

Title of project

**Juan Navio Nunez**

**N00200925**

Software Project

Ticket selling WebPage

Year 2 2021-22

DL836 BSc (Hons) in Creative Computing

Table of Contents

[1 Introduction 1](#_Toc96009471)

[2 Business Concept 2](#_Toc96009472)

[2.1 Business Idea 2](#_Toc96009473)

[2.2 Business model 2](#_Toc96009474)

[2.3 Market Research 2](#_Toc96009475)

[2.4 Marketing/Advertising 2](#_Toc96009476)

[2.5 Suppliers 2](#_Toc96009477)

[2.6 Competitors 2](#_Toc96009478)

[2.7 Employees 2](#_Toc96009479)

[2.8 Environmental Impact 2](#_Toc96009480)

[3 Requirements 3](#_Toc96009481)

[3.1 Introduction 3](#_Toc96009482)

[3.2 Requirements gathering 3](#_Toc96009483)

[3.2.1 Similar applications 3](#_Toc96009484)

[3.2.2 Interviews 3](#_Toc96009485)

[3.3 Requirements modelling 3](#_Toc96009486)

[3.3.1 Functional requirements 3](#_Toc96009487)

[3.3.2 Non-functional requirements 3](#_Toc96009488)

[3.3.3 Use Case Diagrams 4](#_Toc96009489)

[3.4 Feasibility 4](#_Toc96009490)

[4 Web application Design 6](#_Toc96009491)

[4.1 Layout 6](#_Toc96009492)

[4.2 Interaction 6](#_Toc96009493)

[4.3 Colour schemes 6](#_Toc96009494)

[4.4 Font choices 6](#_Toc96009495)

[4.5 Wireframes 6](#_Toc96009496)

[5 Database Design 7](#_Toc96009497)

[5.1 Description 7](#_Toc96009498)

[5.2 Business Reporting Requirements 7](#_Toc96009499)

[5.3 Textual Representation of Data-Set 7](#_Toc96009500)

[5.4 Business Rules 8](#_Toc96009501)

[5.5 Entity Relationship Diagram 8](#_Toc96009502)

[5.6 Tables 9](#_Toc96009503)

[5.7 Database Dictionary 10](#_Toc96009504)

[6 System Design/ Architecture Overview 11](#_Toc96009505)

[6.1 Introduction 11](#_Toc96009506)

[6.2 Model View Controller 11](#_Toc96009507)

[6.3 User Authenticaion 11](#_Toc96009508)

[6.4 Routing 11](#_Toc96009509)

[6.5 Templating 11](#_Toc96009510)

[7 Testing 12](#_Toc96009511)

[7.1 Introduction 12](#_Toc96009512)

[7.2 Functional Testing 12](#_Toc96009513)

[7.2.1 Login/Registration 13](#_Toc96009514)

[7.2.2 Navigation 13](#_Toc96009515)

[7.2.3 Calculation 13](#_Toc96009516)

[7.2.4 CRUD 13](#_Toc96009517)

[7.2.5 Discussion of Functional Testing Results 14](#_Toc96009518)

[7.3 User Testing 14](#_Toc96009519)

[7.4 Conclusion 14](#_Toc96009520)

[8 Project Management 15](#_Toc96009521)

[8.1 Introduction 15](#_Toc96009522)

[8.2 Project Phases 15](#_Toc96009523)

[8.2.1 Requirements 15](#_Toc96009524)

[8.2.2 Design 15](#_Toc96009525)

[8.2.3 Implementation 15](#_Toc96009526)

[8.2.4 Testing 15](#_Toc96009527)

[8.3 SCRUM Methodology 15](#_Toc96009528)

[8.4 Project Management Tools 16](#_Toc96009529)

[8.4.1 Github Project 16](#_Toc96009530)

[8.4.2 GitHub 16](#_Toc96009531)

[9 Reflection 17](#_Toc96009532)

[9.1 Your views on the project 17](#_Toc96009533)

[9.2 How could the project could be developed further? 17](#_Toc96009534)

[9.3 Assessment of your learning. 17](#_Toc96009535)

[9.4 Completing a large software development project 17](#_Toc96009536)

[9.5 Technical skills 17](#_Toc96009537)

[9.6 Further competencies and skills 17](#_Toc96009538)

[10 References 18](#_Toc96009539)

# Introduction

The aim of this project is to create a web application which will help me to develop my skills using different languages as PHP, MySQL, Bootstrap, CSS, Sass, html and js.

