DUBLIN INSTITUTE of TECHNOLOGY
Institiúid Teicneolaíochta Bhaile Átha Cliath

Total GAA (GAA Pitch Finder)
Final Year Project Report

DT228
BSc in Computer Science

Martin Quinn
Richard Lawlor

School of Computing
Dublin Institute of Technology

06/04/17

Abstract

People who are involved in the Gaelic Athletic Association of Ireland whether it be players, supporters, managers or coaches are faced with the weekly dilemma when travelling to matches, of finding the various playing grounds of the numerous clubs and teams around the country.

The time in which it takes to find the location and research the directions forces people to depart earlier to games, which may impede on their preparations for that said game. The reason why people have to leave early is quite simple, they are compensating for the time that they will inevitably spend lost, aimlessly driving around a rural town in order to find a local pedestrian to set them in the right direction. From my own personal experience and from my research this happens too often to too many people and needs to be addressed. I feel like I have found a solution to this widespread issue. This application was developed to help people find their way, in a clear and accessible manner .

The aim is to target all GAA audiences and make this application as user friendly as possible. Each user will have the option to search the playing grounds by 'Club name' or by 'County'. To put it simply; this application is going to save people valuable time, it will hopefully decrease the amount of stress and confusion they face and with any luck will allow those to furthermore enjoy their time spent watching and playing GAA.

I also want the application to provide an added service for those managers and players who seek to keep their GAA affairs (i.e. training times and locations for teams, results and fixtures for games) in one clear easily accessible location online within the application "GAA Pitch Finder". This application has the potential to grow and develop furthermore to suit the needs of all involved in the GAA on a club level.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Martin Quinn

06/04/17

Acknowledgements

I would first like to thank my supervisor Richard Lawlor who was very helpful during the time developing this project. A huge thanks also to everyone who did testing for me during the project, especially my hurling manager, Mark Lynch who tested and used the application. I wouldn't have been able to do it without you.

A special thanks to my friends and family who were there for me when I needed them and helped me keep my focus and complete the project to the best of my abilities.

Contents

Introduction

Since its foundation in 1884 The Gaelic Athletic Association has been and continues to be a huge part of Irish heritage and Irish culture. Irish people are very proud of their national sports, Hurling and Gaelic Football. The GAA is Ireland's largest sporting organisation. It is celebrated as one of the great amateur sporting associations in the world [1].

GAA also carries with it a lot of sporting history and players today inspire youth to become better people and more rounded human beings. In my own experience, being involved with the GAA from a young age has given me the opportunity to grow personally as well as skilfully, It has taught me about teamwork, responsibility, dealing with loss and with victory. Recent players that inspire me, such as DJ Carey of Kilkenny, Joe Canning of Galway, John Mullane of Waterford, are legends of a mere amateur game, yet engrave a true and solid mark of the annals of GAA history and have the undying respect of millions around Ireland and the world. The GAA museum in Croke Park is home to the official GAA Hall of Fame, which celebrates former players who have made a unique and exceptional contribution to hurling and football [2]. The vast numbers of visitors that come and experience what the GAA museum on a regular basis in Croke Park shows enormous amount of people who are interested in and connected to the GAA.

Gaelic games are Ireland's amateur sports, but there's nothing amateur about the style of play. Every September, tens of thousands of fans, bedecked in their county colours, make the pilgrimage to Dublin's Croke Park to see the All-Ireland Football and Hurling Championship finals [3]. This is a massive occasion for every GAA fan to view a spectacle such as an All-Ireland final. Out of all the 32 counties of Ireland at this pinnacle of the player's careers lies a great opportunity to reign as All Ireland champions, to stamp their county firmly on the map and create a legacy that only champions can continue. This journey starts at a young age, at a club level, in the training they join in on a daily basis and the matches they attend on a weekly basis, each of which involve finding the appropriate playing ground. The All-Ireland final is an occasion where everyone is on the edge of their seats no matter what counties are playing on the sacred ground there exists an inner passion for the GAA. The atmosphere, the roars, a concoction of emotion trapped inside a battleground holding over 80,000 screaming fans. It's a true awesome moment in a GAA supporter's life, trekking to Croke Park, bags packed, sandwiches made, face painted, hats, flags, headbands, and a smile from ear to ear to see their county perform to the highest standard.

A counties' journey to the top tier begins small and continues to grow through the clubs that inhabit that county. Every club is considered when selecting a panel to represent a county team. Development of players starts as young as 5 years old. Donaghmore Ashbourne a club in Meath focuses strongly on the development of these young players so that they have a real chance to reach the biggest prize of playing in Croke Park. Many other clubs follow the same line. There is nothing quite like playing in front of hundreds of loyal club supporters at your home ground who you've grown up with,

played with, or even shared the same house with, sisters, brothers, parents, and friends. All of their support drives a player to victory.

There is an issue that arises though, when club matches take play on the opposing teams' home ground. Many of the supporters are less likely to attend when its an away match, there is a significant drop in the number of supporters when it's an away fixture. The reasons for this can be that transport is an issue, the weather can also affect the turn out, but from my research and experience the most substantial issue is most of all the stress and confusion involved in trying to find the correct playing grounds of the opposing team. For example Castleknock GAA in Dublin have four separate playing grounds. If one was to search for 'Castleknock GAA pitch' on many of the various search engines, Google, Bing, Yahoo, any of the four could come up, supporters could set off for their journey and arrive only to realise that they are in the wrong place. My application offers a solution to this issue, as when 'Castleknock' is searched, each of the four pitches will appear and the user has the option of choosing the appropriate pitch before they set off on their journey.

The aim of this project is to produce a simple 100% accurate pitch location finder application of any club in Ireland, no matter how rural or remote the pitch is, this application will find it. The project is mainly aimed at those supporters who want to be there 100% of the way and see their players grow and develop throughout the younger age groups through to the older age groups. This is not to say that this is the only target audience, if someone is new to the GAA and wants to attend GAA matches, they are going to inevitably face the challenge of finding theses playing grounds also. My app is designed to eliminate any issues involved in finding the playing ground of any club in Ireland. Anyone who is venturing to the grounds of obscure club that is unknown to them whether it be for a challenge match or a league game this application is the one to use. Especially when someone is under pressure with time or they are running late and want to get to a game before throw in, this app is certainly for you, it will identify and produce the fastest route to guide you to your correct destination.

Another part of this application is that managers, players and coaches will be able to keep track of what is happening within their own local GAA club, this will be located within the application, in a detailed and organised fashion. This could facilitate many uses for the players, supporters, and especially managers of a team. To achieve this there will be a web application where a user can login to their account and post the information in a secure environment, similar to a blog for only those who have access with the login. This can be accessed in conjunction with the 'GAA Pitch Finder' application. One of its main uses could be that all members of the team have access and log into the information about the fixtures- upcoming matches at any time. The coaches and managers at anytime have the opportunity to amend and make changes or update the 'Team page' or add new information on matches.

Project Objectives

The main objectives of the project:

- Develop an Android and iOS application to locate any GAA pitch in Ireland.

- Develop a Web application to maintain a user's posts.

Additional objectives of the project:

- Create a section in the mobile versions for allowing posts.
- Create a section in the web application and mobile applications to show results and fixtures of selected leagues and teams at inter-county level and club level.

Project Challenges

As with any given project there will no doubt be many challenges to overcome, many issues to address and many problems to fix. These issues are best ~~to resolve~~ resolved sooner rather than later. It is my intention to reflect on these issues continuously throughout the applications development and when faced with a challenge I consider all practical solutions and then begin creating the most suitable solution.

The first set of challenges surface when acquiring the information of geographical location of each club with their exact co-ordinates. The main points of vital information needed is that of:

- the club name,
- the county,
- and the co-ordinates of the pitch.

The next challenge is storing this information. The database will be the center of three applications on three different platforms. The information must be flexible and allow modification if needed. The information is all available online the way in which to acquire the data is essential to the progress of the project.

Challenges I expect to find in development:

- Creating an API to link a database to each application without having to create a database for each application.
- Developing a user friendly environment to cater for all age groups and allow a simple flow to the application, find the club, locate the club.
- Making the information on the posts relevant to a player, manager, supporters' needs

Risks

The main risk that accompanies this project is that there is a lot of platforms targeted for development, which may leave the additional objectives behind. There may not be enough time or attention paid to the side objective of the project, such as the fixtures and results section of the applications.

Another risk is if there are too many problems or challenges with one platform the time put into the next may be limited meaning the result would not be to the same standard as the initial platform.

The development of the main database could run into issues if the data becomes compromised each application takes a sever blow when live. This leads to another problem of creating new databases for each platform and application which would be redundant.

Structure of the document

Research & Technologies

This describes the GAA and the problem areas this application is trying to solve. It also shows details of what preparation is needed for the project to succeed, and the technologies used in the development of the project.

Design & Architecture

This describes the design of each application at a high level including use case diagrams for each interface developed, the design of the database, the architecture and the different project components. The design methodology chosen is also described.

Development

This describes the development process of the project. It describes the design and implementation of the Graphical user interfaces for each. It also discusses the problems discovered during developments and provides the appropriate solutions.

System Validation

This part of the document shows how the applications were tested and the feedback provided from users of the system during testing and deployment.

Analysis & Conclusion

This chapter describes how the project has changed and evolved since its first inception, discusses possible future work, and changes to the project. It also discusses and evaluates the technologies and methodologies used in the project.

Research

Problem Research

Traveling in a rush is stressful, traveling when not knowing where to go can also be very stressful, and being late is most of all, stressful. In a survey carried out by “A. Vogel” [4], being late resulted in the number one cause for stress [5]. What happens to your body when you are under stress could be harmful to your health. Being late is one of those things that stresses you out when it really shouldn't, whether you're late for work or to meet a friend, if you're a generally late person, you're probably going to start feeling some stress [6].

Being late can make your body:

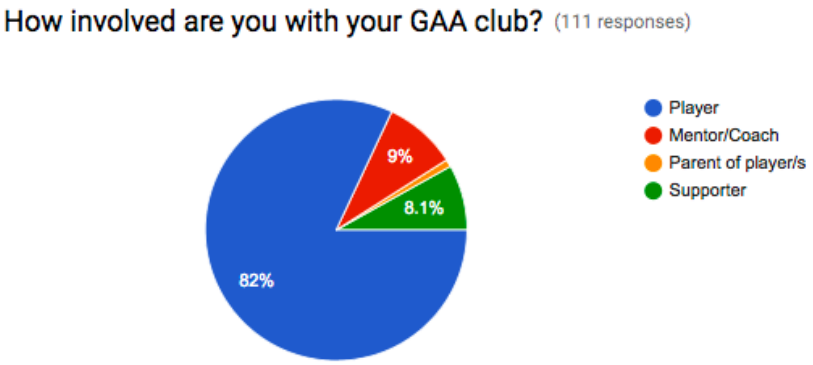
- Go into “fight or flight” mode.
- Your stomach may become upset.
- Your blood pressure will elevate.
- You might get chest pain.
- Your body will start sweating [6].

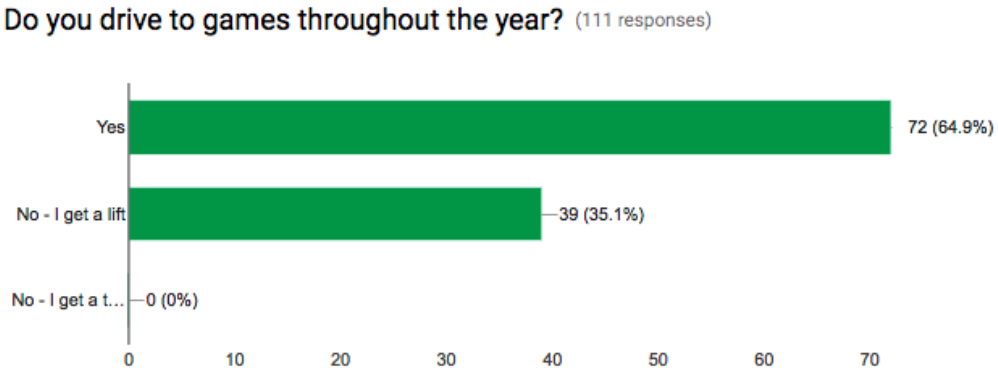
The objective of this project is to eliminate the factor of being late to a match. Which in turn then removes the stress which may accompany the idea of going to see matches. If a person had gone to see a match, in a hard to find place, who then got lost and was rushing to get there on time, this could impact the chances of them going to the next match. People don’t like stressing, and why would someone put themselves through the effort and torture of that experience again. They may just stick to the home games from then on. This application will aid everyone who has this issue.

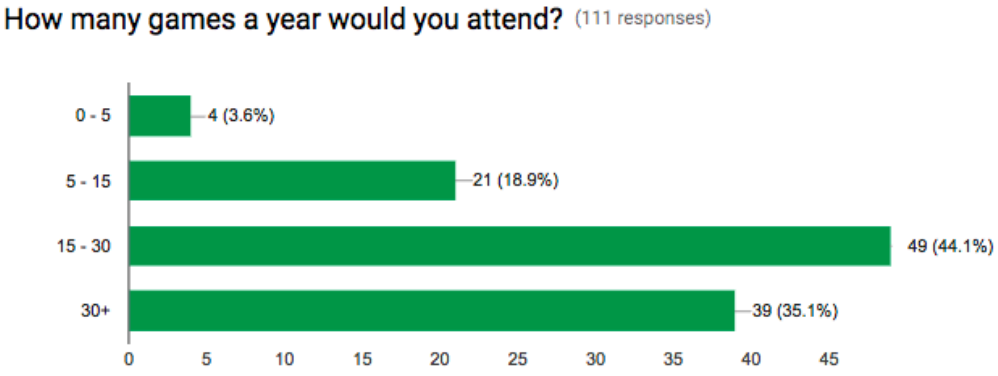
Alison Cullen, A. Vogel nutritional therapist, said: “Ongoing stress causes the body to put everything on hold except immediate survival. Areas such as fertility, detoxing, and immune cells patrolling to check for infections are neglected. The result is more cold and flu infections, which in turn cause more stress. Many people neglect their health because their schedules are so pressurised; ironically, though, spending a little time on your health can save spending a lot of time being ill.” [5]

GAA Survey

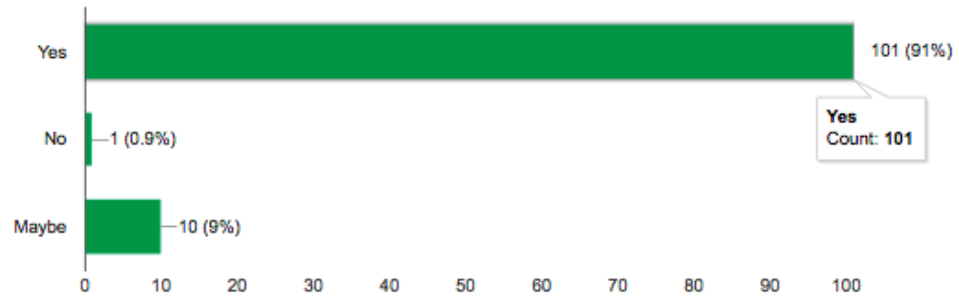
To gain an insight into what the target users would like to see in this application, a survey was carried out, there was a great response from this and seems that people are very interested in having this created to help them through the year to make GAA even more entertaining, less stressful, and fun for everyone involved. [7]



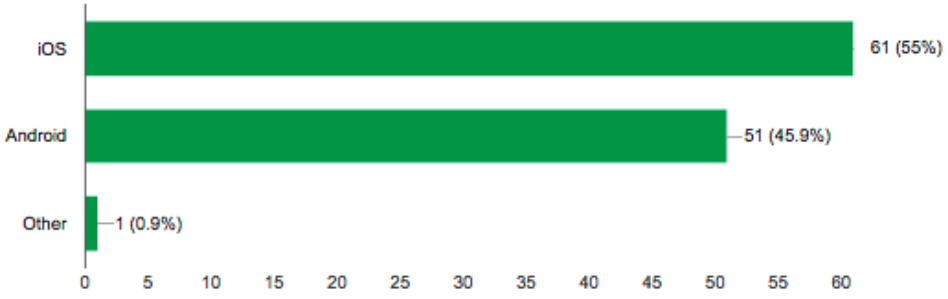




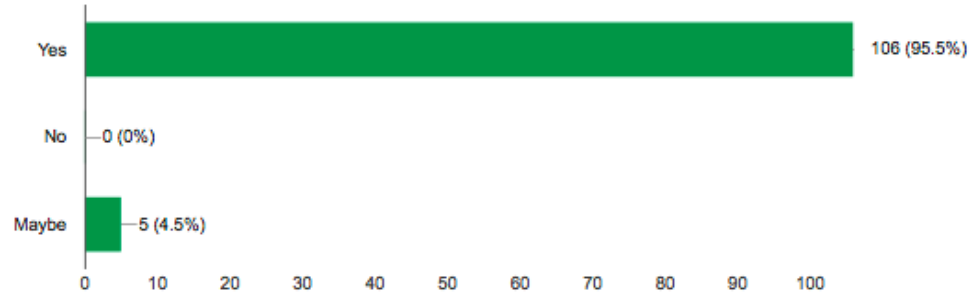
Would you like to see a central location mobile app for all club locations in your County?
(111 responses)



What mobile device do you use? (111 responses)



Would you like the app to include information on your Local Club team and their upcoming fixtures?
(111 responses)



The results show that just under 80% of the people who participated in the survey attend over 15 matches a year. many of these matches would be away from home grounds meaning they would have to either all leave together in a convoy or ask someone who knows the way for directions if they do not know where exactly the pitch is. There are many occasions where when travelling around the country for different games when playing with the County teams that you get lost most of the time, this means being late and potentially being in the manager's bad book for not preparing yourself correctly which in turn stresses you out. [7]

It is very hard trying to travel to a pitch in the middle of nowhere during the year either after work or college, when you're rushing and trying to figure out where exactly to go. This app would be a life saving tool in most cases for many players in particular and the percentage of people who attend these matches may increase. [7]

Another finding from the survey was that the split in devices is divided between iOS and Android, in Apples favour so an application in iOS would be a good approach. If the application was to be on just one platform many users would miss out and it would frustrate them, knowing there is something out there for them to use but are unable to use it. [7]

The most important finding from the Survey is that an overwhelming amount of players and followers of the GAA would like to see this application implemented. In total 101 out of a total 111 would like to see this app surface. A section in the app for the user to 'favourite' a team and then they will be able to see either the results or fixtures of that team should be implemented. This can be done by using the various County GAA websites that provide that information, the application should display this information and also include notifications and updates to the user, this is however a low priority feature but when the app's main objectives are complete this development can begin. The survey shows that more than 95% of the responders would like to see this implemented. [7]

Ease of Use

Ease of use is at the bottom of the list in IT application development. But if you do it well, you will do well with your users [8]. This application needs to be very easy to use and ideally the user should not spend longer than a minute trying to find their destination. To achieve this there are many factors to consider and evaluate.

