Final Report

Easy Donation Project

Jay Rai (149054719)
Supervisor: Dr Hongxia Wong

Word Count: 7843

Project Definition Form

Student name: Jay Rai (Computer Science with Multimedia)

Project title: Easy Donation

About: The project is about designing an on-line donation management system to enable every person/household to donate anything easily. As many charity organisations are making a big effort to collect donations. It is quite costly and inefficient for charity organisations to do on-door envelope distribution and collection. While many people have unwanted things at their homes, they couldn't donate due to time limitations. The basic idea is to help people to select preferred charity organisations and send the donation information on line to inform charity organisations. Also to help charity organisations to provide better management of their resources and only make necessary door to door collections.

Project deliverable: The project deliverables for this project is a web application showing the donation management system, and a report to describe the stages and process of this project.

Originality: This project will enable me to design a site where people can choose their preferred organisations and send donations to them; similar to a "compare the market" website. This will not only help the people who use the site but also the charity organisations as it will cut down on door to door collections and make smaller organisations more noticeable.

Timetable:

Research (29/09/16 – 20/10/16) – This will involve researching similar donation/charitable sites, making note of the similarities and differences in the design and navigation. This will help me structure my own web application and improve on the flaws and also incorporate design ideas from other sites. This is one of the most important stages of the project as it will help spark ideas and create an initial proposal.

Wireframes / Initial Design (20/10/16 – 10/11/16) - I will be designing low fidelity wireframes (paper based or some online tool) for the initial design of the web application. I will also be brainstorming ideas on how the structure should be laid out through the research I have done. This will help me design a suitable prototype after getting feedback.

Prototyping (10/11/16 – 08/12/16) - When I finish these I can then design some high fidelity wireframes in Photoshop or some similar program. This will lead to the design of the final layout which I can implement into the web application, however this may change due to resources or testing. Prototyping is important as designs are going to change over time and new ideas will arise so it's best not to jump straight into development.

Development (15/12/16 -) – The development is the largest and longest stage, as this is the implementation of the design which can change due to resources and/or response from the testing of the initial design. The development stage may start earlier or later depending on how the project evolves, within this, I will be using a Laravel framework to help build my web application in PHP – therefore I will need some time to learn and practice using this framework.

Testing (After development) – Testing is the last stage and will be done once the implementation has finished, this is perhaps the most important stage as this will ensure that the site meets the needs of the parties using it. I will be performing some; functionality testing to verify the flow of the system and there are no dead pages. Some usability testing to test the navigation, interface and compatibility testing, and finally performance testing. I may do some security testing also as there may be payment transactions through this application however this will be through a secure channel such as PayPal or another secure package.

Acknowledgments

I would like to offer my special thanks to Acorns and Sue Ryder Care charity shops located in Great Barr, for giving me an insight to the problems faced by charity shops which therefore helped me during the research and planning stages of this project.

I would also like to express my appreciation to my supervisor Dr Hongxia Wong for making the outline of this project much clearer, and generously making time so I was able to contact her if I had any issues.

Finally I would like to thank my family and friends for their support throughout my study.

Contents

Abstract

This report will explain why there is a need for a donation application, addressing the issues of why such an application is needed. Issues, such as, smaller charities not getting the traffic of customers that other charities are getting because the market is too big.

There is an increasing amount of donations being made by people in the UK since 2014 has increased significantly. Research shows that people aged between 45 and 64 in higher socio-economic are more likely to donate, furthermore the majority of people tend to be female.

There are not many platforms where people can target charities directly to request donations, barring general social media websites. If people request to charities on social media platforms, there is a high risk of their request not being seen because of the amount of traffic on the website, therefore, an application strictly for the purpose of requests and donations is ideal especially for smaller charities.

EasyDonation is an application that will attempt to resolve these issues. This report details the production of this application, going through the stages of; research, design, development and evaluations.

This report will outline how this web application EasyDonation will help people donate to charities, send request to charities, allow smaller charities to be more recognisable and demonstrate the justifications behind the application and descisions made throughout.

Chapter 1: Introduction / Background

This report will justify the ideas behind this project and the need for such a web application; furthermore it will explain the aims and objectives of how this project was constructed as a whole.

Many people donate to charity, and many of them make regular donations whether its money, clothes or other items. Also, many charity organisations provide a door to door collection service for donations however this is something businesses are trying to reduce as it is not environmentally friendly. Charities issue bags to houses in certain areas near to them, then they have to make door to door collections in that area and sometimes there is nothing to collect; therefore a website where users can donate items and send requests would benefit both the charity and user.

This project has an interdisciplinary approach, exploring areas in human computer interaction and web development, and therefore relevant to a number of areas within computer science and multimedia.

The main areas in this report are;

Design & Human Computer Interaction – In order to develop an intuitive GUI, I will need to establish the requirements of the target audience and create an interface that is effective and simple.

Programming and Software Architecture – A number of programming concepts will be required in order for the application to work efficiently. Furthermore, software architecture, such as, the model view controller will be needed for the structure of the application.

Database – This project will require a carefully designed database, allowing data to be stored safely.

Web Development – Basic front end and back end web technologies will be used, such as, HTML/CSS, JavaScript, PHP and MYSQL. PHP Framework 'Laravel' will also be used.

Testing – A very important stage in this project and in any software development project. This will be done to ensure the application is working and find any issues that need to be fixed.

The overall aim of this project is to build an application that helps charities (large or small) to gain more donations, and to reduce door to door collections.

To achieve this aim it will involve completing the objective of designing and developing a web application.

Chapter 2: Preparation

2.1 Requirements Analysis

2.1.1 Target Users

Before any design ideas can be created, it is important to understand the type of user that will be using the application, and what the application has to do in order to satisfy their needs.

In this type of development, the purpose of the application or features designed will completely change in this project. This is especially true while taking an evolutionary acquisition approach, in using the iterative model; this will hopefully result in a much better design.

The intended users of this application will be charities and people who want to donate. Charities will want to manage/accept donations, while users will just want to be able to donate and keep a track of their donations. This website will also be useful for smaller charitable organisations that want to make a name for themselves.

2.1.2 Primary Research

2.1.2.1 Questionnaires

A questionnaire was conducted in order to get a better understanding of the target users. This was sent out to the general public (via: Reddit) and got many responses. Furthermore, I went around with a questionnaire aiming it at charity shops to see how their door to door collections work and if this process is suitable, the results were collected and shown in the pie charts below.

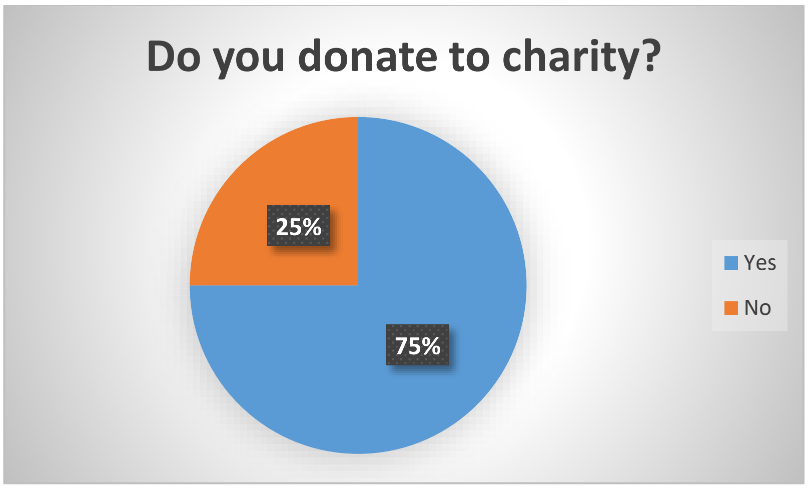


Figure 1: Do you donate to charity?

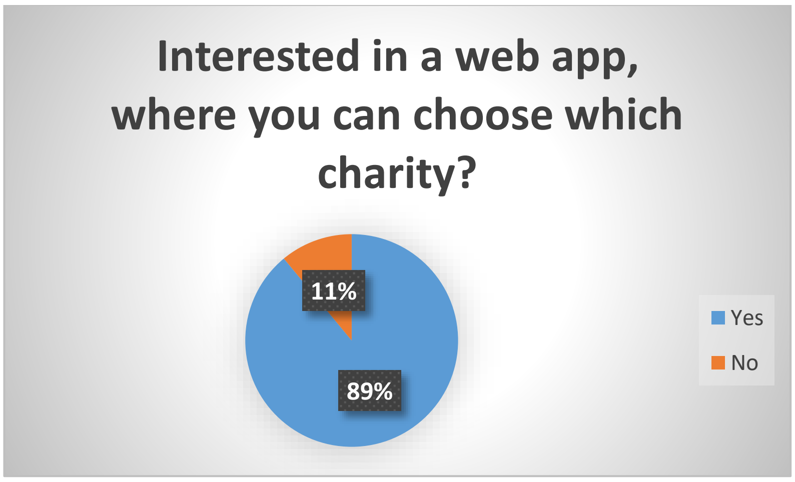


Figure 2: Would you be interested in a web app that allows you to choose a charity to donate to?

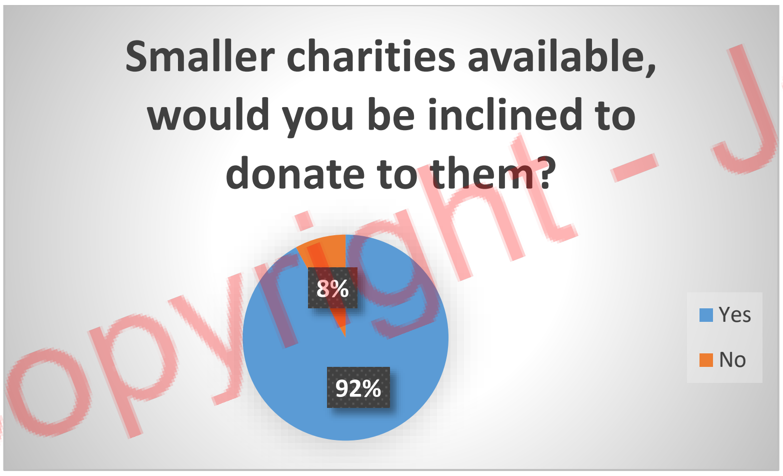


Figure 3: If smaller charities are made available on the web app, would you be more inclined to donate to them?

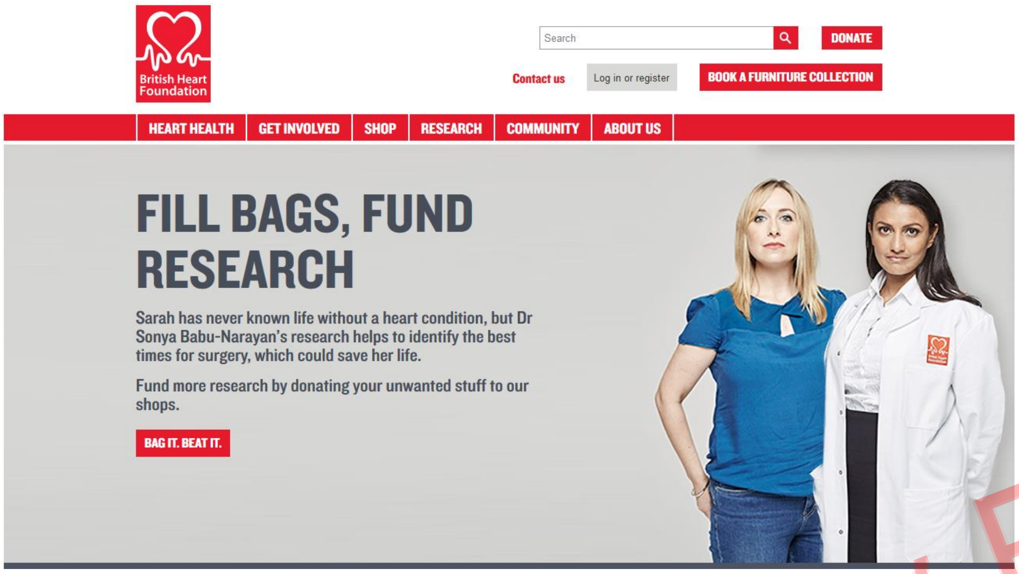
These are the three main questions that will provide a general idea of how easy it is for people who donate, and also whether a web application is suitable.

The first question asked if people donate to charities, as times and trends change this result can differ. As shown by the results still 75% of people who took this questionnaire said that they still donate to charities so a web application can still be useful. The second question asked if a web application is needed to choose a charity from and nearly 90% said that an application such would be a very useful tool, this shows that a web app is needed so people can easily find the charities they want to donate to. The third question asked was to see if a web application would benefit smaller charities against the more popular ones when it comes to donating, and over 90% of people said they would be more inclined to donate to smaller charities if they were available on a web application.

2.1.3 Secondary Research

Secondary research involved, looking at similar web applications that are already on the internet. For this project, this would involve, charity sites and any kind of application where the user is able to donate.

2.1.3.1 British Heart Foundation [1]

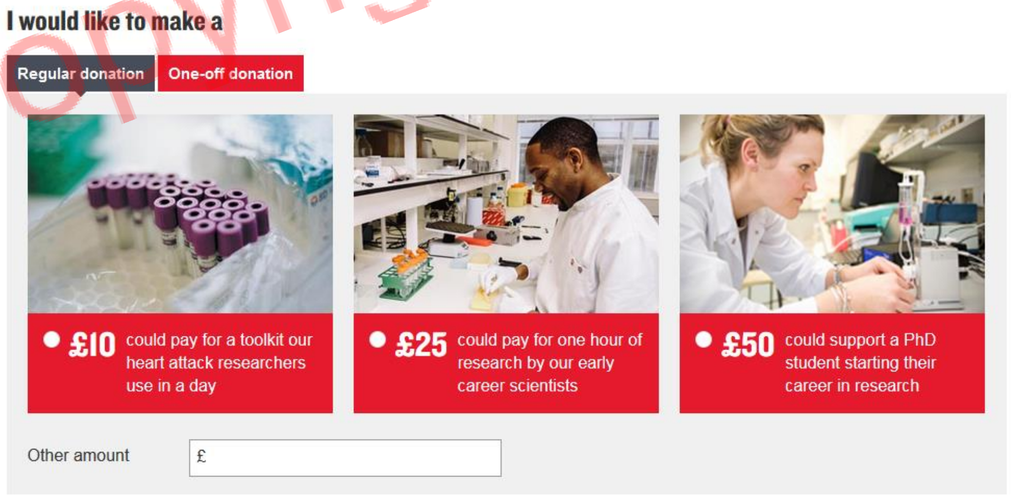


Sarah has never known life without a heart condition, but Dr Sonya Babu-Narayan's research helps to identify the best times for surgery, which could save her life.

Fund more research by donating your unwanted stuff to our shops.

Figure 4: Home page of British Heart Foundation.

This website has a consistent red colour scheme, a maximum of around three colours is used to draw the user's attention; too many colours can do the opposite effect. When booking a free collection, it is a very simple process of filling in your personal details and then collection dates and times, the user also does not have to be a member in order to donate – this is a good idea because many users don't want to make an account for just a one-time donation.



● £10 could pay for a toolkit our heart attack researchers use in a day

● £25 could pay for one hour of research by our early career scientists

● £50 could support a PhD student starting their career in research

Figure 5: Donation page of British Heart Foundation

The donation process is easy, and the website provides you with easy visual links to how much the user wants to donate.

2.1.3.2 American Heart Foundation [2]

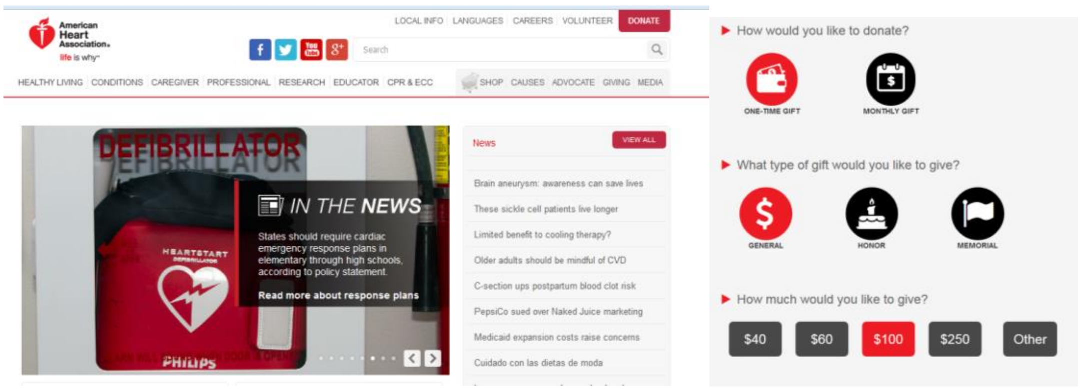

American Heart Association
life is why
LOCAL INFO
LANGUAGES
CAREERS
VOLUNTEER
DONATE
Search
HEALTHY LIVING
CONDITIONS
CAREGIVER
PROFESSIONAL
RESEARCH
EDUCATOR
CPR & ECC
SHOP
CAUSES
ADVOCATE
GIVING
MEDIA
DEFIBRILLATOR
IN THE NEWS
States should require cardiac emergency response plans in elementary through high schools, according to policy statement
Read more about response plans
News
VIEW ALL
Brain aneurysm: awareness can save lives
These sickle cell patients live longer
Limited benefit to cooling therapy?
Older adults should be mindful of CVD
C-section ups postpartum blood clot risk
PepsiCo sued over Naked Juice marketing
Medicaid expansion costs raise concerns
Cuidado con las dietas de moda
How would you like to donate?
ONE-TIME GIFT
MONTHLY GIFT
What type of gift would you like to give?
GENERAL
HONOR
MEMORIAL
How much would you like to give?
$40
$60
$100
$250
Other

Figure 6 & 7: American Heart Foundation – Home & Donation process

The American Heart Foundation only allows the user to donate money, from the homepage it does not look spectacular, but when the user goes through the donation process, it is designed intelligently with buttons and minimal text. As shown from figure 4, the items the user has to select are chosen from an icon instead of standard text button.

2.1.3.3 Next Gen Climate [3]

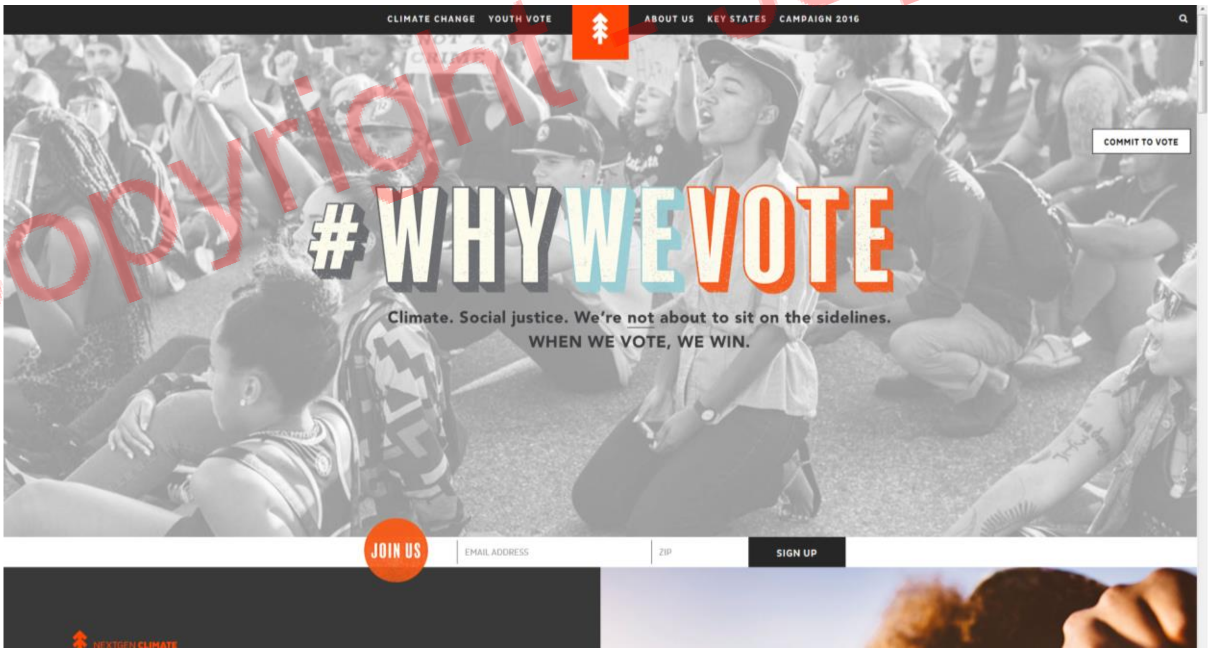


Figure 8: Next Gen Climate homepage.

This websites design is very elegant, as it stripped all the information (text) and populated the site with images that link to different parts of the site; the only text used is on the navigation bar for quick access. It can be quite tedious for new users to navigate on the website as it is 'in-your-face'. The downsides to this website is that you have to be a member before accessing most features, this will not be implemented into the web application however some of the design features will be taken into account.

