MAJOR PROJECT REPORT

Interactive Multimedia Design – 2015/16

Gareth Frazer – B00613978

Peer Support Group 10 – Giuseppe Trombino

Table of Contents

Acknowledgements

I would like to thank my mentor Giuseppe Trombino for helping me develop the idea of SermonShare. I would also like to thank the AV Browne Web Team for introducing me to Laravel and helping me understand it during my placement year as the knowledge that I gained during this time was very insightful.

1 – Introduction

This report will look at the process of creating a sermon sharing web application called SermonShare and how it has become a fully functional product.

1.1 – The Challenge

The challenge was to create a project that showcased the learning and skills gained over the 4 years of the IMD course. Building a Web Application would help to demonstrate the skills obtained over this period of time. After much consideration this developer decided to create an application that allows churches to share their sermons with their congregations online. This was completed after looking at the process of idea generation which led to a number of different concepts that best showcased the skills a web developer should have gained during this period of study.

1.2 – Aims and Objectives

Aims

The Aim of this Web Application was to produce a system that would enable churches to connect with their congregations and wider communities by communicating interactively through recorded audio and video of church services from the comfort of their own homes. Another aim of the web developer to achieve a First Class Honours in Interactive Multimedia Design.

Objectives

Below are some of the primary objectives created by the developer in order to achieve the aims that have been established.

- Design and Develop a user friendly experience across all devices that will allow comfortable user engagement.

- Design and Develop the platforms that will allow users to upload media to be shared with other users e.g. churches uploading sermons for congregation/community members.

- Develop a user account system that is both capable of creating a sole user and also creating a church account user.

- Develop a search feature that will allow users to search for churches/sermons and engage with the results of the search.

- Develop a follow feature that allows specific content to be generated for users based on what churches they are following.

The secondary objectives show what could be done after the Web Applications initial release.

- Advertise and demonstrate the product to try and get more churches to use it.

- Develop the application into a native application for popular mobile Operating Systems such and Android and iOS.

- Provide different paid plans for users/churches that would allow them more functionality within the product e.g. Bronze, Silver, Gold packages.

- Have more languages available to allow the product to be used globally.

1.3 – Overview of Work Undertaken

In order to make SermonShare become a reality there was a lot of preparation and work that needed to be carried out. Firstly, the idea of SermonShare had to be thought over carefully to ensure that it was physically something that could be achieved in the given time frame and also something that the developer wanted to create and would enjoy creating it.

The next area that needed to address was the design of the project. This involved imagining the user experience, thinking about the system design and designing the logic and data of the project. Once this process was completed the developer was then in the position to create a functional prototype that they believed would be one of the most challenging areas of the project to complete. This would then give them a greater sense as to whether the project was manageable or not. After looking at these results the developer believed that they would be able to move forward.

The developer still had a considerable amount of work to be completed following on from the prototype stage to enable him to complete it as he had intended to before the deadlines approached.

1.4 – Overview of Report

This report was created to detail the process that the developer went through when creating SermonShare. As he has used a waterfall methodology when creating SermonShare he decided use the same methodology to keep this report consistent. This enabled him to have the organized structure that this report follows.

2 - Concept definition and testing

This section of the report acknowledges the concepts that were considered and demonstrates how these selected concepts were then tested to ensure it feasibility.

2.1 - Idea generation

Throughout the idea generation stage, the developer had multiple ideas that he then narrowed down to three possible concepts and conducted further research to decide which of these concepts would be feasible and enjoyable for him to work on, as well as present some welcomed challenges.

2.1.1 – 1st Concept – FlawlessFootball

The initial concept for this was to create a one stop place for booking your next Premier League football trip. The idea behind this concept was that when a user registers for this service and selected their favourite football team it would give location based results that would inform them of all the possible methods they could use to get to a match. This would have included travel arrangements, hotels and hostels along with day-trip options. The system would also provide external links as to where you could purchase match tickets.

This was not the developers chosen concept and their reasoning for this was that after some research was conducted it was simply not feasible. The developer was excited about this concept but in terms of making it successful he decided that it would be extremely difficult and at times even impossible.

2.1.2 – 2nd Concept – DaysOff

The second idea was to create an online based leave management system that could be used by either small or large businesses to keep track of employee's holidays. A company would simply register with the site then they would be able to add their employees who could then request annual leave. This system would also have shown how many holiday hours each employee had and would have identified time periods that may have been requested by other staff. These results would have been visually represented on a calendar type interface.

Although this is quite a practically idea, it was not the developers chosen concept. The reasoning behind this was not due to feasibility as this project was very feasible and within the developers range of skills to build it. The developer believed that it would not have sustained his interest over a long period of time and may have affected the production of it.

2.1.3 – 3rd Concept – SermonShare (Chosen Concept)

The developers third and final concept was SermonShare. SermonShare is an online space for churches to share their sermons via audio or video for their congregation/community members. SermonShare also allows users to sign up to the service to follow their church so they can catch up on missed materials.

The developer decided to build SermonShare because it was very feasible and because this concept was something he was enthusiastic about and wanted to see become a reality. SermonShare is something that he feels could really benefit his own church as well as others across the country or even the globe.

2.2 - Contextual research

2.2.1 - Problem or gap addressed

The problem that the developer is addressing for this project is that there are no services out there that offer an all-in-one space for both listening to and sharing sermons. The developer was trying to achieve an all in one space were users can follow their church or even other churches and interact with the content that they are producing from their sermons. The developer has targeted a few gaps in the market as well as this could be a useful application for many different people. An example would be a sick or elderly congregation member who was unable to attend church, as long as they had a reliable internet connection they would still be able to interact with their church and be up to date with what sermons are being preached.

Another gap that will be addressed is one which see's people scouring the internet for sermons. I will have this in one place with SermonShare and hopefully this will require less interaction from the user as they will find what they are looking for in one place.

2.2.2 - Existing solution

As the developer documented above there are no solutions exactly like mine, however there are solutions that can do the job that SermonShare will effectively carry out. Services such as YouTube, SoundCloud and iTunes all contain sermons in some form. His personal experience of this would be in his own church who upload their sermons to iTunes. These are solutions to the media handling side of uploading and sharing sermons however with SermonShare the developer is trying to achieve a richer experience with more features that will be of benefit to the user.

With these existing solutions in terms of media handling, there could be a use for them. With the likes of YouTube and SoundCloud having API's there is a possibility that he could use these to host his media for sermons, however this is not something that he has decided on entirely yet and it could maybe even just be an option to the user.

Another existing solution he came across was Sermon Audio. Sermon Audio is an online library of Sermons from churches and ministries worldwide. After looking into it I do feel it is not the same as what I am trying to achieve as my focus is more on the churches and their congregations being able to connect with them at any time. Another few examples which offered the same things are Sermon Central and preaching.com.

One thing the developer noticed about all of these sites he had researched is that they have a lot of information on every page you visit and in terms of the market he believes this would not be a very straightforward user experience for many and with SermonShare he feels that he can build a product that has high regard for the user's experience.

With SermonShare he wanted to create a solution to the problem of people having to use many different services to listen to sermons. He would like to create this solution to allow them to have one space for this rich experience to take place. This would save the hassle of using various services and even having to remember multiple passwords for accounts.

2.2.3 - What has gone before

In terms of what has been and gone in this market the developer is not too sure. He believes the end product that he is attempting to create is not something that even if unsuccessful would just go away. He believes this is because products like SermonShare can remain viable even if users aren't actually engaging with the service.

Of course the service would be pointless if no one were to be engaging with it, however he does not feel that it would be something that would simply slip away. As there aren't too many platforms out there just like SermonShare It has been quite difficult to find something that has been and gone. When viewing services such as Sermon Audio I feel that there is a real chance for SermonShare to be successful as it will focus more on the user experience and ensuring that users are comfortable using the service.

If SermonShare were to fall under the category of a social media tool then of course there would be multiple examples of services that have been and gone, but as of now I can't see what similar services to SermonShare have been and gone.

2.2.4 - Emerging solutions or context

An emerging solution for the problem he has addressed is SermonShare. With SermonShare he will be able to address the issue of people having to scour the Internet to find sermons. They will now be able to find them in one place and they will be more relevant as their own churches can both share and connect with their congregation/community members.

Other potential emerging solutions could be the likes of iTunes, SoundCloud or YouTube if they decided to have an area that was dedicated to sermons. Other solutions would include the likes of Sermon Audio, Sermon Central and preaching.com. Whilst these are established solutions, the developer had been unaware of them before conducting my contextual research into SermonShare. Of course this isn't to say just because he didn't know about them doesn't mean nobody else does but the chances are that they may not be as well-known as some of the bigger media services such as iTunes, SoundCloud or YouTube.

2.3 - Requirements specification

2.3.1 – Stakeholders

He decided that the main stakeholders of SermonShare would be churches and their congregation members. There could also be other students within the University that may make use of the application also.

2.3.2 – Tasks

The developer created a set of tasks in order to better present how he was going to complete the project within the given time frame and also to show that is feasible. He then followed up on this with the constraints that the project will bring and also the potential risks that could cause some issues.

Design

- **User Experience** - Create a user experience that suits the targeted audience and makes the process of using SermonShare flawless. This will involve creating paper-prototypes and wireframes for the application.