User involvement

No one knows better than your end user if an application is easy to use—or whether its workflow should be modified for improved ease of use. This is why more IT departments are actually dividing their QA functions into two sections:

- A team that checks the technical excellence of an app
- An end-user team that verifies ease of use and usability

No matter how well developed an app is, if it isn't easy to use, it will sit on the shelf [8].

Fit with the business process/environment

One of the first usability principles for a new app or technology is its fit with the environment or work process it's being designed for. For instance, the first impulse might be to look at tablets or mobile phones for all field-based business—but a police officer or a cab driver gets best reliability in unpredictable situations from a ruggedized laptop. And handheld units (even though they are more expensive) are often the best ruggedized option for a warehouse or yard worker [8].

Ease of navigation

Users will give up on an app if they have to navigate through three or four levels of menu choices. They'll also give up if the app has a busy display with no clues as to what the workflow is. As part of your application prototyping process with end users, cover the proposed workflow of the application as well as its features and functions [8].

Pleasing displays

Pleasing displays are important to users. What "pleasing" means to most users is a web-like interface for an app that is laid out and orchestrated like most other web apps and that's intuitively easy to understand and use. The aesthetics of the display (e.g., the number of images and text lines and the amount of blank space—and how they're arranged), as well as the display load time are all important application touches that end users should review and agree upon before the application goes live [8].

Recognizable error messages

Most of us have received canned error messages like: "Error 207, data incompatibility" — or some other cryptic message that is generated from the depths of an internal programming routine that no end user can possibly understand or take action on. As part of your QA process, carefully test the error message generation from the application. Error messages should be in plain English and the end user should know immediately what he or she must do to get out of the error situation and complete the work that was underway [8].

Thorough testing

To meet tight deadlines, IT has a tendency to cut its test and checkout of apps short—but you want to avoid an application breaking or a deluge of phone calls from users saying that certain functions in the app are missing or don't work. To prevent those occurrences, never take time away from QA. In addition, include end users in quality assurance, final checkout, and signoff. The calls you save at the help desk are well worth it—and a consistent application that is predictable and works every time will go far in establishing high usability with end users [8].

"Fit to form" applications

Enterprises develop applications for a plethora of platforms that reflect the many different devices their business users are using these apps on. Consequently, a new product configuration application must be able to perform well on a desktop computer, a laptop, and potentially any number of Android, iPhone, Blackberry, and other mobile devices. The ability of a device to support an application varies on an individual platform basis. A desktop or a laptop computer will be able to carry a more fully featured version of an app than an Android, iPhone, or tablet with a smaller technology footprint and a smaller screen. For this reason, any application that will be deployed on multiple devices should tested for ease of use and form and fit on each device [8].

Centralized control for security, lockdown, and new software releases

Any app deployed on mobile devices and laptops should come with security and control that enable headquarters to track devices that get lost or displaced. If the device can't be located, central IT should have the technology to lock down the device and disable it. The same centralized control should be exerted over in-field devices by pushing out new releases of software that is either updated automatically or by permission of the end user on his/her device. That way, all devices stay current with the same levels of software and IT technical support and IT has to manage only one version of software at a time [8].

Auto fill-in fields

If end users enter a customer record, it's convenient if the entire record detail can appear on the screen. This gives them the choice to accept that record or to modify it. It is far more tedious if they have to keep entering repetitive data fields. Meet with end users during app design to determine which data fields should be automatically supplied by the system. This will save user frustration, time, and key strokes [8].

Favoured applications as models

Many times, IT is asked to replace older applications that end users are displeased with—and there is a deliberate intent to make the new application markedly different in what it does, how it looks, and how it behaves. But there are also cases where end users are happy with their existing applications and simply want a new application that can conform to the basic workflows and styles of the original apps [8].

The main users of this application will not have time to stress about some bug or inconsistency within the app they would rather not use this application. This frustration

is often found with old style people with simple demands and a slight disregard for technology. This can lead them to getting frustrated with technology and just avoiding it completely.

Frustration with Technology

A research project was conducted by three American universities and according to the results up to 45% of employees' time spent using technology is wasted because things can so easily go off the rails. And although these frustrations have common roots, employee reactions to tech issues vary widely [9].

The Apologizer

This personality type submits and sends documents with a callout apologizing for low-quality work, refuses to ask for help, doesn't take advantage of technology investment, and paints technology as the scapegoat.
Business Impact: Sending incomplete, incorrect, unformatted work negatively impacts the work-product. Weak, apologetic communication of tech frustration harms business relationships [9].

The Interrupter

They disrupt concentration by asking others to accomplish routine technology tasks. This personality also interrupts other employees' workflow and spends more time figuring out small tech issues than on substantive work solutions.
Business Impact: Constant requests for help decrease productivity of the employee. Constant interruptions to colleagues' workflow impacts enterprise productivity, and wasted focus on small, easily-avoided tech issues takes time away from critical business issues [9].

The Delegator

They pawn off work on co-workers, asking them to complete the task instead and waste two employees' time at once by refusing to learn new technology and frustrating others [9].
Business Impact: Delegation keeps colleagues and employees from performing their own business-critical tasks, which results in a duplication of efforts without the reward [9].

The Quitter

This personality gives up completion of critical tasks because of tech complaints, which creates inefficiencies. They view technology as non-essential and cause others to follow suit [9].
Business Impact: They refuse to learn and their unwillingness to adopt technology slows business growth. Employee withdrawal from their task creates abandonment of business goals, absenteeism and even turnover. They refuse to pursue new ideas out of fear that technology will become a barrier and prevent individual and enterprise-wide innovation [9].

The Exploder

They view technology as the enemy, throwing tech-fuelled temper tantrums through furious key-smashing, laptop computer-slamming, violent phone-shaking and forceful button-pressing [9].

Business Impact: Aggressive displays of frustration halt productivity and Create an unhealthy work environment [9].

The personality that will mainly feature when using this application will be “The Exploder”, as the user interacts with the application the goal is too limit the complexity of the layout and make the task very apparent and obvious. If a deviation occurs from the main objective, it may be too late, phones and laptops could go flying. Another aspect that could frustrate the user is the performance of the application, any blip or dip in performance could trigger the user into thinking the device is broken or not responding. This could then in turn lead them to get stressed and ditch the idea of the Application.

Time Management as a Manger, Coach, or Player.

The importance of planning and preparing as an athlete in Ireland has increased greatly. There are more players of the GAA spending hours of their own free time to build and prepare themselves for a season. The players are doing more in the gym, more on the pitch, and much more in the nutritional field. These amateur athletes are acting and playing like full blown professionals.

Some teams who struggle needs guidance and that's where a manager can come in and help with their preparation and guide then to what's right and wrong in regards to their sporting careers. It would be nice to have a centralized location for all matters the team could have, where all players could speak out and have a voice without being distracted by other variables such as social media and irrelevant clutter. To have a place where time management is listed for the players to see and all the season plans are drawn up and in place a hub of all things GAA. This application aims to provide this luxury.

There are some key aspects to time management and in regards to being a manager of any team, these are the most important of all.

Goal Setting

Time management is not a standalone skill. You can only manage how you use your time and how you use your time should be driven by effective goal-setting. When a team is first distinguished a set of players are at a manager's disposal. As a manger their job is also to set realistic goals with the present talent. Given the time you have with your team it could be from November to September when the season most likely ends a manager will have to set goals with the time they have. The most fundamental of time management skills is the ability to use your time in a manner which serves your goals [10].
When making decisions about what to focus your time on, you should always be knowledgeable of your goals and how each action is aimed at bringing you closer to achieving those goals [10].

Prioritisation

The biggest reason that most people struggle with prioritisation is that they start too late in the process. They attempt to prioritise the items that are on their task list. However, if you look closely at most task lists, you will find that they contain items which never should have made it on to the task list in the first place. Managers may have instructions from a club to improve a certain aspect of their current team, but this could be a problem hiding elsewhere so a manager may have to dig deeper and pick out the smaller problem that will gradually improve the larger problem [10].
As strange as it may sound, prioritising should not begin with a focus on getting more work done. Prioritisation should always begin with avoiding/eliminating the tasks which you should not be performing. Once this has been done, you can switch your focus to completing the most valuable work you can with the time and resources available to you. Prioritisation is one of the most misunderstood and misused of the time

management skills. When you get it right, you will find that your time management improves rapidly [10].

Focus

Regardless of what you are trying to do, there will always be something else competing for your attention. It's not easy to shut everything out and focus on the task at hand. Focus is one of those time management skills where you don't realise how important it is until you struggle with it [10].
It is important to remember that no matter how many tasks need to be done, you can only work on one task in any given moment. The myth of multi-tasking causes many problems for those who wish to improve their time management but if you want to get results, you must learn to focus on one task at a time and block out all distractions. As a manger this may be difficult with many outside opinions and inputs but it is crucial for them to stay focused and complete the objectives [10].

Decision making

It would be nice to think that you could just sit down as manager and choose your team without having to put any serious thought into it. Alas, there are few jobs that fit that description. You will have to make important decisions and will have to ask yourself some questions, for example:

What team are we playing?
What players are available? (injuries etc.)
What is the best approach?
Who needs most attention/guidance as players?
How will an outside force affect the team, how do I react? (Media etc.) [10].

If your decisions only affected you, it wouldn't be such a big deal but as a manager your actions will impact the team and those involved. Almost every task has a knock on effect on another person, or task, which means that every decision that you make has consequences both for you and for others [10].
Decision making is one those time management skills which if you are not good at it, you will notice the negative impacts in every area of your life. It is imperative that you are able to consider the consequences and make effective, clear decisions [10].

Planning

Tasks will overlap and be dependent on each other. There will often be times when one task cannot be started until another task is finished. Such as a fitness session can not be performed until a team has adjusted to playing again, this could risk injury and affect your team's morale. Your schedule will also be impacted by the schedules of others. An example of this is if pitches are not available due to not planning and booking a pitch in advance, as there are many other teams in any given club that need to train. These factors need to be considered at the beginning of each project and, monitored throughout. Failure to do so can lead to delays and missed deadlines [10].

Planning is one of the essential time management skills because it allows you to foresee all of the tasks which will be required to complete a project and, how they will best fit together. A well made plan will save you a great deal of time [10].

Communication Skills

You will have to work with others on a daily basis when managing a GAA team, there will be a couple of mentors with you and of course the constant contact with the team looking to you for any guidance or with any issues. It is unlikely that you will perform every aspect of your work so you will need to enlist the help of others. Strong communication skills will enable you to build supportive relationships with those whom you work with. You will be able to work better together and achieve more than you ever could apart [10].
When you require another person to do some work for you; you will want to communicate in a manner which will enable them to perform the work to the desired standard, in the fastest time. Should any errors occur, you will want to raise the issue quickly and explain clearly about the adjustments that need to be made, for example a drill which involves heavy lifted or speed work as these are areas that are very important in modern GAA. In these situations, the quality of your communication directly impacts the quality of the work that gets done, using a group plan would help this greatly [10].

Working effectively with others

Working effectively with others is often the quickest way to get a job done properly. This is about more than just communication and delegation. It is important to understand how others like to work; their goals and expectations. As you get to know people better, you build positive relationships where you can work together for the benefit of all concerned. An example of this would be the way in which players like to warm up and eat before games, this can be different for others as some foods don't work well with some people and some warmups don't fit well with other people's routines. It is important to reach a happy medium with all players and management, where everyone works to their potentials [10].

Record Keeping

When you are on top of everything and you know exactly what is going on; you can make effective decisions and provide information quicker. Regardless of your subject area, accurate information is essential. You do not have to know everything off the top of your head but you would be surprised at how much time you can save when you know where to find the necessary information at the moment you need it.
You must determine what information you need to have and put systems in place to ensure that it is collected and stored. This application will provide the means of all records being kept safe and secure for all those authorised to view them [10].

Patience

Many people think that time management skills are all about getting more work done and winning matches and championships is the only indication of success. That is not the case. Time management skills are about ensuring that you get the important work

done that will guide you to the ultimate goal. You could try to focus on getting more done but you end up rushing things, making mistakes and consequently stunting progress. By the time you have rectified the mistakes you will have spent more time than if you'd taken your time and done the job properly [10].
Patience isn't just a virtue; it is a skill. It is something which you have to practice. The very best time managers do not rush things. They have patience and take precisely the amount of time required to do the job properly [10].

With all these aspects considered this application will create an environment to cater for all the needs of the manager. Each session, each match, each play, each squad will be accessible, documented, securely stored for those involved in the team.

Existing Solutions

GaaPitchLocator.net

GaaPitchLocator is a website that allows the user to locate all clubs around Ireland. It has a section in which you can choose the county from a map in an animated Ireland map and then from there find the club in a list on the next page. On the home page it gives the user information on how many clubs are in each province and how many clubs are in total. At the very top of the website it has a slide show of a couple of photos in relation to the pitches in Ireland. You can log in to the website also and can buy merchandise such as posters. The main part of the website is the map of where the clubs are [11].

