**Daniel Morrissey**

**X21118701**

**HDWD\_SEPOL**

**Project Report**

**Contents**

1. Introduction…………………………………………………………………………………………..2
2. Purpose………………………………………………………………………………………………….3
3. Design and Planning………………………………………………………………………………4
4. Homepage……………………………………………………………………………………………..7
5. Band page………………………………………………………………………………………………9
6. New bands page…………………………………………………………………………………..10
7. Test………………………………………………………………………………………………………11
8. External gems………………………………………………………………………………………12
9. Custom gem…………………………………………………………………………………………13
10. Implementation and Deployment…..…………………………………………………..14
11. Conclusion
12. Bibliography/references

**Introduction**

For this Cloud Application Development project, the task was to create a web application in which was to be deployed on a cloud-based platform. We are to pick a topic and then to choose a programming language and framework in order to create this web application that is to be deployed to a cloud-based platform.

Once the topic, the programming language and framework has been chosen, preparation is to be made in order to complete this project. The first step is to make at least one wireframe in order to design what the web application will look like. This can be done using online tools or applications like Figma as an example.

The next step is to design a database so as to have an optimal database with no redundant data. After the design of the database, the next step is to choose the language used to query the database via the use of object-oriented programming. Once that has been set up, choosing external libraries is to be next. To choose a library, a certain level of research is needed in order to use the external libraries that can be chosen. (Add something about creating a gem)

The type of tests that are to be used is also needed to be decided upon in order to ensure that the web application works as intended.

Once all these steps are completed the next step is to create the web application in question and to choose a suitable cloud platform such as Heroku or Amazon Web Services.

Graphical user interface, text, application

Description automatically generated

Screenshot of Heroku Website

**Purpose**

The purpose of this web application is to create a website that a person can go to and simply add a band or an artist to a database so that other people can look at bands and artists that other people have recommended to listen to. One advantage to this music recommendation site is that there is no need to register and only the band’s name or the artists name and the genre of music is needed.

This type of web application was chosen with the goal of creating a non-committal and simple website that someone can just enter a bands or artists details or check out the other bands and artists that people have recommended.

While it is understood that there are other websites that also recommend music such as Gnoosic (screenshot shown below), it felt like that there that some effort is involved in getting those music recommendations such as Gnoosic would ask a series of questions before you get a music recommendation of a band or an artist and it seems to be one at a time.

Graphical user interface, text, application

Description automatically generated

(Screenshot of Gnoosic homepage)

In order to create a music recommendation website, the Ruby programming language and the framework Ruby on Rails 7 were chosen to be the tools to create this web application.

**Design and Planning**

Before attempting to create this web application of a music recommendation site, a certain level of planning is needed. To plan for this web application, a use case diagram and entity relation diagram was created.

Diagram

Description automatically generated

(Use Case Diagram for web application, created using Visual Paradigm)

For the use case diagram above, a user (represented by Actor) would visit the site and then visit the Band list webpage and/or add a new band to the listing. Checking the list of bands and adding a new band are not always going to happen as the user may just visit the homepage.

In regard to the web application receiving and showing band and artists, the web application would need to interact with a database. Therefore, a database table needs to be designed to give the web application the ability to receive and show bands and artists.

To design a database table, the types of data required needs to be decided. In the end, it was decided to have five attributes and one of the attributes will be auto incremented. The attributes are: ID, Name, Country, Members and Genre. The ID and Members attributes require integer data while the rest are string data. The Name attribute accepts the name of the band or artist. The Country attribute accepts the country of origin of the band or artist. The Members attribute accepts the number of members in the band and the Genre attribute accepts the music genre.

When a user enters data, only the Name and Genre attributes are required, and the minimum number of members will be 1. The entity relationship diagram is on the next page.

Diagram

Description automatically generated

(Entity Relationship diagram of Band table)

After the database has been designed, the next step is to design a general layout of what the web application will look like. To design the general layout of the web application an application called Figma was used.

It was decided to create a wireframe to outline the general layout of each web page of the music recommendation web application to make a total of 3 web pages. The three web pages will be the homepage, the band and artist listings page and a web page that deals with the addition of new bands and artists.