The original idea is a concert/festival ticket selling webpage, where all the discographic business or independent artist can sell their tickets

# Business Concept

## Business Idea

The original idea is a concert/festival ticket selling webpage, where all the discographic business or independent artist can sell their tickets I’m aiming to have a web application with dynamism and easy to use, a site where you can go in and have a fast buy without too many hesitations.

## Business model

My business will be able to make the ticket selling easy and fast for the sellers and for the users.

The business will take a small percentage of the tickets sold, plus different incomes from advertising, etc…

## Weakness / Threats

The competitors are very weak and not trying enough to make this business bigger than already is, so the only thing I could say is a treat is not being famous enough between the existing business, luckily, I think it won’t take long to change this.

## Market Research

My business marketing would be basically all online, having advertising in other platforms such as YouTube or Twitch where I can find targets in the age range I’m looking for, younger generations.

## Competitors

The main competitors would be the other selling web pages such as TicketMaster or StubHub.

# Requirements

### Functional requirements

The sellers would be able to “post their tickets” and the buyers would be able to see them online so they are able to make a buy fast and save.

### User requirements

User will be able to buy the tickets or in the case they are the sellers they can sell their tickets for their own events.

### Use Case Diagrams

A picture containing text, map, indoor

Description automatically generated

## Feasibility

For my feasibility I will use: GitHub, Figma, Bootstrap and Laravel.

# Web application Design

## Layout

For the layout I used Figma, using basic rectangles to make my page minimalist and clean.

## Interaction

The webpage will have interaction for selling the tickets and for buying them.

## Colour schemes

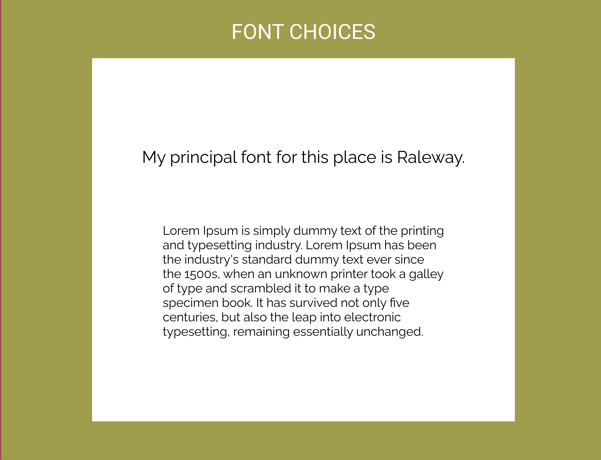
For my color schemes I used 5 colours.

A picture containing graphical user interface

Description automatically generated

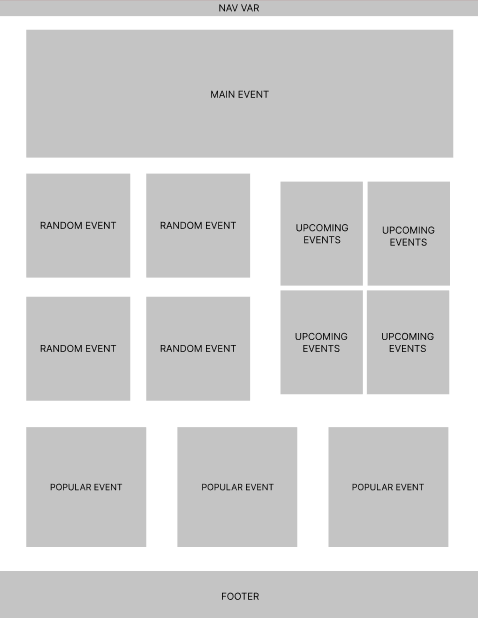
## Font choices

For my font choices I will use Raleway, which I found attractive for my page.

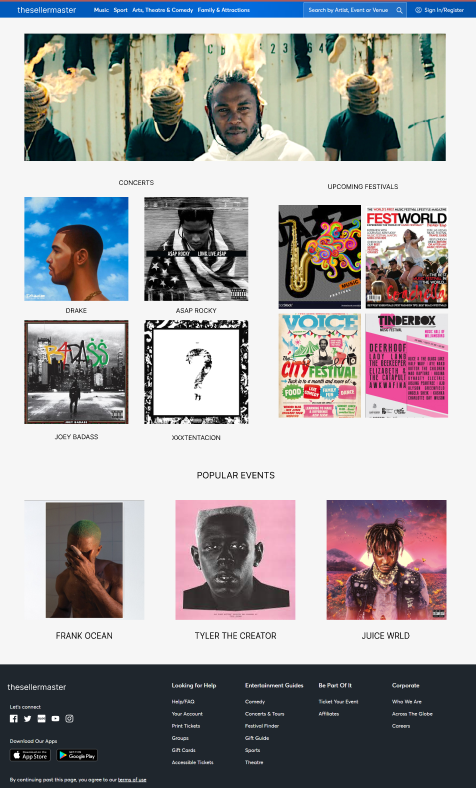


## Wireframes

Homepage mid-fidelity

****

**Homepage high-fidelity**

****

# Database Design

## Description

My application will need a database to store users posts, comments, user information and advertisements. Users will be able to create new posts. Those posts will have a title body name and date. A user can then add a comment underneath the post. The comment will have a title and body. Users will be able to follow a category and add a post to a category.