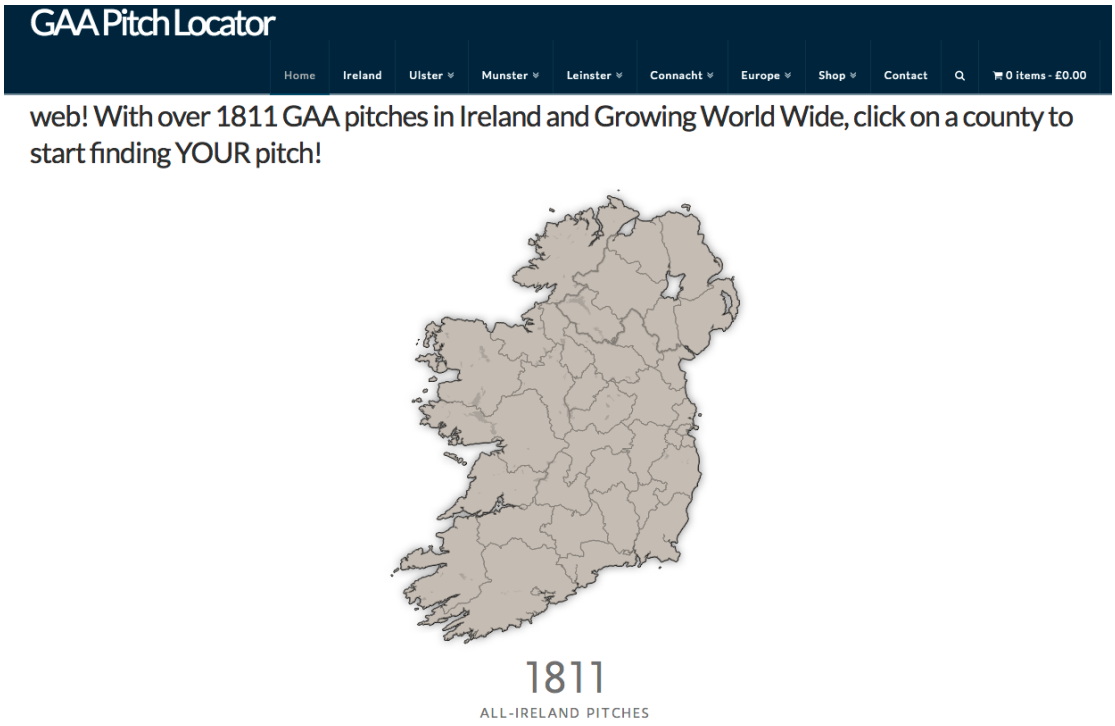


Fig 2.1 GAApitchlocator.net home page (http://www.gaapitchlocator.net) [11]

There are no accurate android or iOS applications that show all clubs in Ireland. Which is quite surprising considering the market that is there for people who follow the GAA.

Technologies Researched.

Final Decision

The Technologies researched includes iOS, android, and Django. Android uses Java with an SQLite database in the Android development kit. iOS which is built in their very own language Swift is based off Objective-C and also uses a SQLite Database. Django which is a framework used to develop web applications using Python and has a SQLite database also. The database should be stored on a server in the cloud with SSD storage. This will be essential for user experience to make sure that there are no performance issues.

MongoDB is the database that suits the project. It being very heavily documented with a large community allows the best usage of the technology. A server that was chosen is “DigitalOcean”, They tick all boxes. The server can be accessed from any PC so that when using the app on mobile the requests will be from this server.

The advantages of using a database on a server is that there is a centralised location for the information to make the applications lightweight and very manageable. Although this may become an issue if at any stage the data becomes compromised, as the data will not be available for any of the three applications.

Additional Technologies Researched

Digital Ocean

DigitalOcean is a simple and robust cloud computing platform, designed for developers. DigitalOcean's command centre makes configuring cloud servers and development frameworks simple and intuitive. DigitalOcean's community pages are continually updated and moderated. Developers will find numerous tutorials, a robust Q & A section, and myriad community projects. DigitalOcean's transparent and simple pricing model starts at $5 per month and offers hourly billing. DigitalOcean's servers use only high-performance Solid State Disks. Their speed directly benefits the performance of hosted web sites and applications. All DigitalOcean cloud servers are live and configurable in less than one minute. [12]

Linux distributions/one-click installations – DigitalOcean offers six popular Linux distributions that can be automatically pre-installed upon deployment of a server: Ubuntu, CentOS, Debian, Fedora, CoreOS and FreeBSD. A multitude of application packages can be installed with just one click, including LAMP, LEMP, and MEAN; also development frameworks Ruby on Rails, Django and Docker; as well as popular applications like eCommerce software Magento, CMS solutions Wordpress and Ghost, and MediaWiki. [12]

MongoDB

MongoDB is a free and open-source cross-platform document-oriented database program that stores flexible document data model similar to JSON. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc. and is free and open-source. Documents

stored may contain one or more fields, includes array, binary data or sub-documents which allows for development teams to gradually develop the data model as the requirements of the application change. Unlike most NoSQL databases, MongoDB provides comprehensive secondary indexes, including geospatial and text search, this will prove very useful with my selected project as well as extensive security and aggregation capabilities. MongoDB provides the features you need to develop the majority of the new applications your organisation develops today. [13]

Pros

- Document validation
- Encrypted storage engine
- Common use cases:
- Mobile apps
- Product catalogues
- Content management
- Real-time apps with in-memory storage engine
- Reduces time between primary failure and recovery. [14]

Cons

- Doesn't fit applications needing complex transactions
- Not a drop-in replacement for legacy applications
- Young solution: software changes and evolves quickly. [14]

I researched many other database technologies before settling for MongoDB, one of which was MySQL, which I have used in the past.

MySQL

MySQL is an open-source relational database management system (RDBMS). The MySQL development project has made its source code available under the terms of the GNU General Public License. MySQL runs on virtually all platforms, including Linux, UNIX, and Windows. Although it can be used in a wide range of applications, MySQL is most often associated with web-based applications and online publishing and is an important component of an open source enterprise stack called LAMP. LAMP is a Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. (Sometimes Perl or Python is used instead of PHP.) [15]

Pros

- Host-based verification
- Can be used even when no network is available
- Flexible privilege and password system
- Security encryption of all password traffic

Library that can be embedded into standalone applications
Provides server as separate program for client/server networked environment. [14]

Cons

- Acquired by Oracle:
- Users feel MySQL no longer falls under free and OS
- No longer community driven
- Members can't fix bugs and craft patches
- Falls behind others due to slow updates. [14]

PostgreSQL

PostgreSQL was another Database technology I looked into using but I felt like it didn't suit my type of project. I felt like there would be many aspects of this technology that would be unused and would be pointless in using.

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL:2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. It has native programming interfaces for C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, and exceptional documentation. [16]

Pros

- Create custom data types and query methods
- Framework enables definition and creation of own custom data types. [14]
- Runs stored procedures in more than a dozen programming languages:
- Java, Perl, Python, Ruby, Tcl, C/C++, and its own PL/pgSQL. [14]
- GiST (Generalized Search Tree) system
- Brings together different sorting and searching algorithms:
- B-tree, B+-tree, R-tree, partial sum trees, and ranked B+-trees. [3]
- Creation of extensions like CitusDB for more parallelism without modifying Postgres code. [14]

Cons

- MVCC system requires regular "vacuuming"
- Problems in high transaction rate environments. [14]

- Development is done by broad community
- Fair amount of effort for improvements added. [14]

SQLite

I have used SQLite in my prototype for the Pitch Locator section of the app and felt like it was difficult to get the information from the database in comparison MongoDB

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. Think of SQLite not as a replacement for Oracle but as a replacement for fopen(). [17]

Pros

- No separate server process
- File format is cross-platform
- Compact library: runs faster even with more memory
- Transactions are ACID compliant
- Professional support also available. [14]

Cons

- Not recommended for:
- Client/Server applications
- High-volume websites
- Large datasets
- High concurrency. [14]

Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. [18]

Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source. Django was designed to help developers take applications from concept to completion as quickly as possible. Django takes

security seriously and helps developers avoid many common security mistakes. It is also exceedingly scalable. Some of the busiest sites on the Web leverage Django's ability to quickly and flexibly scale.
Development of Django is supported by an independent foundation established as a 501(c)(3) non-profit. Like most open-source foundations, the goal of the Django Software Foundation is to promote, support, and advance its open-source project: in this case, the Django Web framework. [19]

Android

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast—every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content. [20]

Android gives you a world-class platform for creating apps and games for Android users everywhere, as well as an open marketplace for distributing to them instantly. [20]

Android is continuously pushing the boundaries of hardware and software forward, to bring new capabilities to users and developers. For developers, the rapid evolution of Android technology lets you stay in front with powerful, differentiated applications. [20]

Android gives you access to the latest technologies and innovations across a multitude of device form-factors, chipset architectures, and price points. From multicore processing and high-performance graphics to state-of-the-art sensors, vibrant touchscreens, and emerging mobile technologies. [20]

J2ObjC

For a developer, it is critical to recognize that both Android and iOS are different platforms. Android uses Java as the programmable language whereas iOS uses ObjectiveC. Due to this, it is hard to convert an app built for Android to iOS or vice versa. For a developer who intends to build an app for both the platforms, require using a multi-functional platform such as HTML5 that is capable of running on different devices irrespective of the operating system. However, with the help of Google's J2ObjC project, it is now possible to convert Java source code to Objective-C for the iOS platform. [21]

J2ObjC is an open source command-line utility tool developed by Google in order to allow developers to convert Java source into Objective-C coding. With the help of the utility tool, a developer building an Android app will be able to convert it to iOS app with the support of this tool. The main aim of the project is to enable a developer to utilise non-UI code such as data models and application logic in Java and then share it using web apps such as Android apps and iOS apps. The most engaging part of the plan is the support it provides to Java runtime features and language required for a particular client-side application. Additionally, it even helps in anonymous classes, exceptions, threads and reflection, and generic types. Before proceeding further, it is

important for a developer to have JDK 1.8 or higher, Mac laptop or workstation, Mac OS X 10.11 and above, and Xcode 7 and above. [21]

Objective C

Objective-C is the primary programming language you use when writing software for OS X and iOS. It's a superset of the C programming language and provides object-oriented capabilities and a dynamic runtime. Objective-C inherits the syntax, primitive types, and flow control statements of C and adds syntax for defining classes and methods. It also adds language-level support for object graph management and object literals while providing dynamic typing and binding, deferring many responsibilities until runtime. [22]

When building apps for OS X or iOS, you'll spend most of your time working with objects. Those objects are instances of Objective-C classes, some of which are provided for you by Cocoa or Cocoa Touch and some of which you'll write yourself. [11]

If you're writing your own class, start by providing a description of the class that details the intended public interface to instances of the class. This interface includes the public properties to encapsulate relevant data, along with a list of methods. Method declarations indicate the messages that an object can receive, and include information about the parameters required whenever the method is called. You'll also provide a class implementation, which includes the executable code for each method declared in the interface. [22]

Swift

Swift is a general-purpose programming language built using a modern approach to safety, performance, and software design patterns. The goal of the Swift project is to create the best available language for uses ranging from systems programming, to mobile and desktop apps, scaling up to cloud services. Most importantly, Swift is designed to make writing and maintaining correct programs easier for the developer. [23]

The most obvious way to write code should also behave in a safe manner. Undefined behaviour is the enemy of safety, and developer mistakes should be caught before software is in production. Opting for safety sometimes means Swift will feel strict, but we believe that clarity saves time in the long run. [23]

Swift is intended as a replacement for C-based languages (C, C++, and Objective-C). As such, Swift must be comparable to those languages in performance for most tasks. Performance must also be predictable and consistent, not just fast in short bursts that require clean-up later. There are lots of languages with novel features — being fast is rare. [23]

Swift benefits from decades of advancement in computer science to offer syntax that is a joy to use, with modern features developers expect. But Swift is never done. We

will monitor language advancements and embrace what works, continually evolving to make Swift even better. [23]

Swift includes features that make code easier to read and write, while giving the developer the control needed in a true systems programming language. Swift supports inferred types to make code cleaner and less prone to mistakes, and modules eliminate headers and provide namespaces. Memory is managed automatically, and you don't even need to type semi-colons. Swift also borrows from other languages, for instance named parameters brought forward from Objective-C are expressed in a clean syntax that makes APIs in Swift easy to read and maintain. [23]

JavaScript

JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites. It was invented by Brendan Eich, co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation. [24]
JavaScript is incredibly versatile. You can start small, with carousels, image galleries, fluctuating layouts, and responses to button clicks. With more experience you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more! [24]
JavaScript itself is fairly compact yet very flexible. Developers have written a large variety of tools on top of the core JavaScript language, unlocking a vast amount of extra functionality with minimum effort. These include:
Browser Application Programming Interfaces (APIs) — APIs built into web browsers, providing functionality like dynamically creating HTML and setting CSS styles, collecting and manipulating a video stream from the user's webcam, or generating 3D graphics and audio samples. [24]
Third-party APIs to allow developers to incorporate functionality in their sites from other content providers, such as Twitter or Facebook. [24]
Third-party frameworks and libraries you can apply to your HTML to allow you to rapidly build up sites and applications. [24]

Design

Scrum software methodology

The Scrum software methodology will be used in the development of this project. This is an Agile methodology which involves breaking a project down into sections/phases based on its requirements and doing weekly sprints where aspects of these phases are implemented in order to complete the phase requirements. Each finished phase should include a shippable project. Latter phases will have additional functionality than their predecessors. The project's plan is to divide the project into five phases which will aim to be completed using biweekly sprints over the course of the year. The sprints may extend over the period so giving some time at the end to buffer out any kinks is essential. [25]

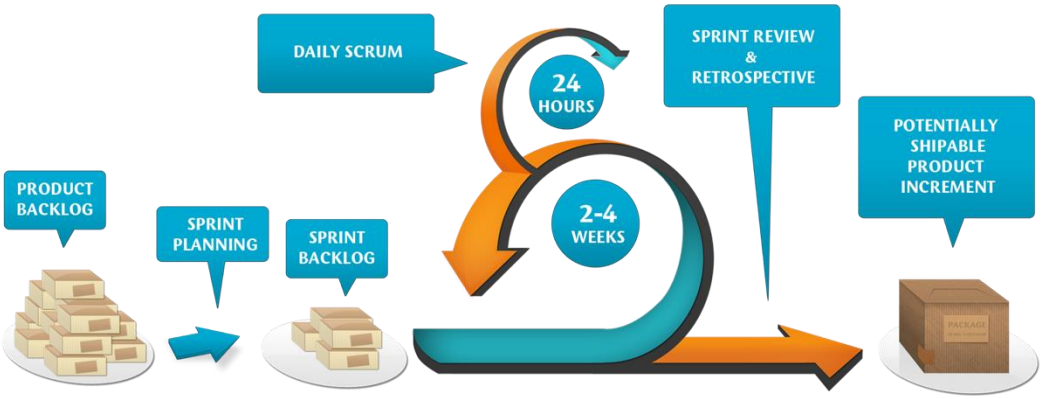


fig 3.1 Scrum software methodology (http://scrummethodology.com/) [25]

Use Case Diagram

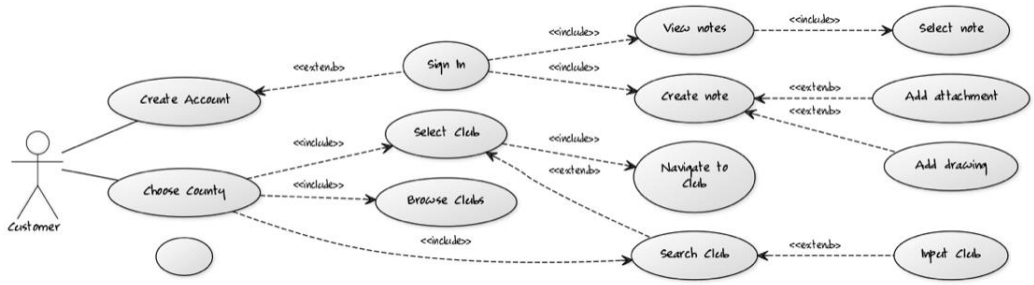


Fig 3.2 Use case of the system with mobile app and web application

In Fig 3.2 shows the main application use case that include the sign up to the application separate to the Pitch Finding Section as the user may only want to know where the Pitch is and maybe Log in to see more of the app at a later stage. the option will always be there to log in to see more and explore the possibilities of the app. The main function of the application is to find the destination of the pitch. To do this the user will either choose county to filter from or search a club by name. once done they can select the club and navigate to the club. The extra parts of the application include to note taking, which requires the user to login or sign up as the note will need a user to designate the ownership to.

The UI Design

Android and iOS

Home Page

Fig 3.3a Home page of the mobile application

The UI Layout in Fig 3.3a would be appealing to any user who interacts with it. The design in it would be an interactive animation where the user will be able to choose the county on the map and then be brought to another view where it would show the clubs that are in that county. In this case the highlighted county is Meath and when tapped it will open a view pass the information to the next screen and query the database with the given club name.