- **Branding** - Create a highly recognisable, solid brand that communicates what it is SermonShare does to its user base.

- **Database Design** - Take database design into account as SermonShare will have a user account system and will require database design to show the relationships that are required between tables.

Development

Development tasks fall under two categories, front-end and back-end. By separating these into sub-tasks, the developer can better manage his time as some tasks in this area will be more time consuming that others.

Front-end

- **User Interface (UI)** - The user interface will require coding and using a framework such as Bootstrap could help speed up this process.

- **User Experience** - The user experience will need to be carefully considered as I may be catering for some older generations who may not be as computer literate as a younger generation, this is not to say that they won't be either. It is a very important area as it could sway some of the user's opinions.

Back-end

- **Database** – A database will need to be set up in the back-end. The database will store user details, uploaded sermons, churches details and various other information.

- **User Accounts** – A user accounts system will be developed that will have two options. Firstly, you can register as a church if you are signing you're church up or secondly you can register as an everyday user that can follow their church and the content they are supplying.

- **Search Function** – A search function will be developed that allows users to search for content such as sermons, users and churches.

- **Tagging** – A tagging system will be created that allows users to tag uploaded sermons to make the search experience greater.

- **User Area** – A user's own personal area will be developed were all the content they are following or have listened to/watched will be in one convenient place unless they remove it.

- **Bible Integration** – For Sermons, an onscreen Bible will be available with various translations to allow users to follow along if they wish to do so.

Testing

- **Functionality** - The functionality of the application will need to be tested to ensure everything works as it should.

- **Usability** – Tests will be carried out to ensure the application is useable and that the user base will be able to navigate it with no issues.

2.3.3 – Risks

There are several risks that the developer will need to address when creating SermonShare. By addressing these risks, he is able to foresee some of the obstacles that will come my way and in turn create a solution that may help solve the problems.

Area	***Risk***	***Likelihood***	***Level***	***Possible Solution***
Skills	Knowledge of the Laravel Framework	Medium	High	Not knowing how to work with the Laravel Framework would mean that the developer would struggle to develop the application. There is a lot of documentation for Laravel and also screen casts on Laracasts that would help him understand the framework better.
	Creating the upload feature for Audio and Video	Medium	High	The developer has never created an upload feature before this could prove a very hard task. As this is a core part of the application it would not properly function without this. A possible solution would be to use other services such as SoundCloud or YouTube to plug-in to SermonShare and host from these services instead.
	Not knowing how to Integrate a Bible API	Medium	Medium	Look into API tutorials and how to implement it into the project. The overall project would still function without the

				inclusion of a Bible, however the quality would suffer as a result.
	Using Bootstrap for front-end development	Low	Low	Bootstrap would be ideal to help the product design on the front-end. If it became unfeasible to use, there are other frameworks out there that he could potentially replace it with such as Foundation.
Application	Creation of a tagging system for Sermon uploads	High	High	This could be something that may be too challenging as he can't foresee how it would be done. If sermons were untagged the system would still function but this would have a knock on affect with the search as it would not be as fluid. He could instead have sermons put into categories that would work instead of the tagging. The tagging would allow more flexibility though and would be preferred.

	Creating different User Account Types	Medium	Medium	Setting up a user account system should not be too difficult however making two versions of this system that will show a different view of the application could prove to be. If this were to occur the developer could have the same view for everyone throughout the application and all users could have the option to upload media.
	Creating the User Area (Homepage for user)	Medium	Low	Not having this part of the system would not stumble the functionality in any way. It would make it less impressive and the user experience would not be as enjoyable. A possible solution would be to do away with the user area if it became unfeasible.
	Search Functionality	Low	Medium	The search functionality will scour the application for keywords and also target the tagging system to find the best matches to the search terms. If a search feature couldn't be implemented it would seriously limit the user

				experience. A possible solution would be to navigate users in another way without the use of search although this yet again would limit the application.

2.3.4 – Functional Requirements

Standard User

Requirement #: 1

Description: Being able to log into their account.

Rationale: Keeping user data secure and comply with the Data Protection Act (DPA - 1998)

Dependencies: Will require a user to register and set up an account.

Requirement #: 2

Description: Being able to edit their account.

Rationale: A user can edit and update fields in their account. This could be for the purpose of changing their name or email address.

Dependencies: The user will need to edit the fields within their account area and update any relevant information.

Requirement #: 3

Description: Being able to delete their account.

Rationale: A user can delete their account. This could be because they no longer want to be registered with SermonShare or they no longer benefit from it's services.

Dependencies: The user will need to login to their account and then delete it.

Requirement #: 4

Description: Having the ability to follow their church or other churches and the content that they supply.

Rationale: A user can follow their church or other churches to stay up to date with the sermons they are uploading.

Dependencies: The user will need to search for their church to see if they are registered with SermonShare and then follow them.

Requirement #: 5

Description: A user will have an area were all content they are following will be posted with the latest additions first.

Rationale: This user area will act as a home page for the user were they will be able to see updates from the churches that they are following. This may also reduce the amount of searching that a user will need to carry out.

Dependencies: The user will need to follow their church or other churches to be supplied with their content.

Church User (Admin Role)

Requirement #: 1

Description: Being able to log into their account.

Rationale: Keeping user data secure and comply with the Data Protection Act (DPA - 1998)

Dependencies: Will require a user to register and set up an account as a church.

Requirement #: 2

Description: Being able to edit their account.

Rationale: A church user can edit and update fields in their account. This could be for the purpose of changing their name, email address or updating their logo.

Dependencies: The church user will need to edit the fields within their account area and update any relevant information.

Requirement #: 3

Description: Being able to delete their account.

Rationale: A church user can delete their account. This could be because they no longer want to be registered with SermonShare or they no longer benefit from it's services.

Dependencies: The user will need to login to their account and then delete it.

Requirement #: 4

Description: Having the ability to upload sermons in a video or audio format.

Rationale: A church user can upload a video or audio sermon to share with their followers.

Dependencies: The church user will need to upload the content in either a video or audio format. Users will then need to listen or watch this content to make it of any benefit.

Requirement #: 5

Description: A church user should be able to delete content.

Rationale: A church user can delete content they have uploaded if they wish to do so or if it was uploaded by mistake.

Dependencies: The church user will need to go to the piece of content they wish to delete and remove it.

Requirement #: 6

Description: A church user should have the ability to tag content they are uploading.

Rationale: A church user should be able to tag content they are uploading with as many tags as they wish. Recommended tags will come from previous tagging that has taken place for other content.

2.3.5 – Non-functional Requirements

The non-functional requirements are requirements that are not directly carried out within the system but are still of importance to consider.

Requirement #: 1

Description: A home page for non users to get a taste of the service.

Rationale: This home page will act as an 'advertisement' for SermonShare showing the potential site users can have if they register.

Dependencies: The site will need to have a user base visiting it both non-registered and registered in order to promote the service.

Requirement #: 2

Description: Network connection on mobile devices needs to be considered for loading.

Rationale: If a user is on a slow network connection some features should be made unavailable such as uploading content.

Dependencies: N/A

Requirement #: 3

Description: The site will be English based for its first release.

Rationale: The site will be in English for the initial release and maybe further down the line more languages could be added.

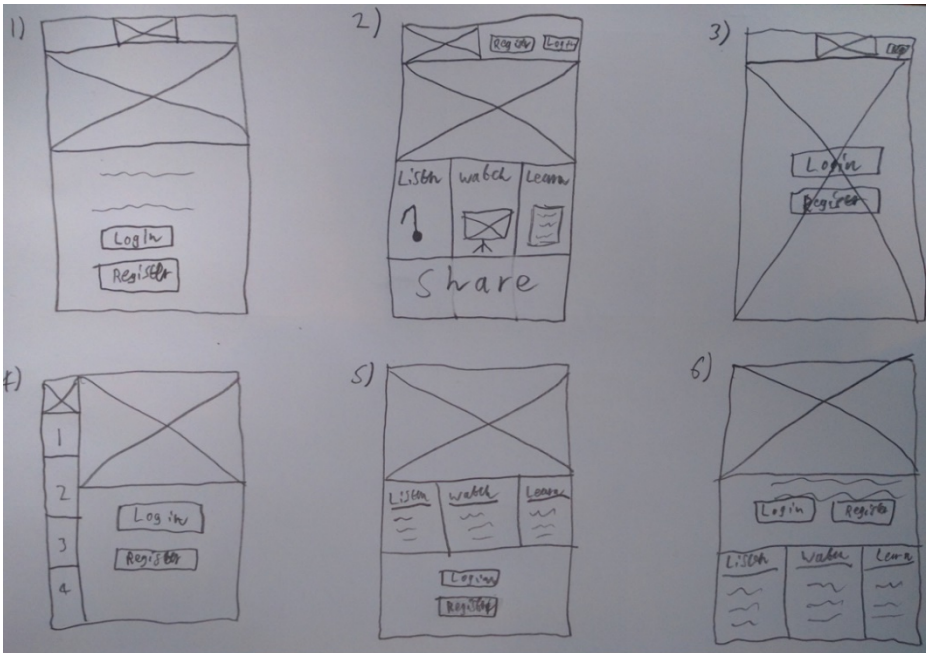
Dependencies: N/A

2.4 - Paper prototyping

The next stage in the process was to put onto paper how the application would function. This is essential in order to meet the requirements of the application and also gives a good insight as to how each stage of the process can be accomplished.

