

Assignment 1 - React JS App

Software Frameworks

Conor Walsh

20057989

Software Systems Development

Document detailing the development of a Single Page Application in React.JS on the context of my choice.

Table of Contents

Introduction	2
Features	2
Data Models	3
Video and CurrentOwner	3
User	3
Home Page	4
Edit Videos	5
Register New Users	6
View a Movie's Details	7
An About Page	8
Architecture	9
Extra Features	9
Independent Learning	9

Introduction

As part of the Software Frameworks module, I was asked to create a Single Page Application (SPA) using React.js. I decided to build an SPA for a fictional video rental store called ExtraVision. The SPA would be used by employees to store data on videos. The employees can also use the app to signup customer's names and passwords to the database.

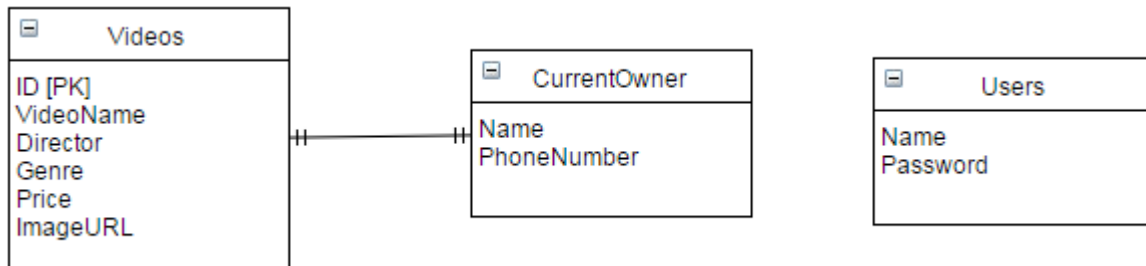
Features

- Video and User data is stored on a JSON Server.
- Users can add, update, and delete videos.
- Users can view and change the name and phone number of the person currently renting the video.
- Users can search for videos.
- Videos can be sorted alphabetically or by price.
- Customer's names and passwords can be added to the JSON server through the app. This could be implemented as a login system in future development.
- All forms use database and HTML5 validation.
- The app has a responsive design thanks to the use of Twitter Bootstrap.
- Parameterised URLs has been implemented for the ability to send and open links.

Data Models

1. Video – this stores data on the videos – name, genre, director, imageURL.
2. Current Owner – this is a nested collection inside of Video.
3. User – this stores data on the video - username and password.

In a relational view, the data model would be represented as follows:



Video and CurrentOwner

```
{
  id: 1,
  videoName: "Jaws",
  director: "Steven Spielberg",
  genre: "Thriller",
  price: "3",
  imageURL: "http://www.filmposters.com/jaws.jpg",
  currentOwner: {
    name: "Conor Walsh",
    phone: "0987654321"
  }
}
```