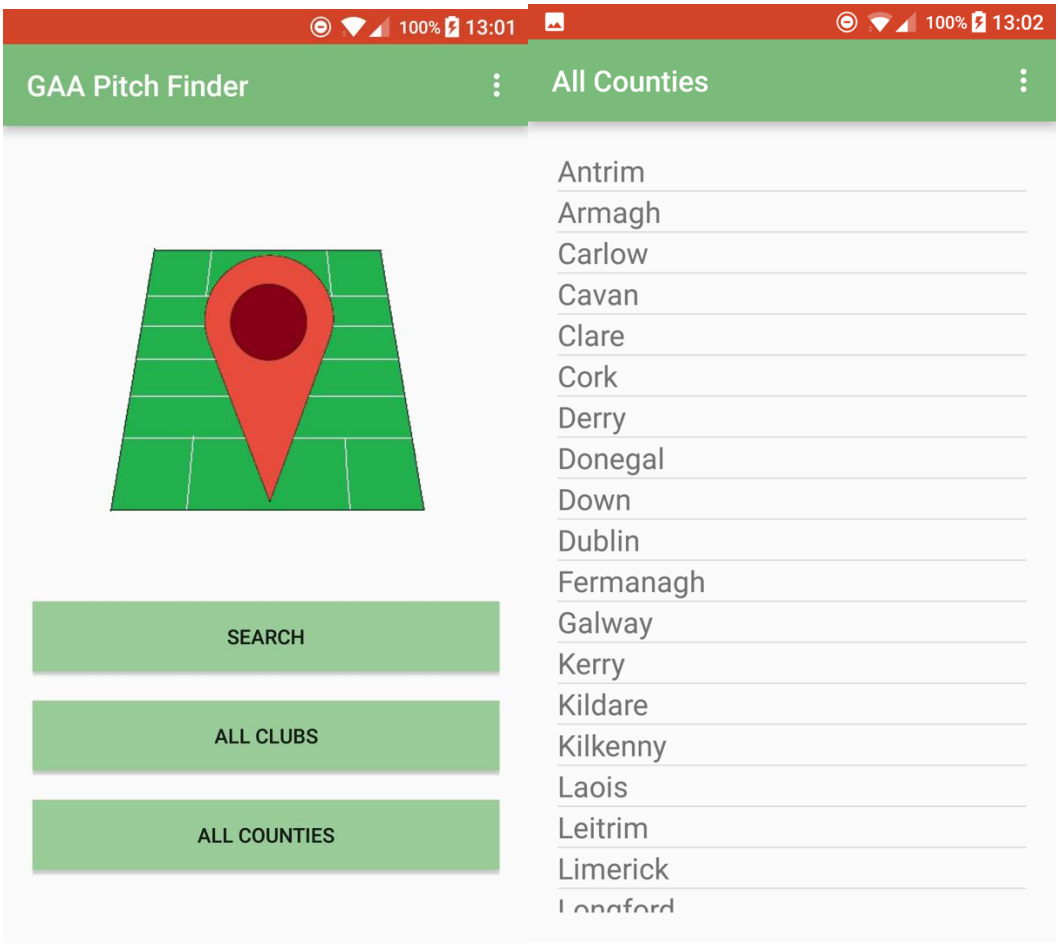


Fig 3.3b Fully developed All county display.

The changes from the proposed look was to make sure the user knew what county they were looking for as their geographical knowledge may be weak. In Fig 3.3b the counties are displayed in a list vie on the right. The home screen on the left includes the options to search, view all clubs, and view all counties.

Clubs display

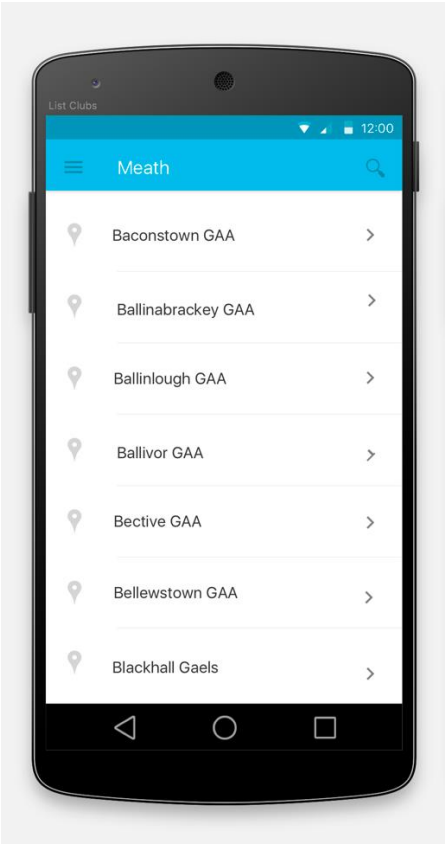


Fig 3.4 Club list of a specified county

On this page shown in Fig 3.4, it shows the Meath clubs in a list view and each object is clickable. If the user would like to narrow the results they can do so by clicking the the search icon in the top right hand corner opening a search bar to filter the results of that current county.

Search Functionality

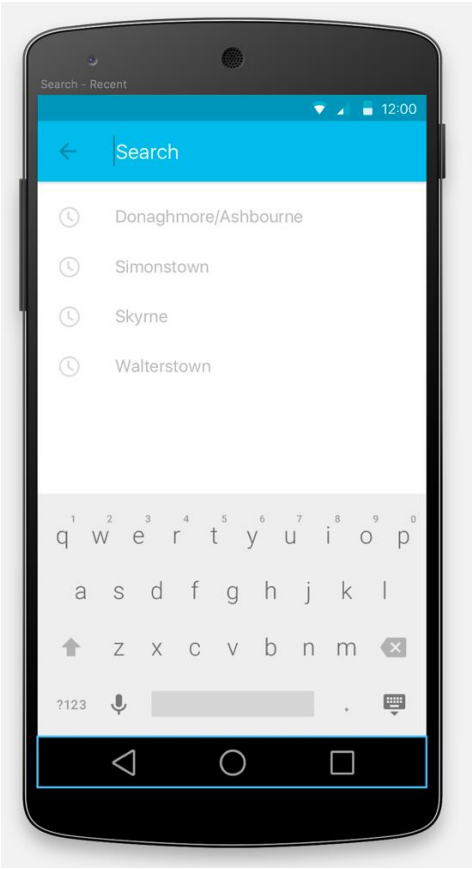


Fig 3.5a Search functionality

Fig 3.5a shows the intended functionality of the searching element in the android application. When the search icon is pressed, the search bar will come into view and the recent searches will be available for the user to choose from. Once enter is pressed on the keyboard the list of clubs that are closest to the search query will display in a list view.

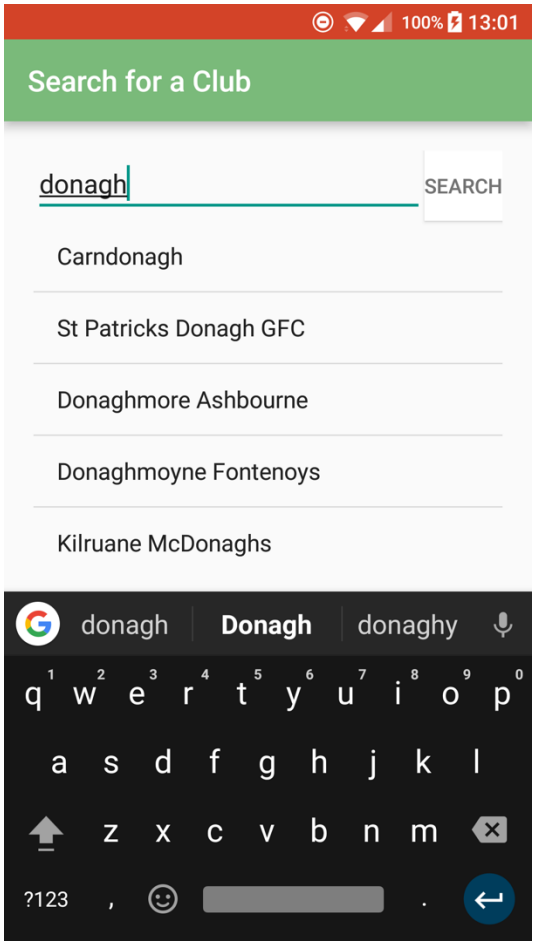


Fig 3.5b Developed search functionality

Fig 3.5b, shows the fully developed functionality of the search include the "clubs display" list view in the same screen as the search to quickly be able to change the search query. It shows the clubs in a list view under the search bar.

Club Information

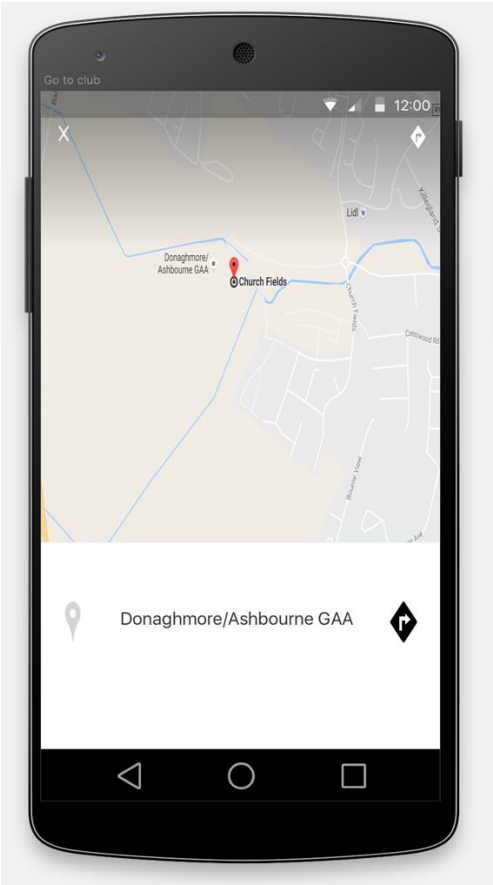


Fig 3.6a Club information and Map

The Club information should be where the user can locate the club easily and smoothly. In Fig 3.6a, when the club name is tapped at the bottom of this page then the information stored in the location value will be passed to google maps where an accurate route will be generated from the user's current position.

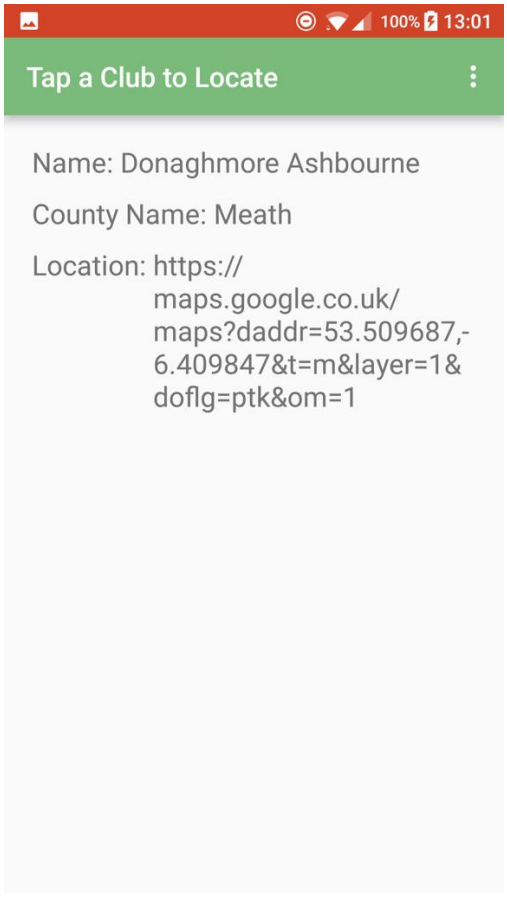
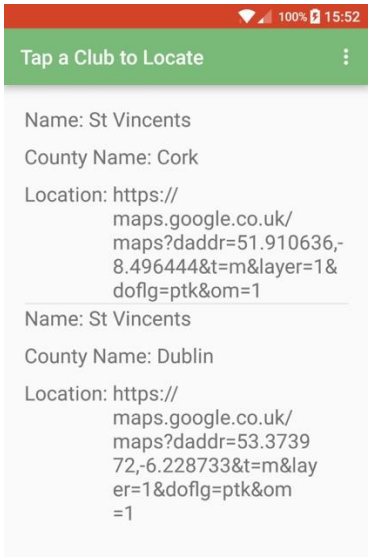


Fig 3.6b Developed Club information

The fully developed Club information shown in Fig 3.6b shows the Club information with the county information as well as the location link. When the club is pressed the location link get executed and google maps opens. The view is a list view as many club names are spelled the same so when selecting a club for example “St. Vincent’s” there are two clubs, one from Cork and one from Dublin. This caters for the user getting mixed up when selecting what county, the club is from.



Navigation

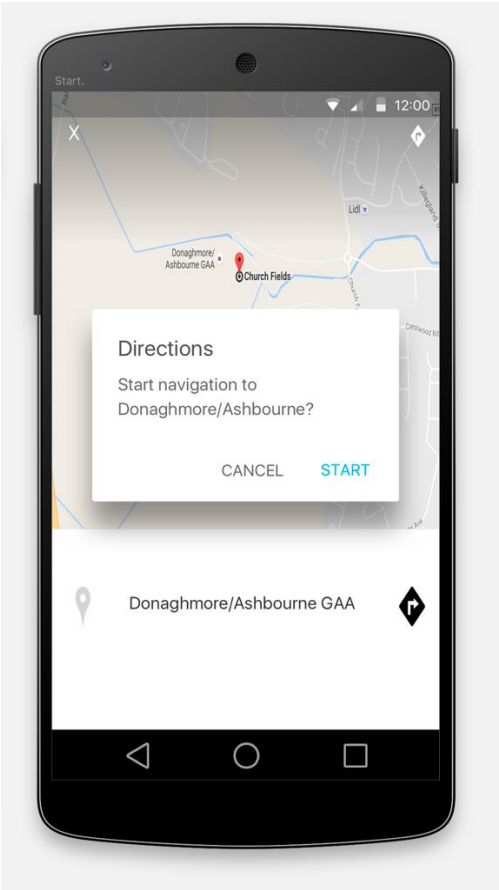


Fig 3.7a Navigation to club through google maps

Fig 3.7a shows the section of the application where it passes over the information to google maps and starts the navigation. The user will be prompted to choose whether to start navigation or cancel and go back to choose a different club from the list.

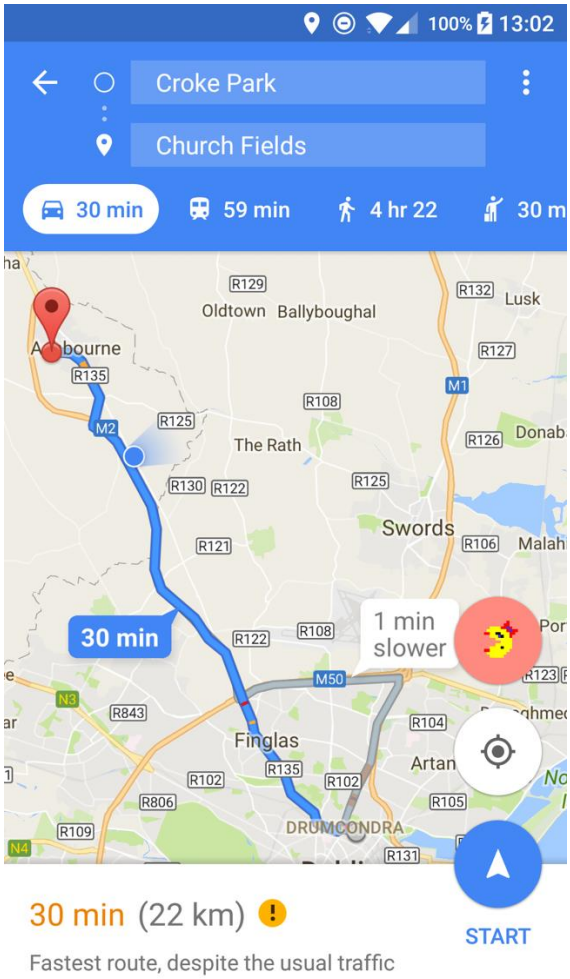


Fig 3.7b Navigation within google maps

Fig 3.7b the route google maps chooses to the destination from the club information.