## Textual Representation of Data-Set

**category** (id, category\_name)

**comments** (id, title, body, user\_id, post\_id)

**posts** (id, title, description, name, body, category\_id)

**users** (id, name, email, password)

**user\_post** (user\_id, post\_id)

**user\_advertisement** (id, user\_id, advertisement\_id)

**advertisement** (id, title, description, body, business\_name)

**roles**  (id, name, description)

## Entity Relationship Diagram

A picture containing diagram

Description automatically generated

## Tables

*Chart, box and whisker chart

Description automatically generated*

## Database Dictionary

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| user | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 100 |  | PK |  |
| Name | unsigned | varchar | 255 |  |  |  |
| email | unsigned | varchar | 255 |  |  |  |
| password | unsigned | varchar | 255 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User\_post | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 100 |  | PK |  |
| User\_id | unsigned | Int | 100 |  | FK | Post |
| Post\_id | unsigned | int | 100 |  | FK | post |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| post | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 0-100 |  | PK |  |
| Title | unsigned | varchar | 2555 |  |  |  |
| body | unsigned | varchar | 255 |  |  |  |
| description | unsigned | varchar | 255 |  |  |  |
| Category\_id | unsigned | int | 255 |  | FK | category |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| comment | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 0-100 |  | PK |  |
| Title | unsigned | varchar | 255 |  |  |  |
| body | unsigned | varchar | 255 |  |  |  |
| User\_id | unsigned | int | 100 |  | FK | Post |
| post\_id | unsigned | int | 100 |  | FK | post |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| category | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 100 |  | PK |  |
| Category\_name | unsigned | varchar | 250 |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User\_advertisement | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 100 |  | PK |  |
| User\_id | unsigned | Int | 100 |  | FK | User |
| Advertisement\_id | unsigned | int | 100 |  | FK | use |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Advertisement | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 0-100 |  | PK |  |
| Title | unsigned | varchar | 250 |  |  |  |
| description | unsigned | varchar | 250 |  |  |  |
| body | unsigned | varchar | 250 |  |  |  |
| Business\_name | unsigned | varchar | 250 |  | FK | category |
| User\_role | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 100 |  | PK |  |
| User\_id | unsigned | int | 100 |  | fk | user |
| Role\_id | unsigned | Int | 100 |  | Fk | roles |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Roles | Attribute | Datatype | Range | Required | PK/FK | FK Ref Table |
| id | unsigned | int | 100 |  | PK |  |
| name | unsigned | varchar | 250 |  |  |  |
| description | unsigned | varchar | 250 |  |  |  |

# System Design/ Architecture Overview

* 1. Introduction

This section will describe the internal functionality of the web framework that you have chosen for the implementation. Add further sections if required by the specification of your web application

* 1. Model View Controller

A user requests a page from the server. The controller handles the request. The controller will check the model for the data. The model is responsible for interacting with the database and will validate the data requested by the controller. Once the controller has the data it will display it in the view. The view is responsible for the Ui of the website. The controller interacts with the model and the view, however the view and model never interact.

* 1. User Authentication

When I setup my Laravel project I created 2 roles, Admin and User. Users can read and write to the app, and admins have full crud functionality. Each user is given a user role which is stored in my database in the ‘user\_role’ table. The User Seeder acts as a template for the user. The user model will check the user’s role and see if they are authorized to do an action.

When a user registers an account, they are automatically assigned the regular user role. Users are given a user Id, which is linked to a role id in the role table. A user with a role id 1 will have admin permissions and a user with an Id of 2 will have only user permissions.

When a user goes to a page, the pages controller and middleware will check if the user is authorized to visit the page.

* 1. Routing

If a user logs into the webapp they will be directed to the ‘user/home’ page. This is done with a route. When the user clicks on the link to go to the home page, the route in web.php gets the ‘user/home’ page, it then goes to the home controller and calls the index function. The index function then returns the users correct view.

* 1. CRUD Functionality

For my project, regular users will only have create and read functionality. Admins will have full crud functionality. Users will be allowed to visit a create page with a forum where they can enter data to the database. They can also view their posts all on one page, or view each post by Id.