2.1.3.4 Just Giving [4]

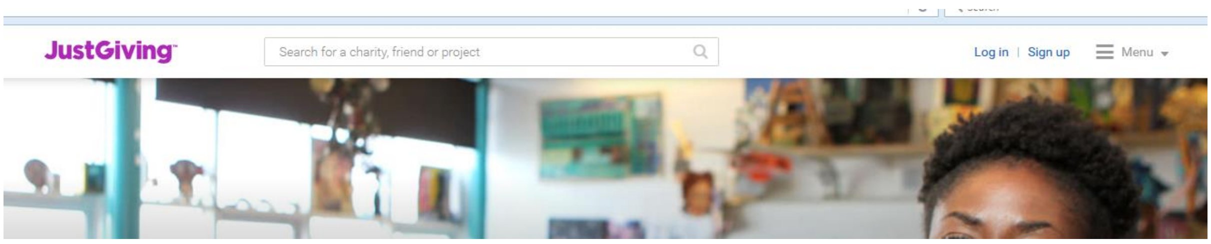
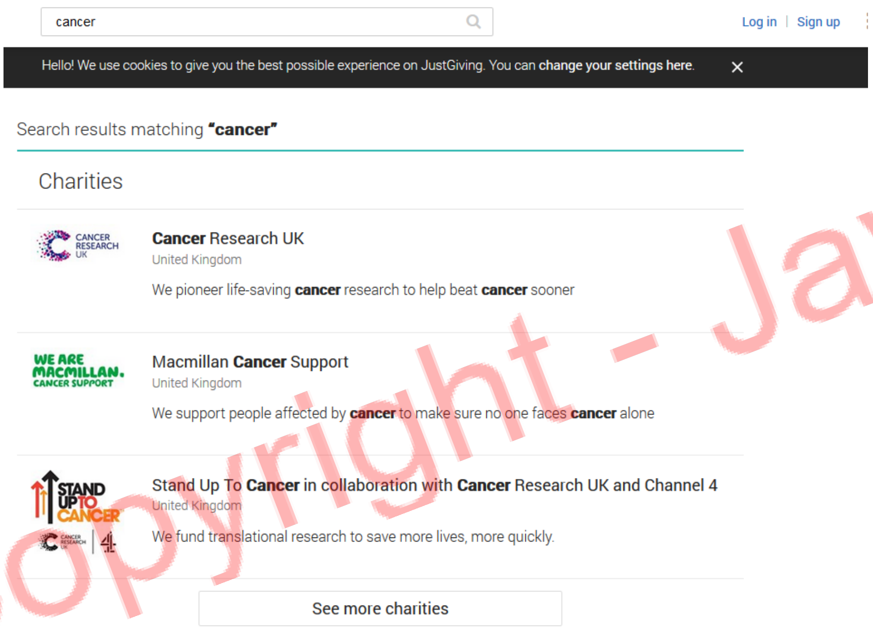


Figure 9: JustGiving homepage.

The JustGiving website is a very similar application to the one that is to be built, as it has a search base for all charities to donate to.

The donation process is very simple and the type of style that I will be implementing into my web application.

Once on the chosen charity, there is a link to their social platform and an option to make a fundraise also. This project should have a donation process similar to this and may implement a place to leave a message.



cancer

Log in | Sign up

Hello! We use cookies to give you the best possible experience on JustGiving. You can **change your settings here**.

Search results matching **"cancer"**

Charities

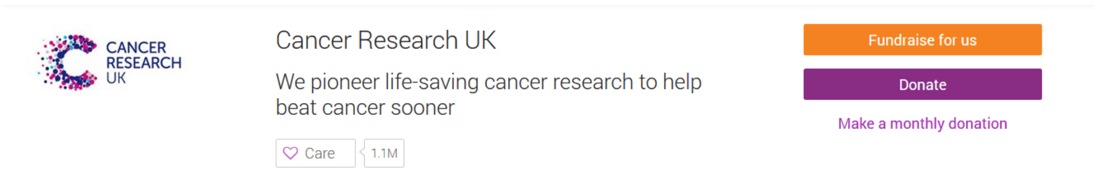
Cancer Research UK
United Kingdom
We pioneer life-saving **cancer** research to help beat **cancer** sooner

Macmillan **Cancer** Support
United Kingdom
We support people affected by **cancer** to make sure no one faces **cancer** alone

Stand Up To **Cancer** in collaboration with **Cancer** Research UK and Channel 4
United Kingdom
We fund translational research to save more lives, more quickly.

See more charities

Figure 10: Just Giving search page.



Cancer Research UK

We pioneer life-saving cancer research to help beat cancer sooner

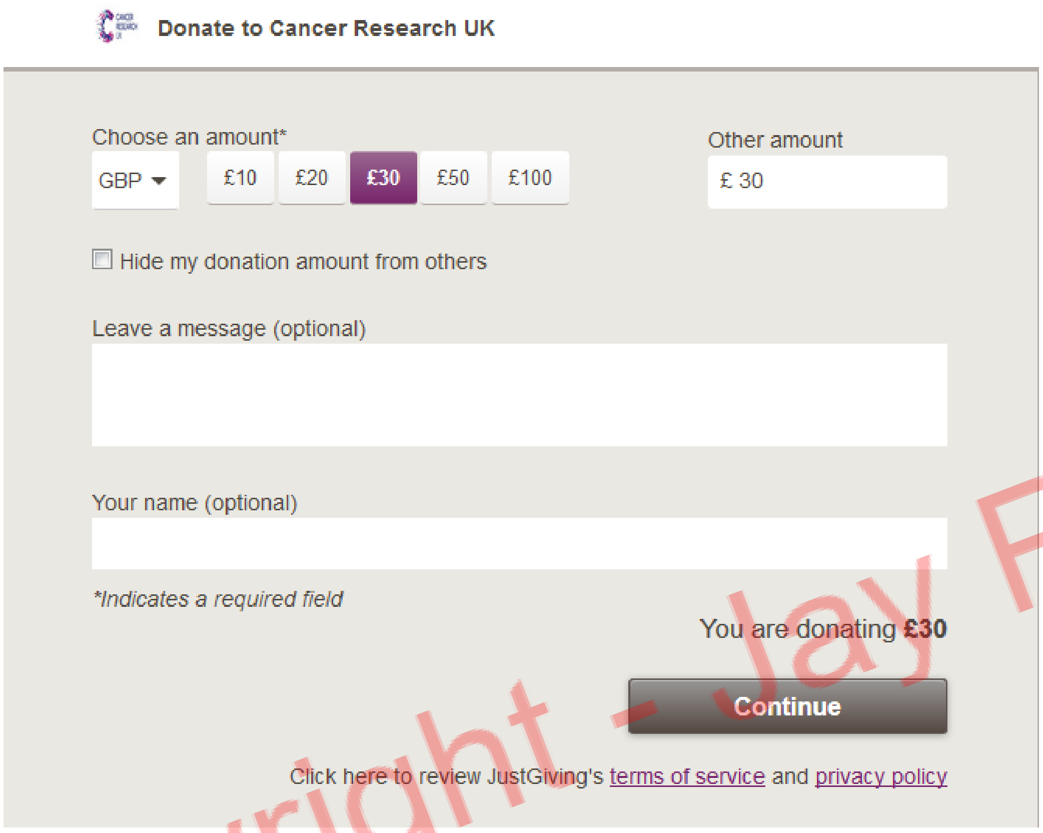
Care 1.1M

Fundraise for us

Donate

Make a monthly donation

Figure 11: Just Giving search page – charity page.


Donate to Cancer Research UK

Choose an amount*

GBP £10 £20 £30 £50 £100

Other amount

£ 30

Hide my donation amount from others

Leave a message (optional)

Your name (optional)

*Indicates a required field

You are donating **£30**

Continue

Click here to review JustGiving's terms of service and privacy policy

Figure 12: Just Giving donation page.

Figure 11 and 12 show how to donate to a charity on the website, it is a very simple process with a clean interface, which is something that should be taken into account when building the application.

Secondary research was important in this project because conducting primary research was restricted due to time, therefore gathering as much information from external sources really gave an insight to how this application could be developed. As most charities have their own websites including the smaller ones, exploring different applications can trigger new ideas.

2.1.4 Personas

Personas are a good way to be able to design to the type of user and to understand/communicate with how my application will be useful to these types of users. Below is an example of a persona.

Name	Kevin Jericho
Age	32
Description	Kevin has recently moved out of his parents' house in Leicester after finding a well-paid job in Birmingham City Centre. Kevin is now living in an apartment alone. Kevin is settling in nicely in his new apartment however he has a lot of old clothes and other items that Kevin wants to dispose of, however it is too much effort to have them collected by the council. Also, there are not many charity shops near him that are trustworthy. Kevin would like an application where he can choose the charity he wants to donate to and then have them collect the items at a time that best suits him.

	Kevin is not ‘Tech-savvy’ and owns 1 laptop and 1 phone, and sometimes gets confused with new technology and some applications.
Needs	Kevin’s needs are as follows: • Web application that has a simple interface. • Quickly dispose of the items he wants disposing. • More space in his apartment.

2.1.5 Overview

Primary and secondary research will help understand the type of application that will be built in more detail by looking at existing solutions and using similar design techniques.

A number of different models will be followed in order for this project to be successful, including; agile, iterative and v model, these will be discussed further on in the report.

2.2 Use Case Diagrams

These are some use case diagrams, designed to show what the main functions of the application will be.

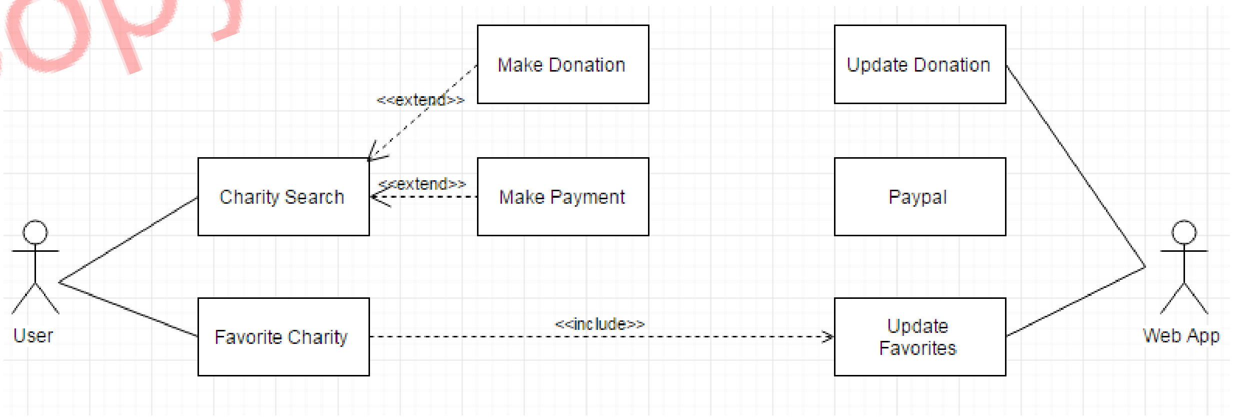


Figure 13: Use Case diagram to show some of the functions of a user searching for a charity.

This use case diagram shows the main functions of a user that searches for their chosen charity in the search bar. The user will search for the charity and then have a choice to add the charity to their favourites, or make a payment donation or other donation to the charity. The web application “actor” will update the donation, show the user a PayPal system, and update the user’s favourite list if they added the charity to their favourites.

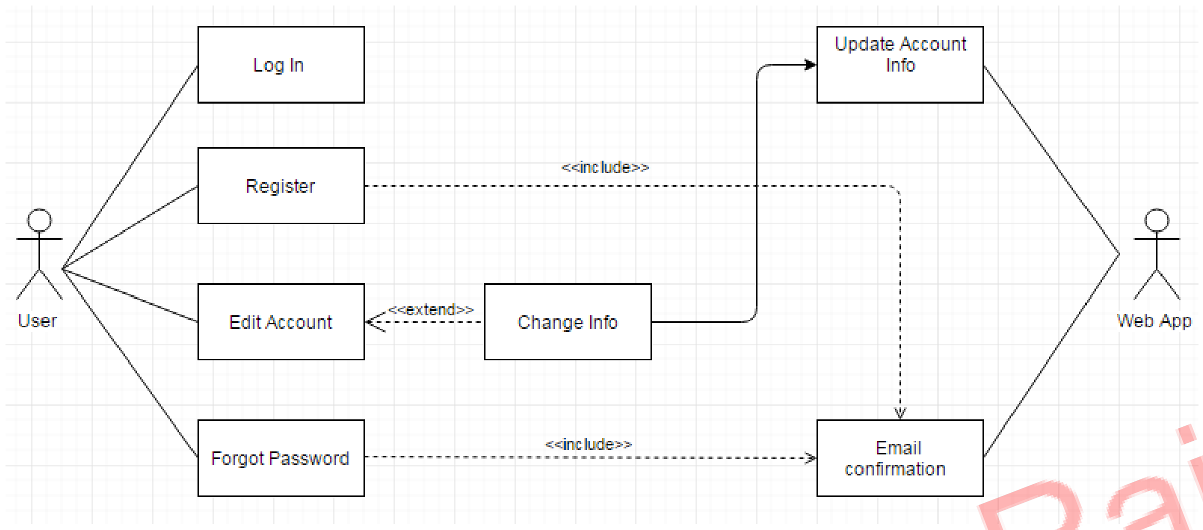


Figure 14: Use Case diagram to show a new user making an account and logging in.

This use case diagram shows the options that a new user has when presented with this system, they have a number of options, such as; log in with an existing account, register for a new account or forgot password. The web application will present the user with an email confirmation if they decide to register or click forgot password, or update the account information if the user decides to edit their account after they have logged in.

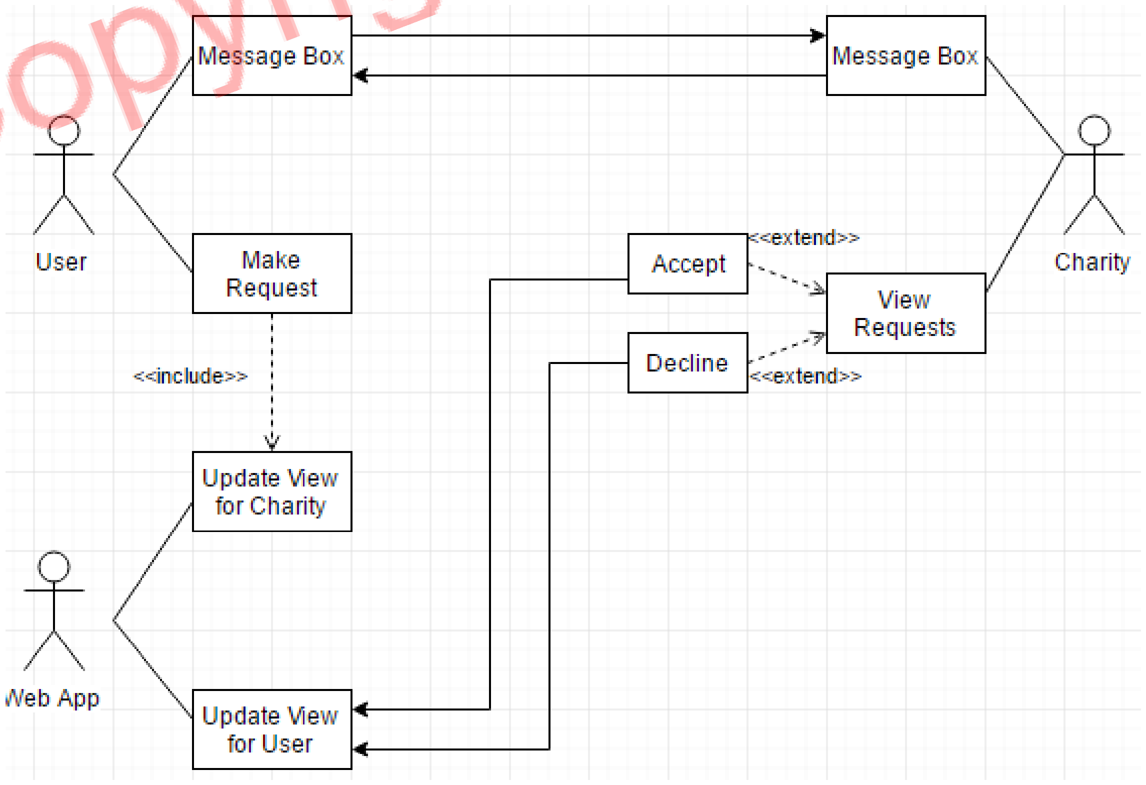


Figure 15: Use Case diagram to show a user making a request to the system, and a charity accepting or declining.

This diagram shows the action of the user making a request on the application, which charities can view and decide whether to accept the request or decline it. If the charity decides to accept it, they will be able to contact the user regarding details. The web application will only show the latest requests made by users to keep interface updated.

2.3 User Requirements

The user requirements help when building this application, to make sure the design is suitable to the user and this app will be beneficial. These have been split up into functional and non-functional requirements.

2.3.1 Functional

Functional requirements describe what my system should do, and what the user expects from it. These have been divided into; general web application requirements which discuss the general functionality that the app should deliver to the user. The user actions define the general things that the user can do on the application. Finally, the donation requirements is what the application should offer when the user is going through the donation process.

2.3.1.1 General Web Application Requirements

1. *The application should allow users to sign up/ create account.*
2. *The application should allow users to delete their account.*
3. *The application should allow users to modify their account/data.*
4. *The application should present the user with a forget password option.*
5. *The application should allow the user to see their donation history.*
6. *The application should allow the user to see their favourites list.*

2.3.1.2 User Actions

1. *The user should be able to search for available charities.*
2. *The user should be presented with information about the selected charity.*
3. *The user should be able donate items or money.*
4. *The user should be able to favourite chosen charities.*
5. *The user should be able to delete charities from their favourites list.*
6. *The user should be able to make a request to the other charities.*

2.3.1.3 Donation Requirements

1. *The application should present the user with a donate money option.*
2. *The application should present the user with a donate items option if the charity accepts items.*
3. *The application should ask the user how much they want to donate.*
4. *The application should update the requests to show the charity.*
5. *The application should allow a charity organization to accept or decline donation requests..*
6. *The application should allow the user to make a secure payment.*

2.3.2 Non-Functional

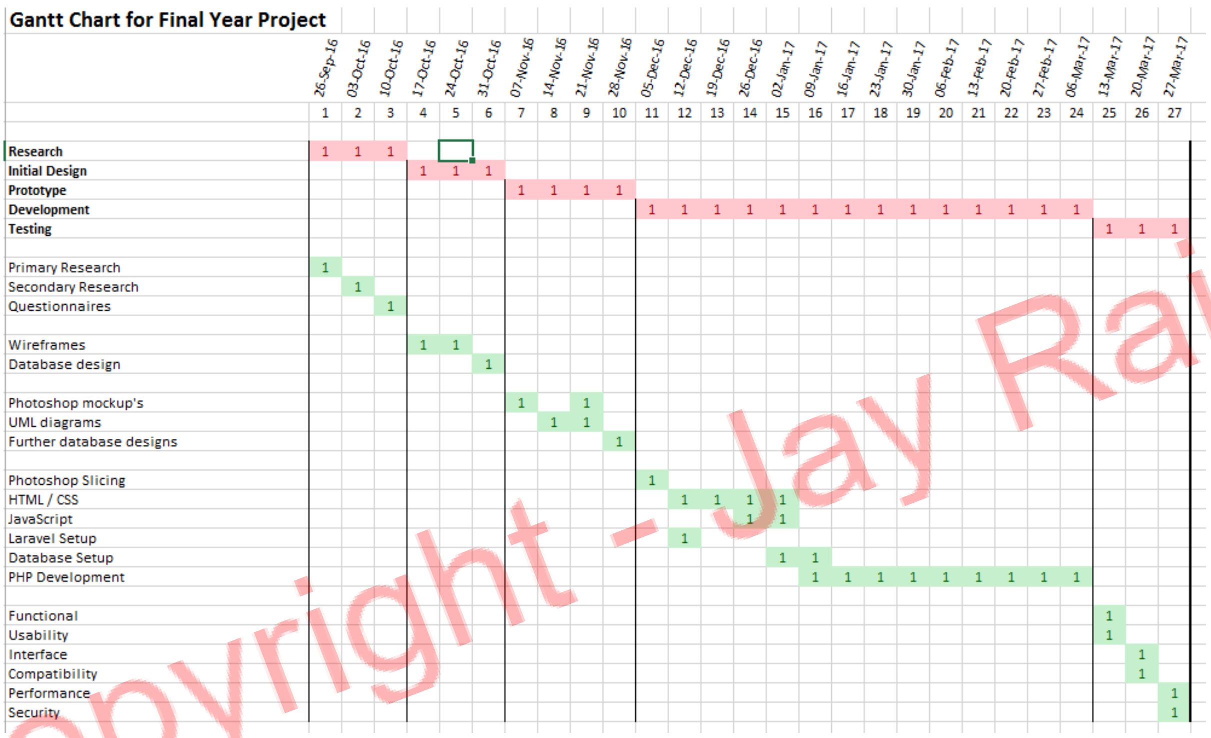
The non-functional requirements describe how the system works, and how the system should behave.

1. *The application should run smoothly across all platforms (Internet Explorer, Chrome, Safari, Firefox, Opera etc.)*
2. *The page load time should be kept to a minimum.*
3. *The GUI should be easy to use, with a well-designed clean interface.*
4. *The application could be functional on a mobile device.*

2.4 Schedule

This project will be completed in logical steps, throughout the project the dates in the Gantt chart are considered as deadlines and work should be completed before these deadlines so it can be checked over.

The Gantt chart will be updated if the project falls behind, or some parts are moved forward.



Gantt Chart for Final Year Project

	26-Sep-16	03-Oct-16	10-Oct-16	17-Oct-16	24-Oct-16	31-Oct-16	07-Nov-16	14-Nov-16	21-Nov-16	28-Nov-16	05-Dec-16	12-Dec-16	19-Dec-16	26-Dec-16	02-Jan-17	09-Jan-17	16-Jan-17	23-Jan-17	30-Jan-17	06-Feb-17	13-Feb-17	20-Feb-17	27-Feb-17	06-Mar-17	13-Mar-17	20-Mar-17	27-Mar-17
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Research	1	1	1																								
Initial Design				1	1	1																					
Prototype							1	1	1	1																	
Development											1	1	1	1	1	1	1	1	1	1	1	1	1	1			
Testing																									1	1	1
Primary Research	1																										
Secondary Research		1																									
Questionnaires			1																								
Wireframes				1	1																						
Database design						1																					
Photoshop mockup's							1		1																		
UML diagrams								1	1																		
Further database designs										1																	
Photoshop Slicing											1																
HTML / CSS												1	1	1	1												
JavaScript														1	1												
Laravel Setup												1															
Database Setup															1	1											
PHP Development																1	1	1	1	1	1	1	1	1			
Functional																									1		
Usability																									1		
Interface																										1	
Compatibility																										1	
Performance																											1
Security																											1

Figure 16: A Gantt chart showing the main stages of the project and the time periods they should be completed in. The main headings are at the top in red, and the green stages are subheadings accordingly.