Manager Section

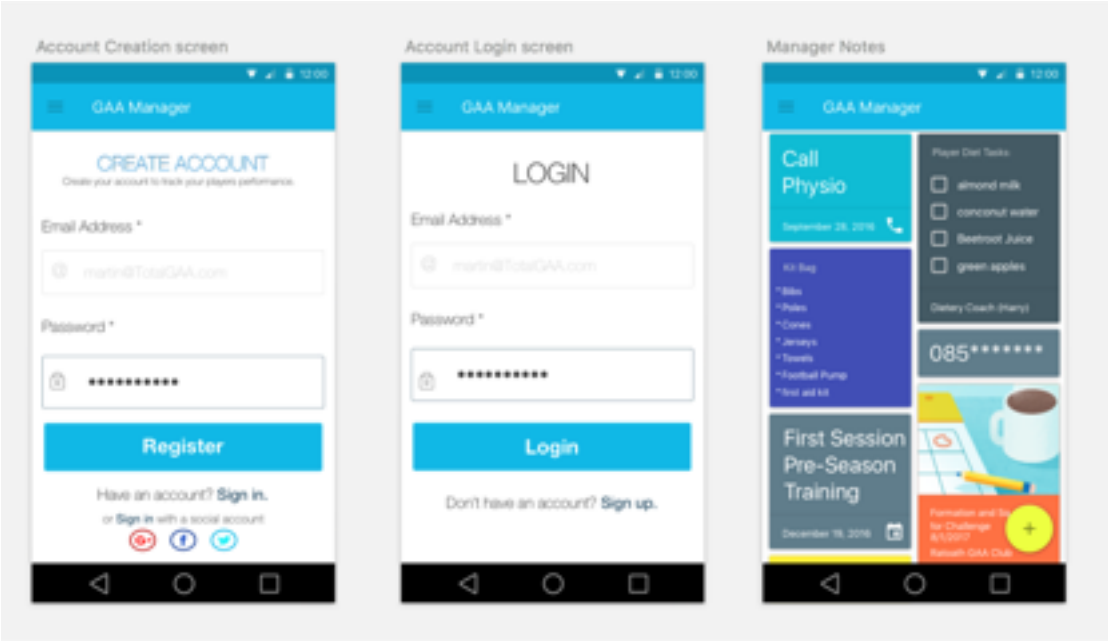


Fig 3.8 Manager Section of the app showing sign up, login, and notes screens

Fig 3.8 shows where the user will be able to login to the application by creating an account or by using social accounts and then proceed to be able to take note and create interesting layouts. The notes would resemble those of “Google Keep”. Where they appear as blocks of information which is visibly clear and easy to use.

This section is still in development and will not be added to the final submission. Although should be implemented in a later version of the application.

Web Application (Django)

Home Page

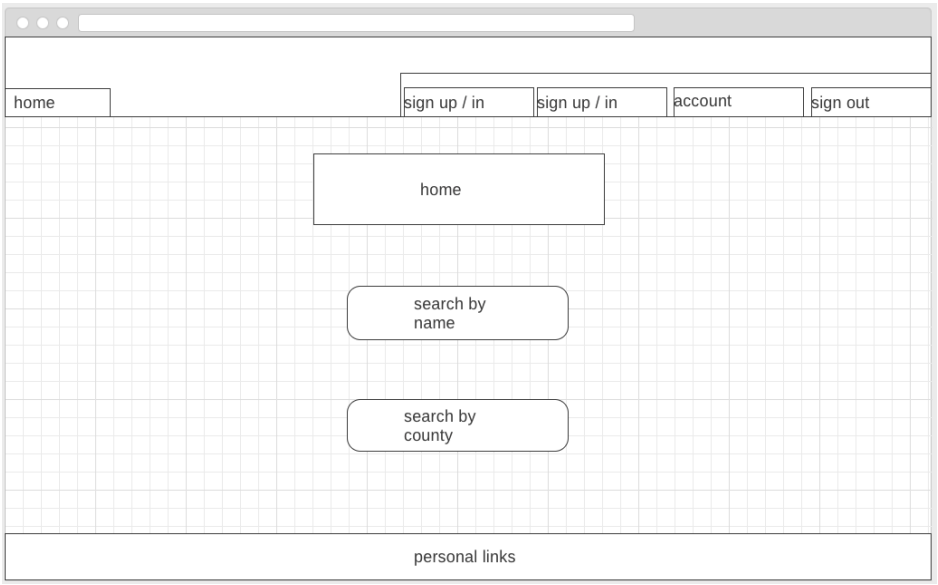


Fig 3.9a Home page of the web app developed in Django (https://wireframe.cc/aqULV4)

In Fig 3.9a, the webpage consists of the home template which would contain the links back to the home page labelled “home”, the login and sign up page links labelled “sign up / in”, the account page link where the user’s posts would be stored labelled “account”, the sign out button labelled “sign out”, and any personal links provided labelled “personal links”. The body of the page would contain the search functionality labelled “search by name”, and the filter by county functionality labelled “search by county”.

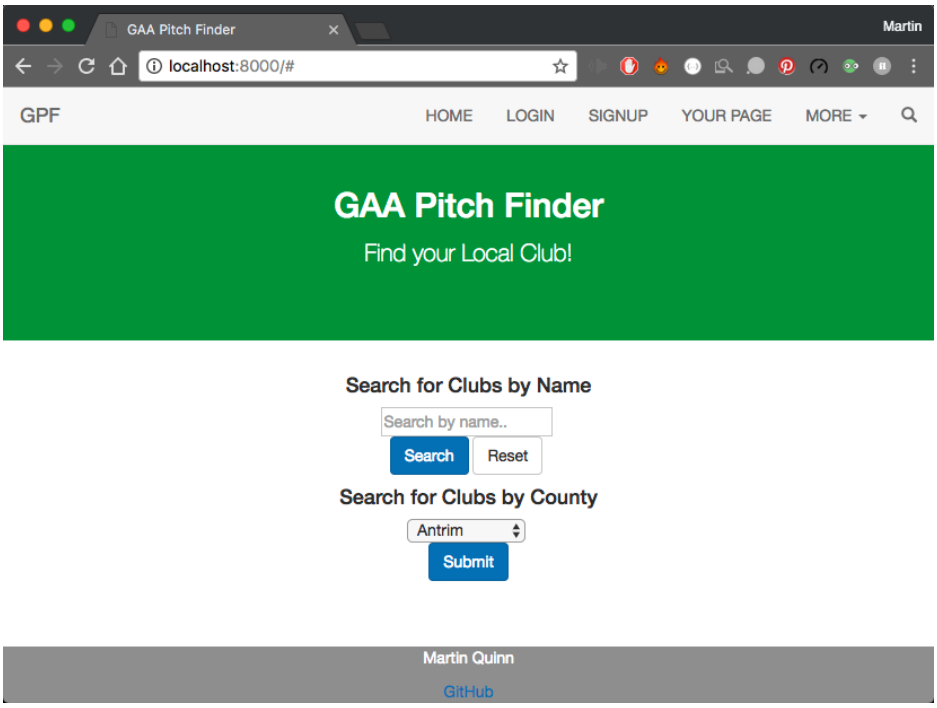


Fig 3.9b Fully developed home page

Search by name

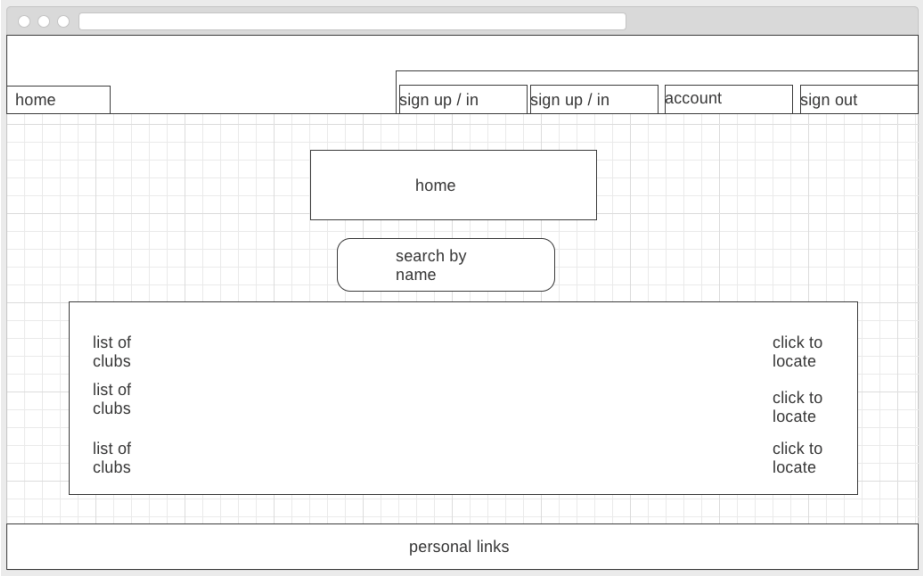


Fig 3.10a Searching clubs by name (https://wireframe.cc/d5gIBO)

In Fig 3.10a, the page will include the home template described in the "Home page" section. The user will be able to type in a club's name in the box labelled "search by name" and then once submitted the clubs matching closest to the query will display labelled "list of clubs". The user can then click the section labelled "click to locate" to go to the google maps location.

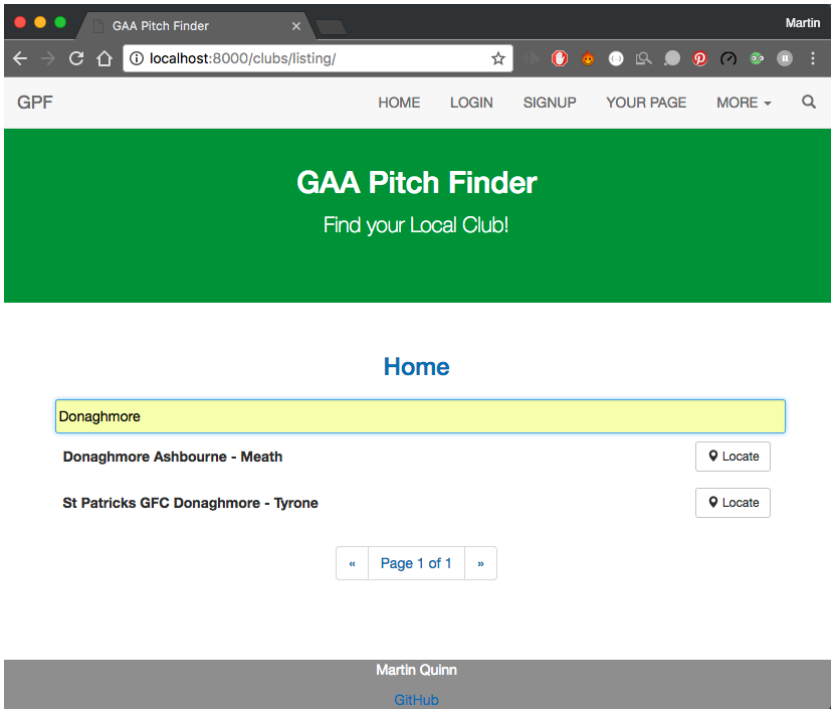


Fig 3.10b Fully developed search functionality

Search by county

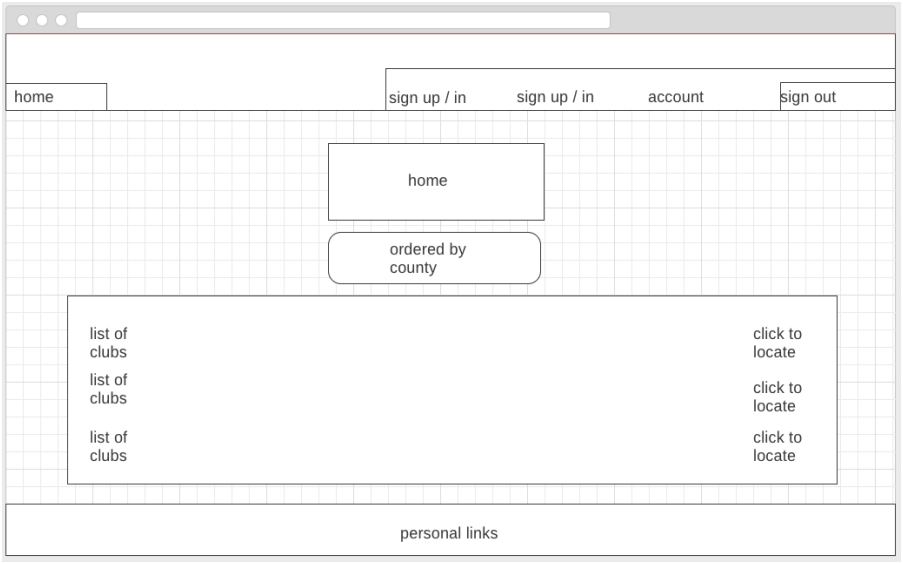


Fig 3.11a searching clubs by county (https://wireframe.cc/QptSa8)

In Fig 3.11a, the page will include the home template described in the "Home page" section. The user will be able to type a county in the box labelled "ordered by county" and then once submitted the clubs matching the county will display labelled "list of clubs". The user can then click the section labelled "click to locate" to go to the google maps location the exact same functionality as "search by name".

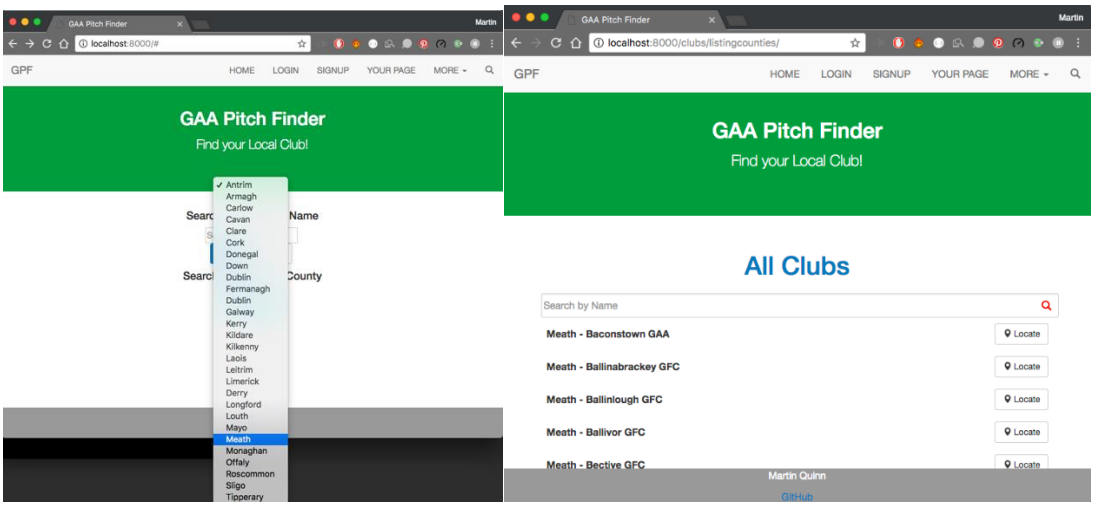


Fig 3.11b Fully developed search by county.

Sign up

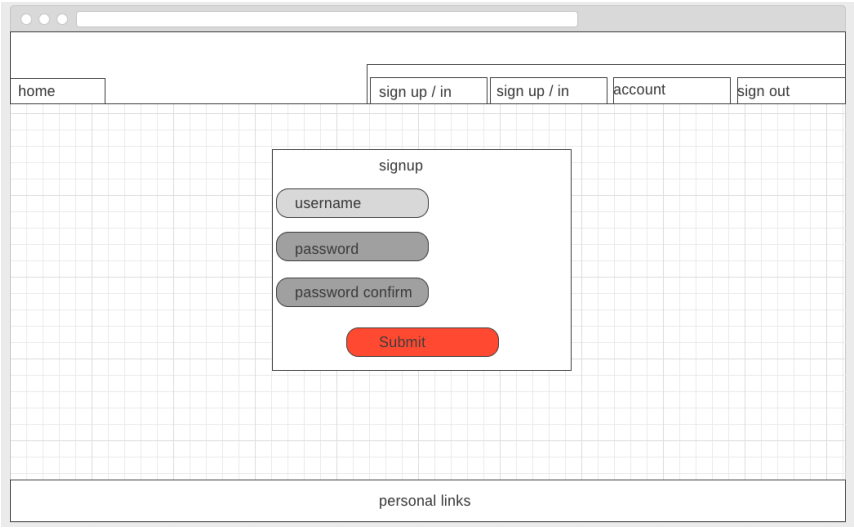


Fig 3.12a sign up page for access to the accounts section. (https://wireframe.cc/AlM1uK)

In Fig 3.12a, the page will include the home template described in the "Home page" section. The body of the page will include a box in which the user will need to enter information on their account. The user should not be able to use usernames that have been chosen before, the username has to be unique. The user will then have to enter a password and then verify the password is correct by typing it again to ensure they don't wrongly type the password the first time. Once complete the user can then submit and will be taken straight to the account page accessible through label "account".