6-ups

This process involves creating 6 different sketches of a page within the application. The developer is targeting the homepage for this as it is always a great place to start and will then help set the tone for the rest of the application.



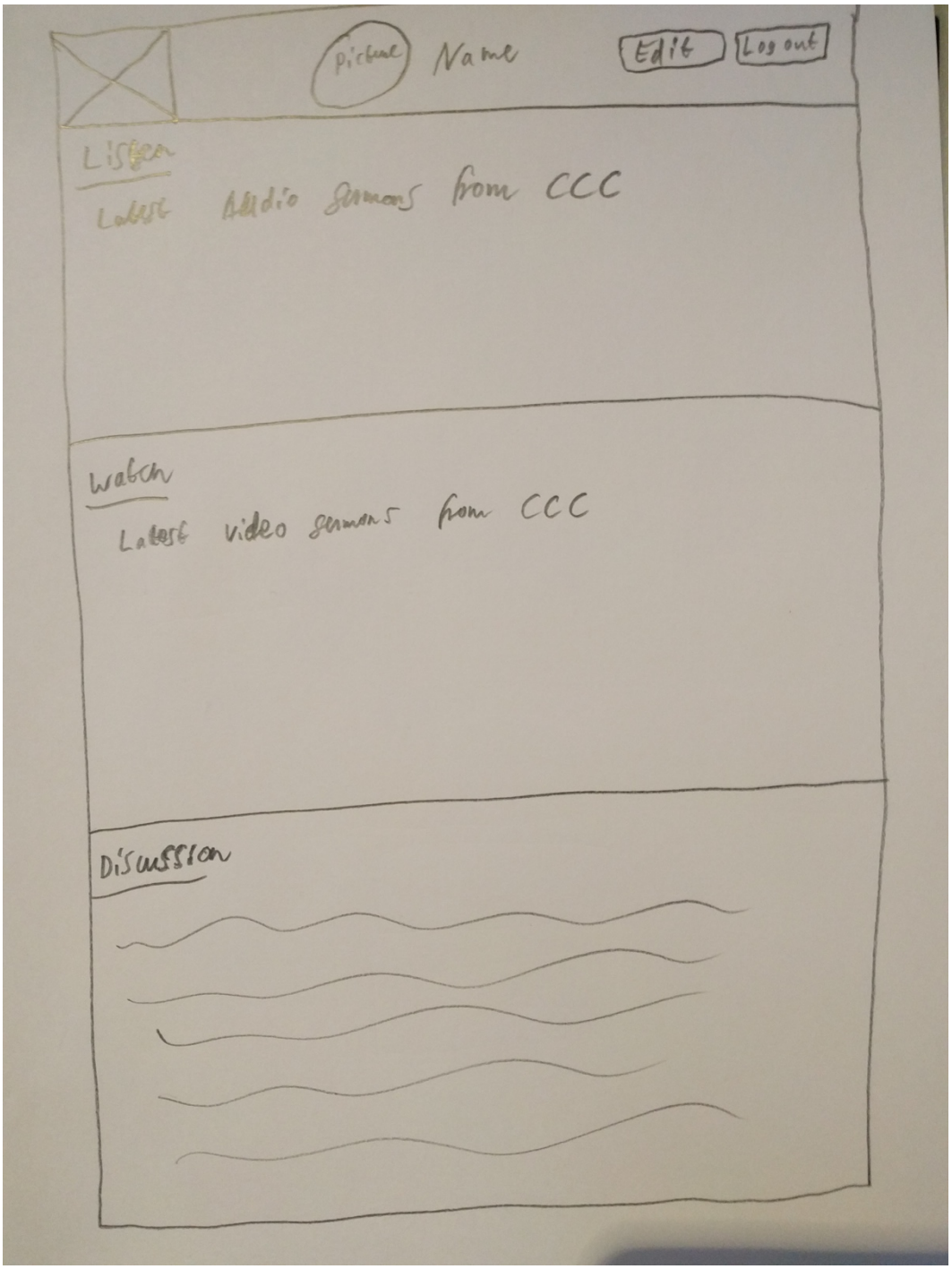
Homepage 6-ups

Once the developer completed the 6-ups for the homepage of the site he then picked the one which he felt was the best, number 5. The developer believes that the layout is

the best of them all and allows space for everything that a homepage needs. He then drew a larger mock-up of number 5.

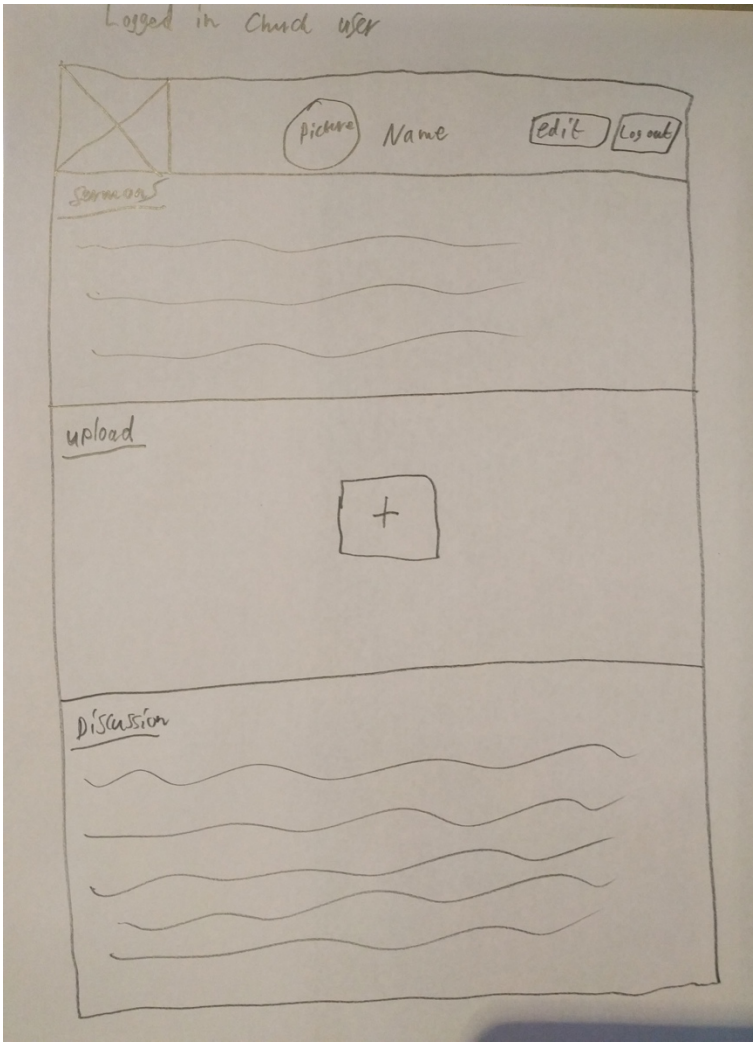
Chosen Homepage

He then created a logged-in mock-up of the homepage to show what differences will take place whenever a user logs in.



Logged in view of homepage for standard user

The developer then done another quick mock-up, this time it was of a logged in church user. The features are different as a church user has more options than a standard user.



As you can see from the final sketch the options available will change according to the user type. Within the sermons section the church user will also see what a standard user sees in both the listen and watch section. Within the upload section the church user will be able to upload video or audio that will then be viewable by both sets of

users. The discussion area is where comments will appear for each sermon and link users to those discussions.

2.5 - Feasibility testing

The feasibility testing is in place to help assert the project and ensure that it can go ahead. At this stage the developer has now created a functional prototype. To choose what he created for this prototype he went back to his risks table and selected the risk that he felt would be the most difficult for me to create. The developer decided to choose the upload feature that a church user will have the capability of doing.

As this is just a functional prototype the developer did not need to finalise this code or have it designed appropriately. This is simply at stage where the developer can see if something is feasible or not and he was pleased to say that it is and was able to create the prototype with no issues.

The developer had successfully created an upload feature that allowed a church user to upload either an audio or video file. The uploads themselves were stored within the file system as having them in a database could prove to be difficult later on. If there is a better method of uploading files available during further development of SermonShare the developer will of course take this into account. After he had created the prototype he was now confident that he would be able to produce an upload feature for the church users in the final release.

2.6 - Methodology selection

The next phase in the project was to then select a suitable methodology. This methodology had to take into consideration the way in which the developer was

approaching the project whilst also taking into consideration the report itself and how it flows.

There are many different methodologies you can choose from including Agile, Waterfall, Rapid Application Development (RAD) and Prototyping. One methodology is not necessarily better than the other one however it is important that one is chosen that suits the needs of the application he is am developing.

The developer decided to choose the Modified Waterfall Methodology. The Waterfall Methodology is great in that it allows you to do something and then move on to the next part, once all parts are complete then the process is complete, however he needed to use a methodology that allowed me to go back any time that he needed to and fix or update parts of the project. This is why he selected the Modified Waterfall as it allows for more flexibility whilst also allowing you to create a part of the application at one time.