The homepage is an introduction to the music recommendation web application that describes what the web application is and what it does. The band and artist listings page lists all the artists and bands that other users have recommended starting with the most recently added band or artist. The list of bands and artists show the name of the band or artist, the country of origin, the number of members and the music genre that the band or artist plays.

A picture containing diagram

Description automatically generated

(Image of the wireframe that shows the general layout of the web application, not to scale)

To carry out this cloud application development project, the Ruby on Rails framework was chosen in order to use the model-view-controller approach. Ruby on Rails itself was chosen as this is the framework being taught. The use of the model-view-controller approach was chosen as it comes with a number of advantages such as supporting test-driven development, allows multiple views with less duplication of code and organises a web application (https://www.geeksforgeeks.org/benefit-of-using-mvc/).

**Homepage**

The homepage would normally be the first page that a user would see when they visit the web application. The homepage of the web application for this cloud application development continuous assessment project is split into the navigation bar on the top of the page, a slide show of images of random objects from my own home. The reason random objects were chosen was to avoid any copyright issues.

A picture containing text, indoor, screenshot, different

Description automatically generated

(Screenshot of the navigation bar and the slideshow of the web application)

Just after the slideshow is a paragraph that instructs the user on how to use the website which has been previously discussed in this report.

Text

Description automatically generated

(Screenshot of web application introduction and quote)

Below the slideshow is a quote from *Edward G. Bulwer-Lytton* about how music stays with a person.

A picture containing text, different, screenshot, several

Description automatically generated

(Screenshot of the bootstrap cards of objects from my home and the footer)

The bottom third of the homepage is the bootstrap cards and the footer. The bootstrap cards contain images of objects from my home and labelled as a band at a concert.

Finally, below the bootstrap cards is the footer which contains links to Facebook, Twitter, Ticketmaster and Tickets.ie. The links only link to the homepage of each website. The aim is to give an impression of a music recommendation web application.

The navigation bar at the top of the web application contains the links to the homepage, the band and artist listing page and the form is the third link.

**Band Page**

The band page can be accessed via the navigation bar and would be the second link in said navigation bar. The Band page has the Bootstrap carousel and the Bootstrap cards like the other pages.

Table

Description automatically generated with medium confidence

(Screenshot of the main part of the Band page)

After the Bootstrap carousel, is a brief introduction of what this page is to the user followed by a Bootstrap table that draws data from the database. The data comes from users adding new bands and artist information. This will be discussed later. The band or artist name and the genre will not be blank and the number of members will default to 1. Country is optional so a blank space will be possible as shown in the screenshot above (it is important to note that the data shown in the screenshot is data inputted by me). After the Bootstrap table are the Bootstrap cards that are the same as the Homepage. This is then followed by the footer which has been discussed previously.

The Bootstrap table contains data that pertains to the band or artists name, the country of origin, number of members, music genre and a link that’ll lead to a new tab displaying a YouTube search based on the name of the band or artist.

**New Bands Page**

The New Bands page can be accessed via the navigation bar and would be the third link in said navigation bar. The New Band page has the Bootstrap carousel and the Bootstrap cards like the other pages. The link is written as “Add a New Band”.

Graphical user interface, text

Description automatically generated

(Screenshot of the main content of the New Bands page)

After the Bootstrap carousel is the instructions on how to fill out the form, followed by the actual form. There are four input fields, three text fields, one number field and a “Save” button. This form was created and linked to the database via the controller using the “simple\_form” gem. There is JavaScript and back-end code that validates this form so as to make sure that the band name and genre fields are filled in and that the minimum number of members defaults to one if any wrong or no data is entered in the members number field. However, despite the default value for the members field being one, the insertion of data with the number of members is optional.

**Tests**

For testing this Cloud Application Development continuous assessment project, the RSpec gem was installed to carry out unit testing on the model (database) to ensure that the model is working as it should be.

There are a total of seven tests created to test the functionality of the model. The first test tests the model by checking if that data can be inserted in the database without the country and members fields. The second test tests that data can be inserted in the database with all fields filled in.