Admins will be able to delete posts from the database or update them. They will have an update page with a forum that they can enter data to. The admins page controller will call the relevant edit, store, update and destroy functions

* 1. Edit Functionality

Once the edit button is clicked on the index page, the user then gets routed to the Edit route in Web.php. Then routes you to the Edit function In the Controller. Edit Function Loads edit.blade.php. The customer id gets passed through the function. The View and Form then gets populated with the customer's Id data. The submit button goes to the update route. Then goes to Web.php to find out where to go. The update route goes to the update Function in the controller. The update function validates the data and saves it to the database. Once saved, index.php is then loaded again.

* 1. Templating

Chart, box and whisker chart

Description automatically generated

# Testing

* 1. Introduction

My web application will undergo functional testing. I will be testing various parts of the website such as Crud functionality, user login and registration, and navigation. These are all functional requirements for my website. It is important to test that they all work as expected so my web app can function correctly. For this chapter I am doing functional testing and user testing.

* 1. Functional Testing

I will be testing the login, registration, crud, and navigation functionality. I will be recording my test results using a Black Box. I will log the test number, input, expected output and actual output. It is important to check that the output is what you expected. If its not, you must record it and debug the problem.

* + 1. Login/Registration

To test Login functionality, I first checked to see if a user can register an account. This will require an email and password which will be stored in the database. The register page must be loaded and then the data entered. Once submitted I check the database to see if that user was added to the records.

Then to test login functionality, I use the same credentials and login. The data for logging in and registering is passed through the Login Controller and Register Controller.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No | Description of test case | Input | Expected Output | Actual Output | Comment |
| 1 | Testing if a user can register. | User enters an email and password, submits the data. | An account is created in the database | An account is created in the database |  |
| 2 | Testing if a user can login. | A user logs in with the registered email and password | The user can login | The user can login |  |

* + 1. Navigation

Navigation is important to check that the flow of the web app is as intended. For my WebApp I want to test can a user get from the home page to creating a post and view it. This will require routes for the user and the correct pages to be loaded.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No | Description of test case | Input | Expected Output | Actual Output | Comment |
| 1 | The user can navigate from the home page to the create page | A user will click on the create button and the page is loaded | The user can navigate to the correct page | The user can navigate to the correct page |  |

* + 1. CRUD

For the Crud functionality testing I tested my create function first. I went to the web app’s create page and added data to the database through a forum. I then submitted the data and checked the database to see if the correct data had been added and a post was created. When the data is being submitted it is checked in the store function of my controlled. It uses a validate function to make sure the data that the user is submitting is valid.  
Text

Description automatically generated

For testing the read functionality, I was able to load my webapp and go to the post I submitted to the webapp and find its Id. I was then able to load that data by id to a web page.   
To check if update functionality was working, I went to the update page and edited the data with a forum and resubmitted it to the database. It then also uses a require function to check if the data is valid. I then check in the database if the data has changed correctly and then check in the webapp if it is displayed correctly.  
I can then check to see if delete functionality is working by deleting the data, I added earlier by using the web app to call the controllers destroy function and passing it the correct id. I can then check in the webapp and database to see if the data has been deleted.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No | Description of test case | Input | Expected Output | Actual Output | Comment |
| 1 | Create function works | The user can create a post | A post is created | A post is created |  |
| 2 | The user can see the post they created | The user views the post by Id | The user can view the post by Id | The user can view the post by Id |  |
| 3 | The user can update the post in a forum | Data is entered into the forum | The data in the database is updated correctly | The data in the database is updated correctly |  |
| 4 | The user can delete the post from the database through the webapp | The user calls the destroy function | The post is deleted from the database | The post is deleted from the database |  |

### Discussion of Functional Testing Results

While testing the functionality of my web app I found everything to work as expected. All data that I created or updated was valid. When reading and deleing it everything also worked as intended. All navigation functionality worked as intended. All pages loaded and routed correctly. Login and registration also had no problems when carrying out testing.

* 1. User Testing

When doing the user testing, I made sure the front end of the application was working correctly. I checked the pages loaded correctly. I made sure all buttons for navigation worked correctly. I made sure that all forums worked as expected and a user could interact with them.

* 1. Conclusion

I had no issues when testing and everything worked as expected. I found it useful to record the steps taken, the input, expected output and actual output. Recording these makes it much easier to review and make changes where necessary. Logging each step is also useful in case you must go back and fix other problems or debug.