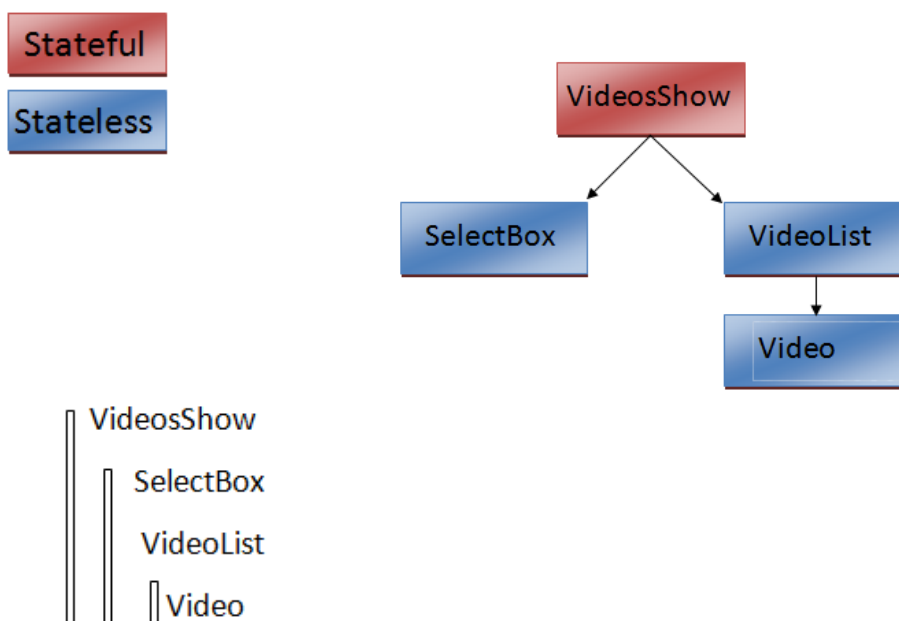
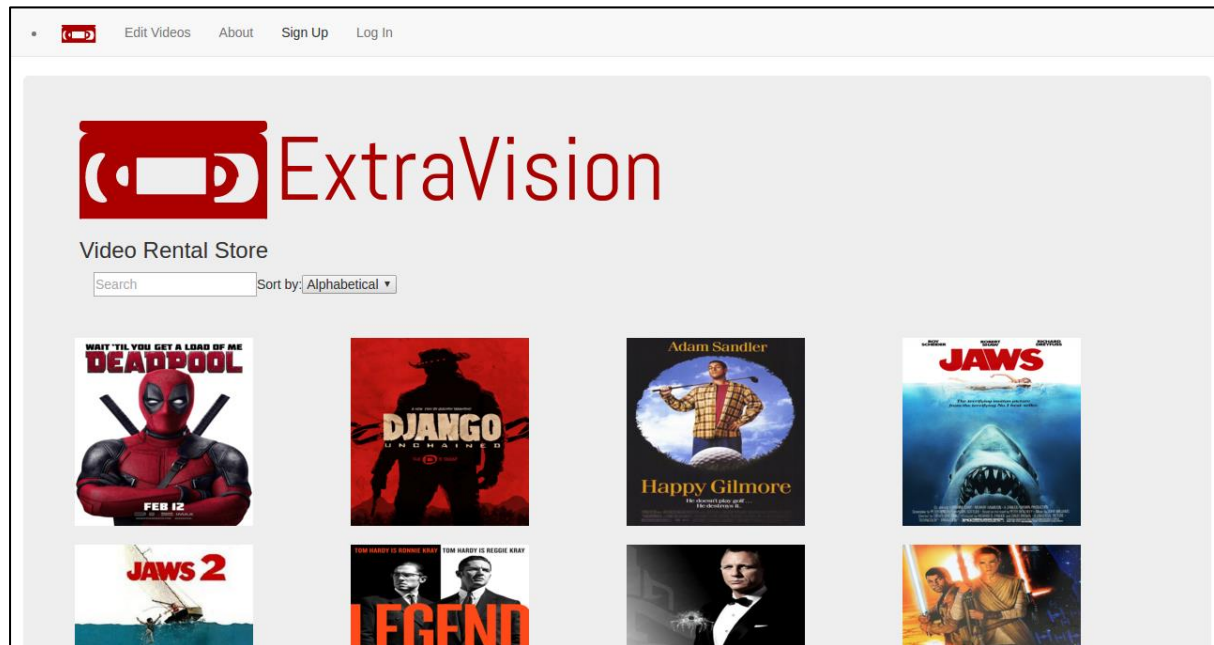
User

```
{
  name: "admin",
  password: "password123"
}
```

Home Page

When a user opens the SPA, they will initially see a grid containing all the videos. They can search for a video through the search box or sort the videos alphabetically, or by price, using the drop down menu.

This URL for this page is the domain name followed by '/' or '/videos/' .