2.5 Web Technologies

The development of this web application will rely on web technologies – both front end and back end frameworks - working cohesively in order for the application to run smoothly.

2.5.1 Front End

The majority of the front end of the web application will be coded using HTML 5 and CSS 3 - CSS will be used for all the styling and positioning. However, for added functionality, animations and interaction JavaScript will be used, in particular the framework JQuery (version: 3.1.1).

2.5.2 Back End

The back end of the web application will be coded in PHP framework Laravel, AJAX requests will also be used so loading of requests can happen in the background. The current version is 5.3 (5.4 releases in January which will be too late to start development). Laravel's key features include: restful routing, composer (tool that manages third-party packages easily), built in unit testing and an ORM (called Laravel) which means working between database objects and relationships is a lot more eloquent with easy syntax.

I have chosen to use a framework because as well as making development a faster process, it makes the code well organised and clean, which makes it more reusable. Most frameworks, including Laravel use a MVC architecture which ensures separation of presentation and logic. Also, learning Laravel will be a challenge as I have not used it before, therefore the end application will be more satisfying.

I chose to use Laravel because it has sufficient documentation which makes the learning process easier, also makes managing databases simpler. Furthermore it has a templating engine called 'Blade' which allows me to write plain PHP in the templates.

Finally, Laravel has a very unique architecture, making it easier for developers to create their own infrastructure specifically designed for their application; and therefore Laravel can be used for big or small projects.

2.6 Design

Before any development can begin, design must be completed in order to set the overall tone of the project and some structure can be followed when developing.

The first thing to do when designing is to brand the application, which can be the most difficult part because a lot of people judge the application solely on the branding (this includes; the name and logo). Therefore choosing a name that is boring or hard to pronounce is really bad for the project before it has even begun.

Names that were brainstormed were: AceDonate, LoveCharity, EasyCharity, and EasyDonation.

Eventually the name 'EasyDonation' was chosen because it's easy to remember and it is also the whole point of the application – to make donations simpler. Having also shown the names to other people, the majority thought 'EasyDonation' sounded better.

Once the name was chosen, the logo could be designed. Initial designs were designed on paper and taken further in Photoshop.

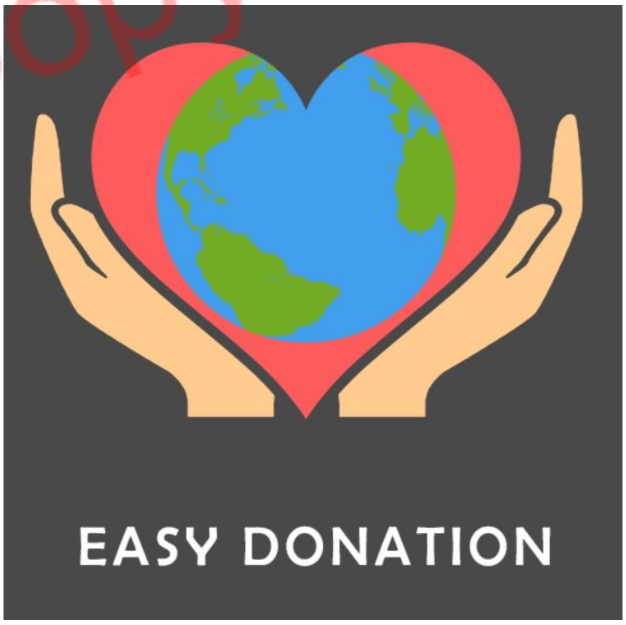


Figure 17: Easy Donation final logo.

This logo was picked after researching other charity logos and they all followed a similar design, with similar objects in each one, i.e. a heart. Therefore, after taking ideas from other logos, this was sketched and then transferred into Photoshop. This logo works well with the application as there are a variety of neutral colours which will work with any other colours. These colours were chosen purposely not too bright otherwise they would clash with other colours in the logo, these colours together create a warm feeling that you expect from a charity.

2.6.1 Wire-framing

When first starting to design a web application or any kind of interface, it is important to produce wireframes, as doing so can generate new ideas. Wireframes block out the main functionality of the application and allowed me to concentrate on what the design should look like to benefit user experience. The wireframes developed on paper can be seen below:

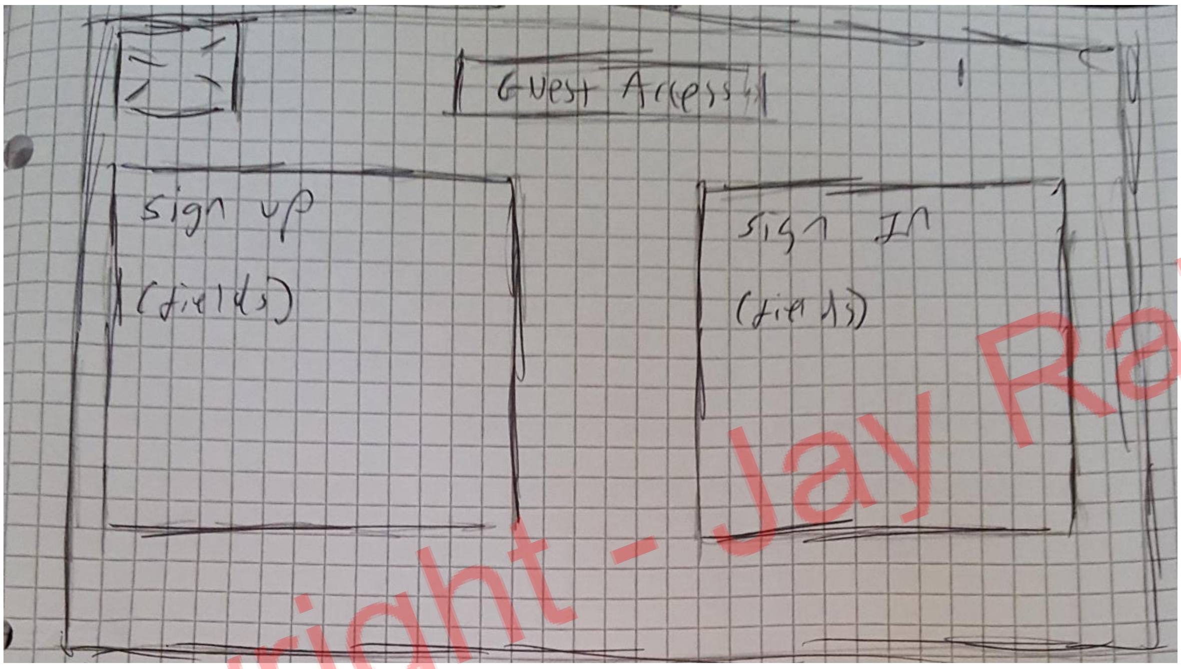


Figure 18: EasyDonation Wireframe (Welcome Screen)

This is the initial screen the user will be faced with, where there are three options, sign up to access all features, log into their account or continue as a guest where some features will be locked.

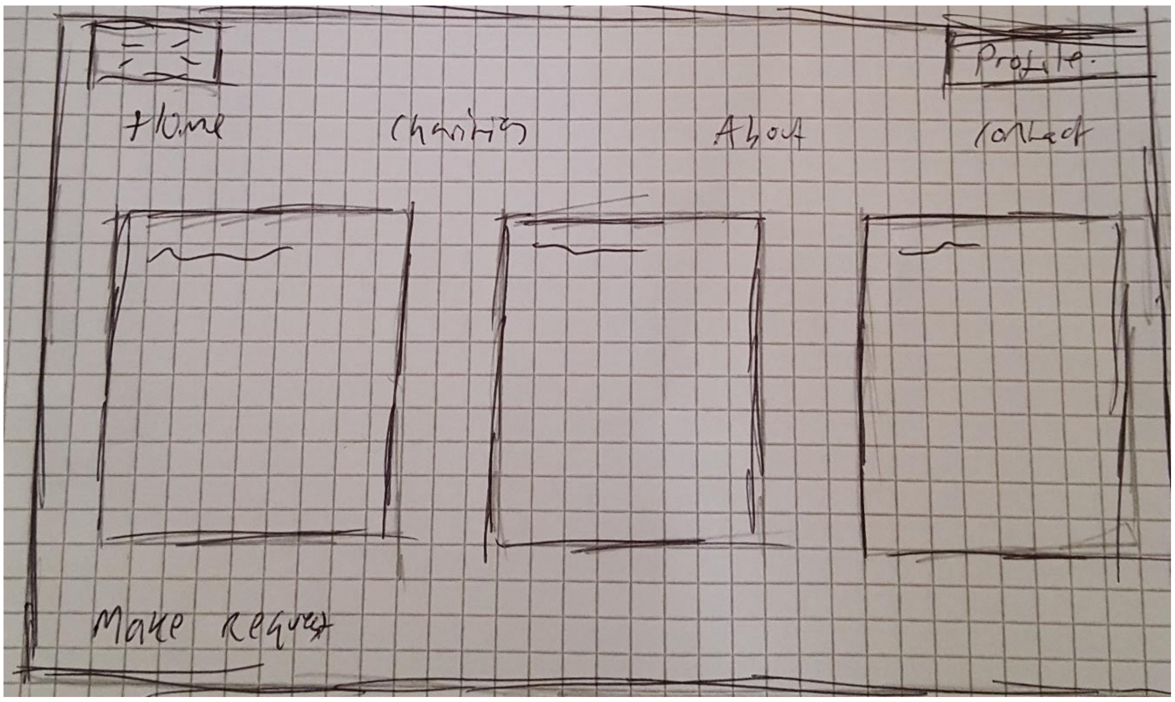


Figure 19: EasyDonation Wireframe (Dashboard)

This wireframe shows the screen the user will see when they log into the application, the three panels will show the users favourites, recent donations and requests respectively. Below there is an option for the user to make a request.

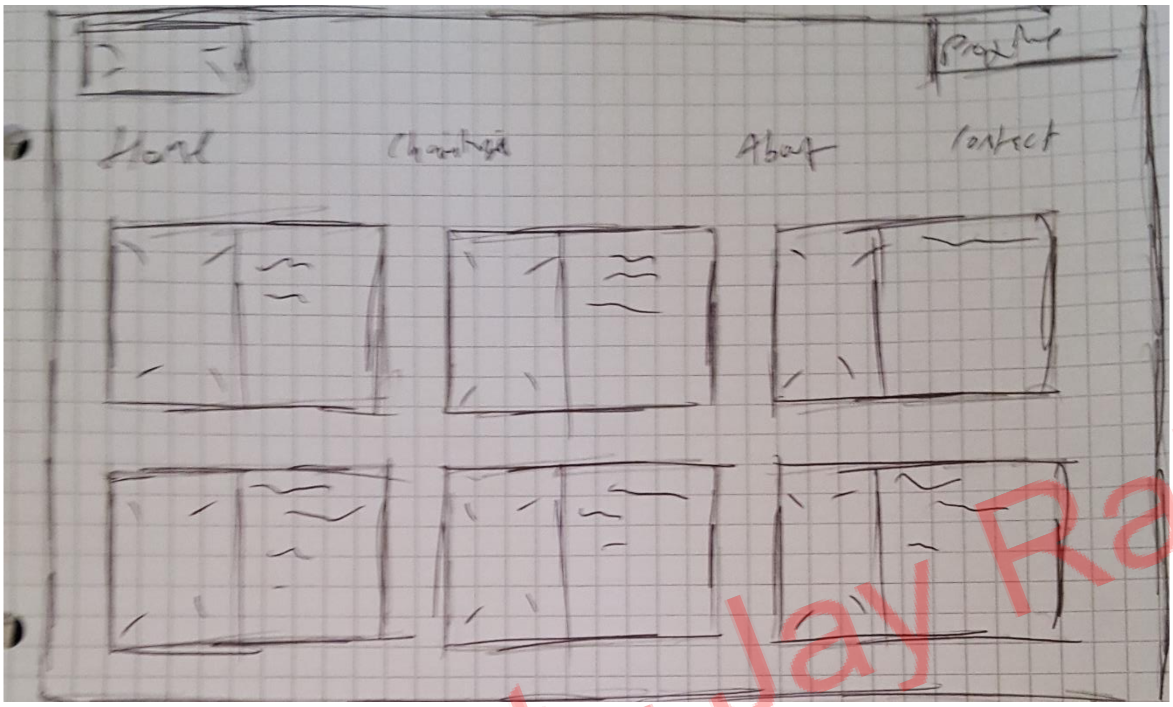


Figure 20: EasyDonation Wireframe (Charities Screen)

This screen will have a maximum of 6 charities on per page, to keep the scrolling to a minimum. There will be an image placeholder and the accompanying text will allow the user to donate.

After designing these wireframes, it's important to see what the designs will actually look like when put onto the web, therefore some of the screens can be seen in InVision [6] below:

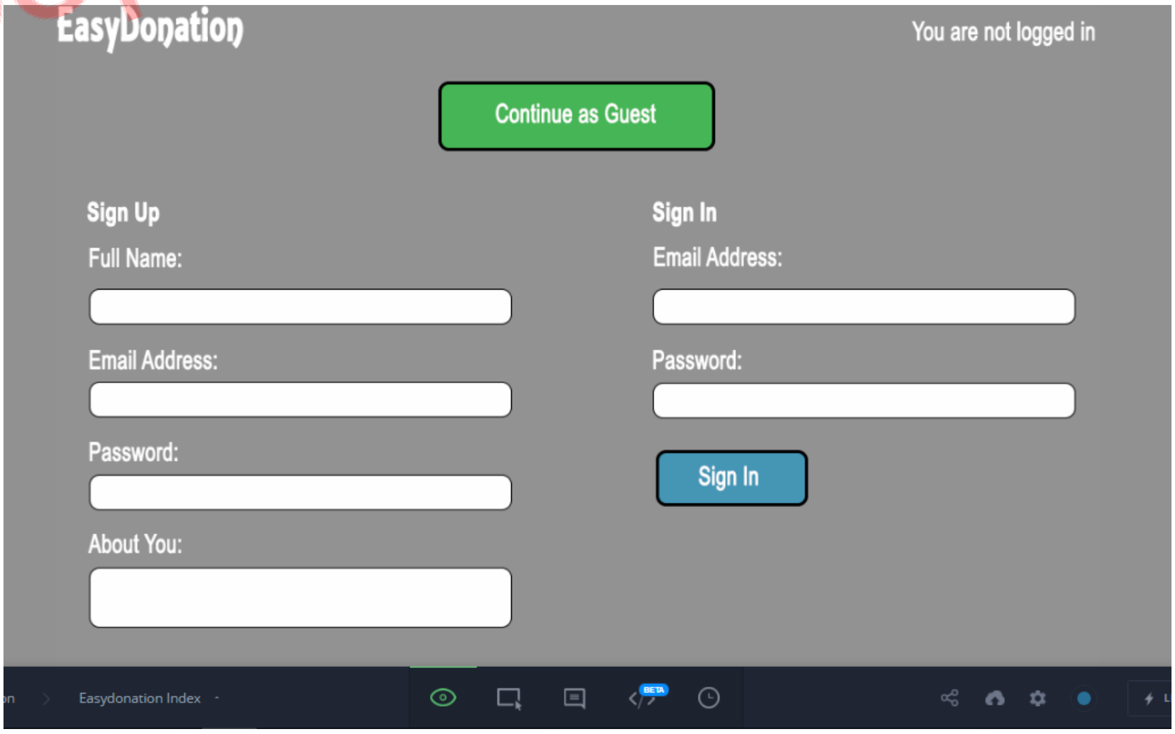


Figure 21: InVision Prototype of the Index Page.

With the added colour and structural layout, this looks a lot cleaner to the user, the options are clear and there is not a clash of colour or too much information.

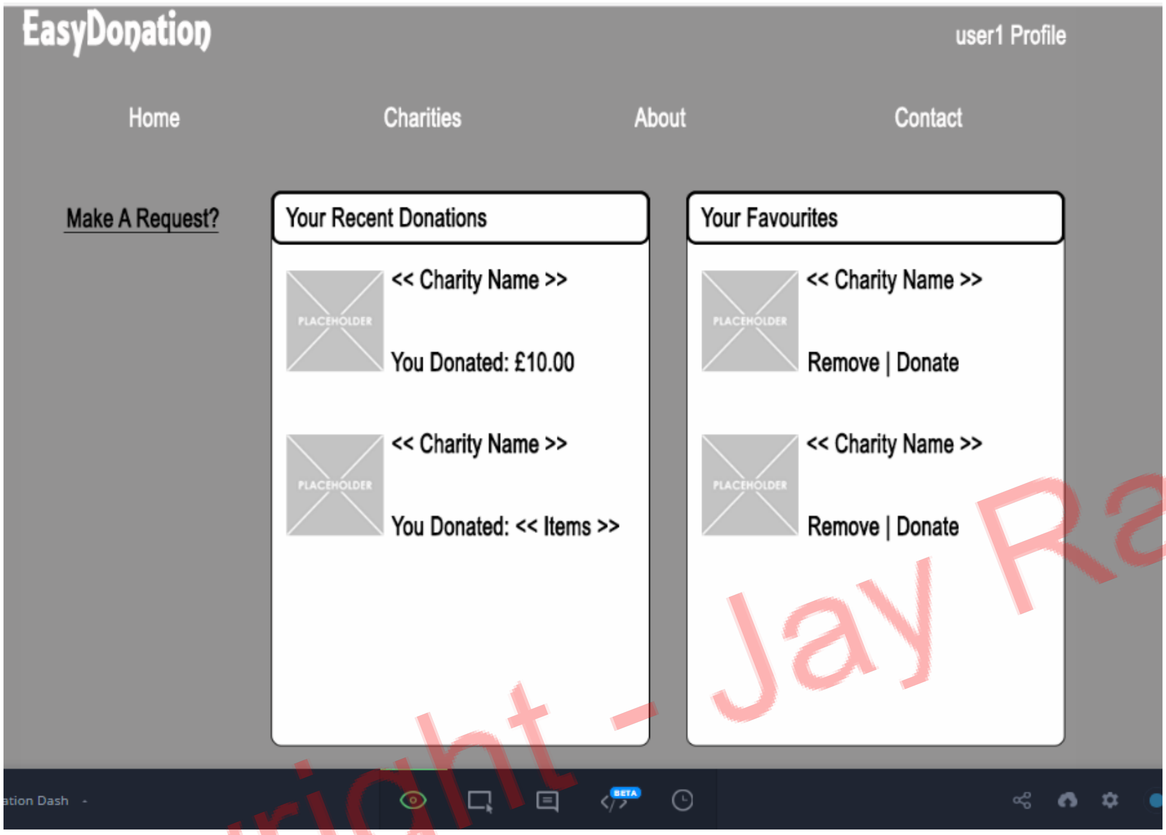


Figure 22: InVision Prototype of the Dashboard.