# Project Management

## Introduction

Project management is important to so that your project is managed well, and its resources and scheduling is balanced so your projects expected outcomes can be finished on time and deliver on your set goals. If a project is not properly managed, it is very easy for it to run over time, waste resources such as time and money, and deliver an unfinished project. There are four project phases, Requirements, Design, Implementation and Testing.

## Project Phases

### Requirements

Requirements is everything you will need for your project. It can be broken down into functional and nonfunctional requirements. For example, crud functionality maybe be required for my project, but dark mode may not be required for it to still function.

### Design

Design includes wireframes, type scales, colour palettes and other front-end components for my web app. Planning all of these out is essential for the project to be delivered on time. I first started with a wire frame, then slowly built up a high fidelity in Figma using my fonts and colours.

### Implementation

Project management is very important for the implementation stage. If the implementation stage isn’t managed properly then it is very easy to brake schedule. Having a backflow system and Kanban board is very useful for the implementation stage.

### Testing

It is important to leave enough time for testing in case some part of the project isn’t functioning properly and must go back to the implementation stage. Testing will be important to make sure your final product is working as expected

It is important to define the scope of the project and what will be done. This is done by talking to your employer and agreeing on the expected outcomes for the project. Laying out a plan is very important and having a contingency plan if something was to go wrong.

## SCRUM Methodology

Scrum is often used in project management. It is a framework for developing a project. Scrum has three roles, the product owner, scrum master and the development team. Scrum is quite effective for group projects but not necessarily the best method to product development for a solo project. Scrum projects will have a daily scrum to communicate with each other quickly and keep up to date. It is not meant for detailed discussions.

## How it is used

In my project I am using several useful project management tools are used such as, GitHub Project, Kanban boards and Gantt Charts. In GitHub project I have my project repo where I backup my code to. I also use the GitHub Kanban board. I use this to have a to do, In progress and Done list.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

Text, table

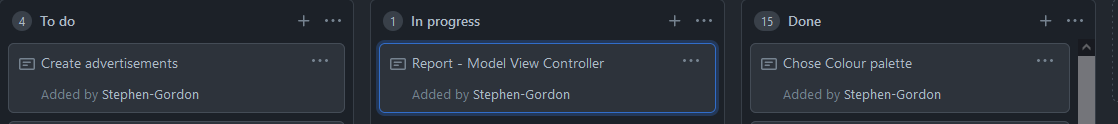
Description automatically generatedDiagram

Description automatically generated with medium confidenceThis is useful to have a backflow of work that can be done when some part of your project is finished or ready to move on. I also have setup a Gantt Chart to schedule tasks and estimate the time taken to complete those tasks. This is better for time management and budgeting as you your schedule is well documented. The Gannt chart is useful as you can be working on several tasks at once.

## Project Management Tools

### Github Project

Github project is used to manage your project. I was able to create a project and use their Kanban board. Using their Kanban board is very useful as I can plan out what I need to do. I can then keep track of what I am doing and what has been done. I can also make a back log of work to do.



### GitHub

GitHub was useful for storing my code repository. Every time I work on code, I can commit the changes at the end of the session. Once committed I push it up to the GitHub repo. I can also pull down the project on GitHub if working on another pc.

# Reflection

## Your views on the project

I think the project went well. I found staying organized was a big help. Using the tools like GitHub to manage my project made a big difference when staying organized and keeping everything on track.   
I struggled a little bit with entity relationships and foreign keys within Laravel.

## How could the project could be developed further?

I would like to have worked more with SQL queries in Laravel, for example having sort or filter functionality.

I would like to have done more user testing to see if I could make my application’s front-end smoother.

## Assessment of your learning.

I feel my back end and front-end development skills are a lot better. It was useful to develop my Laravel skills further from the advanced web development module. My understanding of the model view controller design is a lot better than before starting the project.  
I think my project management skills are a lot better. I now understand how to use and implement the necessary project management skills such as requirements, design, implementation, and testing. I now know how to use backflow, Kanban boards, Gannt charts and sprints to help with project management.

## Completing a large software development project

I learned that it’s important to stay on top of your work and not fall behind is very important. Keeping track of what you have done and what needs to be done is also important. Creating a plan and sticking to it is also important so you are not repeating steps.

## Technical skills

My HTML CSS JS and SQL skills are a lot better than before the project. Also, my understanding of working with a web development framework is better. Working with a framework makes more sense when you have multiple dependencies, languages and roles.

## Further competencies and skills

Further developing my research and design skills with Figma and Miro was also important. Having wireframes ready to go makes building the front-end CSS a lot faster as you have something to work directly from. Also researching and understanding other businesses was important when creating mine

# References

## Frameworks:

**Laravel:**

<https://laravel.com/>

**Bootstrap:**

<https://getbootstrap.com/>