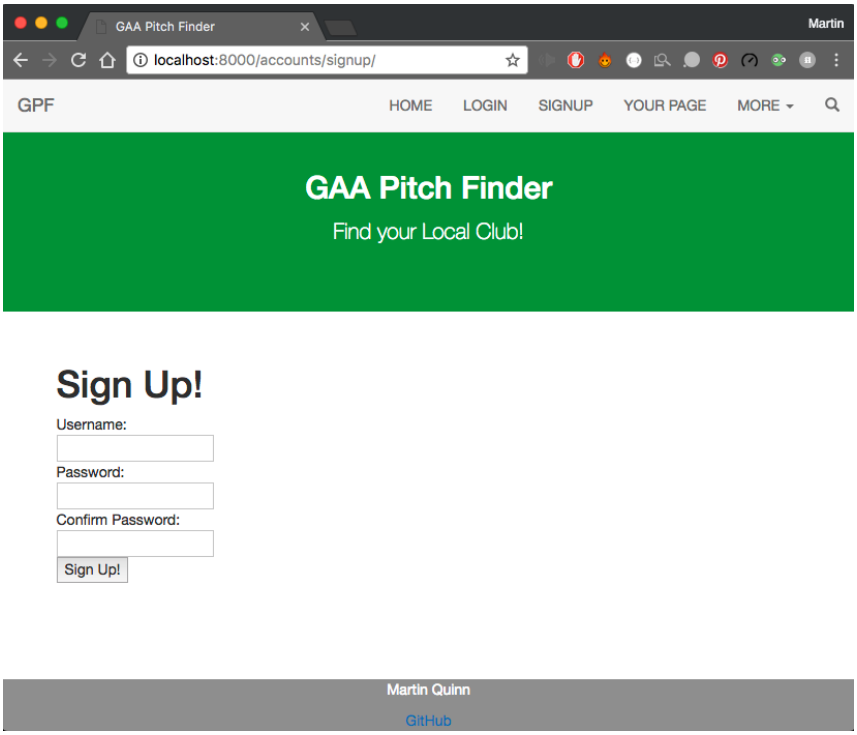


Fig 3.12b Fully developed Sign up page

Sign In

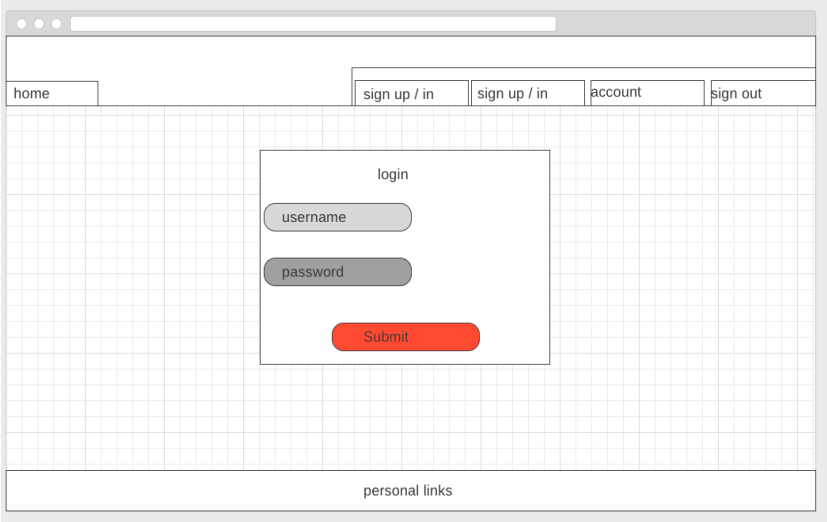


Fig 3.13a Sign in page for access to the accounts section. (https://wireframe.cc/9BTkWn)

In Fig 3.13a, the page will include the home template described in the "Home page" section. The body of the page will include a box in which the user will need to enter information on their account to sign in. Once the correct information is entered by the user then they will be sent to the accounts section to view or create new posts. To sign out of an account, the user will need to click the sign out button in the top right of any screen. This will log the user out and redirect them to the home page.

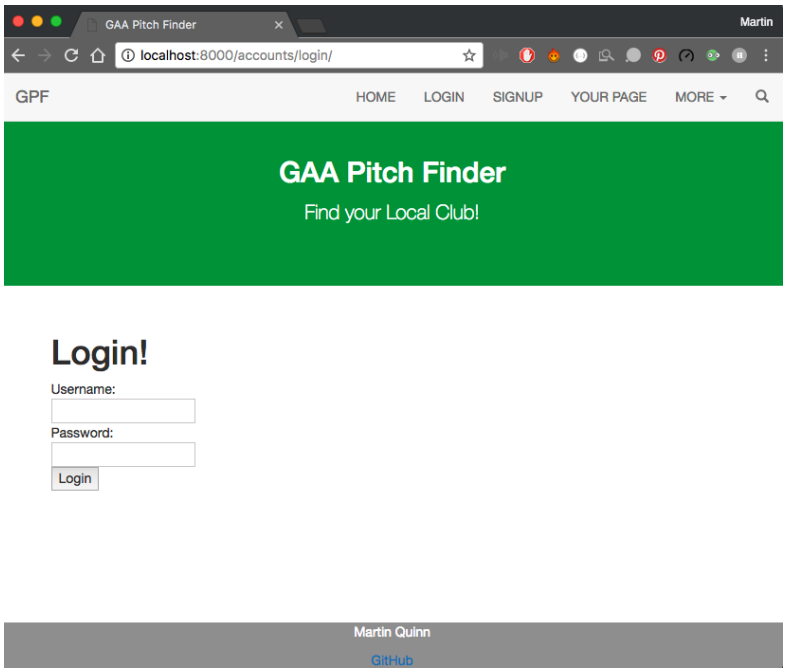


Fig 3.13b Fully developed login page

User account

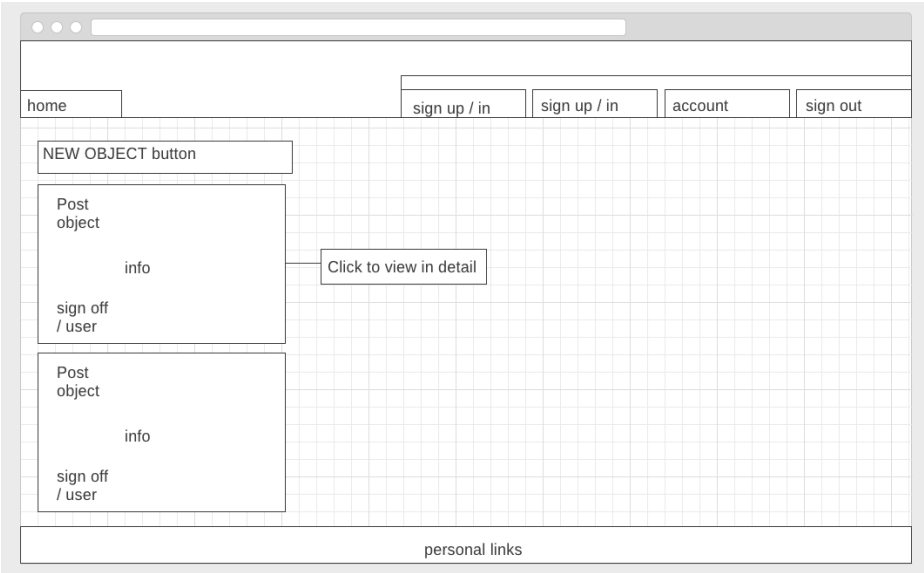


Fig 3.14 User account section including post objects (https://wireframe.cc/2Gtil0)

In Fig 3.14, the page will include the home template described in the "Home page" section. The body of the page will include a new post button labelled "NEW OBJECT button", and will also include each post object name labelled "Post object" and a small piece of information about them labelled "info", it will then have the username who posted it and the date labelled "sign off/ user". To get more information on the object the user will be able to click into the object.

This section is not fully developed, it is being tested and will not be featured in the final submission.

Post object

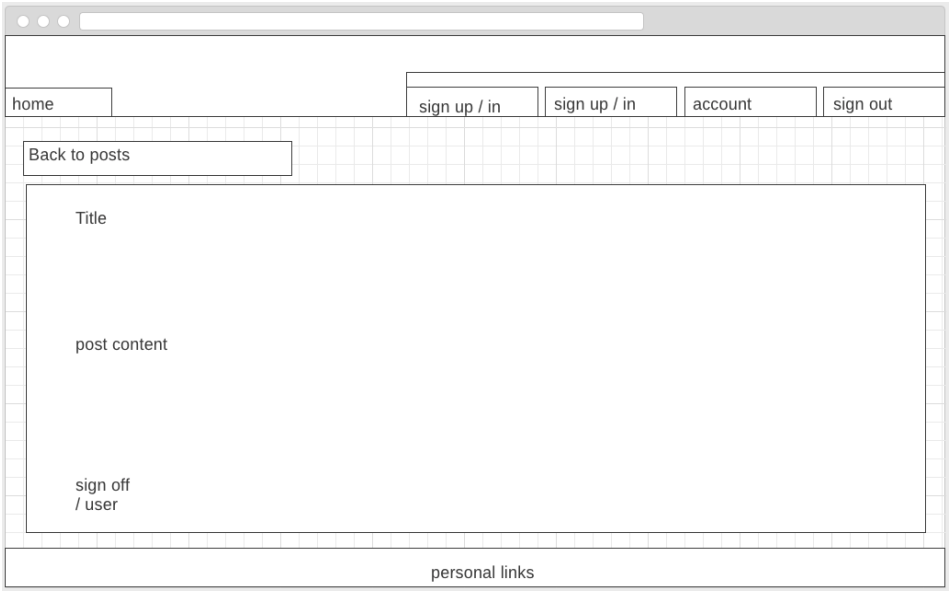


Fig 3.15 Post object detailed information. (https://wireframe.cc/xw3aLQ)

In Fig 3.15, the page will include the home template described in the "Home page" section. The body of this page will go into detail of the post object. Show the full title labelled "Title", The actual content of the post labelled "post content", then the username and date labelled "sign off / user". To return to the full list of post the button at the top of the page's body labelled "Back to posts" will take the user there.

Creating a post

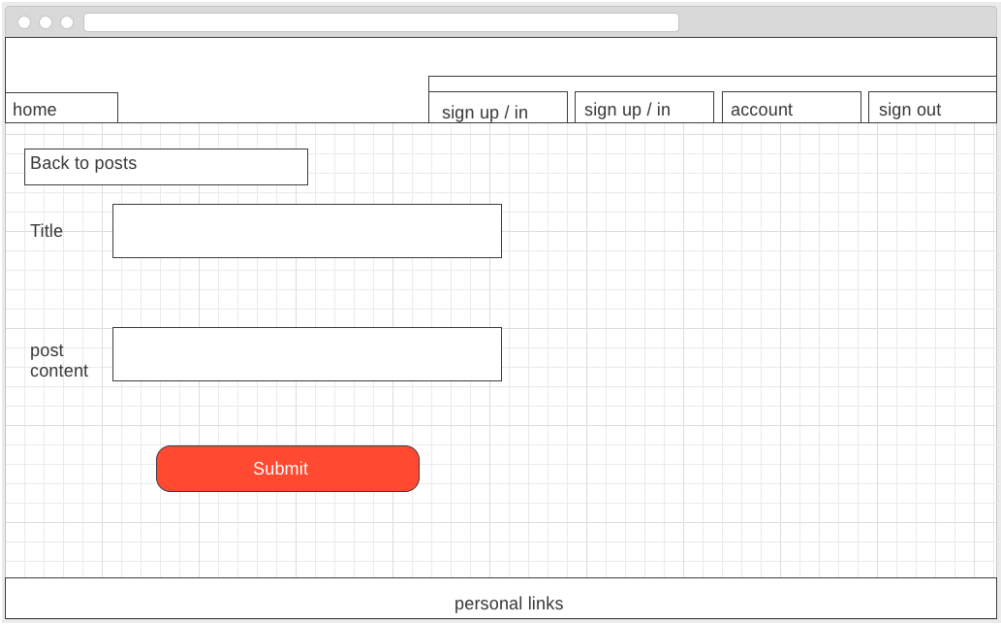


Fig 3.16 Creation of a post for a user account. (https://wireframe.cc/ia8BFy)

In Fig 3.16, the page will include the home template described in the "Home page" section. The body of this page will allow the user to enter the title of the post in the box, "Title", then the information of the post can be entered in the box "post content". When completed the user can post the object by clicking Submit or to return to the full list of post the button at the top of the page's body labelled "Back to posts" will take the user back there.

Technical Architecture Diagram

Three-Tier Architecture Model

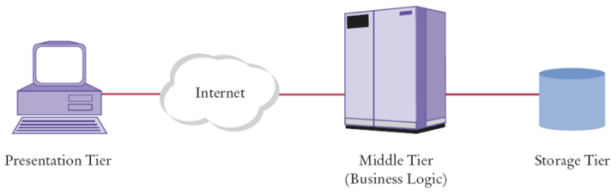


Fig 3.17 Three-tier architecture model [26]

The three-tier architecture model, which is the fundamental framework for the logical design model, divides an application's components into three tiers of services. These tiers do not necessarily correspond to physical locations on various computers on a network, but rather to logical layers of the application. How the pieces of an application are distributed in a physical topology can change, depending on the system's requirements. Following are brief descriptions of the services allocated to each tier:

The presentation tier, or user services layer, gives a user access to the application. This layer presents data to the user and optionally permits data manipulation and data entry. The two main types of user interface for this layer are the traditional application and the Web-based application. Web-based applications now often contain most of the data manipulation features that traditional applications use. This is accomplished through use of Dynamic HTML and client-side data sources and data cursors. [26]

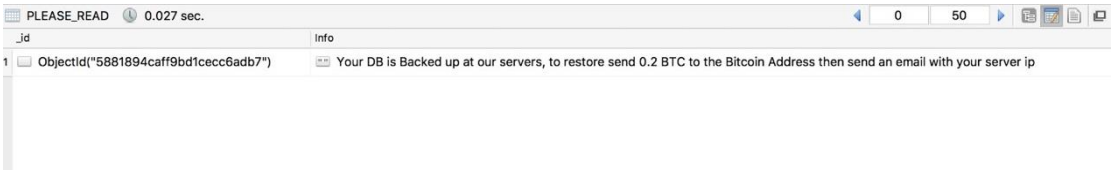
The middle tier, or business services layer, consists of business and data rules. Also referred to as the business logic tier, the middle tier is where COM+ developers can solve mission-critical business problems and achieve major productivity advantages. The components that make up this layer can exist on a server machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and legal or governmental regulations, and data rules, which are designed to keep the data structures consistent within either specific or multiple databases. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require. For example, simple edits can be placed on the client side to minimize network round-trips, or data rules can be placed in stored procedures. [26]

The data tier, or data services layer, interacts with persistent data usually stored in a database or in permanent storage. This is the actual DBMS access layer. It can be accessed through the business services layer and on occasion by the user services layer. This layer consists of data access components (rather than raw DBMS connections) to aid in resource sharing and to allow clients to be configured without installing the DBMS libraries and ODBC drivers on each client. [26]

Development

Solution to problem

The Solution to the application will be the usage of a database on a server, and that database was initially to use a “MongoDB” stored on a server located on one of Digital Ocean’s servers. The data would be stored on this server allowing requests from the three applications. The data would be easily accessible. However, the data was compromised during the development of the android application. The security of the server had been breached, and the data stored on the server was replaced with a single file stating that the data had been stored elsewhere and a sum of bitcoin would ensure the safe return of the data.



This was a huge setback in the progress of the project. Having set up the structure of the database and the research involved in doing so it seemed to be all for nothing. The security that is needed to combat the threats from these people, is much superior to the basic set up that the Digital Ocean server that was purchased.

With great deliberation the best option was to start fresh at a new database setup. This being to individually set a database for each application. The databases that are default with the applications were SQLite, which was helpful due to prior knowledge of SQLite. To populate the database a Python program was run to extract all the clubs and locations in an easy layout to directly insert into the databases. The clubs, all being available in some shape or form on the internet was a great help, considering the setback. [11]

Code Structure and classes

Both applications follow the Model View Controller Design pattern.

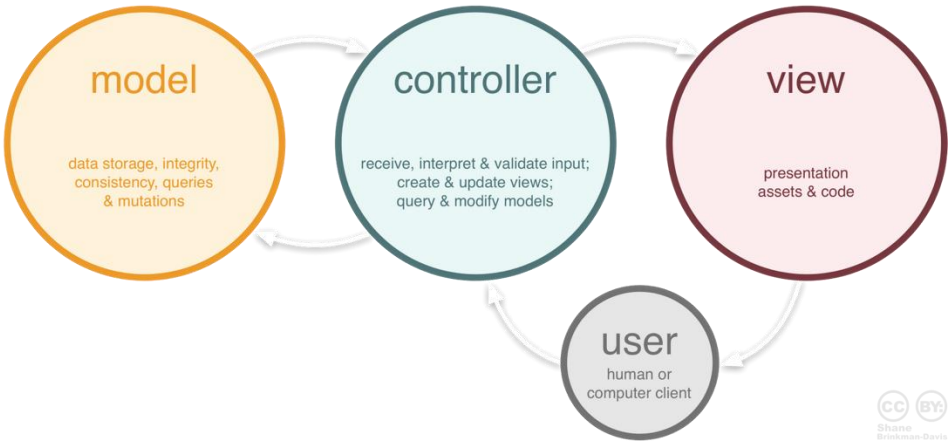


Fig 4.1 Model View Controller design pattern [27]

In the android application the code is structured to fit the MVC design pattern. The model, controller and view layers are each kept separate each other. This aids in code readability and reusability.
Controllers are the activities themselves which contain all the business logic done in the application.
Models are entities which describe the components of the apps.
Views Can be done in XML layouts.