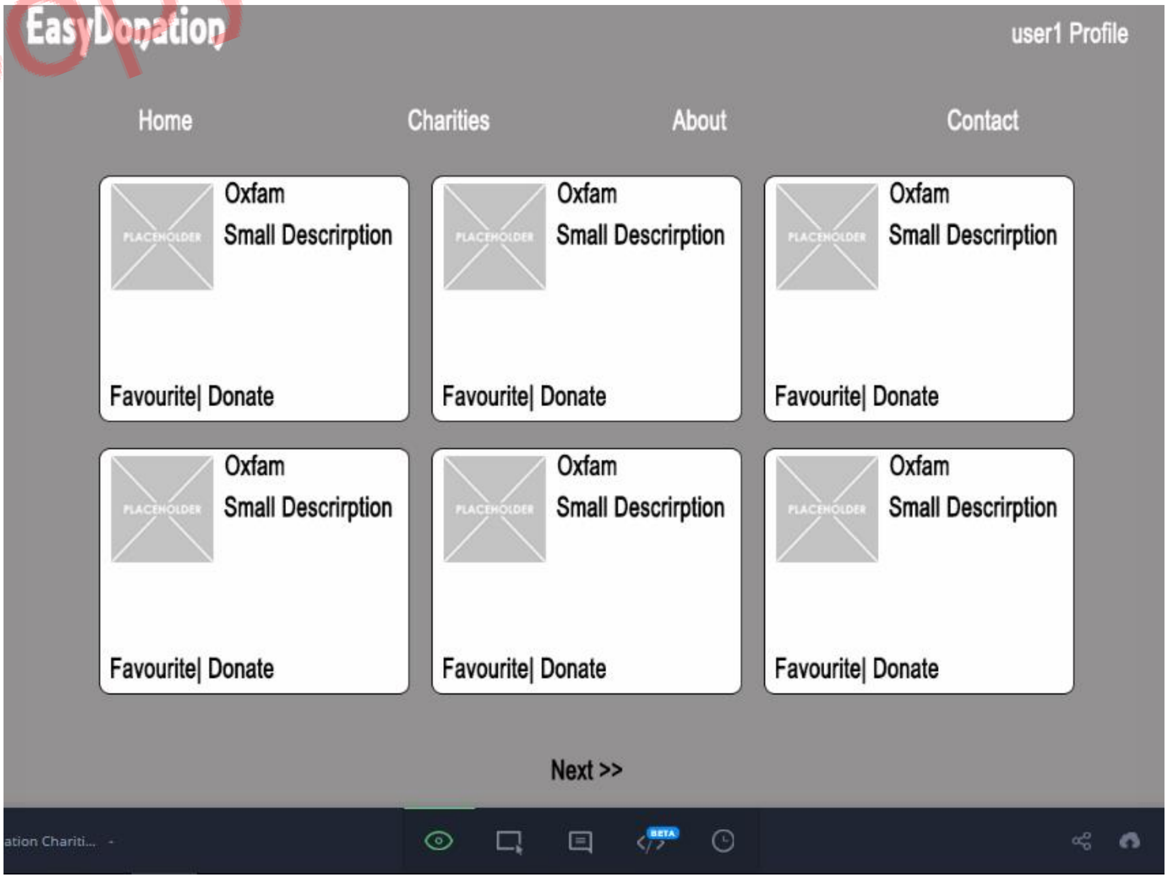
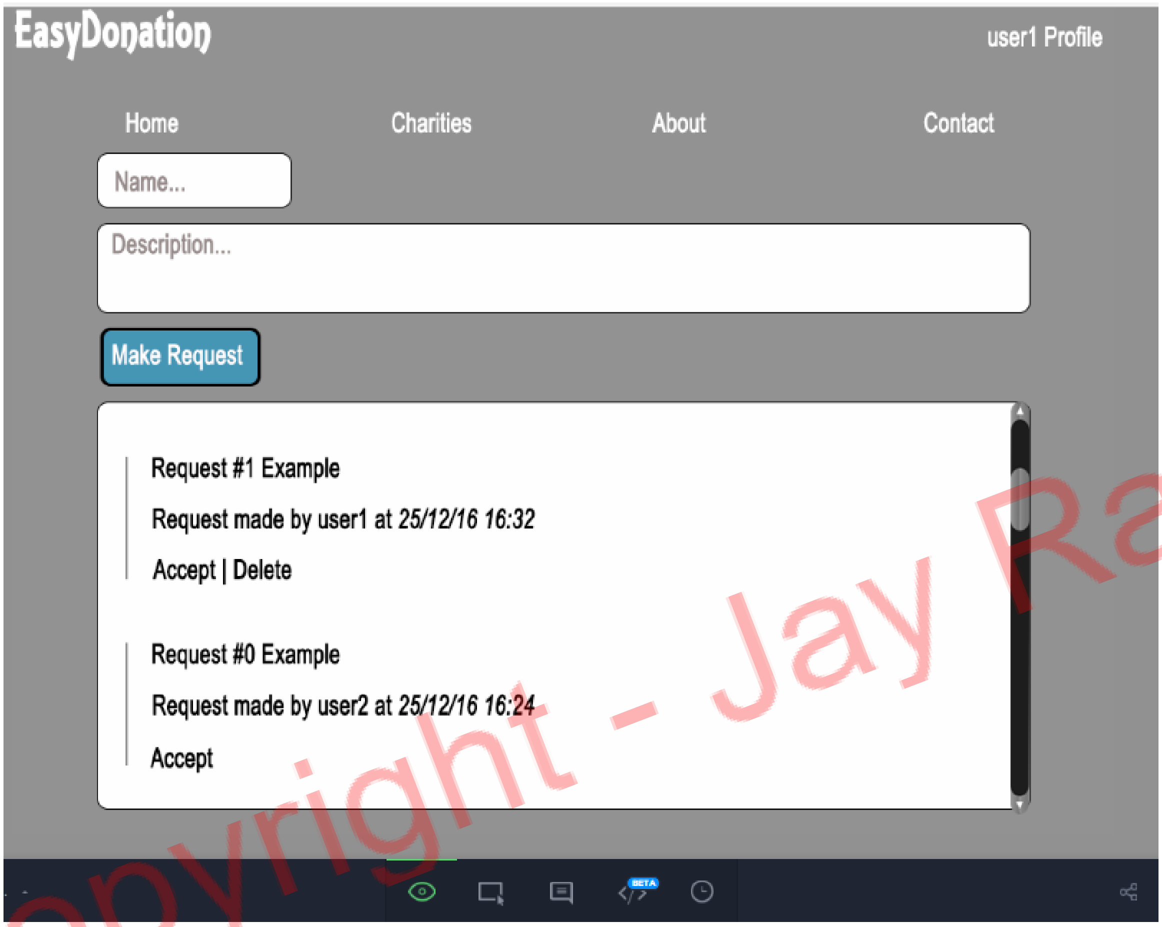


Figure 23: InVision Prototype of Charities page.


EasyDonation

user1 Profile

Home Charities About Contact

Name...

Description...

Make Request

Request #1 Example

Request made by user1 at 25/12/16 16:32

Accept | Delete

Request #0 Example

Request made by user2 at 25/12/16 16:24

Accept

Figure 24: InVision Prototype of Requests Page.

There is already a big difference in the systems designs, and from the prototypes a real-life application can be seen with some clear functionality.

After experimenting in InVision, the design was best as simple and minimal, there is no need for an overload of information on each page. This design had to be consistent throughout the whole application. It is interesting to see the changes the design has taken and progressed from the initial wireframes to the mock ups; however, this may also change when development is in progress.

2.6.2 Responsive Design

The application will have various pieces of information on each page, however there are not many pages to navigate to, therefore a simple navigation bar located at the top of the screen will be most useful. When initially designing this at the beginning of the project, a side navigation bar was going to be added.

The application will have several different pages, where the users can perform different actions, therefore to keep the design appealing the application will have a side navigation bar, furthermore scrolling will be kept to a minimum because a lot of information is not needed on every page.

As the front end will be designed using the Bootstrap framework, the panels, navigation bar and other elements will be responsive. This means that even when the window has its size manipulated, the applications functionality will still be available and clearly presented to the user.

2.6.3 Initial Database Design

As I am following an agile development style, the UML diagram does not represent the final state of the application, the following diagram is a starting point from which it can be built upon.

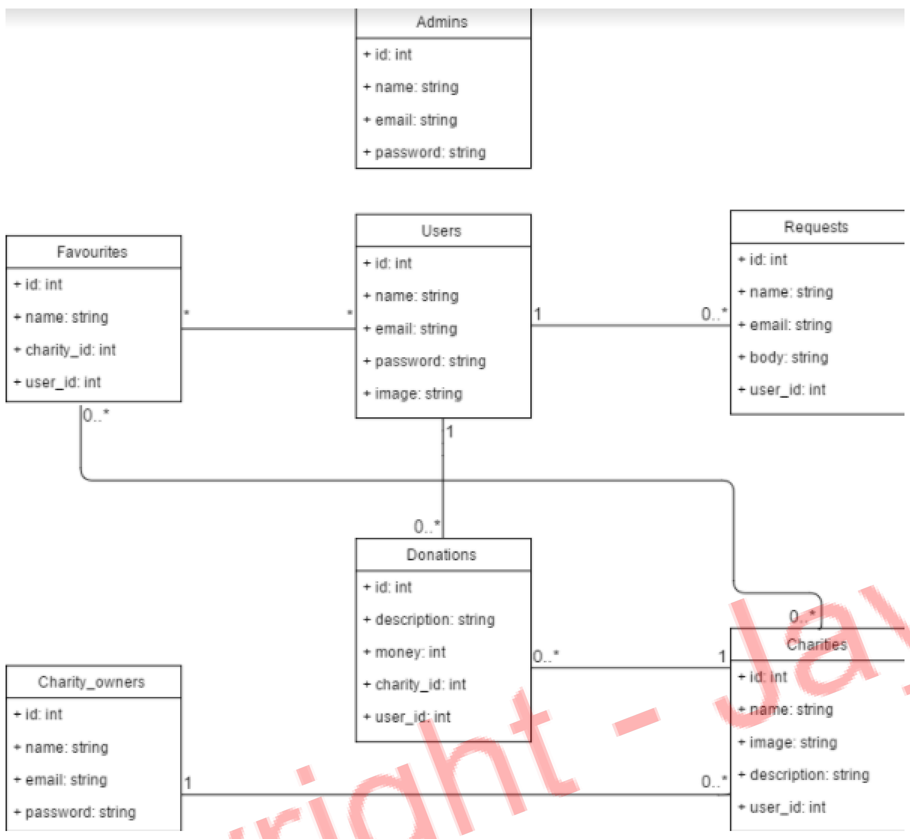


Figure 25: Database Diagram

.

Chapter 3: Deliverable

3.1 HCI & Interface

3.1.1 Animations

To further enhance the overall user experience and draw the user's attention, well placed and simple animations can be very effective, it also provides feedback to the user if anything goes wrong in some functionality.

Within this application, animations have been well used as well as some transitions. Some CSS animations were used as well as JavaScript to give the application a nice smooth feel for the user.

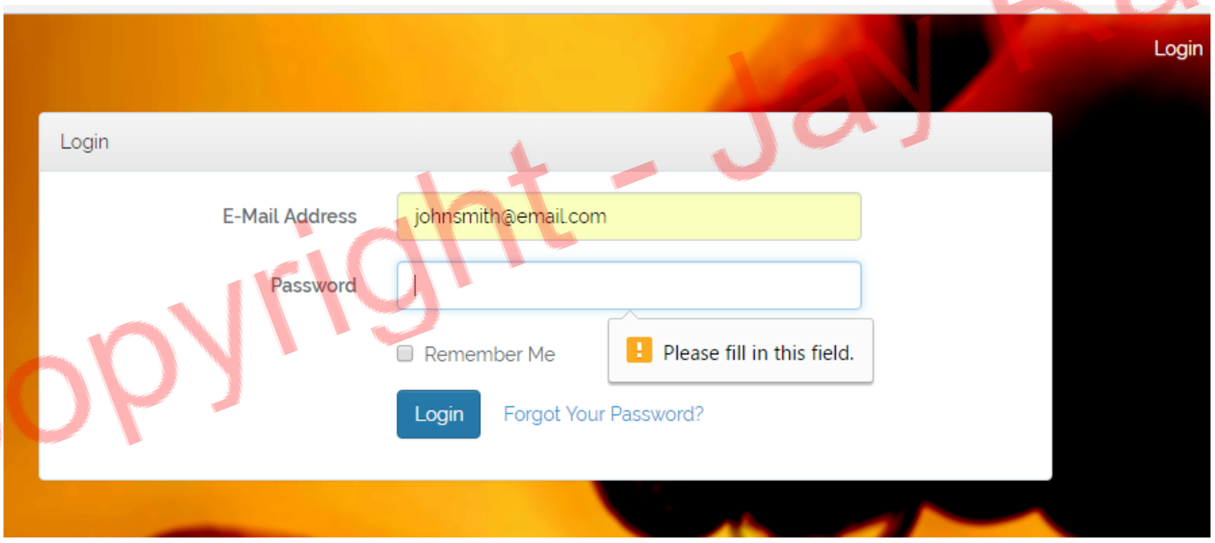


Figure 26: Animations for errors.

When a user doesn't fill in a field, the application shows a small pop-up by the field they haven't filled in with a warning. Animations like this make the application look nicer and also is very clear to the user where they have gone wrong.

3.1.1 AJAX

When attempting to implement the favourite charity functionality, AJAX requests were used so everything happens in the background and does not disrupt the user experience too much, this was done cleaner in Laravel with the use of routes ('*{{ route('favourite') }}*').

3.2 Development Method

The overall aim of this project is to create a web application that works effectively and beneficial to the end users. Initially the project would be of a relatively small scale however the application needs to allow for scalability and enable to work as a large scale application. Therefore, I will be implementing a few development methods; agile, iterative and v model.

3.2.1 Agile Model

Agile was the main development strategy that will be followed. The agile model is a type of incremental model and areas of the site will be developed in rapid cycles. It is useful for a long project like this to ensure the application will be useful to the end users. This model has many benefits to my type of project.

Working software (parts of the application) are delivered on a frequent basis, so the application is being design and developed correctly. It is very easy to adapt part of the application when there is a change in circumstances, even if there are late changes – these can be implemented. It is both suitable to fixed or changing requirements. Finally, not much planning is required making the whole project easy to manage.

3.2.2 Iterative Model

An iterative process model will also be followed. This model does not attempt to start with a full specification requirements, development begins by specifying and implementing just part of the application, which can then be reviewed in order to identify further requirements.

This model can be used in cases where a small part of the application is developed which can then be tested. This process would be repeated (develop – test), to ensure every part of the application is working as intended to.

The iterative model has many benefits including; detecting flaws in the project early to avoid a bigger problem later on in the building. The model also supports changing requirements, also testing and debugging is made simpler during small iterations.

3.2.3 The V Model

The V model is essentially very similar to the waterfall model – more specifically the V model is an extension of the waterfall model. Phases only start when the previous phase is completed, this is used when in design to development phases, i.e. every phase in the design has to be done sequentially (wireframe – mock ups – interface implementation) before a start can be made on the actual development of the application.

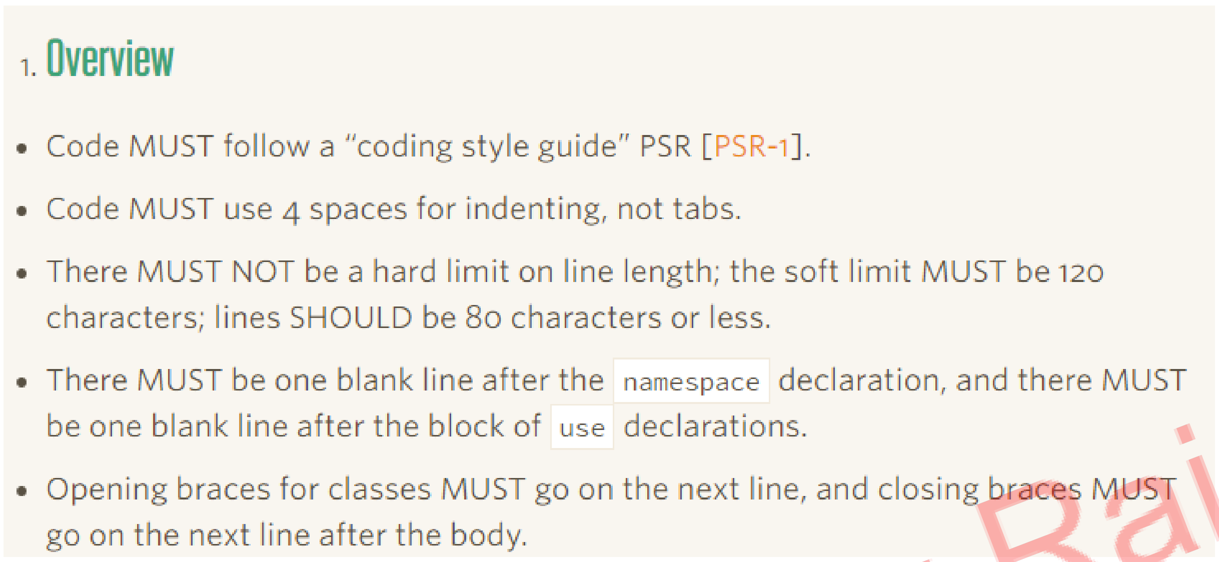
There are some benefits to using this model in my project, including; phases are completed one a time – such as research, design, development and testing, this is useful because my requirements were shown to me at the beginning of the project and I was able to follow them one by one using the V model.

3.2.4 Application Architecture

Application architecture is very important in web development and it varies depending on the type of application. It is fundamental in keeping a neat coding style so it is clear to understand and easy to maintain. As said above the Laravel framework will be used for backend development which encourages the use of a MVC architecture, however it does not force you into this scheme as the application will still work if, for example, code that should be placed in the model is placed into the controller instead.

3.2.4.1 Code Styling

Using a consistent coding style is essential in large applications especially when it comes to testing or modifying certain areas. It improves the overall readability of the code therefore if another developer came in part way to inspect, it would be easy to understand what is happening. For these reason, certain PHP coding standards will be used – ‘de-1’ and ‘PSR-2’. PSR-2 is essentially an extension of PSR-1, the guide includes:

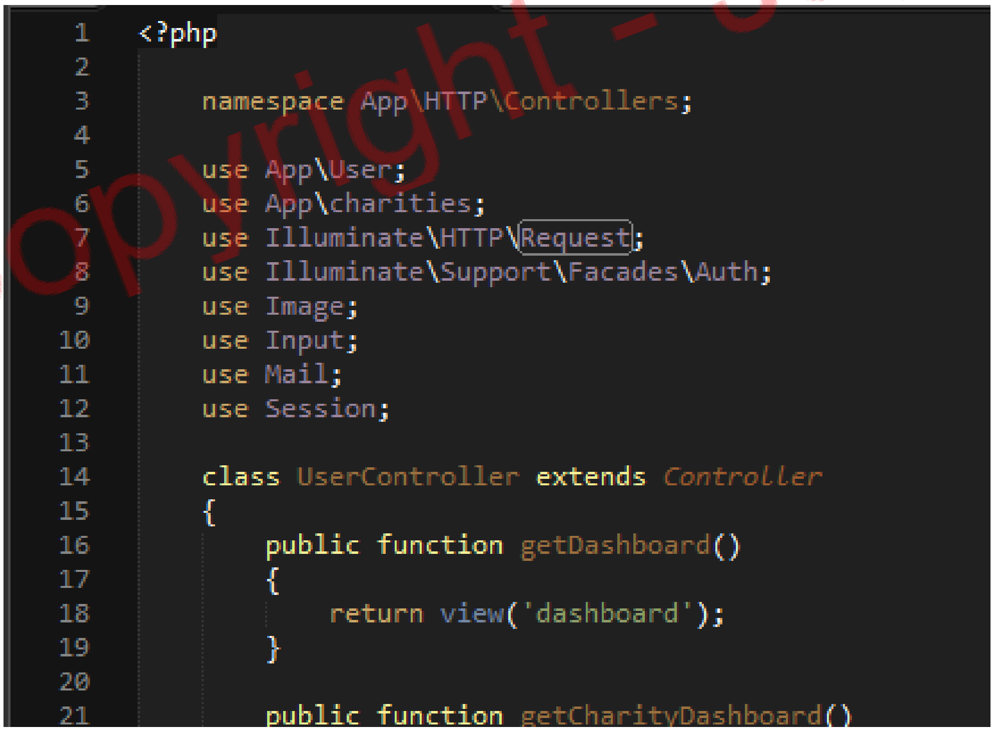


1. Overview

- Code MUST follow a "coding style guide" PSR [PSR-1].
- Code MUST use 4 spaces for indenting, not tabs.
- There MUST NOT be a hard limit on line length; the soft limit MUST be 120 characters; lines SHOULD be 80 characters or less.
- There MUST be one blank line after the `namespace` declaration, and there MUST be one blank line after the block of `use` declarations.
- Opening braces for classes MUST go on the next line, and closing braces MUST go on the next line after the body.

[5] *Figure 27: PSR-2 Code Styling Overview.*

I have followed this style of coding throughout my application, as shown below:



```php
<?php

    namespace App\HTTP\Controllers;

    use App\User;
    use App\charities;
    use Illuminate\HTTP\Request;
    use Illuminate\Support\Facades\Auth;
    use Image;
    use Input;
    use Mail;
    use Session;

    class UserController extends Controller
    {
        public function getDashboard()
        {
            return view('dashboard');
        }

        public function getCharityDashboard()
```

Figure 28: Example block code from 'UserController' that follows the PSR2 guideline.

3.2.4.2 Version Control

Version control is a type of a system that records changes to some files so the programmer is able to recall specific versions later. This is important in this project as the application is quite large, therefore a mistake can be costly to other parts of the application, and version control will allow files to be reverted back to their original state. For this reason, Git version control was used throughout in order to track changes in the project.

3.2.4.3 S.O.L.I.D Design Principles

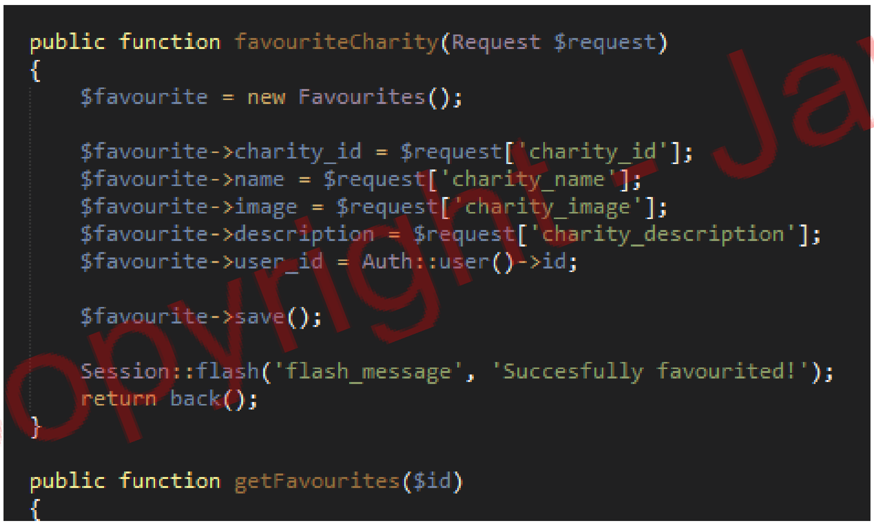
S.O.L.I.D stands for five principles – Single responsibility, open closed, Liskov substitution, interface segregation and dependency inversion – respectively. These five principles relate to the design of the finished application, design is important due to quality and it is the only way to effectively and accurately translate requirements into a finished software product. These principles combined together should encourage the development of a maintainable and understandable application.

3.2.4.3.1 Single Responsibility

"*A class should have one and only one reason to change.*" [6]

In the single responsibility principle, each responsibility should be in a separate class because each responsibility is an axis of change. Following this technique in the development of this application will make maintaining and debugging it much easier.

Within my project, a practical example would be that within the User model, there should only be functions related to the user. This responsibility can also be applied to functions, such as, a function that stores the charity as a favourite for the user should only do that. The output of their favourites would be done in another function.



```
public function favouriteCharity(Request $request)
{
    $favourite = new Favourites();

    $favourite->charity_id = $request['charity_id'];
    $favourite->name = $request['charity_name'];
    $favourite->image = $request['charity_image'];
    $favourite->description = $request['charity_description'];
    $favourite->user_id = Auth::user()->id;

    $favourite->save();

    Session::flash('flash_message', 'Succesfully favourited!');
    return back();
}

public function getFavourites($id)
{
```

Figure 29: favouriteCharity function.