The third test tests that the name field needs to be filled in. The fourth test tests that the genre field needs to be filled in and the fifth test tests that both the name and genre fields need to be filled in at the same time. The sixth and seventh tests test that data can be inserted with only the country field empty and only the members field empty respectfully.

Text

Description automatically generated

(Image of RSpec tests passing)

As shown in the image above, all seven tests pass therefore showing that the database is working as designed and that the validation of the data is working correctly.

**External Gems**

For this Cloud Application Development project, the use of external gems was required in order to aid in the creation of this project. A number of gems were picked in order to aid in this endeavour each serving a specific purpose.

For this project, the external gems that were selected are as follows:

1. CSSBundling - rails
2. Simple form
3. Meta-tags
4. RSpec
5. Ffi

CSSBundling – rails

[https://GitHub.com/rails/cssbundling-rails?fbclid=IwAR0or3K4CeWypWzEQZvqJoHgORMoVkarLao65\_ljgLeiZpVQ5\_M04mJGHhk](https://github.com/rails/cssbundling-rails?fbclid=IwAR0or3K4CeWypWzEQZvqJoHgORMoVkarLao65_ljgLeiZpVQ5_M04mJGHhk)

<https://www.theodinproject.com/lessons/ruby-on-rails-css-bundling?fbclid=IwAR0_BtaTjdr9EjxseO8yYmsLVUQfjWo9Bqm73bkYE1jGodeq-19qLfK48sY>

The CSSBundling – rails gem is an external gem that is used to bundle and process CSS and can be installed to include Bootstrap CSS. By installing this gem, I was then able to utilise Bootstrap on my project. However, there was a requirement to add the Bootstrap JavaScript separately which was done by using the CDN.

Bootstrap was used to style the navigation bar at the top of the web page, the Bootstrap Carousel (alongside the Bootstrap JavaScript CDN), the Bootstrap cards and the footer at the bottom of the web page.

This external gem was chosen as it was found to be easier to install into the Cloud Application Development project than the method that was taught which required further configuration to use.

Simple form

[https://GitHub.com/heartcombo/simple\_form?fbclid=IwAR3iwno8DWrieaUy5ehaPIbYDeFQabDxRhvbyyP-ishUtSlWcLMERSPvp50](https://github.com/heartcombo/simple_form?fbclid=IwAR3iwno8DWrieaUy5ehaPIbYDeFQabDxRhvbyyP-ishUtSlWcLMERSPvp50)

The Simple Form gem is an external gem that simplifies the use of creating forms using embedded ruby and linking to a database. All labels, styling, the input field and attributes for that input can be created using a single line of embedded ruby as long as the Simple Form gem is installed.

This external gem was chosen to simplify the creation of the form on the new bands web page that will be used to insert band and artist recommendations into the database.

Meta-tags

[https://GitHub.com/kpumuk/meta-tags?fbclid=IwAR3bqg6-hM-spcleiT6gpGZMOSK8gRrGOE04oR6lIXACKuOXMP15GigS\_0c](https://github.com/kpumuk/meta-tags?fbclid=IwAR3bqg6-hM-spcleiT6gpGZMOSK8gRrGOE04oR6lIXACKuOXMP15GigS_0c)

The Meta – tags gem is an external gem that adds meta elements into the head element of the webpage via the controller. This gem was chosen to provide an faster way of adding meta data into the web page to help optimise the SEO.

FFI and RSpec

[https://GitHub.com/rspec/rspec-rails?fbclid=IwAR2MBB-SoJ0dOfSICVNFisCR\_wX5f82ClZn6f-Bb75syN8f6Lx0UjZrSAyM](https://github.com/rspec/rspec-rails?fbclid=IwAR2MBB-SoJ0dOfSICVNFisCR_wX5f82ClZn6f-Bb75syN8f6Lx0UjZrSAyM)

The FFI and RSpec external gems are gems that were used for testing Ruby on Rails projects. The FFI gem was required in order to carry out any testing on Ruby on Rails in Windows. The RSpec gem was used to carry out behaviour driven unit tests that were used to test the form that was created using the Simple Form gem on the new bands web page.