Django follows the MVC pattern closely, however it does use its own logic in the implementation. Because the “C” is handled by the framework itself and most of the excitement in Django happens in models, templates and views, Django is often referred to as an *MTV framework*. In the MTV development pattern: [28]
M stands for “Model,” the data access layer. This layer contains anything and everything about the data: how to access it, how to validate it, which behaviours it has, and the relationships between the data. [28]
T stands for “Template,” the presentation layer. This layer contains presentation-related decisions: how something should be displayed on a Web page or other type of document. [28]
V stands for “View,” the business logic layer. This layer contains the logic that accesses the model and defers to the appropriate template(s). You can think of it as the bridge between models and templates. [28]

Android Application

Main activity “GAAactivity.java”

This is the main activity of the android application it contains three buttons which allows the user to navigate to the search activity, all club view, and all county view.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_gaa);

    // Search Club Button to start the search activity
    searchClubButton = (Button)findViewById(R.id.searchclub_button);
    searchClubButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Start getClubActivity

            Intent searchClubActivity = new Intent(GaaActivity.this, SearchActivity.class);
            startActivity(searchClubActivity);

        }
    });

    // Get Club Button to start the Display club activity
    getClubButton = (Button)findViewById(R.id.getclub_button);
    getClubButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Start getClubActivity

            Intent getClubActivity = new Intent(GaaActivity.this , AllClubDisplay.class);
            startActivity(getClubActivity);

        }
    });

    // Search The counties Button to start the County list activity
    countyButton = (Button) findViewById(R.id.button_county);
    countyButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent getCountyActivity = new Intent(GaaActivity.this , CountyDisplay.class);
            startActivity(getCountyActivity);

        }
    });

}
```

Search activity “SearchActivity.java”

This is the search activity which passes the user input into the database to find the closest results to their query. Search button is clicked and then the information is passed, the query can be empty or can have the data inside.

```
public class SearchActivity extends AddClub {

    private Button searchClubButton;
    ArrayAdapter<String> mArrayAdapter;
    List<ClubsClass> mClubsClasses;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search);

        // This is the array list of clubs from the search results
        // when clicked the information will be passed again to get all information
        mClubsClasses = new ArrayList<>();

        //to view the array list
        mArrayAdapter = new ArrayAdapter<>(this,android.R.layout.simple_list_item_1,
android.R.id.text1);
        final ListView listClubs = (ListView) findViewById(R.id.listView);
        listClubs.setAdapter(mArrayAdapter);

        // the search field information
        clubName = (EditText) findViewById(R.id.club_search);

        // the button which then passes the query into the db.
        searchClubButton = (Button)findViewById(R.id.button);
        searchClubButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                // Start getClubActivity
                if(clubName == null){
                    try{
                        // This code will run if a no name is searched in the search box.
                        db.open();
                        Cursor result = db.getAll();
                        MyCursorAdapter cursorAdapter = new MyCursorAdapter(SearchActivity.this, result);

                        db.close();

                    }catch(SQLException e){
                        e.printStackTrace();
                    }
                }else{
                    try {
                        // This searches the database of clubs and returns the closest results
                        db.open();
                        mArrayAdapter.clear();
                        mArrayAdapter.notifyDataSetChanged();
                        listClubs.setAdapter(mArrayAdapter);

                        // opens db and uses function inside DBmanager.java
                        mClubsClasses = db.searchLikeName(clubName.getText().toString());

                        for(int i = 0; i < mClubsClasses.size(); i++) {
                          mArrayAdapter.add(mClubsClasses.get(i).getName());
                        }

                        db.close();
```

```
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
});

listClubs.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> av, View view, int position, long arg) {

        //intent to single club with details of all aspects of the club
        ClubsClass clubsClass = mClubsClasses.get(position);
        // The club information is passed with the intent to find the club in the database
        Intent ClubInfoDisplay = new Intent(SearchActivity.this, ClubInfo.class);
        // name of club
        ClubInfoDisplay.putExtra("clubInfo",clubsClass.getName());
        // location info
        ClubInfoDisplay.putExtra("locInfo",clubsClass.getLocaiton());
        startActivity(ClubInfoDisplay);
    }
});
}
```

County Display “CountyDisplay.java”

This is the county display activity, it displays all the counties in Ireland and when a county is selected it will pass the county id over to the all club display (AllClubsDisplay.java).

```
public class CountyDisplay extends AddClub {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_county_display);

      final ListView listClubs = (ListView) findViewById(R.id.listView_counties);

      // This gets all the counties from the database.
      try {
         db.open();

         //uses function stored in "DBManager.java"
         Cursor result = db.getAllCounties();
         MyCursorCountyAdapter cursorCountyAdapter = new
MyCursorCountyAdapter(CountyDisplay.this, result);

         listClubs.setAdapter(cursorCountyAdapter);
         db.close();
      } catch (Exception ex) {
         // catch if error with database opening

         Context context = getApplicationContext();
         CharSequence text = "Error opening database";
         int duration = Toast.LENGTH_LONG;

         Toast toast = Toast.makeText(context, text, duration);
         toast.show();
      }

      listClubs.setOnItemClickListener(new AdapterView.OnItemClickListener() {
         @Override
         public void onItemClick(AdapterView<?> av, View view, int position, long arg) {

            Cursor mycursor = (Cursor) av.getItemAtPosition(position);
            int countyQuery = mycursor.getInt(0);

            // starts the display of all the clubs within the selected county
            Intent ClubInfoDisplay = new Intent(CountyDisplay.this, AllClubDisplay.class);

            // putting the extra information with the intent
            ClubInfoDisplay.putExtra("countyID", countyQuery);

            // start the clubs display with the selected county
            startActivity(ClubInfoDisplay);

         }
      });
   }
```

All Club display “AllClubDisplay.java”

This Activity will display clubs depending on what is passed with an intent from a previous activity. It first gets the extra from the intent and then checks what information it was given. Once this is clarified the clubs will then display.

```
public class AllClubDisplay extends AddClub{

    MyCursorAdapter cursorAdapter;

    private static final int CLUB = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_all_club_display);

        //this sends The name and id of a certain club
        Intent intent = getIntent();
        int countyID = intent.getIntExtra("countyID", 0);
        String clubInfo = intent.getStringExtra("clubInfo");

        final ListView listClubs = (ListView) findViewById(R.id.listView_clubs);

        // If the intent comes from CountyDisplay.java this will be executed
        if(countyID>0){
            try {
                db.open();
                //this searches a club with the id of a certain county
                Cursor result = db.getClub(countyID);
                cursorAdapter = new MyCursorAdapter(AllClubDisplay.this, result);

                listClubs.setAdapter(cursorAdapter);
                db.close();
            } catch (SQLException ex) {
                // If there is an error opening database
                ex.printStackTrace();
                Context context = getApplicationContext();
                int duration = Toast.LENGTH_LONG;
            }

            // If the intent is coming from SearchActivity.java this will be executed
        }else if (clubInfo != null) {
            try {
                db.open();
                //This Gets the club names with the closest to the inputted field
                Cursor result = db.getClubNames(clubInfo);

                cursorAdapter = new MyCursorAdapter(AllClubDisplay.this, result);
                listClubs.setAdapter(cursorAdapter);
                db.close();
            }
            catch (SQLException ex) {
                // If there is an error opening database
                ex.printStackTrace();
                Context context = getApplicationContext();
                int duration = Toast.LENGTH_LONG;
            }

            // if both "clubInfo" and "CountyID" are null
            // this will execute to display all clubs in the database
            } else {
                try {
                    db.open();
                    // This gets all the Clubs in the database
                    Cursor result = db.getAll();
```

```
                cursorAdapter = new MyCursorAdapter(AllClubDisplay.this, result);
                listClubs.setAdapter(cursorAdapter);
                db.close();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }

    // When a club is clicked / tapped
    listClubs.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> av, View view, int position, long arg) {

            // This is the information that gets passed to the club info activity which
            // displays the name, county, and location of the club in question

            Cursor cursor = (Cursor)cursorAdapter.getItem(position);
            String locQuery = cursor.getString(cursor.getColumnIndex("location"));
            String clubQuery = cursor.getString(cursor.getColumnIndex("club_name"));

            // starting the ClubInfo activity
            Intent ClubInfoDisplay = new Intent(AllClubDisplay.this, ClubInfo.class);
            Bundle extras = new Bundle();

            // Putting forward the two pieces of information
            extras.putString("clubInfo",clubQuery);
            extras.putString("locInfo",locQuery);
            ClubInfoDisplay.putExtras(extras);
            startActivityForResult(ClubInfoDisplay,CLUB);
        }
    });
}
```

Club information “ClubInfo.java”

This is the activity where the club information is displayed in full for the user to see. The user can then click on the list view object and they will be navigated instantly to google maps because the link stored in the location variable gets executed. An instance of google maps will open outside the application itself.

```
public class ClubInfo extends AddClub {

    MyCursorInfoAdapter cursorAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_club_info);
        final ListView listInfo = (ListView) findViewById(R.id.listView_Info);
        final Intent intent = getIntent();

        // getting the club name to query
        final String clubInfo = intent.getStringExtra("clubInfo");

        try {
            db.open();
            // this will get the information in the database with the club name matching it
            // there could be more than one club with the same name
            Cursor result = db.getClubNames(clubInfo);
            cursorAdapter = new MyCursorInfoAdapter(ClubInfo.this, result);
            listInfo.setAdapter(cursorAdapter);
            db.close();
        } catch (SQLException ex) {
            // if an error occurs when opening the database
            ex.printStackTrace();
            Context context = getApplicationContext();
            int duration = Toast.LENGTH_LONG;
        }

        // when clicked will send you straight to the location of the Club through google maps.

        listInfo.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> av, View view, int position, long id) {

                Cursor cursor = (Cursor) cursorAdapter.getItem(position);
                final String location = cursor.getString(cursor.getColumnIndex("location"));
                String webURL = "";
                webURL += location;

                Uri gmmIntentUri = Uri.parse(webURL);
                Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
                startActivity(mapIntent);
            }
        });

    }
```

Database Manager “DBmanager.java”

This is the activity where all the queries take place from other classes. The table creations and inserts take place here.

```
public class DBManager {
   private static final int DATABASE_VERSION = 1;

   private static final String DATABASE = "gaa.db";

   private static final String TABLE_COUNTIES =
         "Counties";
   private static final String KEY_COUNTY_ID =
         "_id";
   private static final String KEY_COUNTY_NAME =
         "county_name";
   private static final String TABLE_CLUBS =
         "Clubs";
   private static final String KEY_CLUB_ID =
         "_id";
   private static final String KEY_CLUB_NAME =
         "club_name";
   private static final String KEY_COLOURS =
         "colours";
   private static final String KEY_CLUB_DESCRIPTION =
         "club_description";
   private static final String KEY_CLUB_LOCATION =
         "location";
   private static final String KEY_CLUB_WEBSITE =
         "website";
   private static final String FOREIGN_KEY_ID =
         "county_id_fk";

   private static final String CREATE_COUNTY_TABLE =
         "CREATE TABLE Counties " +
               "(" +
               "_id INTEGER PRIMARY KEY autoincrement, " +
               "county_name TEXT);";

   private static final String CREATE_CLUB_TABLE =
         "CREATE TABLE Clubs " +
               "(" +
               "_id INTEGER PRIMARY KEY autoincrement, " +
               "county_id_fk INTEGER, " +
               "club_name TEXT, " +
               "colours TEXT, " +
               "club_description TEXT, " +
               "location TEXT, " +
               "website TEXT," +
               "FOREIGN KEY(county_id_fk) REFERENCES Counties(_id));"
         ;
```

ETC… database Inserts.

```
   private final Context context;
   private MyDatabaseHelper DBHelper;
   private SQLiteDatabase db;

   // we must pass the context from our class that we called from
   public DBManager(Context ctx) {
      this.context = ctx;
      DBHelper = new MyDatabaseHelper(context);
   }
```

```
private static class MyDatabaseHelper extends SQLiteOpenHelper {

    public MyDatabaseHelper(Context context) {
        super(context, DATABASE, null, DATABASE_VERSION);
    }

    // when the app is initially created the tabe creates and inserts will execute.
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_COUNTY_TABLE);
        db.execSQL(INITIAL_COUNTY_INSERTS);
        db.execSQL(CREATE_CLUB_TABLE);
        db.execSQL(INITIAL_ANTRIM_CLUB_INSERTS);
        db.execSQL(INITIAL_ARMAGH_CLUB_INSERTS);
        db.execSQL(INITIAL_CARLOW_CLUB_INSERTS);
        etc…     }

    //When upgrading the database the tables will be dropped to cater for the new
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CLUBS);
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_COUNTIES);
        onCreate(db);
    }
}

// opens the database safely when called
public DBManager open() throws SQLException
{
    db = DBHelper.getWritableDatabase();
    return this;
}

// closes the database safely when called
public void close()
{
    DBHelper.close();
}

// If the intent comes from "CountyDisplay.java" to "AllClubDisplay.java"
public Cursor getClub(int countyQuery)
{

    Cursor mCursor = db.rawQuery("SELECT * FROM Clubs JOIN Counties ON
Clubs.county_id_fk = Counties._id WHERE county_id_fk = '"+ countyQuery +"' ORDER
BY club_name;",null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

// gets club names based on the county pressed from "CountyDisplay.java" to
"AllClubDisplay.java"
public Cursor getClubNames(String clubQuery)
{
    String selectQuery = "SELECT * FROM Clubs JOIN Counties ON Clubs.county_id_fk =
Counties._id WHERE Clubs.club_name = ?;";
    Cursor mCursor = db.rawQuery(selectQuery, new String[] {clubQuery});
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}

// gets all clubs to display from "GaaActivity.java" to "AllClubDisplay.java"
```

```
public Cursor getAll()
{

    Cursor mCursor = db.query(TABLE_CLUBS, // a. table
            null, // b. column names to return
            null, // c. selections "where clause"
            null, // d. selections args "where values"
            null, // e. group by
            null, // f. having
            KEY_CLUB_NAME, // g. order by
            null); // h. limit

    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;

}

// get all counties to display from "GaaActivity.java" to "CountyDisplay.java"
public Cursor getAllCounties()
{

    Cursor countyCursor = db.query(TABLE_COUNTIES, // a. table
            null, // b. column names to return
            null, // c. selections "where clause"
            null, // d. selections args "where values"
            null, // e. group by
            null, // f. having
            KEY_COUNTY_NAME, // g. order by
            null); // h. limit

    if (countyCursor != null) {
        countyCursor.moveToFirst();
        Log.d("MQ", KEY_COUNTY_NAME);
    }
    return countyCursor;

}

// searches the database for a club with the name like a search query from
"SearchActivity.java"
public List<ClubsClass> searchLikeName(String likeName) {

    List<ClubsClass> clubNames = new ArrayList<>();

    Cursor cursor = db.rawQuery("SELECT * FROM Clubs WHERE club_name LIKE " + "'%"
+ likeName + "%'" + ";", null);
    if (cursor.getCount() > 0) {
        cursor.moveToFirst();
        do {
            ClubsClass clubsClass = new ClubsClass();
            clubsClass.setId(cursor.getInt(cursor.getColumnIndex(KEY_CLUB_ID)));
            clubsClass.setName(cursor.getString(cursor.getColumnIndex(KEY_CLUB_NAME)));
            clubsClass.setColors(cursor.getString(cursor.getColumnIndex(KEY_COLOURS)));

clubsClass.setDecription(cursor.getString(cursor.getColumnIndex(KEY_CLUB_DESCRIPTIO
N)));

clubsClass.setLocaiton(cursor.getString(cursor.getColumnIndex(KEY_CLUB_LOCATION)));

clubsClass.setWebsite(cursor.getString(cursor.getColumnIndex(KEY_CLUB_WEBSITE)));
            clubNames.add(clubsClass);

        } while (cursor.moveToNext());

        cursor.close();
```

```
    }
    return clubNames;

}
```

Web Application (Django)

Accounts – Views “view.py”

This is where the logic behind the user signing up and signing in is, the information is is passed using POST and has error checking in that in the Signup function it will check if a. the passwords match and b. there is a check to see if the user has been already created.