In figure 29, shows the single responsibility principle in action. This function strictly only stores the charities information into the corresponding table in the database when the user clicks on 'Favourite'.

3.2.4.3.2 Open Closed

"Software entities (classes, modules, functions) should be open for extension, but closed for modification." [7]

Essentially this means that a class should be easily extendable without modifying the class itself. Therefore it prevents breakages and code rot caused by modifying existing code which forces planning ahead for extension.

In simple terms an example would be having a method that sums up the areas of more than one shape, therefore an if statement would be implemented in order to check what shape it is. This can be resolved by attaching the logic to calculate the area of the shape's, and then create an interface in order to check whether the object passed is actually a shape.

3.2.4.3.3 Liskov

"Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program."[8]

This principle can be thought of as an extension of the open closed principle and therefore means that any new derived classes are extending the base classes without any change in behaviour.

“Subtypes must be substitutable for their base types.”[9]

3.2.4.3.4 Interface Segregation

This means that a client should never be forced to implement an interface that it doesn’t use or clients should not be forced to depend on methods they don’t use. This essentially means that if you are implementing interfaces and the implementation does not implement all methods of that interface, this could mean that the interface is bloated and may need redesigning.

An example may be, if in a workplace there is a manager that sets work to the workers. There is a standard worker that works and takes a lunch break, and then there is a robot that works but does not need to take a lunch break. Originally there would be one interface that the worker implements that contains a work and eat method, if the robot implements this then there would be an unused method (eat), which violates this principle. Therefore, the robot class should implement its own work interface that does not contain an eat method.

It is better to use an interface containing only one method rather than writing useless code in order to satisfy the implementation.

3.2.4.3.5 Dependency Inversion

“Depend upon abstractions. Do not depend upon concrete classes.”[10]

Basically this means that high level modules should not depend upon low-level modules, both should depend on abstractions. Abstractions should never depend upon details, details should depend upon abstractions.

This is important because changes are very risky, therefore depending on a concept instead of an implementation, it reduces the need for change at call sites.

An example in this project is how every model extends the ‘Eloquent’ class, which essentially allows the models to have no knowledge of how they actually interact with the database – this can allow swapping of the database without any changes to the actual models.

3.2.4.4 Namespacing

Namespacing is a relatively new feature in Laravel, it allows classes to be of the same name. This is important because we may use third party libraries that contain, for example, a ‘User’ class. By using a namespace at the top of a class, it means that everything that is happening in that class is relative to that namespace, also, any classes created within this file will live inside that namespace. Then in order to instantiate a class within a namespace, you can prefix it with the name of the namespace.

3.2.4.5 External Libraries

3.2.4.5.1 Mailtrap & Gmail

MailTrap was chosen as the email service instead of a traditional email service provider because, even though it being fake SMTP server which is used when testing applications; it also allows all emails to be viewed and then forward them to a real email address if necessary.

I used this mail service because it is easy to setup as there is no need to tune my mail server, no need to clean up my database from the users' email addresses. Also the fact that it is platform independent so it can be integrated into any programming language and framework.

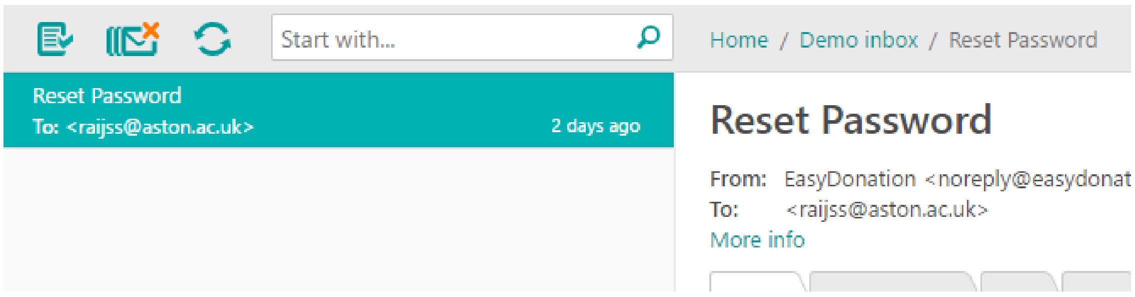
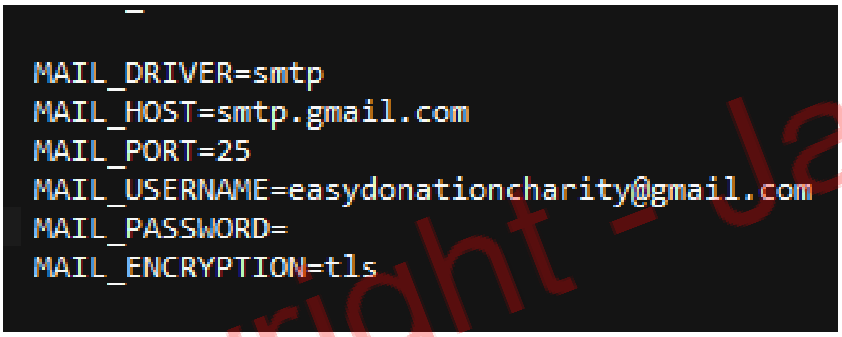


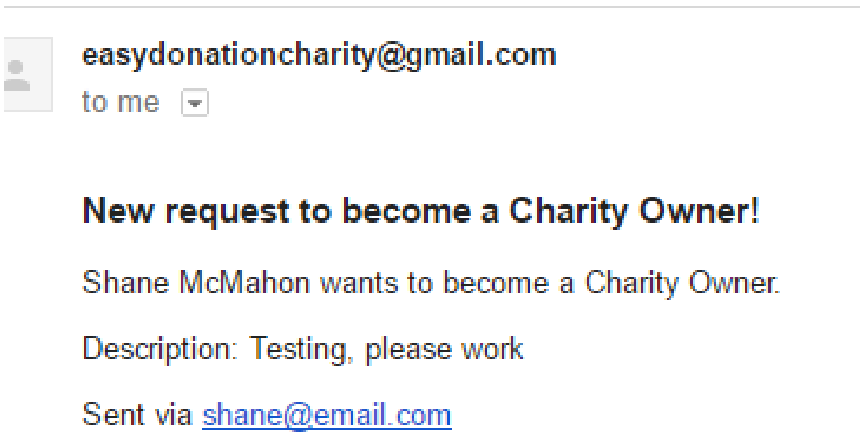
Figure 30 Mailtrap[11]



```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=25
MAIL_USERNAME=easydonationcharity@gmail.com
MAIL_PASSWORD=
MAIL_ENCRYPTION=tls
```

Figure 31 .env file.

The environment file contains required Gmail account information, so users are able to request to become a user or enable password resets.



easydonationcharity@gmail.com
to me

New request to become a Charity Owner!

Shane McMahon wants to become a Charity Owner.

Description: Testing, please work

Sent via shane@email.com

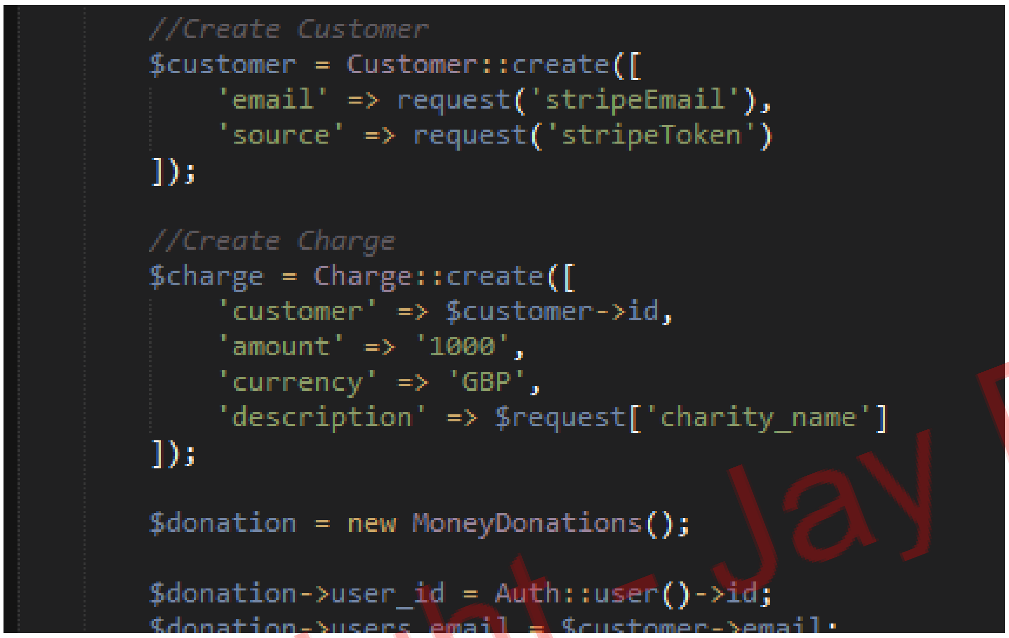
Figure 32 Example of mail.

Figure 32 shows a request being made to become a charity owner via a form on the application, this form is sent directly to easydonationcharity@gmail.com with the form inputs. For this work, the Gmail account has to allow access to less secure applications, as Localhost is not secure.

3.2.4.5.2 Stripe Payment

The Stripe payment package was chosen over PayPal as there was an error on the PayPal servers, which would not allow me to make an API app in time for this project. Stripe also has very well written documentation making it easier to understand and therefore easier to use. Stripe is also becoming very popular worldwide and has many test cards that can be used.

Stripe is a very secure payment package because of the use of 'Stripe.JS', using this on the application means that credit card data entered by the user is never sent to the server, instead it is sent directly to Stripe. Thus, if there is a breach in the servers then no credit card data will be stolen.



```
//Create Customer
$customer = Customer::create([
    'email' => request('stripeEmail'),
    'source' => request('stripeToken')
]);

//Create Charge
$charge = Charge::create([
    'customer' => $customer->id,
    'amount' => '1000',
    'currency' => 'GBP',
    'description' => $request['charity_name']
]);

$donation = new MoneyDonations();

$donation->user_id = Auth::user()->id;
$donation->users_email = $customer->email;
```

Figure 33: Stripe implementation..

Figure 33 shows some of the implementation for the Stripe payment. Essentially, it creates a new customer where the current user logged in is passed through. This is then stored to the Stripe Dashboard through the API key, then this charge is saved into the database using the '$charge' array.

Due to limitations of using this library, the user is unable to specify the amount they wish to donate, and is hardcoded to £10.

The modal in figure 34 shows the user interface of paying with Stripe, test cards are used which can be found in the Stripe documentation [12].

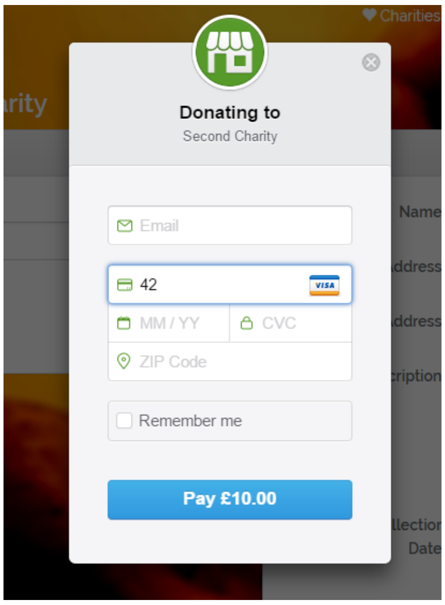
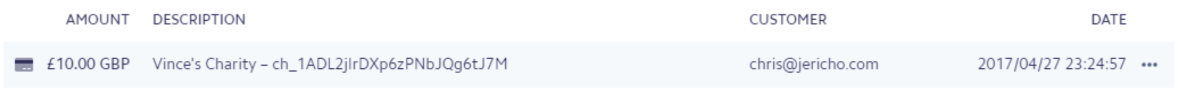


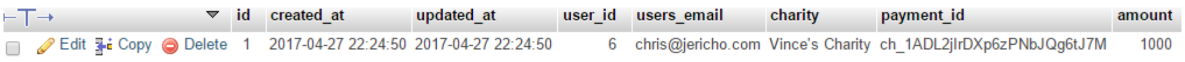
Figure 34 Stripe Payment Modal.

When a user inputs valid information, the payment is sent to the Stripe server and stored on the applications dashboard as well as the applications database, as shown in figures 35 and 36.



AMOUNT	DESCRIPTION	CUSTOMER	DATE
£10.00 GBP	Vince's Charity – ch_1ADL2jlrDXp6zPNbJQg6tJ7M	chris@jericho.com	2017/04/27 23:24:57

Figure 35 Stripe Dashboard payment.



	id	created_at	updated_at	user_id	users_email	charity	payment_id	amount
Edit Copy Delete	1	2017-04-27 22:24:50	2017-04-27 22:24:50	6	chris@jericho.com	Vince's Charity	ch_1ADL2jIrDXp6zPNbJQg6tJ7M	1000

Figure 36 Database payment stored.

3.2.4.5.3 Twitter Integration

A Twitter page for the application was made, which can be accessed through the homepage as shown in figures 37 and 38. This is available to all that visit the system even if they are not registered yet.

Figure 37 Twitter Link

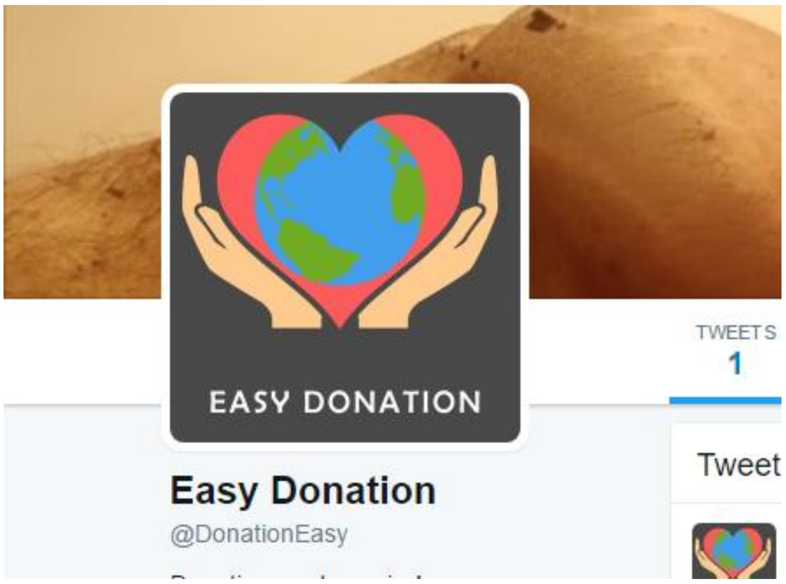
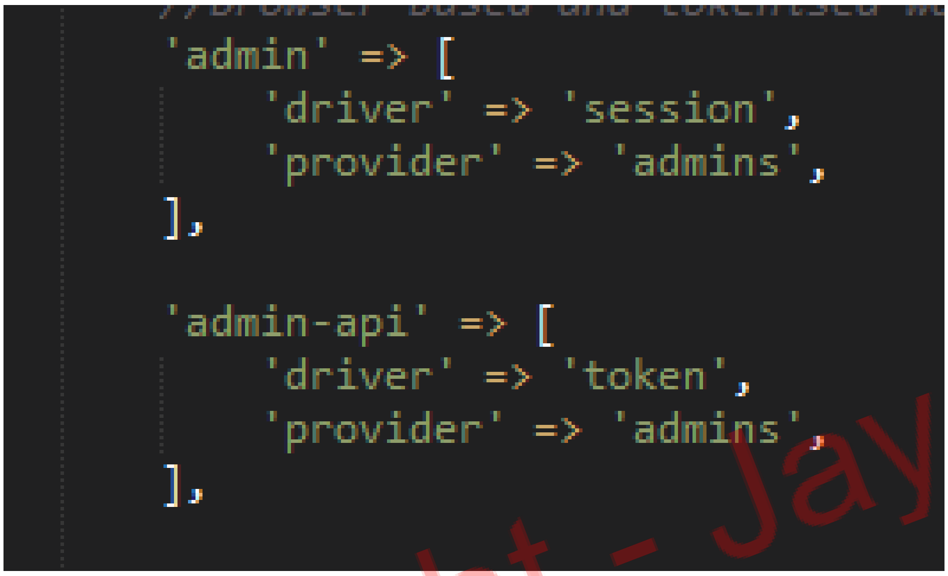


Figure 38 Twitter Page.

3.3 The Application

The web application produced allows users to register and log in, within a secure environment. The application does not allow users to see content that they shouldn't otherwise. If a user tries to directly access the administrator's page, they will be automatically sent to their own log in page.

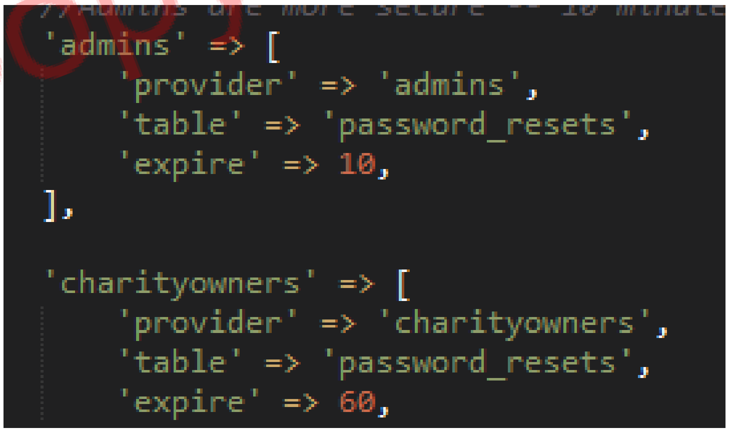
There are three types of users, the administrator, the charity owner and the regular user. Each of these users have different roles and see different views. Using the framework Laravel, the authentication was implemented using guards, where the default is the normal user, thus administrator and charity owner guards were setup respectively.



```
'admin' => [
    'driver' => 'session',
    'provider' => 'admins',
],

'admin-api' => [
    'driver' => 'token',
    'provider' => 'admins',
],
```

Figure 39: Shows the implementation of the admin guard.

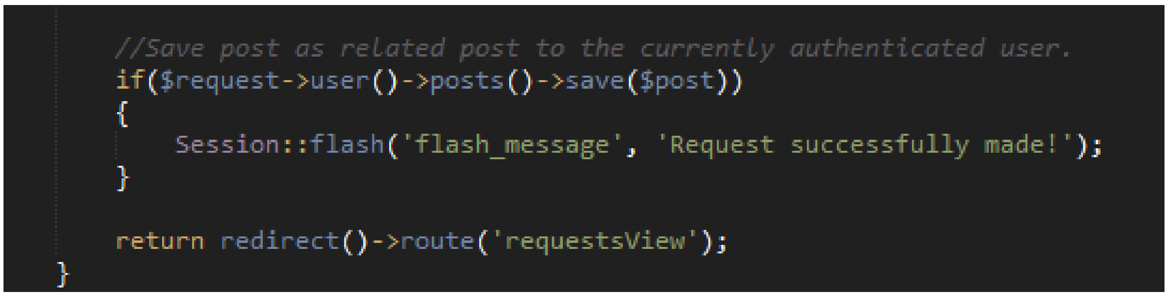
There are 2 ways for the users to be authenticated, either browser based (where sessions are used as the driver) or they can be tokenised (where tokens are used).



```
'admins' => [
    'provider' => 'admins',
    'table' => 'password_resets',
    'expire' => 10,
],

'charityowners' => [
    'provider' => 'charityowners',
    'table' => 'password_resets',
    'expire' => 60,
```

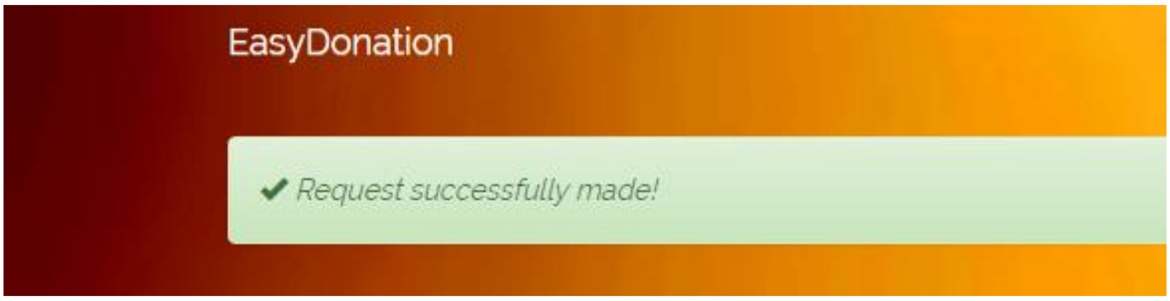
Figure 40: Password resets.

Figure 40 shows the implementation for resetting passwords in the “auth.php”. Admins are given 10 minutes before the reset token is considered invalid, whereas the regular users and charity owners are given an hour. This is to improve security for the administrators.



```
    //Save post as related post to the currently authenticated user.
    if($request->user()->posts()->save($post))
    {
        Session::flash('flash_message', 'Request successfully made!');
    }

    return redirect()->route('requestsView');
}
```

Figure 41: Flash Messages.


EasyDonation

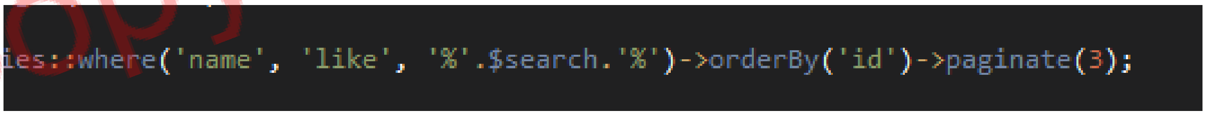
✔ Request successfully made!

Figure 42: Flash message appearance.