3 - Design

3.1 - UK design evolution

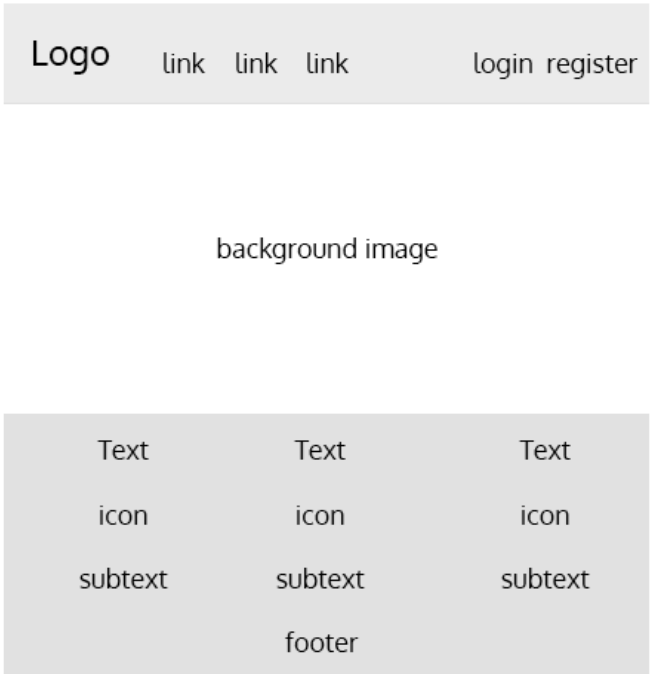
This area will look at the process behind the design of SermonShare and why certain design choices were made.

3.1.1 – Design Choices

When considering the design for SermonShare a lot had to be taken into consideration. Most importantly the developer needed to think of the target audience. The target audience for SermonShare ranges from all ages. This is because people of all ages attend Church. When thinking about this, the developer decided that it would be best to keep the design to a minimum. This didn't mean that the design would be poor or forgotten about, it rather meant that it would be best to keep it as simple as possible.

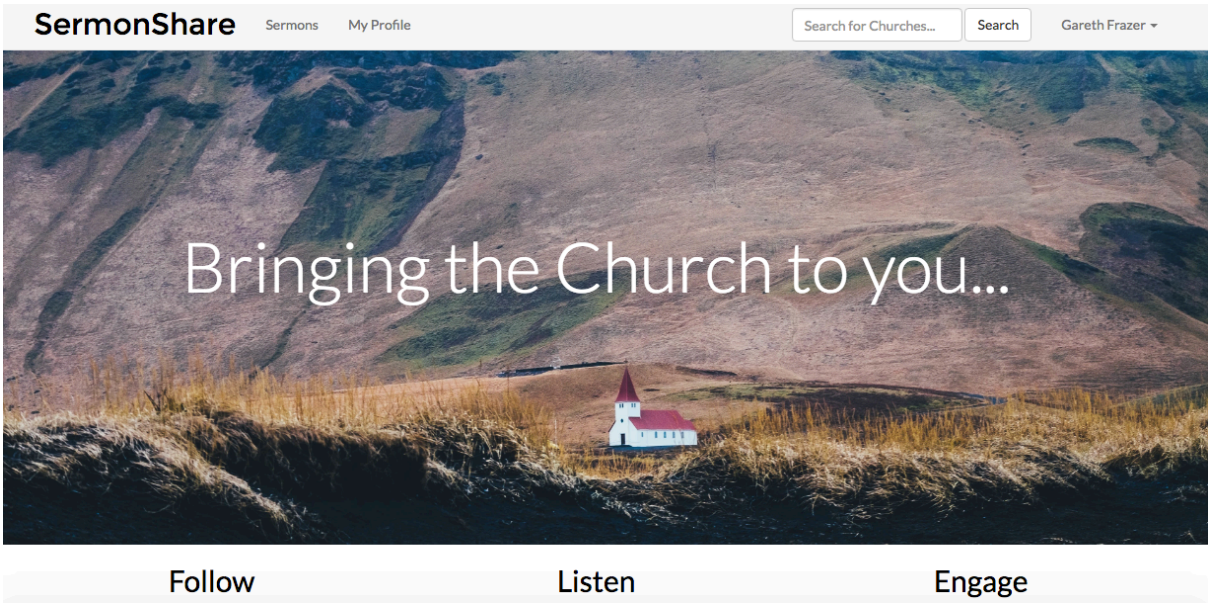
As the developer was using the Bootstrap framework for styling. It was decided that a lot of what Bootstrap offers was simple enough to use for the design of SermonShare.

The developer decided to make a mock-up of how they felt the site should look. The mock-up shows the homepage and how it was envisioned.



As you can see from the above mock-up the design would be very simplistic in order to cater to the target audience. As the target audience may include elderly people as they are unable to make it to Church services, the design needed to be kept simple as some older people may struggle to use the service.

Below is a screenshot of the homepage in production. As you can see it has changed a fair bit from the developer's initial mock-up. The design however is still simplistic and doesn't complicate the system.



3.1.2 – Branding

The branding for SermonShare would also be kept to a minimum as the developer didn't want it to be anything that would not communicate well with some of the target audience. It was decided that a simple font from Google Fonts would be used to act as the branding for SermonShare.

This decision was made based on the developer's notion that keeping the site as simple as possible in terms of design was the best way to go.

When browsing through Google Font's in order to find the correct font, a few different styles were considered.

Oxygen font

Pacifico font

Montserrat font (chosen)

The Montserrat font was chosen as it was both simple to look at and the writing was clear. The developer did prefer the pacifico font however it may be hard to understand for some of the target audience. Certain letters in the pacifico font are hard to make out.

Overall the branding catered well to SermonShare, it is simple, yet effective. As the developer the branding is what brings it all together as the product that has just been developer is held under this brand. The simplistic nature in terms of the design of SermonShare, is in place as a wide target audience needs to be catered for and sometimes to do this, things must be kept simplistic.

3.2 - System design

System design is all about showcasing what technologies and tools have been used to create SermonShare.

Technologies

In order to create the application successfully the developer needed to research some technologies that he would be using to build the application. There are a lot of different approaches that he could take however as this is a web application, most of the languages will reflect that in order to work with web browsers and apply the latest web standards.

Front-end Languages

HTML 5

- *HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It was finalized, and published, on 28 October 2014 by the World Wide Web Consortium (W3C).* **(Wikipedia)**

- Hypertext mark-up language version 5 is the latest version of HTML available on the web. Many of it's features will be of huge benefit within my application. HTML 5 will be used to create the basic views on the front-end and some of its features will be key to the application such as the video element.

CSS

- *CSS3 is the latest evolution of the Cascading Style Sheets language and aims at extending CSS2.1. It brings a lot of long-awaited novelties, like rounded corners, shadows, gradients, transitions or animations, as well as new layouts like multi-columns, flexible box or grid layouts.* **(Mozilla Developer Network)**

- Cascading style sheets version 3 will be used in styling the application. CSS is used to style elements on the page in order to add colour and a meaningful user experience.

JavaScript

- *JavaScript is a cross-platform, object-oriented scripting language. It is a small and lightweight language. Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.* **(Mozilla Developer Network)**

- JavaScript is a client-side scripting language. It allows web pages to become dynamic and interactive. It will be useful in my application as there will be times that I need to make some dynamic things happen on screen. There may also be some JavaScript libraries that the developer might make use of for some of the applications features.

jQuery

- *jQuery is a fast, small, and feature-rich JavaScript library.* **(jquery.com)**

- jQuery could be very useful within my application for tasks such as event handling and advanced dynamic changes happening throughout the application.

Back-end Languages

PHP

- *PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.* **(php.net)**

- PHP will be heavily used within my application as most of the key features will be built with it. As it is a server-side language the code itself cannot be seen from the front-end and most of its tasks involve sending and receiving data from a database. PHP is vital for the application asl be using it to sign users up and save their details in a database amongst other things.

SQL

- *SQL (pronounced "ess-que-el") stands for Structured Query Language. SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems.* **(sqlcourse.com)**

- SQL is used within a database. It is the language that a database is structured on and tells it how to set up a database. It can be used within PHP to make calls to the database to retrieve data and also to send data.

Frameworks

Bootstrap

- *Bootstrap is a free and open-source collection of tools for creating websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional*

JavaScript extensions. It aims to ease the development of dynamic websites and web applications. **(Wikipedia)**

- Bootstrap would be used to help in styling the application as it contains elements that can be used over and over again to keep consistency.

Laravel

- *Laravel is a free, open-source PHP web application framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC) architectural pattern.* **(Wikipedia)**

- The whole application will be built around Laravel which is a PHP framework. Laravel takes away some of the pain that standard PHP has always produced, an example of this would be code repetition. In Laravel you write part of the application once and it knows when to use that part of the application again without the developer needing to worry about adding it elsewhere. Laravel is an MVC (Model, View, Controller) framework that also contains the blade templating system. This system allows you to create repeatable chunks of HTML and then put them into templates for you thus spreading it across the application. Laravel is an extremely useful framework and will really help speed up the process of development once an understanding of it can be fully grasped.

API's

- *In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software and applications.* **(Wikipedia)**

- An API is a tool that developers can use to make use of a service. For this application the developer will require a Bible API for users to make use of. the

developer found an API online that will let me take it and plug it in to my application. API's can take away a lot of headaches and stress for developers as they can make use of things that have already been developed and don't need to create their own. This is also useful in terms of time constraints. After further research he has found a Bible application that can be implemented onto a site through an iframe. This in itself has its pros and cons, however it is one of the best Bible applications available.