VideosShow state {search:'', sort:'videoName', videos[]}

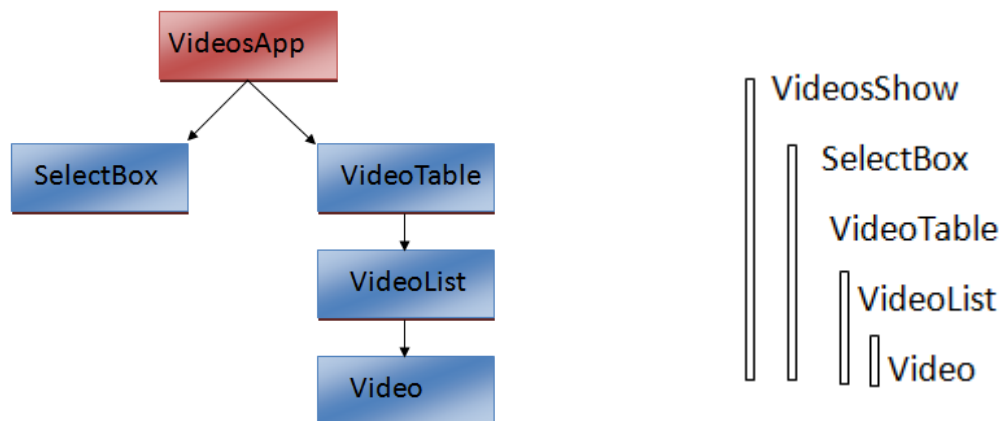
Edit Videos

When the user clicks on the 'Edit Videos' tab, they are brought to the edit page. It can also be accessed by using the URL '/edit/'. On this page the user is presented with a table of videos. From here the user can search for a video, or sort videos alphabetically or by price. Beside each video are two options; Edit and Delete. At the bottom of the table is a form, where users can add more videos.

[Edit Videos](#)
[About](#)
[Sign Up](#)
[Log In](#)

Sort by: Alphabetical

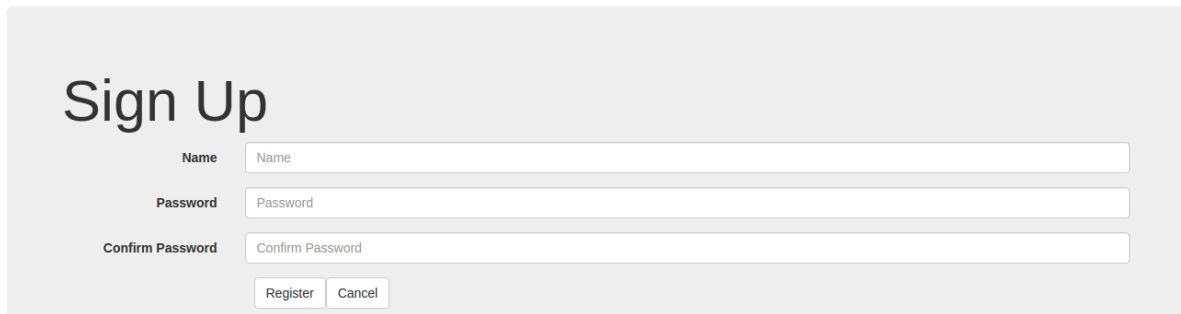
Image	Title	Director	Genre	Price €		
	Deadpool	Tim Miller	Action	€5	Edit	Delete
	Django	Quinten Tarantino	Action	€1.5	Edit	Delete
	Happy Gilmore	Dennis Dugan	Comedy	€3	Edit	Delete
	Jaws	Steven Spielberg	Thriller	€2	Edit	Delete
	Jaws 2	Steven Spielberg	Thriller	€4	Edit	Delete
	Legend	Brian Helgeland	Crime	€6	Edit	Delete