**Custom Gem**

In order to complete this Cloud Application Development project, the creation of a custom gem was required. For this project, it was decided to create a gem that can provide a Bootstrap Carousel template on the command line.

Text

Description automatically generated

(Image of the Bootstrap Carousel template)

Once the gem is installed on the local machine, one would then start the ruby shell in the command prompt and then enter “require ‘carousel’” to use the Bootstrap Carousel template gem. Then to print out the Bootstrap Carousel template enter “Carousel.template”. This would then print out the Bootstrap Carousel template which the user to copy and paste the template into their own Ruby on Rails project.

The only requirement for this Bootstrap Carousel template gem to work is to have Bootstrap already installed with the Bootstrap JavaScript. This custom gem includes instructions on the alterations one would need to make in order to make the Bootstrap Carousel template to work.

This Bootstrap Carousel template gem is useful as once installed, it makes it easier for the user to add the Bootstrap Carousel to their own Ruby on Rails project and know exactly what to add or change to have a functioning Bootstrap Carousel. As it is accessed on the ruby shell in the command line, this template is not restricted to the use of Ruby on Rails. As long as the custom gem is installed on the local machine, the template can be used on any other project such as Laravel.

**Implementation and Deployment**

GitHub repository: [https://GitHub.com/DanielJMorrissey/x21118701\_cloud\_project](https://github.com/DanielJMorrissey/x21118701_cloud_project)

Gem GitHub repository: [https://GitHub.com/DanielJMorrissey/cloudgem](https://github.com/DanielJMorrissey/cloudgem)

Deployed Web Application homepage: <https://x21118701.herokuapp.com/>

To implement the delivery of this cloud application development project the use of GitHub, git bash and Heroku were used. Git bash and GitHub were used for the integration of code from the web application and the custom gem that was created (links to the repositories above).

Git bash was used to create a level of version control by integrating the code on the local machine and ultimately the corresponding GitHub repositories. As Git bash provides a shell and installs Git commands, Git bash is a very useful tool in maintaining version control. Git bash does this by speeding up the integration of code and allows for complex file sharing between the local machine and the GitHub repository.

Graphical user interface, text, application, email, website

Description automatically generated

(Image of GitHub repository for web application)

A screenshot of a computer

Description automatically generated

(Image of GitHub repository of custom gem)

The two images above show the GitHub repositories of both the web application and the custom gem. The most recent commit shows the latest changes and gives an indication of the current version the web application is currently at.

A screenshot of a computer

Description automatically generated with medium confidence

(Image showing past commits of web application GitHub repository)

The above image shows some of the past commits from using Git bash for the web application, showing the most recent commits at the time the image was taken. The commit messages that are shown in the image are showing the attempts at fully deploying the web application to Heroku which ultimately succeeded at the second last commit.

For deploying the web application, the chosen cloud platform was Heroku. The reason for why Heroku was chosen as the cloud platform to host the web application was that the service was free, ease of deployment and provides security to the web application.

A video game screen capture

Description automatically generated with low confidence

(Image of web application on Heroku)

The image above shows a portion of the web application that was deployed on Heroku (Will be shown fully in the video presentation). Heroku provides instructions on how to deploy a web application to its cloud platform providing ease of deployment compared to using AWS which is possible and available to use for this project.

Furthermore, Heroku provides security by using HTTPs without having to configure the certificate and private keys and any necessary coding to utilise HTTPs. The only downside to using Heroku is that the database query language used needs to be converted from SQLite3 to PostgreSQL. To carry out this conversion from SQLite3 to PostgreSQL, the pg and rails\_12factor gems need to be installed and the SQLite3 gem to be commented out as shown in the image below.

Text, application

Description automatically generated

(Image of gemfile with the pg and rails\_12factor gems)

The only issue when trying to upload the web application to Heroku was caused by the database.yml file (shown below) which required reconfiguring to adapt to PostgreSQL. This was found to be the issue after some troubleshooting and research on stackoverflow.

Text

Description automatically generated

(Image of database.yml file after change to PostgreSQL)