3.3 - Logic design

This area looks at the design of the logic within the system. This section will discuss the technologies that have been listed within the system design and show how the requirements can be achieved.

Client Server Model

The client server model shows the technologies that are being used and were they fall into place, whether it be the client, internet or server. As SermonShare is a web application the developer found it appropriate to use this model to represent how the logic will flow. These various technologies will only work were they are represented on the model.

Client	Internet	Server
HTML5	YouVersion Bible iframe	PHP
CSS3		MySQL
JavaScript		Laravel
jQuery		
Bootstrap		

In the above table you can see the technologies presented in the category that they fall under in the Client Server Model. Visual elements are represented within the Client column, external services such as the Bible iframe are represented in the Internet column and unseen technologies that run in the background can be seen in the server column.

3.4 - Data design

In this section the developer will be looking at how the application will use data in various parts of the system. There are various elements on both the front and back-end that could be implemented with repeatable code. This means that the developer would not be writing the same code over and over again and it is better practice to have things this way.

Laravel uses a templating engine known as blade. Blade enables the developer to use repeatable elements of code and also easily iterate over variables in for loops. The advantage of this is that loads of code does not need to be repeated. Blade's syntax is also very easy to understand and efficient to use. Laravel's blades can be used as snippets of code in what can be referred to as partials. These partials can then be called by using blade's @include method. This makes it easier for the developer to create

templates to be used in various parts of the system. The list below shows the parts of SermonShare that used Laravel's blade templating engine.

- Navigation Bar
- Profiles
- Uploads
- Sermons
- Login and Register
- Search

On the back-end of the system there are various elements that are responsible for creating and saving data. This takes place within Laravel's Model's and it can then be used to create some logic in the Controllers. As Laravel is an MVC (Model View Controller) framework designing the back-end of the system is a very structured and organised process and the developer will need to stick to this structure in order to create a system that reflects its requirements.

3.4.1 - Database design

SermonShare required a database in order to store its data. This data would include user information, sermons that have been uploaded and profiles for users. It's important to get the structure of the database correct as it will be vital for the data to have the correct relationships set. To create the database structure an entity relationship diagram (ERD) was created showing the links data required and links between tables as shown below.

users		
primary	user_id	int
	name	varchar(255)
	email	varchar(255)
	password	varchar(255)
foreign	role_id	varchar(255)
	remember_token	varchar(255)
	created_at	timestamp
	updated_at	timestamp

sermons		
primary	id	int
	title	varchar(255)
	body	varchar(255)
	author	varchar(255)
	file	varchar(255)
foreign	user_id	int
	created_at	timestamp
	updated_at	timestamp

roles		
primary	role_id	int
	name	varchar(255)
	description	varchar(255)
	created_at	timestamp
	updated_at	timestamp

profiles		
primary	profile_id	int
foreign	user_id	int
	name	varchar(255)
	location	varchar(255)
	bio	varchar(255)
	twitter_username	varchar(255)
	facebook_username	varchar(255)
	website	varchar(255)
	created_at	timestamp
	updated_at	timestamp

followers		
primary	id	int
foreign	user_id	varchar(255)
	follower_id	varchar(255)
	created_at	timestamp
	updated_at	timestamp

As you can see from the diagram above SermonShare required 4 tables. The developer has not included the migrations and password resets table as these are just for Laravel to manage the database and manage password resets for users. The migrations table looks up all the tables you have created within Laravel and then brings them into the database. Seeds can also be set up in Laravel which hold data to be placed into tables in the data base. They can be thought of as dummy data or fake data to get systems up and running.

The users table was created to store all information for a user. A user will fill out a registration form which will then take the data and store it in the users table. The

primary key of the users table is the user_id which is used throughout the application for various things such as checking if the user's id is authenticated or not.

The sermons table was created to store links to sermons which have been uploaded to the file system. This means that the developer can return the data in the file column which will hold the location of the file. Sermons also holds a foreign key which is user_id. This is where the id of the user uploading the sermon is stored. This is then used to show only the sermons that a user has followed through the followers table which will be discussed later on.

The roles table has a vital part to be in the development of the system. Within the roles table there is 2 role_id's stored. These role_id's define what a user can and cannot do. By using this role_id the developer is able to shut off some of the system to a user depending on their account privileges. The two types of account that are available include a church account and a general user account. The church account can upload content were as with the general user account it is forbidden to do so. The role_id is a foreign key on the users table as each user will be given a role_id when they register an account, this depends on what the user chooses.

The profiles table contains a profile for every user created. When a new user is registered the profile table is also created with the user_id again as its foreign key. The user_id is taken from the newly created user and placed in the profiles table along with other data which is editable to the user. This then allows a nice way for a user to show some data about themselves or not if they choose not to do so.

The followers table is an important table and in particular when it comes to the sermons being shown to a user. The followers table also holds the user_id as the foreign key with users. For example, an account is set up as a general user account. A search can be performed to find a church. If the church is on SermonShare, a link to a profile will be

shown, if they are not, a message will be displayed to inform the user that the church they searched for is not on SermonShare. If the church is on SermonShare, a user can click through and be presented with a follow button on the church users profile. If the user follows this church, the user will then get all the sermons that church has uploaded as the followers table looks at the user_id and matches it to the user_id in sermons as long as the authenticated user (logged in user), is present in the followers table matching to a follower_id. The follower_id then follows the user_id and the relationship ensures that Sermons are displayed depending on who the authenticated user is following.

4 - Implementation

The implementation section looks at the development which took place to create SermonShare. It details the functionality that can be found within SermonShare and how that functionality came to be. The tools that were used throughout the development process will also be reviewed and any significant changes will be documented.

4.1 - Technology/tool review

This section looks at what possible technologies could be used in order to create SermonShare and what would be best in order to get SermonShare to function.

4.1.1 - PHP

To create SermonShare the developer could work with PHP. Some people would refer to PHP as vanilla PHP. This is because there are now many powerful PHP frameworks available which take care of a lot of the code which a developer would often finds themselves writing over and over again.

PHP would be an ideal choice as it caters for all the requirements that SermonShare has. PHP would allow the developer to connect to a MySQL database and query the relationships that were shown in the database design. PHP would also allow a developer to create a web application that will accept data from a user and store it in the database.

The developer would have concerns when vanilla PHP as the code could become uncontrollable and very messy. There would also be countless times were code would be written over and over again. This is in comparison to a PHP framework that would allow the developer to take a more structured approach to building SermonShare.

4.1.2 - Slim

Slim is small PHP framework that would enable the developer to quickly write powerful web applications and APIs. This would be a better choice than vanilla PHP as the framework provides a nicer way for the developer to get post, update and delete data. This could well meet the requirements of SermonShare, however after looking into Slim in more detail, the developer found that it may not be the best choice when building SermonShare.

Slim would provide a nice way of getting and manipulating data however it wouldn't be powerful enough to tackle the requirements of SermonShare head on.

4.1.3 - Laravel

Once Slim had been considered, the developer then looked into Laravel. Laravel is a free, open-source PHP framework. It was created by Taylor Otwell and applications are developed under the model-view-controller (MVC) architectural pattern. - *https://en.wikipedia.org/wiki/Laravel*

Laravel seemed like the perfect candidate as it provided a powerful framework that catered to the requirements of SermonShare. The developer has had some previous experience with Laravel and knew of some of the benefits of using it.

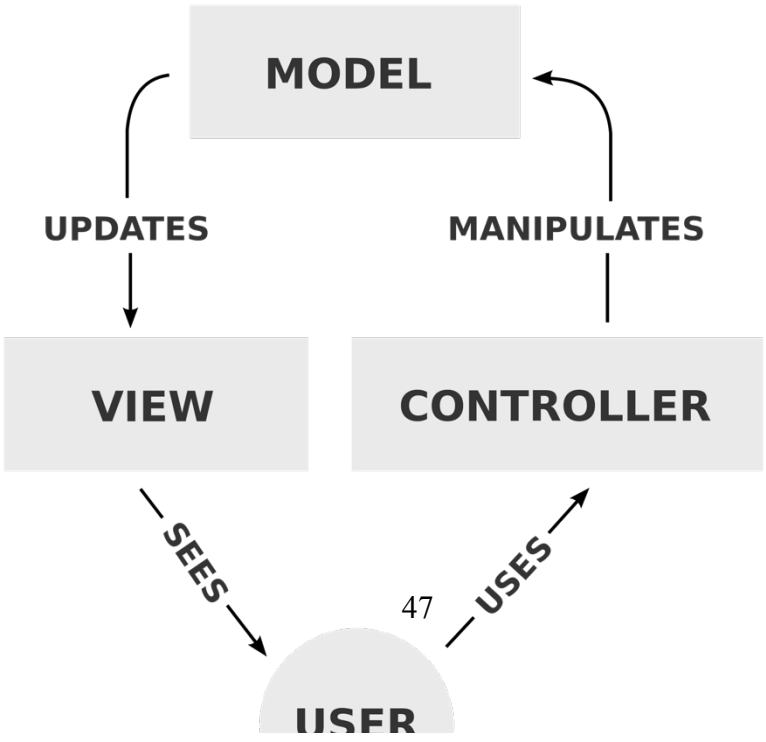
Laravel provides strong documentation on its official website as well as screencasts on laracasts.com. The developer has used some of these screencasts before to learn the basics of the framework and found them very productive and helpful.