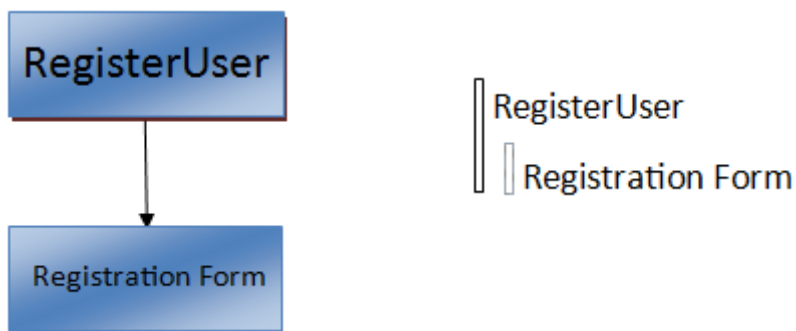
VideosApp state {search:'', sort:'videoName', videos[]}

Register New Users

This page allows the user to save username and password information to a database. There is also validation to check if all fields are completed, to check if the passwords match, and to check if the username has already been taken.



A 'Sign Up' form with a light gray background. The title 'Sign Up' is in large black font. Below it are three input fields: 'Name', 'Password', and 'Confirm Password', each with a label to its left. At the bottom are two buttons: 'Register' and 'Cancel'.

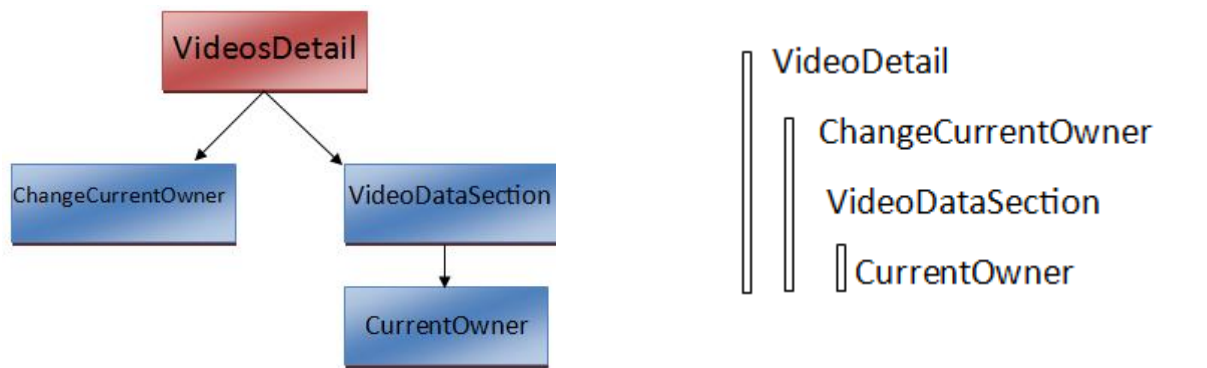
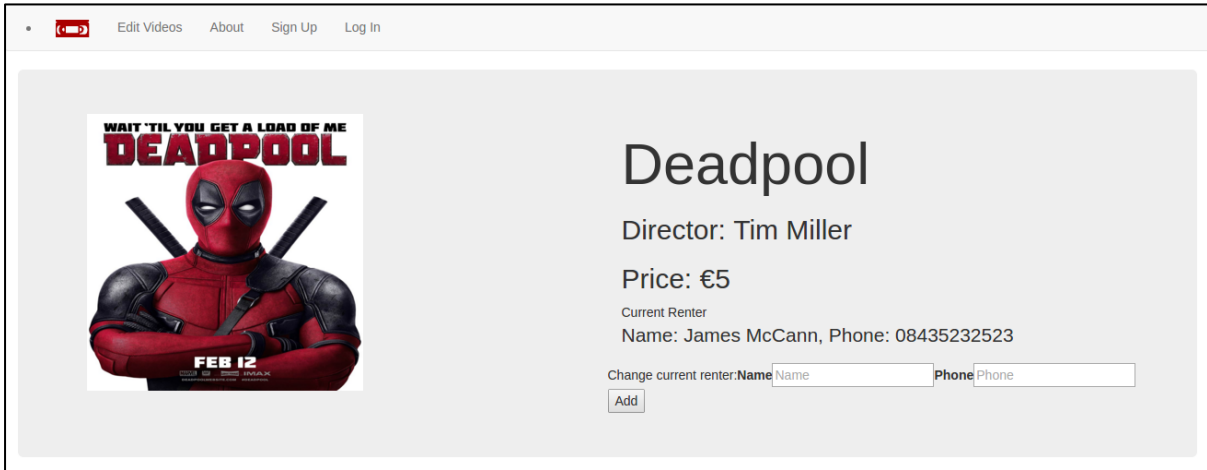