Figures 41 and 42 show the implementation and appearance of the flash messages, which are fully responsive. In this example, a request has been added by the user successfully and therefore this message is displayed as the user is directed back to the previous page. These flash messages also, disappear after around 5 seconds so the user does not need to refresh the screen.

Within the controllers of the application, it was easy to test whether functions were submitted and tell the user whether they are successful or not, this is nice for the user because they know what has happened to their request.

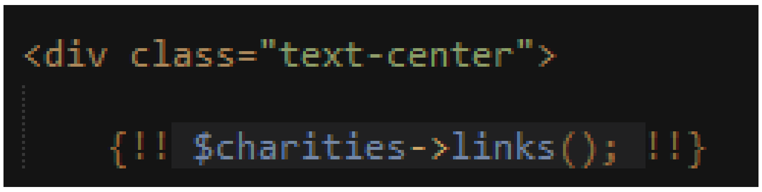
Users are able to donate money securely through Stripe, or fill in a form to make a collection. Charity owners can manage their donations, as well as, add new charities, delete existing ones, and even accept requests made by users who are then notified. Users are able to add favourites, to charities that they donate to regularly – this is in case the database becomes over populated with charities, however there is a search feature on the application. Charities also show 3 per page – to avoid scrolling – this is done via Laravel's pagination method, as shown in figure 43.



```
ies::where('name', 'like', '%'.$search.'%')->orderBy('id')->paginate(3);
```

Figure 43 paginate method.

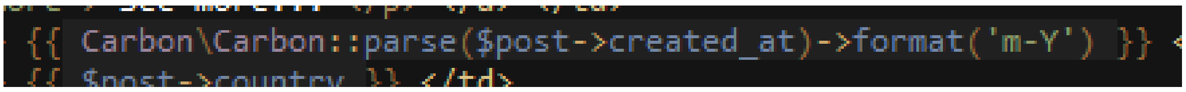
This then uses a links method on the view, figure 44.



```
<div class="text-center">
    {!! $charities->links(); !!}
```

Figure 44 links method

There are several places in the application where dates are used, in the database they are stored in the format: YYYY-MM-DD 00:00:00. The user does not need to see all this on the application, therefore Laravel's Carbon method comes into place, as shown from figure 45.



```
{{ Carbon\Carbon::parse($post->created_at)->format('m-Y') }}
{{ $post->country }} </td>
```

Figure 45 Carbon method to change the date.

Chapter 4: Evaluation

4.1 Testing

Testing is a very big and important part of this project, which is why it is allocated over a few weeks (as shown from the Gantt chart).

In order to develop an effective test; the system must be understood in detail (having implemented it, I do), application and solution domain knowledge, know what testing techniques to integrate and the skills to apply these techniques. From the project definition form, a number of different tests will be implemented; usability tests, functionality tests, performance, interface/compatibility, and security tests.

The system will be tested first, in order to test the complex functionality. However there will be independent testers as well. It would be bad unsatisfactory if I, as the developer to test it because; of mental attitude that the program will behave in a certain way even though it may not, sticking to data set that makes the programs work, for example, only performing the main tasks when in fact naturally users may make mistakes therefore the application needs to react accordingly, as a program may often not work when tried by somebody else.

When testing, a number of black box and white box testing methods were implemented. Black box testing is a method where the internal structure/design/implementation of the item being tested is not known. White box testing is essentially where the items internal structure is known. Some black box testing methods are; acceptance and system testing. A white box testing that I will be implementing is unit testing.

4.1.1 Unit Testing

Unit testing is a testing technique where individual modules are tested in order to determine if there are any issues, this is performed by me – as the developer. The main aim is to isolate each unit of the application in order to identify, analyse and fix the defects.

Below shows the traditional cycle I will be taking.

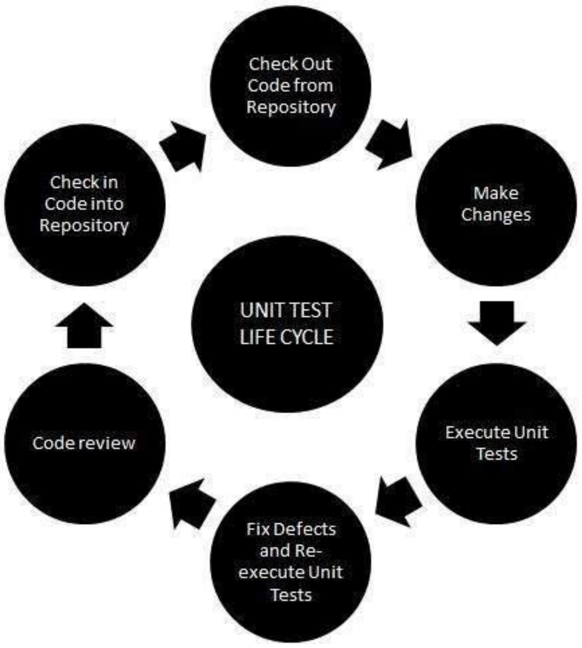

Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Unit Tests
Code review
Check in Code into Repository
UNIT TEST LIFE CYCLE

Figure 46: Unit test life cycle [13]

4.1.2 User Testing

4.1.2.1 Usability Tests

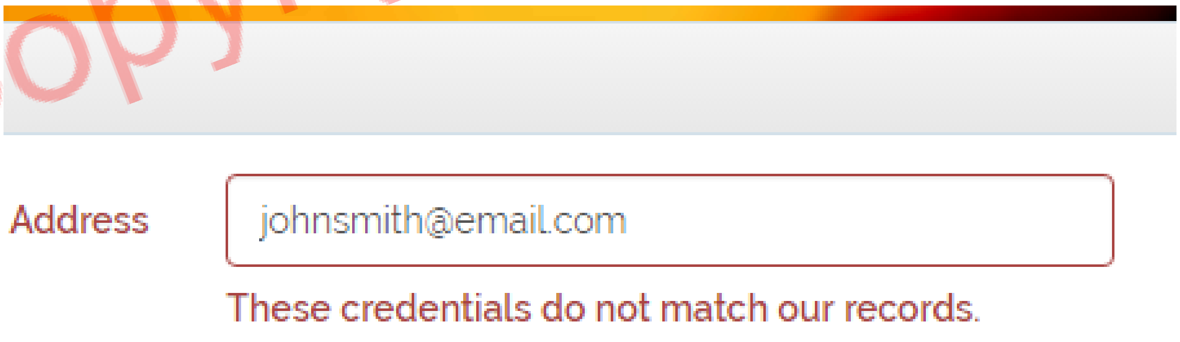
Usability testing is a non-functional testing technique that measures how easily a user can essentially use the system, it can be measured using the following: the level of skill of the user, it should maintain the balance for both novice and expert, and the overall assessment of users' attitude towards using the application.

This test will be performed before implementation, after the initial designs are drawn up, they will also be done after implementation when the final designs are complete. This is an effective test because if the application is not easy to use, then it doesn't matter how complex the functionality is, the application will be useless.

4.1.2.2 Summary of Findings

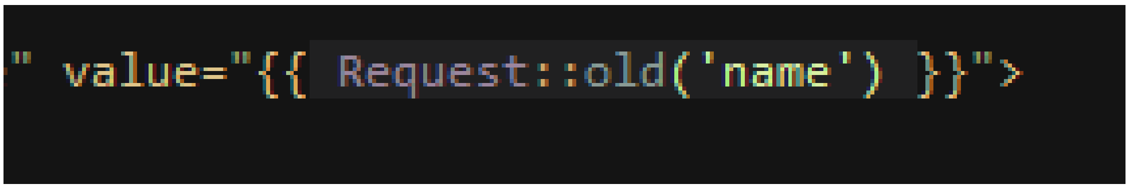
When carrying out tests on the application, there were one major problems found. Also, a number of small issues were raised.

One participant said that when they make a mistake when signing up, it was frustrating having to retype all the information back into the fields and then finally sign up. Therefore the application was changed so user's credentials even if wrong would be remembered for better user experience, as shown from figures 47 and 48 below, it was simple fix by just requesting what they user typed.



These credentials do not match our records.

Figure 47: Screenshot of Application showing it remembering the users credentials even when thrown an error.



```
" value="{{ Request::old('name') }}">
```

Figure 48: Added code for this feature.

Another participant accidently deleted a charity that they had created in the application, when they clicked the delete button it removed the charity straight away. Therefore from figure 49, a pop over was implemented so there is an option for the user in case they accidently clicked it and didn't want to confirm it.

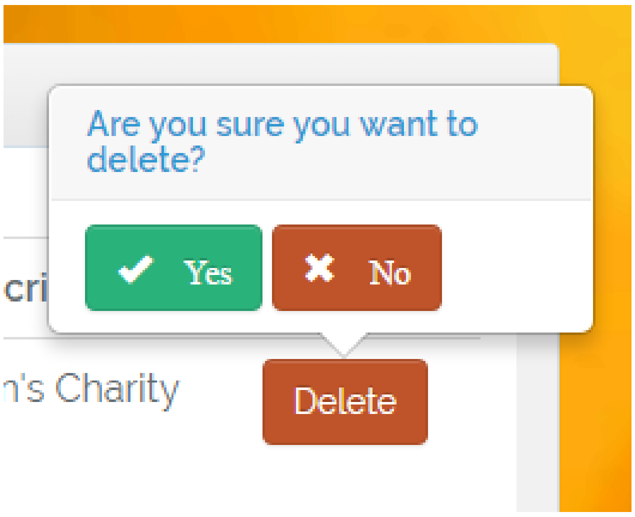


Figure 49:'Are you sure you want to delete option?'

Another issue that was raised, was that a person visiting the application should not be expected to register straight away as they don't know what the website is about. Figure 50 shows the original index page that the user was shown.

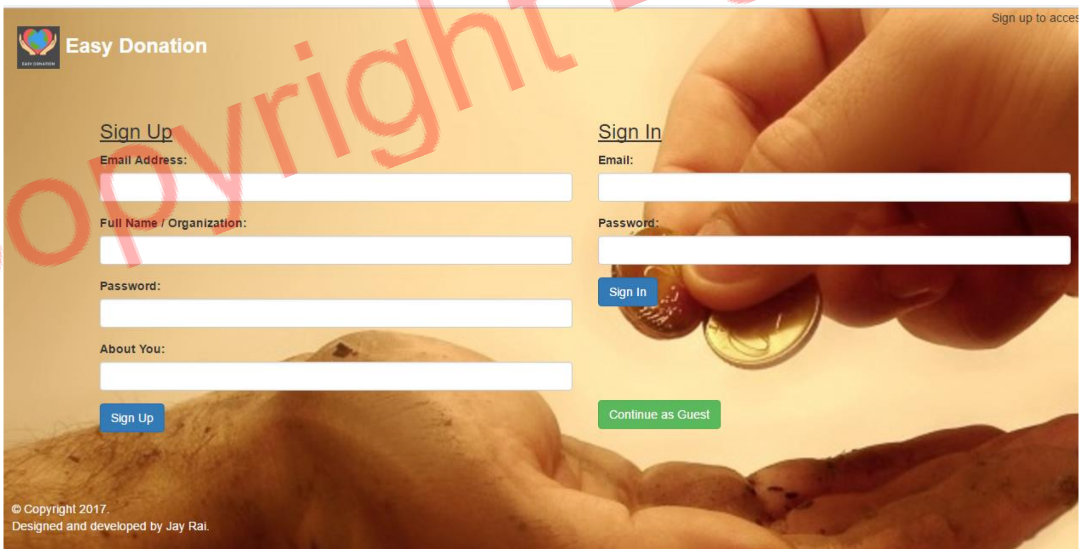


Figure 50: Original welcome screen.

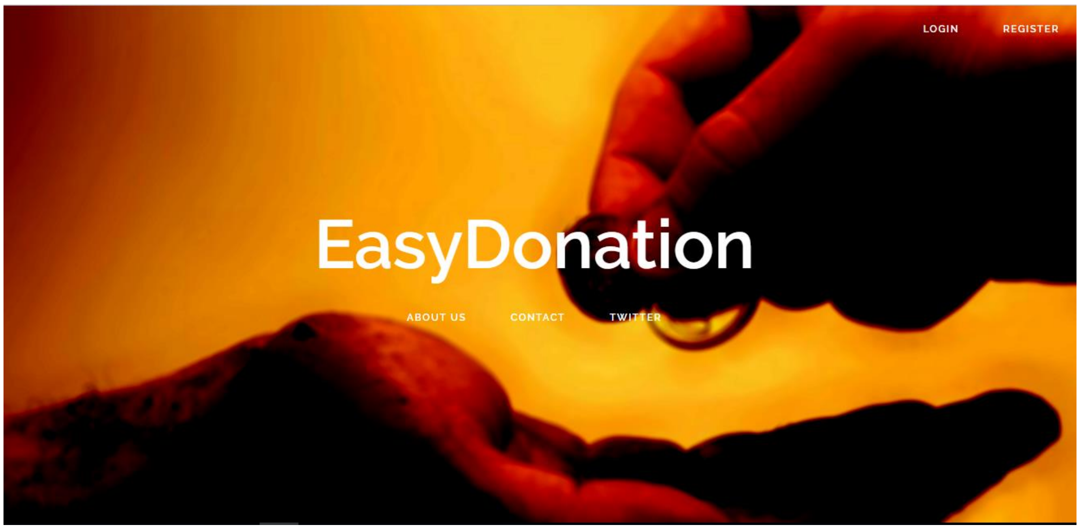


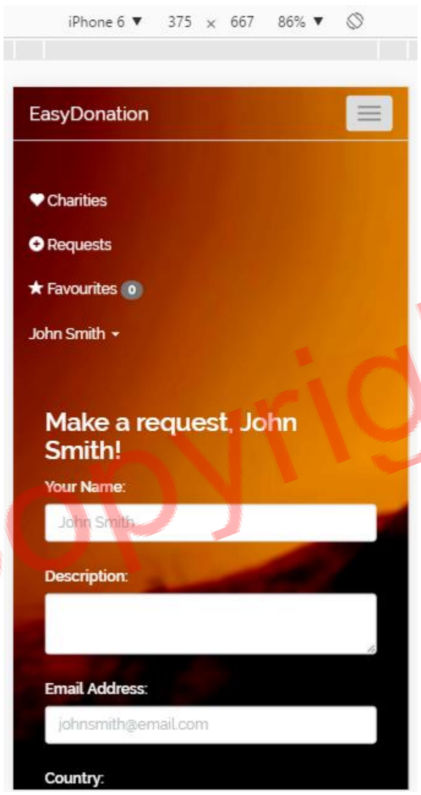
Figure 51: New welcome screen.

Figure 51 shows the new welcome screen, where the visitor has an option to register for an account at the top right, however, the user can now find out about the application through the about link in the centre as well see the applications social network profile.

4.1.2.3 Compatibility Tests

This is a non-functional test that tests what the application is like in different environments, such as browsers and devices.

Within the inspector of Chrome, I am able to test the application on different devices and how responsive it is.



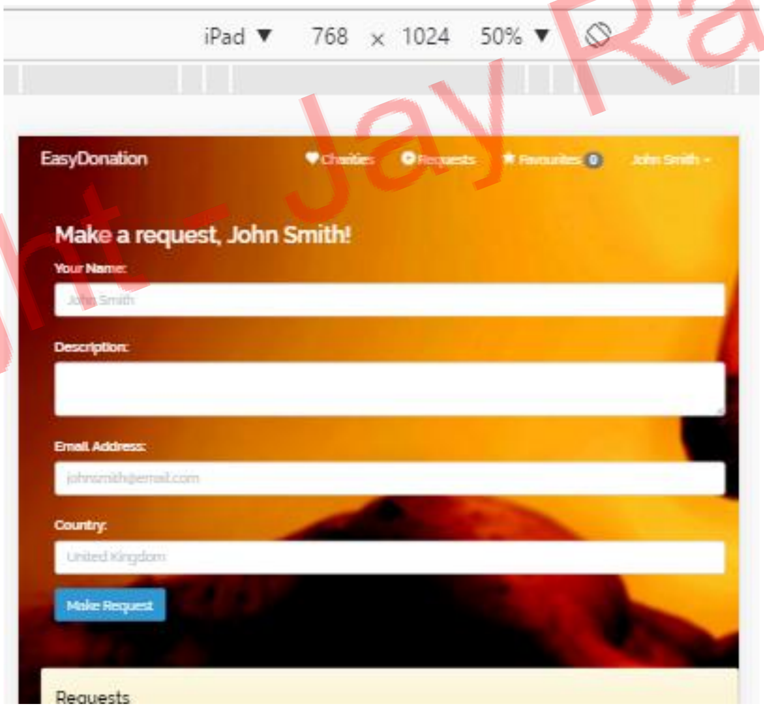
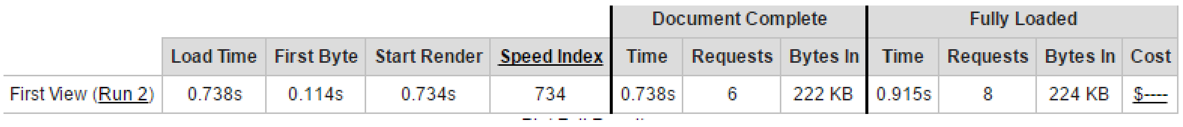


Figure 52 iPhone Layout *Figure 53 iPad Layout*

As shown the application is fully responsive, the navigation bar minimises itself when it is shown on a smaller screen, which can be opened by clicking the icon. The form does not lose its position and users do not have to scroll horizontally when the screen gets smaller, the application moves the components down.

4.1.2.4 Performance Testing

To test the performance of EasyDonation I used an application called 'WebPageTest'[15]. The average ideal time for a web page to be loaded is less than 1.5 seconds, from figure 54, EasyDonation took 0.915 seconds to load everything on the homepage which is fast.



					Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 2)	0.738s	0.114s	0.734s	734	0.738s	6	222 KB	0.915s	8	224 KB	$----

Figure 54 Performance results for homepage

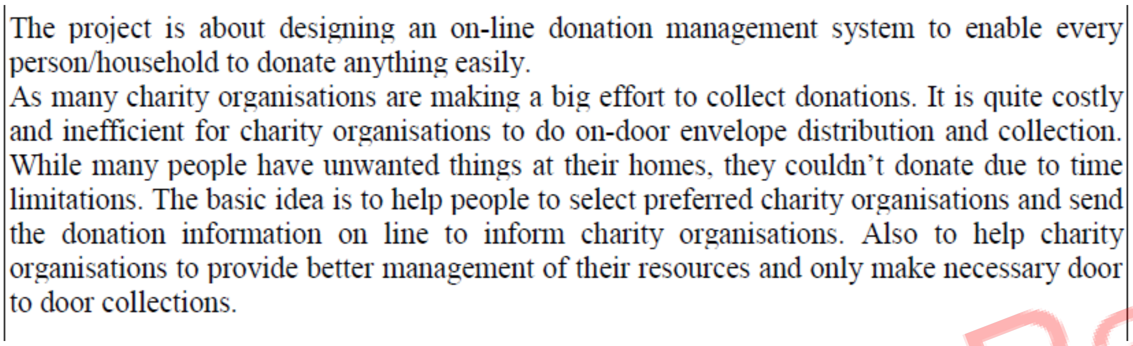
4.1.2.5 Acceptance Testing

For acceptance testing, I looked back over the requirements set, and see if they were achieved or not; some may have changed to meeting decisions.

Task	Achieved	Comments
The application should allow users to sign up/ create account.	Yes	Using Laravel Auth makes it secure.
The application should allow users to delete their account.	No	Wasn't need, however admins can delete users.
The application should allow users to modify their account/data.	Yes	Users can change picture.
The application should present the user with a forget password option.	Yes	Sends a reset email
The application should allow the user to see their donation history.	Yes	
The application should allow the user to see their favourites list.	Yes	
The user should be able to search for available charities.	Yes	
The user should be presented with information about the selected charity.	No	Users are able to see a small description/
The user should be able donate items or money.	Yes	
The user should be able to favourite chosen charities.	Yes	
The user should be able to delete charities from their favourites list.	Yes	
The user should be able to make a request to the other charities.	Yes	
The application should run smoothly across all platforms (Internet Explorer, Chrome, Safari, Firefox, Opera etc.	Yes	
The page load time should be kept to a minimum.	Yes	Retrieving ID's enhances retrieving speed.
The GUI should be easy to use, with a well-designed clean interface.	Yes	UI is not cluttered, positive comments from test users.
The application could be functional on a mobile device.	Yes	It has been tested on a mobile.

Chapter 5: Conclusion

The aim of this project was as follows:



The project is about designing an on-line donation management system to enable every person/household to donate anything easily.
As many charity organisations are making a big effort to collect donations. It is quite costly and inefficient for charity organisations to do on-door envelope distribution and collection. While many people have unwanted things at their homes, they couldn't donate due to time limitations. The basic idea is to help people to select preferred charity organisations and send the donation information on line to inform charity organisations. Also to help charity organisations to provide better management of their resources and only make necessary door to door collections.

I believe this web application fulfils this aim. There are two main users for this application, a regular user and a charity owner. The user is able to donate items and money to selected charities, make requests that the charities can see. Charity owners can accept requests, manage donations, and add their charities.

There are some differences, such as, for the future, this application would pass this information onto the charities, and it would be implemented with live payments not in a test environment.

This application was designed and developed using Laravel 5, HTML/CSS and JavaScript. Laravel was a new framework to me, therefore it took me longer than expected to create an application. Reading through all the documentation, I got a better understanding of how the framework should be used and the available libraries that go with it. For the design I used Laravel's own template called 'Blade', allowing me to use sections and yields to display content. [14].

For payments, the application would not use the test environment in real life however for the purposes of this project, test cards are used to make payments. A limitation to this application is that the user cannot specify the amount they want to donate (it is hardcoded at £10), this is due to time. Originally the application was supposed to take PayPal payments however due to problems with their server, PayPal would not allow me to make an API app. Therefore, I switched to Stripe quite late on in the project and was not able to implement a way for the user to input an amount. However, these payments are stored on Stripe's server as well as in the applications database.