After much investigation it was decided that Laravel would be the technology that would be used to create the SermonShare web application.

4.2 - Technology/tool selection

It was decided that Laravel would be the technology of choice to create SermonShare. Laravel works under the model-view-controller architecture. What this means is the application will be divided into three interconnecting parts. This is so as information can be understood differently from the way in which the application represents it back to a user. - *https://en.wikipedia.org/wiki/Model-view-controller*

Model	The Model handles the logic for the web application
View	The View is where the data is handled for display
Controller	The Controller handles how the user interacts with the web application and what should happen when a user does something



https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

The above diagram shows how the MVC architecture works in terms of a user working with the application.

Laravel is a powerful application and with it's MVC architecture it is the perfect tool to build SermonShare with.

4.3 - Technology/tool use

4.3.1 – Site functionality

Routes

In Laravel the routes of the application are the urls that a user is taken to. In the routes file the developer has set a name for a route and then when the user accesses that route, the route then hits a specified controller and method. What this means is within

the controller various methods can be called and when a user for example navigates to '/sermons' they will be going to a method called index. Within this method all the sermons that have been uploaded by a church that the user is following will then be returned.

```
Route::get('/sermons', 'SermonsController@index');
```

Controllers

Within SermonShare the developer has made use of controllers. The controllers are called into action when a user accesses a route and then an action is called on that route to hit a method on the controller. Below is a list of controllers that SermonShare uses:

- **FollowersController**
    - The Followers Controller manages what happens whenever a user follows another user such as creating a new follower when a user follows another user and deleting a follower when a user unfollows.
- **PagesController**
    - The page's controller manages the static pages for the application. This includes the home page when both logged in and logged out.
- **ProfilesController**
    - The profiles controller manages what happens when a user profile is shown, edited and updated.

- **QueryController**
  - The query controller manages what happens when a user searches for a church or another user.

- **SermonsController**
  - The sermons controller manages what sermons a user can see depending on what churches they follow. It is also responsible for handling the uploading of sermons.

- **AuthController**
  - The auth controller manages anything to do with user authentication and uses what's known in Laravel as a middleware to check fi a user is authenticated or not. It is also responsible for creating a new user as this falls under authentication.

Authentication

Authentication in Laravel is something which the developer found very useful. Out of the box Laravel supports authentication which just means it ensures that users are authenticated before doing something. Within the routes file the developer has defined auth which means any routes which fall under it can only be accessed once a user has either logged in or registered. Authentication is vital to SermonShare and with Laravel ensuring that it is taken care of, users of SermonShare will easily see in the views where they can and can't access based on their current authentication status.

Views

As noted before. Laravel handles its views (html) with the blade templating engine. As the developer this is a very nice syntax to use and it almost instantly makes sense. The great things about views is that you can manipulate data with ease. An example of this would be if the developer wanted to get the current user name, the developer could pass in the user model from the controller and then within blade we can access the data from the database and display it in the view. However, if the data is not passed the application will throw errors as the variable will be undefined.

Blade also allows a developer to include sections from other blade files. This is very useful where templating is concerned as the developer does not then need to change multiple lines of code in multiple places, rather the developer will change it in one location and it will be changed across the application.

4.3.3 – Git

Git is a version control system that allows a developer to store code repositories. A developer can work with several branches within Git. An example of this would be a development branch where the developer would store all code that is in development. Another example would be the production branch where code would be stored that is intended to be the live version of the system. Branches can also be merged. For example, the developer has created a new feature for SermonShare, they aren't sure about having it within the production branch so they create a new branch first and then after testing is successful they can merge it to the production branch and put the changes live.

If a mistake is made or the system needs to be rolled back to an earlier version, Git makes this very simple as a developer can view the commit history and rollback to whatever version of the system they want to. If a developer makes changes to the code they can then push these changes to the repository by running a few commands in the

terminal. The other option is to download a GUI for Git which would allow a developer to do the above but within a user interface and without commands.

4.3.4 – GitLab

GitLab is a code revision system that the University has recently setup on their servers. The system is fairly straightforward to use and allows the developer to set up a new project where they would host the code for SermonShare. This is then known as a code repository. The advantages of having a code repository are that:

- Changes you make to code will be tracked and can be rolled back
- Code is secure in case a software or hardware failure on the development machine occurs
- Other developers can easily view your code and make pull requests if the developer was working with a team
- Code can be deployed from some repositories to a live environment

4.3.5 – PHPStorm

PHPStorm is an IDE (Integrated Development Environment) that allows users to code with intelligent code completion. As the developer is using Laravel, PHPStorm has packages available that can be installed to help with certain frameworks, the developer installed the Laravel package which helps complete some code and can also explain what the code is achieving or looking to achieve.

PHPStorm also made it very easy for the developer to search for files. As the developer is using Laravel this was a good feature as the framework comes with a lot of files and the developer noted that it can be hard at times to find what exactly you are looking for. **-***https://en.wikipedia.org/wiki/Integrated_development_environment*

4.3.6 – Terminal & Artisan

The developer also made use of terminal when developing SermonShare. The terminal is used for many things when working with Laravel. The developer used the php artisan commands within the terminal. These commands can do things within Laravel such as create new controllers and create migrations. This then creates the new files with all the boilerplate that they need.

The developer also made use of the terminal whenever they were first setting up the Laravel application. As the developer had the Laravel installer within their composer installation they were able to run a global command that installed the latest version of Laravel as a new project.

Terminal was also used for git commands. The developer would check for any changes within the code from the terminal and if there were changes then the developer would push those changes to the GitLab repository.

4.4 - Notable challenges

As the developer working with Laravel, there were a few notable challenges as the framework can become very complex at times. Areas that I found difficulty working with were uploading content, following another account and getting the project live on a production server.

4.4.1 – Uploading content

The developer found it difficult to get the concept into my head of uploading files. Initially he thought that the files (sermons) would be stored in the database, however after considerable research it was discovered that this was actually very poor practice. This would cause the database to become very huge in size and generally a database should be kept as small in size as possible.

When researching how to store file uploads it was discovered that the developer could store them within the file system. What this means is that when he uploaded a file, the path to that file is stored in the database and the file itself is stored within the file system so that it can be accessed.

The developer had taken a considerable amount of time to work out how he would create the correct function to create a file upload. This is why it was decided that this would be the part of the system that was tackled for the prototype. The developer was successful in creating the prototype and the file upload system was fully functional.

4.4.2 – Following an account and displaying content

Another area that the developer considered to be very challenging was when an account was followed, how would he then show only the sermons that the account that was followed has uploaded. This took a considerable amount of time to work out as the relationships between tables had to be created in the models to ensure that the correct data could be manipulated.

Once the relationships were working as the developer expected them to, the data was now accessible and a function was created that checked the user_id that the authenticated user was following against the sermons table to see if they had uploaded any sermons and if they had then show that content.

The developer found this challenging and at one point he believed that he would not get it working in time for production, however through further understanding of relationships in Laravel's Model's, the developer was able to create code that functioned as planned.

4.4.3 – Production server

This notable challenge was something that was discovered quite late on. As the developer, the production prototype was hosted in the University's production server, however this came with its own issues. The developer had passed the deadline for uploading the prototype as they were having issues deploying it the University's servers. As this was an issue with Laravel needing to be at the top level, the developer contacted the technicians at the University who managed the servers. They were able to guide the developer into what changes needed to be made to the **htacesss file**.

After some time the developer managed to get the prototype working as expected, however this was after the deadline for the prototype. The developer contacted his mentor who knew of the issues and was understanding of the issues that the developer was facing.

In order to not face similar issues or run over the deadline for the final project, it was decided that SermonShare would be hosted outside of the University with a redirect being placed in the University servers. As the developer it was very difficult to get the prototype up and running, therefore the decision was taken to move away from the University's servers and host SermonShare externally.

4.5 - Notable achievements

There were also some notable achievements made whilst developing SermonShare.

4.5.1 – Learning more about Laravel

The developers understanding of Laravel has increased since development began on SermonShare. By following tutorials on Laracasts, the developer was able to further his knowledge of the framework. Specific achievements within learning more about Laravel were:

- Creating a search feature using a controller to get the query that the user makes and then checking it against the database by using s tutorial
- https://tutorialedge.net/laravel-5-simple-site-search-bar

- Setting up a middleware to set user roles throughout the application by using a tutorial.

- https://gist.github.com/drawmyattention/8cb599ee5dc0af5f4246

4.5.2 – Better understanding of relationships

The development of SermonShare required many relationships to be set in Laravel's Models. Relationships are something that the developer previously struggled with in Laravel as they can become very complex at times. By creating the followers table and having the follower see only content based on who they follow, The developer gained a better understanding of how relationships work within Laravel.

This better understanding of relationships will really help the developer as they continue their learning of Laravel.

5 – Testing

In this section of the report, SermonShare will be tested out in various ways. Performance, responsiveness and usability will be tested.

5.1 - Test approach selection

In order to complete testing, it is suggested that an approach is first taken as to how a product will be tested.