View a Movie's Details

A user can view details about a specific movie by clicking on it in the home page or the edit page, or by using a parameterised URL ('/videos/'+video_id).

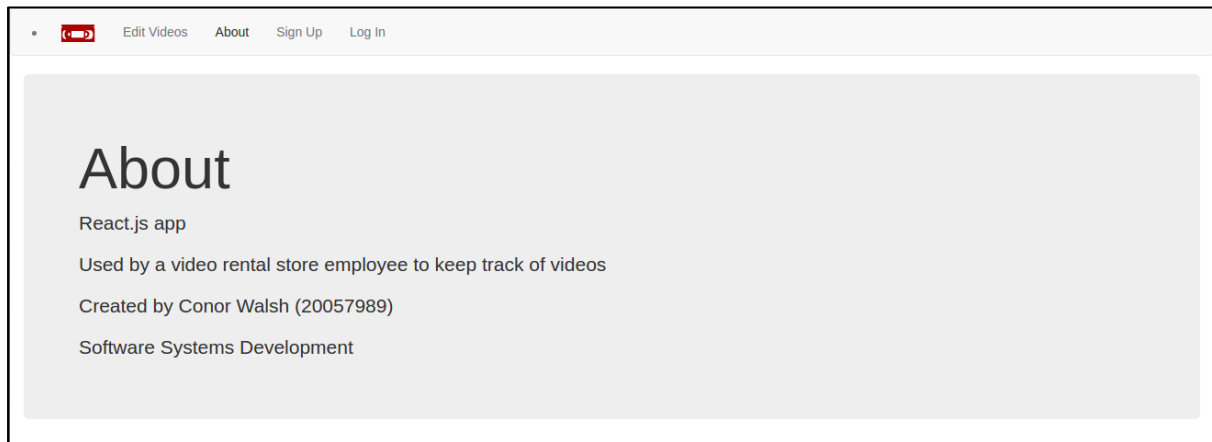
The user can also check who currently has the video. They can also change this using the form provided.

VideoDetail state { video[] }



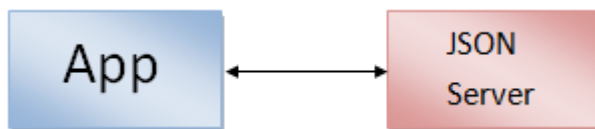
An About Page

There is also an about page which shows my name, course and student number, and a very brief description of what the application does.



Architecture

The app was run on an http server, which interacted with a JSON server, which stored and retrieved all the data.



Extra Features

- Ability to register new users
- Authentication to check if a video or user is already in the database

Independent Learning

As the project went on, I found myself doing a lot of independent learning.

At one point the project stopped entirely, even after a roll back, and turning my laptop on and off again. I discovered the error was to do with local storage and the react lifecycle. The error was being caused due to the videos in the local storage being overwritten by users. The app was then unable to find the videos in the local storage again once I returned to the videos page.

I also had to find out how to do 'PUT', 'POST' and 'DELETE' calls to a JSON server, as we had only covered 'GET' in the labs.

I had to learn about nested 'collections' or objects in the JSON server.