References

[1] British Heart Foundation, (ONLINE). Available at: https://www.bhf.org.uk/?gclid=CLOi8LaA-88CFfYV0wodu-kG2w Accessed on: 10/16.

[2] American Heart Association, (ONLINE). Available at: http://www.heart.org/HEARTORG/ Accessed on: 08/10/16.

[3] Next Gen Climate, (ONLINE). Available at: https://nextgenclimate.org/ Accessed on: 09/10/16.

[4] Just Giving, (ONLINE). Available at: https://www.justgiving.com/ Accessed on: 09/10/16.

[5] PHP-FIG, (ONLINE). Available at: http://www.php-fig.org/psr/psr-2/ Accessed on: 14/01/17.

[6] (BOOK) Robert C. Martin. (2002) Agile Software Development, Principles, Patterns, and Practices.

[7] (BOOK) Bertrand Meyer, Prentice Hall. (1988) Object Orientated Software Construction. Page 23.

[8] (BOOK) Kelt Dockins. (2016) Design Patterns in PHP and Laravel. Page 12.

[9] (BOOK) Robert C. Martin. (2002) Agile Software Development, Principles, Patterns, and Practices.

[10] (BOOK) Elisabeth Robson, Bert Bates, Kathy Sierra (2004) Head First Design Patterns. Page 139.

[11] Mailtrap, (ONLINE). Available at: https://mailtrap.io/inboxes Accessed on: 28/03/17.

[12] Stripe Documentation, (ONLINE). Available at: https://stripe.com/docs Accessed on: 22/03/17.

[13] TutorialsPoint, (2017), (ONLINE). Available at: https://www.tutorialspoint.com/software_testing_dictionary/unit_testing.htm Accessed on: 08/03/17.

[14] Laravel Blade Documentation (2017), (ONLINE). Available at: https://laravel.com/docs/5.4/blade#introduction

[15] WebPageTest (2017), (ONLINE). Available at: https://www.webpagetest.org/

Bibliography

1. https://laravel.com/docs/5.4 Laravel Documentation, 2016/17.

2. Acorns Shop. Great Barr. Questionnaire response. 2016.

3. http://api.jquery.com/ JQuery API Documentation, 2017.

4. https://laracasts.com/ Laracasts Forums, 2017.

5. Ken, L 2003, *Software Development with UML*, Palgrave MacMillan, New York.

Appendices

Project Diary

Minutes (Examples)

DATE:	Thursday, September 29, 2016
TIME:	14:45
LOCATION:	Office 214H

MEETING / PROJECT NAME	Final Year Project #1
MINUTES PREPARED BY:	Jay Rai

1. MEETING NOTES

Introduce the final year project.

Discuss aims and objectives of project, and how best to manage time.

2. ATTENDEES PRESENT

NAME	DEPARTMENT	EMAIL	PHONE
Jay Rai	-	-	-
Dr Hongxia (Helen) Wang	Computer Science	-	-

4. ACTION ITEMS

ACTION	ACTION TO BE TAKEN BY	TO BE ACTIONED BY
Make a timetable	Jay Rai	Next meeting

5. NEXT MEETING

DATE	Thurs, October 6, 2016	TIME	14:00 PM	LOCATION	Office 214H
OBJECTIVE	Progress update				

SUBMITTED	Jay Rai

DATE:	Thursday, October 06, 2016
TIME:	14:00
LOCATION:	Office 214H

MEETING / PROJECT NAME	Final Year Project
MINUTES PREPARED BY:	Jay Rai

1. MEETING NOTES

Discuss Project Definition Form.

Discuss aims and objectives of project, and how best to manage time.

2. ATTENDEES PRESENT

NAME	DEPARTMENT	EMAIL	PHONE
Jay Rai	-	-	-
Dr Hongxia (Helen) Wang	Computer Science	-	-

4. ACTION ITEMS

ACTION	ACTION TO BE TAKEN BY	TO BE ACTIONED BY
Complete Project Definition Form	Jay Rai	Next meeting

5. NEXT MEETING

DATE	Thurs, October 6, 2016	TIME	14:00 PM	LOCATION	Office 214H
OBJECTIVE	Progress update and check project definition form.				

SUBMITTED	Jay Rai

DATE:	Friday, February 17 2017
TIME:	14:30
LOCATION:	Office 214H

MEETING / PROJECT NAME	Final Year Project
MINUTES PREPARED BY:	Jay Rai

1. MEETING NOTES

Show all development made.

Add some authentication for different users.

Clean up the UI.

2. ATTENDEES PRESENT

NAME	DEPARTMENT	EMAIL	PHONE
Jay Rai	-	-	-
Dr Hongxia (Helen) Wang	Computer Science	-	-

4. ACTION ITEMS

ACTION	ACTION TO BE TAKEN BY	TO BE ACTIONED BY
Add some authentication the application.	Jay Rai	Next meeting
Make the UI cleaner, simpler for the user.	Jay Rai	Next meeting
Start new functionality.	Jay Rai	Next meeting

5. NEXT MEETING

DATE	-	TIME	-	LOCATION	-
OBJECTIVE	Show the development and design.				

SUBMITTED Jay Rai

DATE:	Friday, April 07 2017
TIME:	14:45
LOCATION:	Office 214H
MEETING / PROJECT NAME	Final Year Project
MINUTES PREPARED BY:	Jay Rai

1. MEETING NOTES

Add more functionality for the different users.

Add more the deliverable section of the report.

2. ATTENDEES PRESENT

NAME	DEPARTMENT	EMAIL	PHONE
Jay Rai	-	-	-
Dr Hongxia (Helen) Wang	Computer Science	-	-

4. ACTION ITEMS

ACTION	ACTION TO BE TAKEN BY	TO BE ACTIONED BY
Add more functionality	Jay Rai	Next meeting
Users able to donate items, not just money.	Jay Rai	Next meeting
Users can make requests.	Jay Rai	Next meeting

5. NEXT MEETING

DATE	-	TIME	-	LOCATION	-
OBJECTIVE	Show a complete application before submission.				

SUBMITTED Jay Rai

Project Definition Tasks

Tasks	Description
Drafted	The project definition form drafted on 29/09/16 as a plan for the structure of the rest of the project.
Discussed	This form discussed what the project was about, the deliverable, what is original about the project, and the main stages of the work plan – including research, design, prototyping, development and testing. After suggestions from my supervisor, I detailed each key area, such as, the types of software's I will be using, how I will design the application etc. I also took out the end date for development

	and start/end date for testing. This is because they will vary a lot when I am actually doing the implementation and testing.
Completed	I completed the final draft of the project definition form on 14/10/16, and it was signed off on 20/10/16.
Submitted	This was submitted on the 21/10/16.

Term 1 Progress Report Tasks

Tasks	Description
Discussed	I met with my supervisor on 19/01/16 to discuss the term 1 progress report before submission.
Feedback points	I gathered some feedback on the structure of my Gantt chart, as I may have left too much of a big gap for the development and not enough for the testing however this can change when implementing. The relevant sources and packages were discussed also, what language I will be using for the application. Payment packages were also discussed, this is not a necessary feature, this can be further discussed in the report on the limitations I had.
Decisions	Decisions made

Major Stages of Work

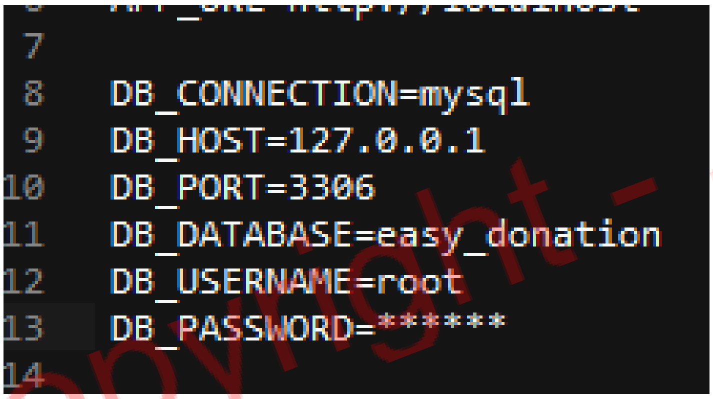
Task	Date	Completed
Primary Research	26/09/16	03/10/16
Secondary Research	28/09/16	10/10/16
Design – Wireframe	15/10/16	22/10/16
Design – Prototypes	05/11/16	21/11/16
Design – Interface	25/11/16	05/12/16
Learning Framework (Laravel 5)	29/11/16	Present
Implementation – Iteration 1 (Register and Login / Security)	18/12/16	23/12/16
Implementation – Iteration 2 (User functionality)	09/01/17	19/01/17
Implementation – Iteration 3 (Charity Owner functionality)	23/01/17	06/02/17
Implementation – Iteration 4 (Admin functionality)	23/02/17	25/02/17
Implementation – Iteration 5 (Cleaning up front end)	08/03/17	30/03/17
Testing – User Tests	01/04/17	07/04/17
Testing – Performance Tests	09/04/17	13/04/17
Testing – Summary of findings	14/04/17	20/04/17

Implementation – Iteration 6 (Changing application due to findings from the tests)	21/04/17	26/04/17
Report (Finalising)	22/04/17	27/04/17

User Instructions

The application can be viewed at: https://app-1493393491.000webhostapp.com/

However to use locally, ensure that a local server is installed, such as, Wamp64. When you download the application there will be a .SQL file which contains all the database information used. You must now log into PHPMyAdmin using your credentials and import the SQL file. In order for this to work with the application, you must configure the .env file.

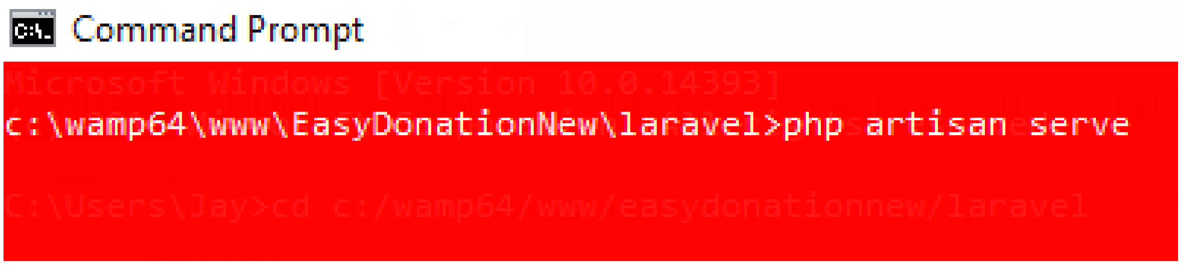


```
APP_URL=http://localhost

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=easy_donation
DB_USERNAME=root
DB_PASSWORD=*******

```

Figure 55 .env database connection

Change this to what you set as the database name, and use your own username and password for PHPMyAdmin. You must restart your server connection everytime you configure the .env file.

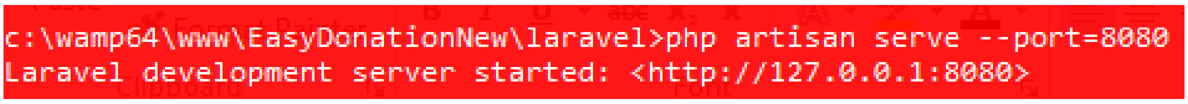
Next run you command prompt, and change the directory to where you stored the application, and run 'php artisan serve' to start Laravel, as shown in figure 56.



```
Microsoft Windows [Version 10.0.14393]
c:\wamp64\www\EasyDonationNew\laravel>php artisan serve

C:\Users\Jay>cd c:/wamp64/www/easydonationnew/laravel
```

Figure 56 starting Laravel.

An important note in case it failed to listen, this command will try and run Laravel on port 8080, if this fails you can try specifying the port, as show in figure 57.



```
c:\wamp64\www\EasyDonationNew\laravel>php artisan serve --port=8080
Laravel development server started: <http://127.0.0.1:8080>
```

Figure 57 changing portl.

You will now be able to use http://localhost:8080/ in your browsers url to view the application.

Test Logins

Below are some test logins that you can use for the charity owner (as you will need to contact the administrator in order to register to be a charity owner), there is also an administrator credentials.

Charity Owner	**Admin**
john@smith.com	jayrai3@hotmail.com
password	password

For users, you will be able to register for an account. As the admin access is not available on the homepage, you will have to edit the url to navigate to the admin's page, to as follows:

http://localhost:8080/admin/login

Final Application Design

This section contains screenshots of some of the main pages within the application, for quicker reference.

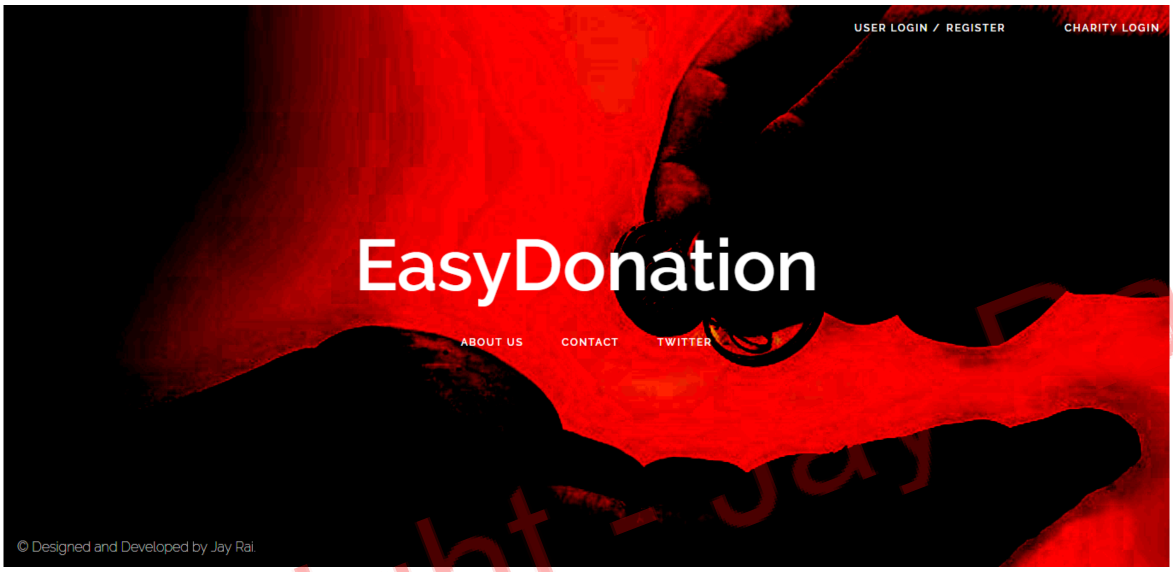
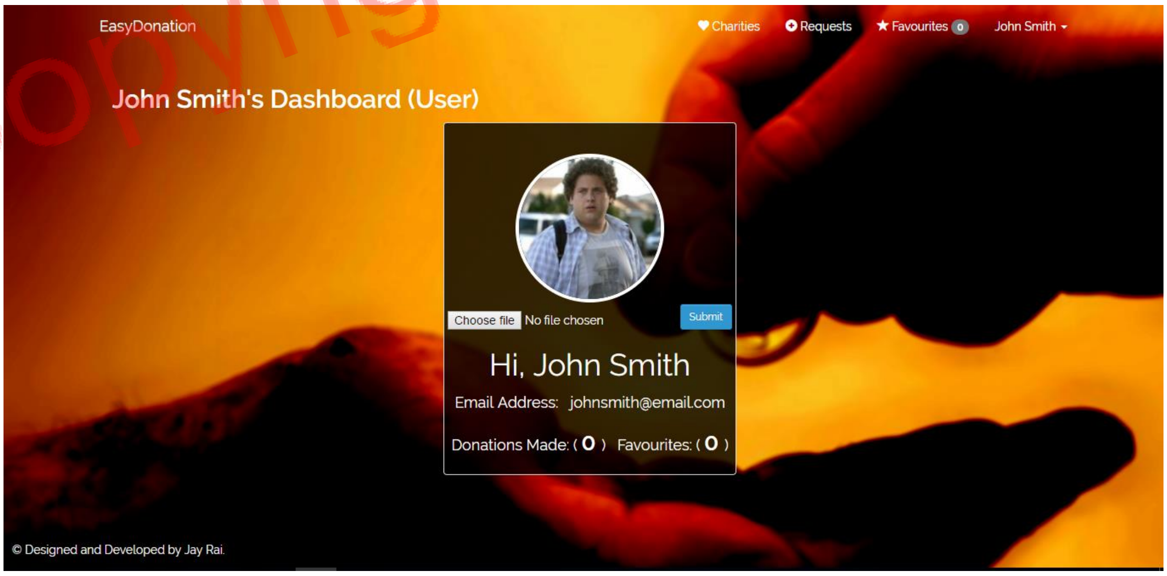


Figure 58 Welcome Page



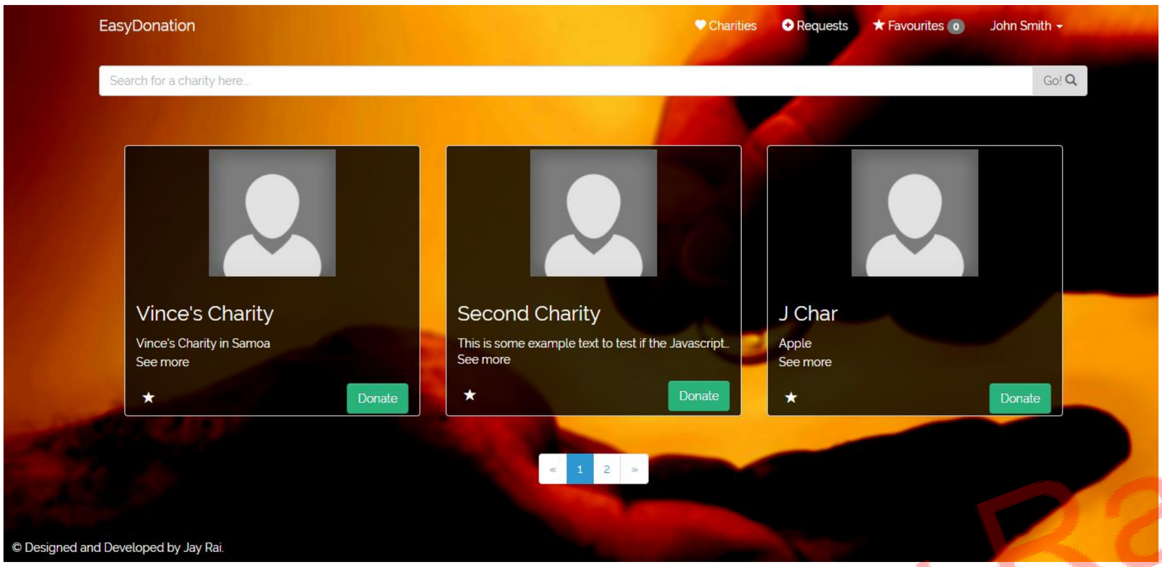
John Smith's Dashboard (User)

Hi, John Smith

Email Address: johnsmith@email.com

Donations Made: (0) Favourites: (0)

Figure 59 Users Dashboard



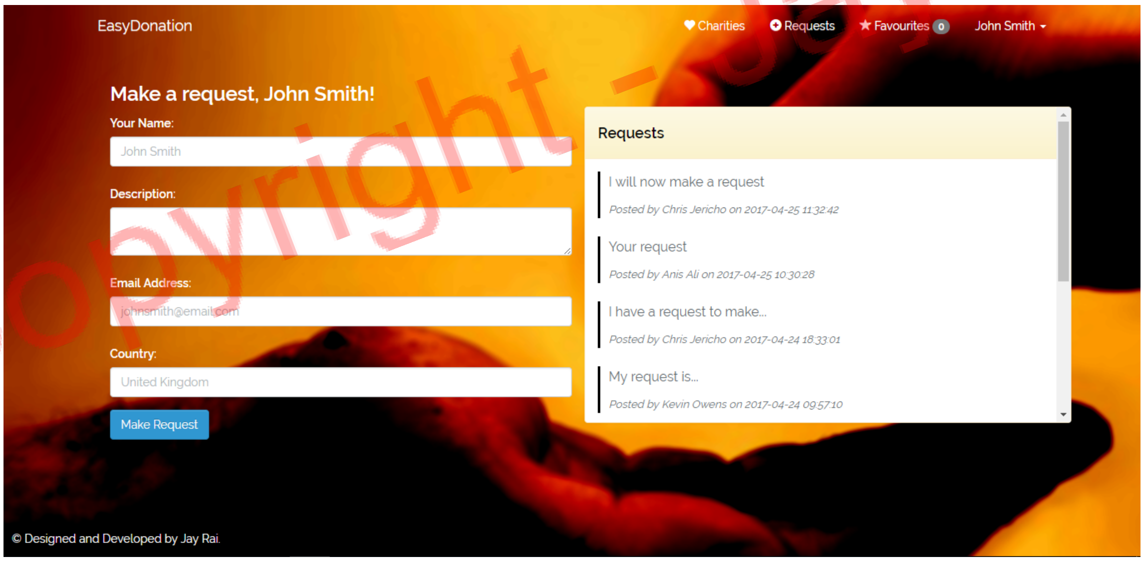
Vince's Charity
Vince's Charity in Samoa
See more

Second Charity
This is some example text to test if the Javascript...
See more

J Char
Apple
See more

© Designed and Developed by Jay Rai

Figure 60 Charites Page



Make a request, John Smith!