5.1.1 – Usability testing

Usability testing may include things such as error messages being provided when an error occurs or if a form isn't valid then a message will appear to let the user know that something wasn't right. This can be essential as users need to know how they can work around the system if something goes wrong.

5.1.2 – User acceptance testing

User acceptance testing ensures that the site works as it is expected to. This could include test with various browsers to ensure everything functions the same way and how it is supposed to.

5.1.3 – Performance testing

Performance testing will put SermonShare under stress tests to ensure that it can cope with demand as it is pushed to its limits.

5.2 - Test process

The test process details how each test is carried out and what is involved when it comes to testing each area.

5.2.1 – Usability testing process

This process simply involved observing a user trying to use the site. The developer watched over the user and made a checklist to see if the user could navigate the site. The checklist included:

- Register as a user
- Search for a church
- Follow a church
- Check out the sermons supplied by the church they followed
- View their own profile

5.2.2 – User Acceptance testing process

SermonShare was tested against all the latest browsers to ensure that it functioned as it should. The responsive design was also taken into consideration.

5.2.3 – Performance testing process

To test the performance of SermonShare I used a tool called nibbler which gives your website a rating out of 10. Nibbler also breaks the score down into various sections. This would give the developer a better idea of how the application could be improved in the future.

5.3 - Test results

The test results for the various tests are shown in this section.

5.3.1 – Usability testing results

The test results came back documenting the users journey and how long it took them to compete the various tasks set out in the checklist. As the developer I can then use the times that I got from the results to improve SermonShare

Register	Search	Follow	Sermons	Profile
Time: 20sec	Time: 10sec	Time: 1min	Time: 30s	Time: 1min 30sec
Comment: the user had a quick look at the site before deciding to register as a user.	Comment: the user searched for a church.	Comment: the user spent some time looking around the user's profile before following.	Comment: the user didn't seem too sure about where to go after following a user.	Comment: the user spent some time editing their account and seemed to understand what they were doing.

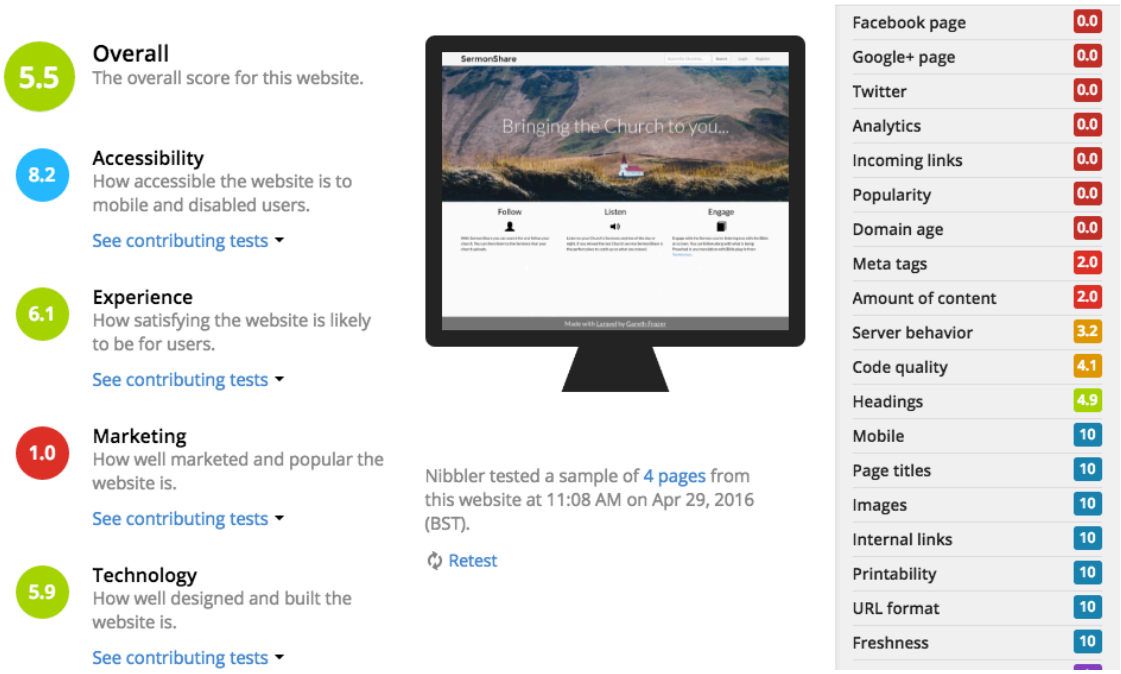
5.2.2 – User acceptance testing results

SermonShare was tested on all modern browsers (Chrome, Firefox, Safari, Internet Explorer, Edge) to ensure that nothing looked out of place and displayed and functioned as it should. Older browsers such as Internet Explorer 8 were not considered as they are too outdated now.

The test results were successful as they showed little or no issues at all when testing SermonShare on all modern browsers.

5.2.3 – User performance testing results

After running SermonShare through nibbler, the results were as follows:



A score of 5.5 is not ideal, however considering the scope of the application as the developer, I was pleased overall. As you can see from the results the accessibility, which is of vital importance to any web application, were very high. The technology score and mobile scores were also very good. This shows that SermonShare is responsive and ready for use on mobile devices.

5.4 - User survey responses

The survey responses can be found in the Appendix. Overall the response was good, however the developer only managed to get a response from 3 users. Two of these users were general users and the other one was a church user. The church user seemed happier with the service than the other two general users. This may indicate that the service could be improved on for general users.

Overall the testing went well and the processes that were followed really helped the application get to its final stage. SermonShare has now been placed in production following the testing process.

6- Evaluation

To evaluate SermonShare a few different areas needed to be considered. The evaluation process helps the developer revisit the testing results, project outcomes, methodology and the plan.

6.1 - Evaluate test/survey results

The test and survey results for SermonShare showed that the system works as expected. There were a few changes that needed to be made along the way which meant that the testing process did take up a considerable amount of time.

The results from the nibbler test showed good promise for SermonShare, however there would still be room for improvement if it was to be worked on in the future.

The user survey results were very insightful as they informed the developer as to what changes could be made in the future and gave some direction as to what users didn't like or didn't understand.

6.2 - Evaluate project outcomes

The initial aim of SermonShare was to create a system that would better allow churches to connect with their congregations by uploading audio and video content online so congregation members could view it from the comfort of their own home.

The aim has been achieved apart from having video being uploaded to the service. This was a decision taken by the developer as they didn't feel it was required, it also allowed more time for the developer to focus on just showing audio sermons to users.

6.3 - Evaluate the methodology

Deciding to use the modified waterfall methodology was a great decision. The developer was able to go back on various parts of the system and tweak things as the methodology allowed them to do so. This became very useful especially when testing out the application as it was after this that tweaks needed to be made throughout the system.

The modified waterfall method allowed the project to be structured and rigid whilst at the same time it allowed the developer to go back and fix or update things that needed adjusted.

6.4 - Evaluate the plan

The plan and schedule were defined after the methodology was chosen. The plan broke the project down into steps.

When investigating the aims and objectives for SermonShare, requirements were specified. These requirements did match up to the final system however some changes could have been made.

The technology selection stage was very useful as it gave an insight into what the best tools to build SermonShare with would be. Once the technologies to use had been decided upon, it made the whole project a lot clearer and development could begin.

The implementation stage was fairly straightforward. During the development of SermonShare there were no real issues that affected production. Having to constantly learn new areas of Laravel is were a lot of time went for the project. This ranged from learning some simple techniques to learning a lot more advanced and time consuming techniques.

7 - Conclusion

This final section will reflect on the project as a whole and discuss what future work may be undertaken in SermonShare.

7.1 - Summarise report

In this report it has been documented the stages that took place to get SermonShare off the ground. The report discusses how the idea was initially created and then researched to see that it was feasible. Once it was feasible, aims and objectives were then set in order to prepare the project for development. After development, SermonShare was tested in several different ways to make sure that it was the project that it set out to be. The project was then evaluated and this helped ensure that the project was created successfully.

7.2 - Reflect on what happened

As the project was completed by a single developer, they had to undertake various roles at different times. This was mostly a developer however there were times when the developer needed to become a project manager, a tester and a researcher. This was a lot of work for the developer to undertake, however it showed how through good structure the project was able to be achieved.

After receiving some user feedback from the testing stages, the developer was tempted to go and make various changes, however this was not feasible within the timescale of the project.

Overall the project was completed to the standard that was set out and the requirements were met.

7.3 - Suggest future work

In terms of future work there are a few things that the developer would like to implement.

The first of these is a church directory. This would be a page that return all users who have a church account. Filtering could be added to find your church more easily. If your church wasn't available, there could be some sort of feature that lets you send an email to your church inviting them to use SermonShare.

The second thing that the developer would like to implement would be video sermons. This was set out in the initial aims and objectives, however it wasn't feasible in the timescale and the developer decided to focus solely on audio support only for now.

Another feature that may be added in future work is an emailing system that lets users know that a church they follow has just uploaded a new sermon.

7.4 – Conclusion

Overall as the developer I have really enjoyed working on SermonShare. It is great to see something come alive that was only ever on paper. As the creator, my hope for SermonShare is to go beyond this project and even be used on a national or even global scale.

As a student I have learnt a lot from working on SermonShare. My knowledge of the Laravel framework has really increased and I feel more comfortable with it now whereas before there would have been times I just couldn't grasp what was happening within the framework.

