Marley	Reggae	CD	9	€12.99					
5	The Bends	Radiohead	Alternative	Vinyl	5	€26.99					
6	OK Computer	Radiohead	Classical	Vinyl	4	€28.99	0473501L	Mandy	Farrugia	99819288	mandy.farrugia.c10352@mcast.edu.mt

KU4.3 - Build the app optimized for production



```
Mandy@DESKTOP-S43VD18 MINGW64 ~/Documents/mf_swd62b_css/frontend/physical_records_sys (main)
$ ng build --configuration production
Initial chunk files | Names | Raw size | Estimated transfer size
main-BH262K73.js | main | 1.82 MB | 440.51 kB
styles-52SRFYN.css | styles | 362.09 kB | 38.86 kB
scripts-I7GEKR7L.js | scripts | 163.22 kB | 40.86 kB
chunk-E6TGZIGP.js | - | 1.15 kB | 508 bytes
chunk-FK6H3RFT.js | - | 272 bytes | 272 bytes

| Initial total | 2.35 MB | 521.02 kB

Lazy chunk files | Names | Raw size | Estimated transfer size
chunk-WAUUKARO.js | html2canvas | 203.21 kB | 38.58 kB
chunk-BTFCGGAP.js | index-es | 158.66 kB | 46.86 kB
chunk-ESHE7SIG.js | purify-es | 22.37 kB | 7.75 kB

Application bundle generation complete. [19.530 seconds] - 2026-01-15T18:02:00.317Z

⚠ [WARNING] Module 'core-js/modules/es.promise.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: https://angular.dev/tools/cli/build#configuring-commonjs-dependencies

⚠ [WARNING] Module 'core-js/modules/es.string.match.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: https://angular.dev/tools/cli/build#configuring-commonjs-dependencies

⚠ [WARNING] Module 'core-js/modules/es.string.replace.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: https://angular.dev/tools/cli/build#configuring-commonjs-dependencies

⚠ [WARNING] Module 'core-js/modules/es.string.starts-with.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: https://angular.dev/tools/cli/build#configuring-commonjs-dependencies
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 64 GITLENS QUERY RESULTS

▲ [WARNING] Module 'core-js/modules/es.string.startsWith.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.array.iterator.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/web.dom-collections.iterator.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.array.reduce.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.string.endsWith.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.string.split.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'raf' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 64 GITLENS QUERY RESULTS

▲ [WARNING] Module 'core-js/modules/es.string.trim.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'rgbcolor' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.array.indexOf.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.string.includes.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.array.reverse.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'core-js/modules/es.regexp.toString.js' used by 'node_modules/canvg/lib/index.es.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'exceljs' used by 'src/components/records-list-component/records-list-component.ts' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER 64 GITLENS QUERY RESULTS

▲ [WARNING] Module 'file-saver' used by 'src/components/records-list-component/records-list-component.ts' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'html2canvas' used by 'node_modules/jspdf/dist/jspdf.es.min.js' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

▲ [WARNING] Module 'sweetalert2' used by 'src/services/alert-service.ts' is not ESM
CommonJS or AMD dependencies can cause optimization bailouts.
For more information see: <https://angular.dev/tools/cli/build#configuring-commonjs-dependencies>

Output location: C:\Users\Mandy\Documents\mf_swd62b_css\frontend\physical_records_sys\dist\physical_records_sys

KU4.3 – Use a free hosting service to host the Angular application