Your Name: John Smith

Description:

Email Address: johnsmith@email.com

Country: United Kingdom

Requests

I will now make a request
Posted by Chris Jericho on 2017-04-25 11:32:42

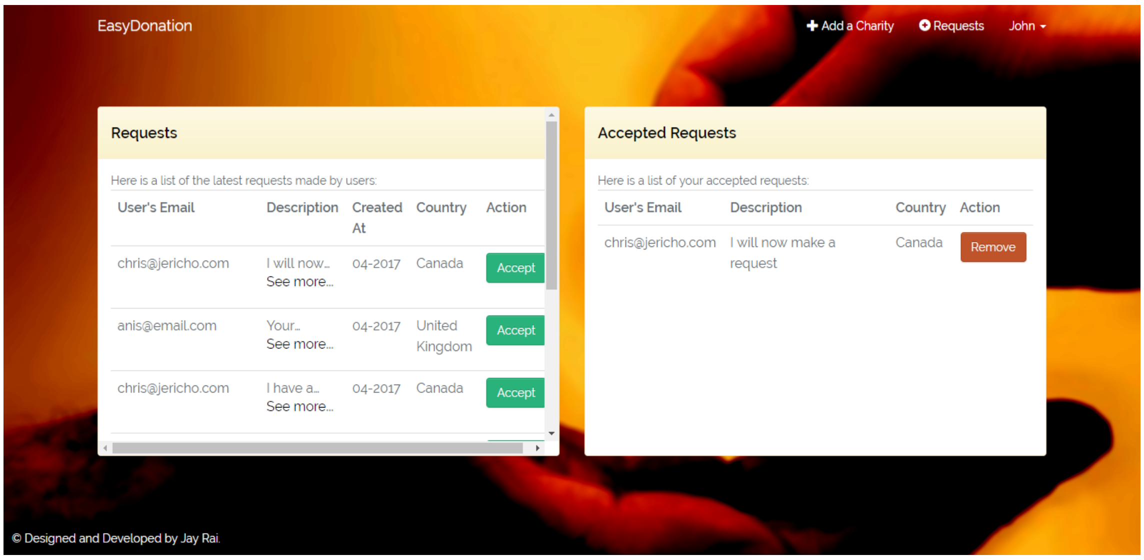
Your request
Posted by Anis Ali on 2017-04-25 10:30:28

I have a request to make...
Posted by Chris Jericho on 2017-04-24 18:33:01

My request is...
Posted by Kevin Owens on 2017-04-24 09:57:10

© Designed and Developed by Jay Rai

Figure 61 Requests Page



Requests

Here is a list of the latest requests made by users.

User's Email	Description	Created At	Country	Action
chris@jericho.com	I will now... See more...	04-2017	Canada	Accept
anis@email.com	Your... See more...	04-2017	United Kingdom	Accept
chris@jericho.com	I have a... See more...	04-2017	Canada	Accept

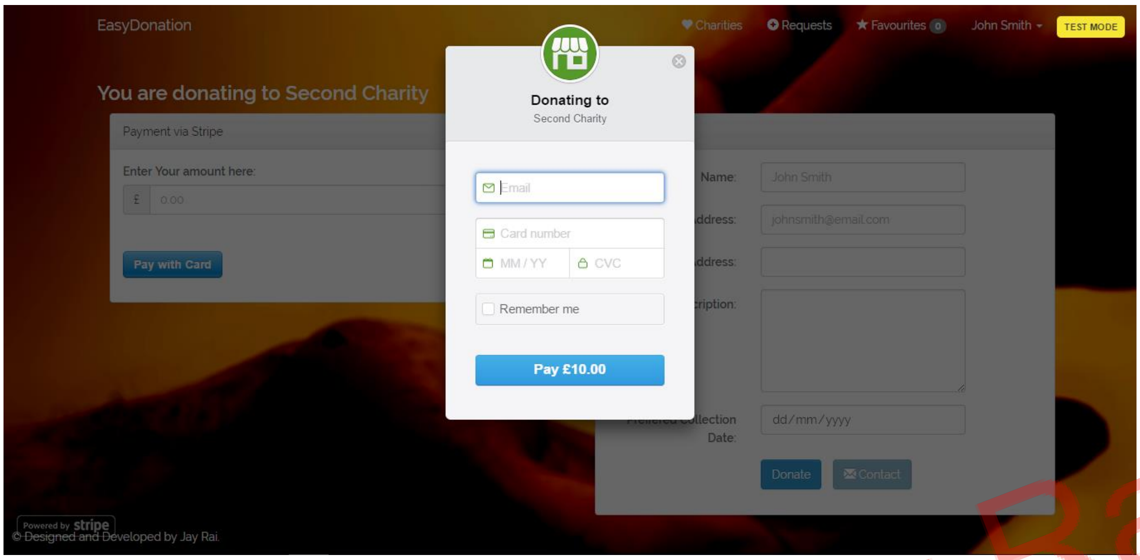
Accepted Requests

Here is a list of your accepted requests.

User's Email	Description	Country	Action
chris@jericho.com	I will now make a request	Canada	Remove

© Designed and Developed by Jay Rai

Figure 62 Accepted Requests – Charity Owner



You are donating to Second Charity

Payment via Stripe

Enter Your amount here:

£ 0.00

Pay with Card

Donating to
Second Charity

Email

Card number

MM / YY   CVC

Remember me

Pay £10.00

Name: John Smith

ddress: johnsmith@email.com

ddress:

ription:

Preferred Collection Date: dd/mm/yyyy

Donate   Contact

Powered by stripe

© Designed and Developed by Jay Rai.

Figure 63 Donating money/items page.

Final Class Diagram

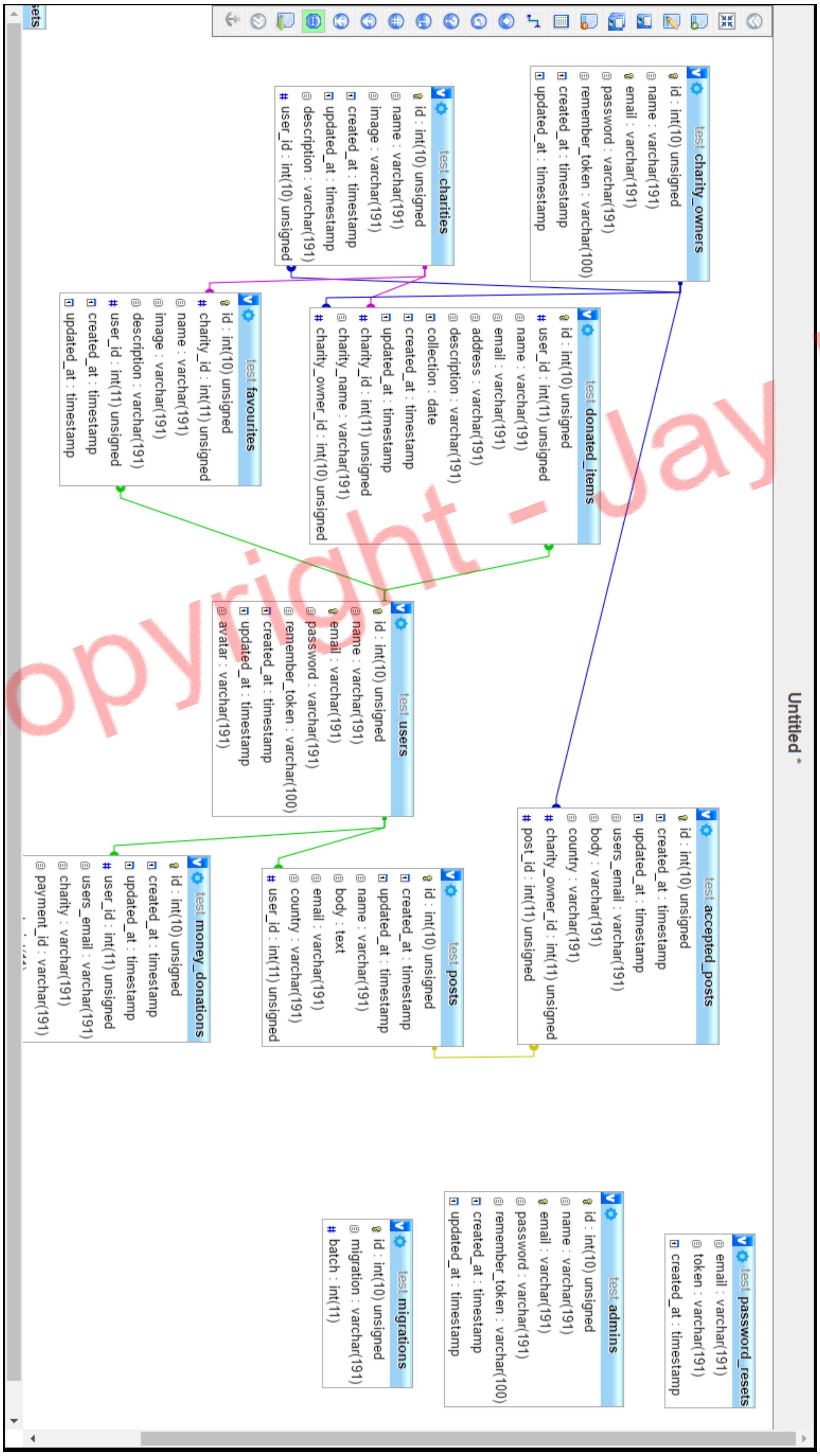
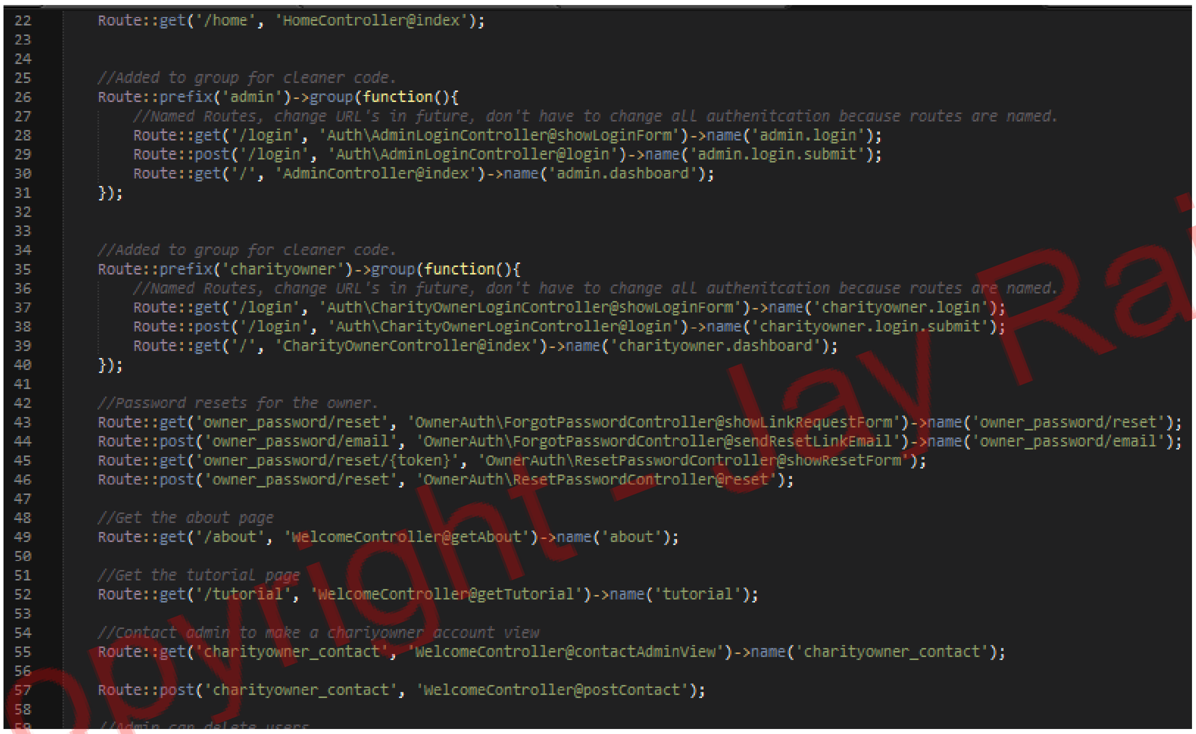


Figure 64

Code Written Examples

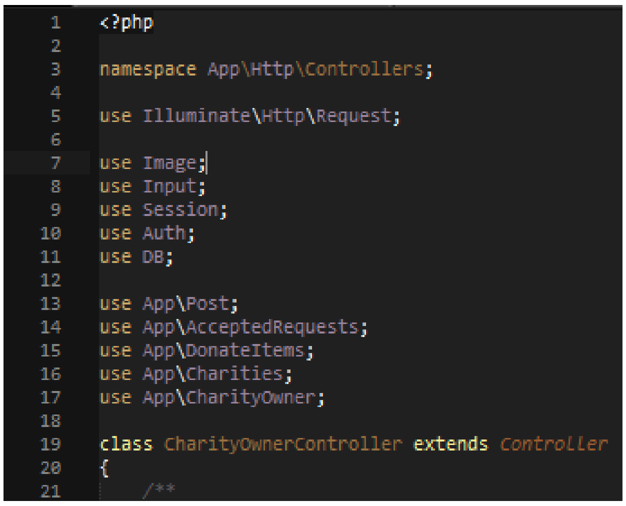
As said in the deliverable chapter, I adapted a PSR-1/2 styling and technique to the code, so it is easily readable and clean, comments are used throughout in case of any necessary changes needed to be done.



```php
22      Route::get('/home', 'HomeController@index');
23
24
25      //Added to group for cleaner code.
26      Route::prefix('admin')->group(function(){
27          //Named Routes, change URL's in future, don't have to change all authenitcation because routes are named.
28          Route::get('/login', 'Auth\AdminLoginController@showLoginForm')->name('admin.login');
29          Route::post('/login', 'Auth\AdminLoginController@login')->name('admin.login.submit');
30          Route::get('/', 'AdminController@index')->name('admin.dashboard');
31      });
32
33
34      //Added to group for cleaner code.
35      Route::prefix('charityowner')->group(function(){
36          //Named Routes, change URL's in future, don't have to change all authenitcation because routes are named.
37          Route::get('/login', 'Auth\CharityOwnerLoginController@showLoginForm')->name('charityowner.login');
38          Route::post('/login', 'Auth\CharityOwnerLoginController@login')->name('charityowner.login.submit');
39          Route::get('/', 'CharityOwnerController@index')->name('charityowner.dashboard');
40      });
41
42      //Password resets for the owner.
43      Route::get('owner_password/reset', 'OwnerAuth\ForgotPasswordController@showLinkRequestForm')->name('owner_password/reset');
44      Route::post('owner_password/email', 'OwnerAuth\ForgotPasswordController@sendResetLinkEmail')->name('owner_password/email');
45      Route::get('owner_password/reset/{token}', 'OwnerAuth\ResetPasswordController@showResetForm');
46      Route::post('owner_password/reset', 'OwnerAuth\ResetPasswordController@reset');
47
48      //Get the about page
49      Route::get('/about', 'WelcomeController@getAbout')->name('about');
50
51      //Get the tutorial page
52      Route::get('/tutorial', 'WelcomeController@getTutorial')->name('tutorial');
53
54      //Contact admin to make a chariyowner account view
55      Route::get('charityowner_contact', 'WelcomeController@contactAdminView')->name('charityowner_contact');
56
57      Route::post('charityowner_contact', 'WelcomeController@postContact');
58
59      //Admin can delete users
```

Figure 65 web.php

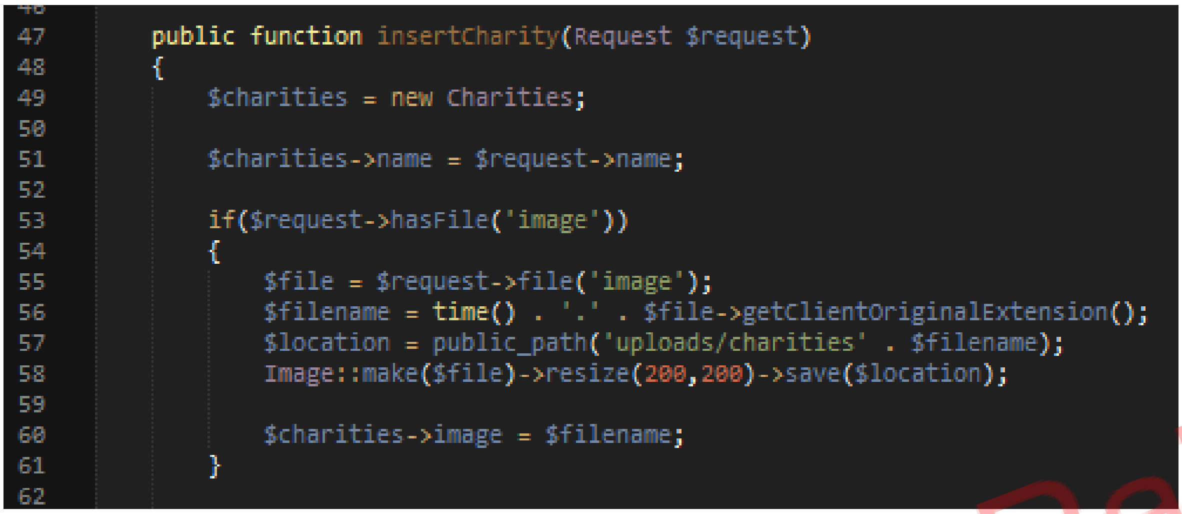
This is my routes file, where all the links happen. This essentially links buttons, forms (anything in the front view) to their functions in the relevant controllers. I have commented throughout, as shown from lines 34 – 40, I have named my routes and even grouped together routes that are for the same user and does similar functionality, such as authentication.



```php
1   <?php
2
3   namespace App\Http\Controllers;
4
5   use Illuminate\Http\Request;
6
7   use Image;
8   use Input;
9   use Session;
10  use Auth;
11  use DB;
12
13  use App\Post;
14  use App\AcceptedRequests;
15  use App\DonateItems;
16  use App\Charities;
17  use App\CharityOwner;
18
19  class CharityOwnerController extends Controller
20  {
21      /**
```

Figure 66 'use' and namespaces.

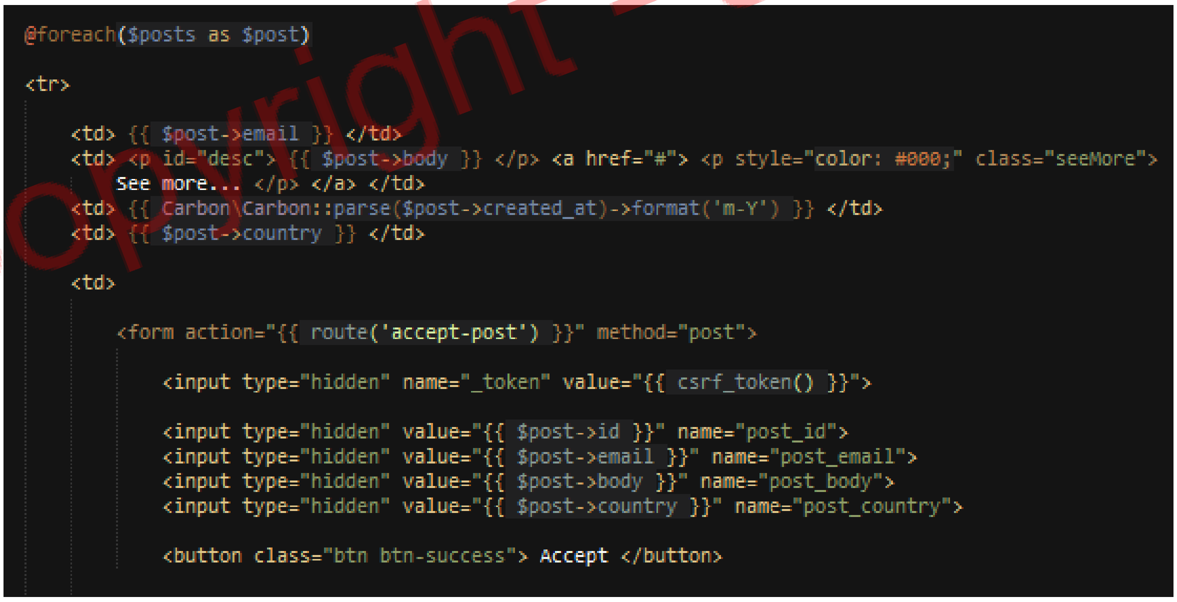
Figure 66 shows an example from the controller where I have included Laravel methods, or models. I have grouped them accordingly, such as, lines 7-11 are built in Laravel methods that are used, and lines 13 – 17 are uses of models that were called in the functions.



```
47      public function insertCharity(Request $request)
48      {
49          $charities = new Charities;
50
51          $charities->name = $request->name;
52
53          if($request->hasFile('image'))
54          {
55              $file = $request->file('image');
56              $filename = time() . '.' . $file->getClientOriginalExtension();
57              $location = public_path('uploads/charities' . $filename);
58              Image::make($file)->resize(200,200)->save($location);
59
60              $charities->image = $filename;
61          }
62
```

Figure 67 Function styling.

Figure 67 shows an example of the styling used in the controller functions, correct indentations are used, and lines separate different actions, such as, line 49 calls everything from the charities table, and line 51 makes a request to form to get the name and store it in the name field in the database.



```
@foreach($posts as $post)

<tr>

    <td> {{ $post->email }} </td>
    <td> <p id="desc"> {{ $post->body }} </p> <a href="#"> <p style="color: #000;" class="seeMore">
        See more... </p> </a> </td>
    <td> {{ Carbon\Carbon::parse($post->created_at)->format('m-Y') }} </td>
    <td> {{ $post->country }} </td>

    <td>

        <form action="{{ route('accept-post') }}" method="post">

            <input type="hidden" name="_token" value="{{ csrf_token() }}">

            <input type="hidden" value="{{ $post->id }}" name="post_id">
            <input type="hidden" value="{{ $post->email }}" name="post_email">
            <input type="hidden" value="{{ $post->body }}" name="post_body">
            <input type="hidden" value="{{ $post->country }}" name="post_country">

            <button class="btn btn-success"> Accept </button>
```

Figure 68 Front view styling

Figure shows the styling in one of the views, using HTML and Laravel’s blade expressions, again with the correct indentation.