As the developer I will be hoping to add to SermonShare and undertake the tasks set out in future work.

8 – References

Laravel - The PHP Framework For Web Artisans. 2016. *Laravel - The PHP Framework For Web Artisans*. [ONLINE] Available at: http://www.laravel.com. [Accessed 07 January 2016].

The Best Laravel and PHP Screencasts. 2016. *The Best Laravel and PHP Screencasts*. [ONLINE] Available at: http://www.laracasts.com. [Accessed 07 January 2016].

Atlassian Bitbucket 2016. *Bitbucket — The Git solution for professional teams*. [ONLINE] Available at: http://www.bitbucket.com. [Accessed 07 January 2016].

Git. 2016. *Git*. [ONLINE] Available at: http://www.git-scm.com. [Accessed 07 January 2016].

YouTube | Google Developers. 2016. *YouTube | Google Developers*. [ONLINE] Available at: https://developers.google.com/youtube. [Accessed 07 January 2016].

SoundCloud Developers. 2016. *SoundCloud Developers*. [ONLINE] Available at: https://developers.soundcloud.com. [Accessed 07 January 2016].

CSC/ECE 517 Fall 2010/ch6 6d bb - PG_Wiki. 2016. *CSC/ECE 517 Fall 2010/ch6 6d bb - PG_Wiki*. [ONLINE] Available at: http://wiki.expertiza.ncsu.edu/index.php/CSC/ECE_517_Fall_2010/ch6_6d_bb. [Accessed 07 January 2016].

YouTube. 2015. *YouTube*. [ONLINE] Available at: http://www.youtube.com. [Accessed 22 October 2015].

iTunes - Apple. 2015. *iTunes - Everything you need to be entertained. - Apple*. [ONLINE] Available at: http://www.itunes.com. [Accessed 22 October 2015].

SoundCloud. 2015. [ONLINE] Available at: http://www.soundcloud.com. [Accessed 22 October 2015].

SermonAudio.com. 2015. *SermonAudio.com*. [ONLINE] Available at: http://www.sermonaudio.com. [Accessed 22 October 2015].

Sermon Central, 2015. *SermonCentral- free sermon preparation, sermon illustrations, church videos, PowerPoints, church media resources, preaching articles for sermon preparation ideas.* [ONLINE] Available at: http://www.sermoncentral.com. [Accessed 22 October 2015].

Home | Preaching.com. 2015. *Home | Preaching.com*. [ONLINE] Available at: http://www.preaching.com. [Accessed 22 October 2015].

9 – Appendices

Feedback Questionnaire

Thank you for using SermonShare. If you have time we would value your feedback on your user experience via the following questions;

(Please highlight your answer for multiple choice)

1) Who are you?

User Account

Church Account

2) How did you hear about SermonShare?

Congregation announcement

Social Media

Other

3) What was your first impression of SermonShare?

Very simple looking in terms of colours and logo, easy to understand.

4) Did you find SermonShare easy to navigate?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

5) Did you find the style and size of the text easy to read?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

6) Did you find it easy to upload your sermons onto SermonShare? **(CHURCH USER ONLY)**

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

7) Were you happy with the quality of the audio of the sermons you have uploaded or listened to?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

8) Did you enjoy and find this service provided by SermonShare useful?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

9) Would you recommend SermonShare?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

10) Do you have any other comments about SermonShare?

N/A

Thank You

Feedback Questionnaire

Thank you for using SermonShare. If you have time we would value your feedback on your user experience via the following questions;

(Please highlight your answer for multiple choice)

1) Who are you?

User Account

Church Account

2) How did you hear about SermonShare?

Congregation announcement

Social Media

Other

3)What was your first impression of SermonShare?

Nice interface

4)Did you find SermonShare easy to navigate?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

5) Did you find the style and size of the text easy to read?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

6) Did you find it easy to upload your sermons onto SermonShare? **(CHURCH USER ONLY)**

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

7) Were you happy with the quality of the audio of the sermons you have uploaded or listened to?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

8) Did you enjoy and find this service provided by SermonShare useful?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

9) Would you recommend SermonShare?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

10) Do you have any other comments about SermonShare?

Makes it easier for or church to share what is happening on Sundays with our congregation. Some of our members can't make it.

Thank You

Feedback Questionnaire

Thank you for using SermonShare. If you have time we would value your feedback on your user experience via the following questions;

(Please highlight your answer for multiple choice)

1) Who are you?

User Account

Church Account

2) How did you hear about SermonShare?

Congregation announcement

Social Media

Other

3) What was your first impression of SermonShare?

Easy to use

4)Did you find SermonShare easy to navigate?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

5) Did you find the style and size of the text easy to read?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

6) Did you find it easy to upload your sermons onto SermonShare? **(CHURCH USER ONLY)**

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

7) Were you happy with the quality of the audio of the sermons you have uploaded or listened to?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

8) Did you enjoy and find this service provided by SermonShare useful?

Strongly Agree

Agree

Don't Know

Disagree

Strongly Disagree

9) Would you recommend SermonShare?

Strongly Agree

Agree

Don't Know

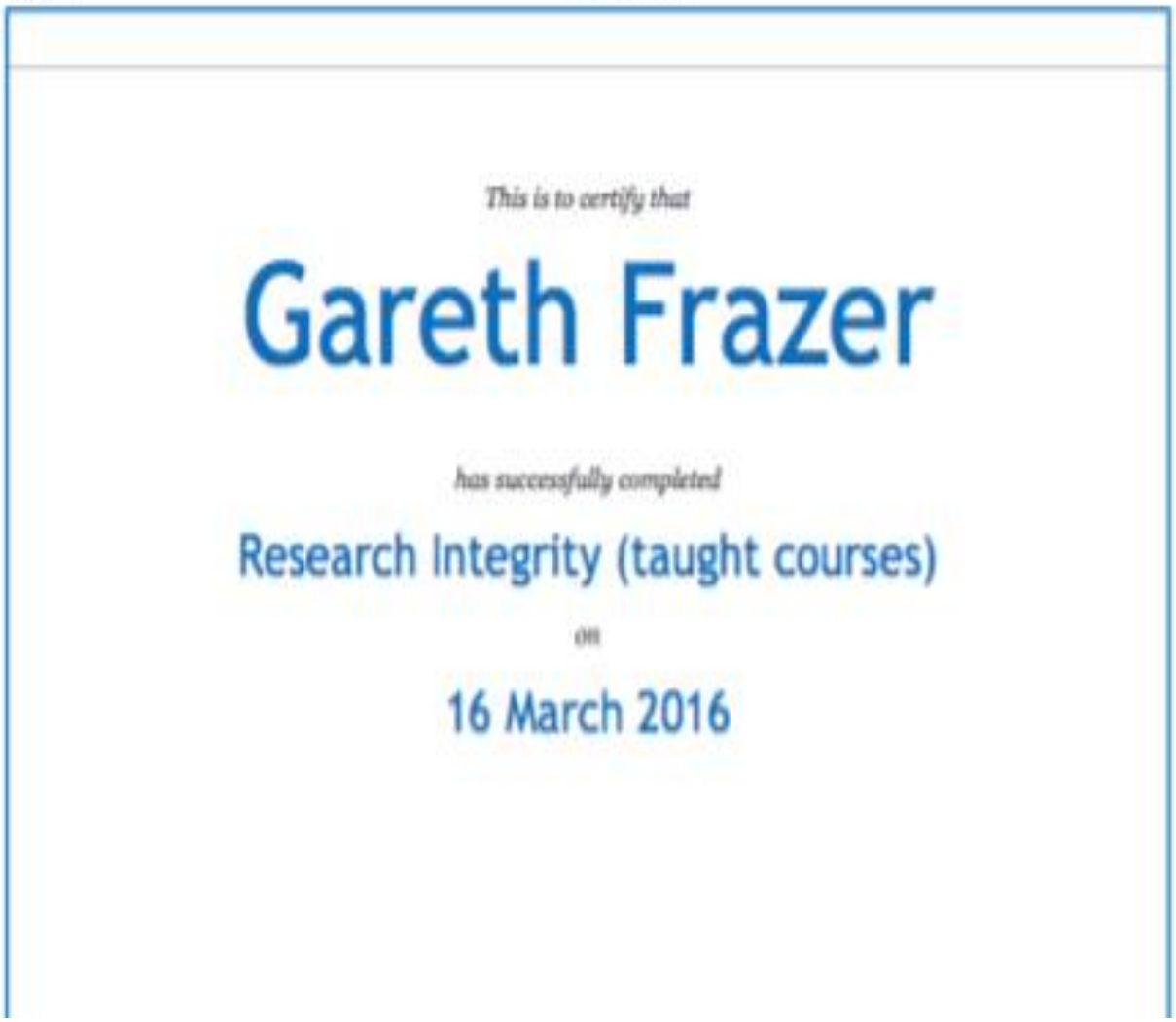
Disagree

Strongly Disagree

10) Do you have any other comments about SermonShare?

I enjoyed using the service and would recommend to other people. Being able to stay in touch with your church is good for when you miss a Sunday.

Thank You



This is to certify that

Gareth Frazer

has successfully completed

Research Integrity (taught courses)

on

16 March 2016