```
# sign up function
def signup(request):
    if request.method =='POST':
        # checking to if passwords match
        if request.POST['password1'] == request.POST['password2']:
            try:
                #checking to see if username is taken
                user = User.objects.get(username=request.POST['username'])
                return render(request, 'accounts/signup.html', {'error':'Username already taken.'})
            except User.DoesNotExist:
                # if the user does not exist it is added to the database
                user = User.objects.create_user(request.POST['username'],password=request.POST['passwo
rd1'])
                login(request, user)
                return render(request, 'accounts/signup.html')
        else:
            #if Passwords dont match it sends an error message
            return render(request, 'accounts/signup.html', {'error':'Passwords don\'t match.'})
    else:
        # if there is no information it will pass the page again to await input "POST"
        return render(request, 'accounts/signup.html')

# login
def loginview(request):
    if request.method =='POST':
        # using Django built in function to authenticate a user logging in
        user = authenticate(username=request.POST['username'], password=request.POST['password'])
        # if the user exists, user gets logged in
        if user is not None:
            login(request, user)
            # if the user wants to get to the accounts section but
            # is not logged in it will redirect the to the login
            # page, but when successfully logged in it will send
            # them to the page they originally wanted to go to
            if 'next' in request.POST:
                    return redirect(request.POST['next'])
            # message to prompt user was successfully logged in
            return render(request, 'accounts/login.html', {'error':'Login successful!'})
        else:
            # if the information was not valid it sends the error message
            return render(request, 'accounts/login.html', {'error':'No valid information.'})
    else:
        # if there is no information it will pass the page again to await input "POST"
```

```
return render(request, 'accounts/login.html')
```

Clubs – Views “views.py”

This is where the logic behind the user searching for clubs is. The home function will just redirect the user to the “home.html” page. The function “listing” will list the depending on what is entered in the search bar on the home page. The query will be set as session variable and then reset when the user the revisits the home page or types a new query. In the “listingcounties” function the clubs are listed based on the county chosen at the home screen. The session variable will be changed to the county name, for example “Meath”, the user can then change this by entering a new county name in the search bar at the top. This may be changed to select from options like on the home page, for now the functionality works as required.

```
# if the user is logged in they can use this functionality where they can post information
# not implemented visually
@login_required
def create(request):
    if request.method =='POST':
        if request.POST['title'] and request.POST['url']:
            post = models.Post()
            post.title = request.POST['title']
            post.url = request.POST['url']
            post.pub_date = timezone.datetime.now()
            post.author = request.user
            post.save()
            return render(request, 'clubs/home.html')
        else:
            return render(request, 'clubs/create.html', {'error':'Sorry, You need to have both a title and a Url t
o create a post.'})

    else:
        return render(request, 'clubs/create.html')

# brings the user to the home page
def home(request):
    # if query has a value, delete it so that it can be set to null for the next query
    try:
        del request.session['query']
    except:
        query = ""

    # send user to home page
    return render(request, 'clubs/home.html')

# lists the counties based on the selection of county on the home screen
def listingCounties(request):
    try:
        # if the method is POST set query to recieve the input and set it as a session variable
        if request.method =='POST':

            query = ""
            query = request.POST.get('County', "")
```

```
            request.session['query'] = query
            # filters the club by the selected county in "query"
            clubs_list = Club.objects.filter(county__icontains= request.session['query'])

        else:
            # filters the club by the selected county already in "query"
            clubs_list = Club.objects.filter(county__icontains= request.session['query'])

    except KeyError:

        #if theres an error reverts back to the query being a name of a club like in "listing"
        query = ""
        clubs_list = Club.objects.order_by('name')

    # shows a max of 15 clubs then overflows onto the next page
    paginator = Paginator(clubs_list, 15)
    page = request.GET.get('page')

    try:

        # sorts the clubs into pages and shows the number at bottom of screen
        clubs = paginator.object_list.order_by('name')
        clubs = paginator.page(page)

    except PageNotAnInteger:
        # If page is not an integer, deliver first page.

        clubs = paginator.object_list.order_by('name')
        clubs = paginator.page(1)

    except EmptyPage:
        # If page is out of range (e.g. 9999), deliver last page of results.

        clubs = paginator.object_list.order_by('name')
        clubs = paginator.page(paginator.num_pages)

    except query:

        clubs = paginator.object_list.order_by('name')
        clubs = paginator.page()

    # sends user to county.html
    return render(request, 'clubs/county.html', {'clubs':clubs})



# lists the counties based on the search of club name on the home screen
def listing(request):
    try:
        # if the method is POST set query to receive the input and set it as a session variable
        if request.method =='POST':
```

```
        query = request.POST.get('search_clubs', "")
        request.session['query'] = query
        clubs_list = Club.objects.filter(name__icontains= request.session['query'])

    else:
        # filters the club by the selected county already in "query"
        clubs_list = Club.objects.filter(name__icontains= request.session['query'])

except KeyError:

    #if theres an error reverts back to the query being a name of a club
    query = ""
    clubs_list = Club.objects.order_by('name')

# shows a max of 15 clubs then overflows onto the next page
paginator = Paginator(clubs_list, 15)
page = request.GET.get('page')

try:

    # sorts the clubs into pages and shows the number at bottom of screen
    clubs = paginator.object_list.order_by('name')
    clubs = paginator.page(page)

except PageNotAnInteger:
    # If page is not an integer, deliver first page.
    clubs = paginator.object_list.order_by('name')
    clubs = paginator.page(1)

except EmptyPage:
    # If page is out of range (e.g. 9999), deliver last page of results.
    clubs = paginator.object_list.order_by('name')
    clubs = paginator.page(paginator.num_pages)

except query:
    clubs = paginator.object_list.order_by('name')
    clubs = paginator.page()

# sends user to club.html
return render(request, 'clubs/club.html', {'clubs':clubs})
```

Club – Models “models.py”

This python file has the models that the web application uses.

- Post model describes how the information for when a user wants to post an item on their blog page will be stored.
- Club model is how the information of all the clubs are stored
- This information when entered will be stored in the SQLite database.

```
class Post(models.Model):
    title = models.CharField(max_length=200)
    image = models.ImageField(upload_to='media/')
    pub_date = models.DateTimeField()
    author = models.ForeignKey(User)
    body = models.TextField()

class Club(models.Model):
    name = models.CharField(max_length= 200)
    county = models.CharField(max_length= 200)
    location = models.URLField()
    province = models.CharField(max_length=50)

    def __str__(self):
        return self.county + " - " + self.name
```

HTML – Main elements and functionality

Base – "base.html"

This file holds the base html structure that is implemented on each html page. It can be updated on this file and then the update spread across all the other files associated with it. It uses "bootstrap" effectively to create an attractive, and responsive design.

- Bootstrap is a free front-end framework for faster and easier web development.
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.
- Bootstrap also gives you the ability to easily create responsive designs [29]

```
<nav class="navbar navbar-default navbar-fixed-top">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="{% url 'home' %}">GPF</a>
    </div>
    <div class="collapse navbar-collapse" id="myNavbar">
      <ul class="nav navbar-nav navbar-right">
        <li><a href="{% url 'home' %}">HOME</a></li>
        <li><a href="{% url 'accounts:login' %}">LOGIN</a></li>
        <li><a href="{% url 'accounts:signup' %}">SIGNUP</a></li>
        <li><a href="#">YOUR PAGE</a></li>
        <li class="dropdown">
          <a class="dropdown-toggle" data-toggle="dropdown" href="#">MORE
          <span class="caret"></span></a>
          <ul class="dropdown-menu">
            <li><a href="#">Login</a></li>
            <li><a href="#">Signup</a></li>
            <li><a href="#">About</a></li>
          </ul>
        </li>
        <li><a href="#"><span class="glyphicon glyphicon-search"></span></a></li>
      </ul>
    </div>
  </div>
</nav>

<div class="jumbotron text-center" >
<br>
  <h2>GAA Pitch Finder</h2>
  <p>Find your Local Club!</p>
</div>

{% block content %}
{% endblock %}
```

```
<br>
<br>

<nav class="navbar1 navbar-default navbar-fixed-bottom text-center">
    <p>Martin Quinn</p>
    <a href="https://github.com/MartinQuinn">GitHub</a>
</nav>
```

The page has an adjustable menu bar for when the page is running on smaller devices such as mobile. This may be useful if the user does not have available storage on their mobile to download the application on the app store. They can use the web app with no issues in the browser, as shown in Fig 4.2.

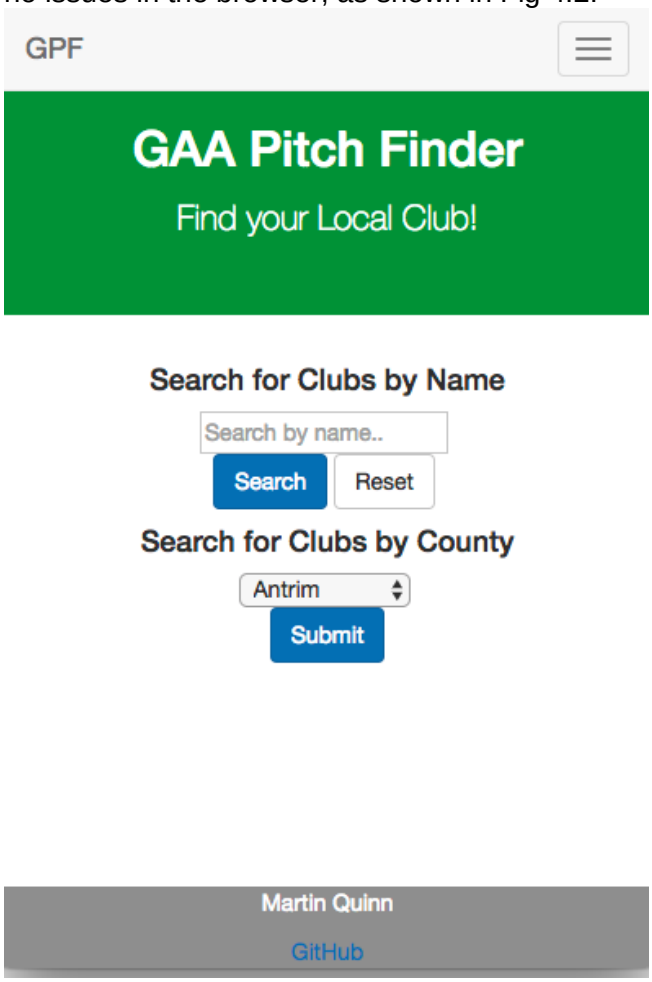


Fig 4.2 Responsive usage of web application

Home page – "clubs/home.html"

This page contains the way in which the user searches for a club and filters by county. Its contains two forms one for the search functionality and one for the filter by county functionality.

- The search functionality POSTs the information to "views.py" and executes "listing".
- The filter functionality POSTs the information to "views.py" and executes "listingcounties".

```
<form action = "{% url 'clubs:listing' %}" method="post">
<div class="container1 text-center" >
   <h4>Search for Clubs by Name</h4>
   {% csrf_token %}
   <input type="text" name="search_clubs" id ="search_clubs" placeholder="Search by name.."></input
>
   <br>
      <button class="btn btn-primary btn-default btn" type="submit" >Search</button>
      <button class="btn btn-default btn" action="{% url 'home' %}" >Reset</button>
   <br>
</form>
</div>
<form method="POST" action="{% url 'clubs:listingcounties' %}">
   <div class="container text-center">
   {% csrf_token %}

   <h4>Search for Clubs by County</h4>
      <select name="County">
         <option name="County" id="County" value="Antrim">Antrim</option>
         <option …> … </option>
         <option …> … </option>
                  etc…
      </select>
      <br>
      <input class="btn btn-primary btn-default btn" type="submit">
   </div>
</form>
```

Club page – "clubs/club.html"

This page displays the clubs in a list using logical code with the html to display the clubs. At the top of the page there is a form that has the exact same functionality as the home page search. Once search a new session variable will be entered in place of "query" and the clubs corresponding to the query will show.

```
<div class="container text-center">
<h3><a href = "{% url 'home' %}">Home</a></h3>
   <form action = "{% url 'clubs:listing' %}" method="post">
   {% csrf_token %}
      <div class="container">
         <div class="row">
            <div id="custom-search-input">
               <div class="input-group col-md-12">
                  <input type="text" name="search_clubs" id ="search_clubs" class="  search-query form-
control" placeholder="Search by Name" />
                  <span class="input-group-btn">
                     <button class="btn btn-danger" type="button">
                  <span class=" glyphicon glyphicon-search"></span>
                     </button>
                  </span>
               </div>
            </div>
         </div>
      </div>
   </form>
```

For each club in clubs it gets the club name first and then labels it with the county name so the user does not get mixed up of what county the club is in. At the far right of the club the button to locate will execute the link and bring the user to google maps.

```
      {% for clubs in clubs %}
      <div class="container1 text-center" >
      <table>
      <tr>
         <th>{{ clubs.name }} - {{ clubs.county }}</th>
         <td>
         <a href="{{ clubs.location }}"><button type="button" class="btn btn-default btn-
sm"><span class="glyphicon glyphicon-map-marker"></span> Locate</button></a>
         </td>
      </tr>
      </table>
      </div>
      {% endfor %}
</div>
```

Paging

This piece of html will order the results into a page system that spreads the results, the amount of clubs allowed on the page is specified in “listing” in “views.py”. Fig 4.3 show the design of the paging system.

```
<div class="container text-center">
   <nav aria-label="Page navigation" >
      <ul class="pagination">
         <li>
            <a href="?page={{ clubs.1 }}" aria-label="Previous">
               <span aria-hidden="true">&laquo;</span>
            </a>
               </li>
               {% if clubs.has_previous %}
               <li><a href="?page={{ clubs.previous_page_number }}">Previous</a></li>
               {% endif %}
               <li><a class="current" href="?page={{ clubs.number }}">Page {{ clubs.number }} of {{ clubs.
paginator.num_pages }}</a></li>
               {% if clubs.has_next %}
               <li><a href="?page={{ clubs.next_page_number }}">Next</a></li>
               {% endif %}
               <li>
            <a href="?page={{ clubs.paginator.num_pages }}" aria-label="Next">
               <span aria-hidden="true">&raquo;</span>
            </a>
         </li>
      </ul>
   </nav>
</div>
```

Fig 4.3 Pagination display

County page – "clubs/county.html"

This page displays the clubs in a list using logical code with the html to display the clubs. At the top of the page there is a form that has the exact same functionality as the home page search. Once search a new session variable will be entered in place of "query" and the clubs corresponding to the query will show.

This is very similar to "club.html", the only difference is that the query will be a county name and not a club name, and it will be POSTed to "views.py" and executed in "listingcounties"

The "paging" works exactly the same as in "club.html", except the information is executed in "listingcounties".

```
<form action = "{% url 'clubs:listingcounties' %}" method="post">
{% csrf_token %}

   <div class="container">
      <div class="row">
         <div id="custom-search-input">
            <div class="input-group col-md-12">
               <input type="text" name="County" id ="County" class="  search-query form-
control" placeholder="Search by Name" />
               <span class="input-group-btn">
                  <button class="btn btn-danger" type="button">
               <span class=" glyphicon glyphicon-search"></span>
                  </button>
               </span>

            </div>
         </div>
      </div>
   </div>
</form>
      {% for clubs in clubs %}
      <div class="container1 text-center" style="margin:0 auto;">
      <table>
      <tr>
         <th>{{ clubs.county }} - {{ clubs.name }}</th>
         <td>
         <a href="{{ clubs.location }}"><button type="button" class="btn btn-default btn-
sm"><span class="glyphicon glyphicon-map-marker"></span> Locate</button></a>
         </td>
      </tr>
      {% endfor %}
      </div>
</div>
```

Security in Django – SQL injection

SQL Injection (SQLi) refers to an injection attack wherein an attacker can execute malicious SQL statements (also commonly referred to as a malicious *payload*) that control a web application's database server (also commonly referred to as a *Relational Database Management System – RDBMS*). Since an SQL Injection vulnerability could possibly affect any website or web application that makes use of an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous of web application vulnerabilities. [30]

By leveraging an SQL Injection vulnerability, given the right circumstances, an attacker can use it to bypass a web application's authentication and authorization mechanisms and retrieve the contents of an entire database. SQL Injection can also be used to add, modify and delete records in a database, affecting data integrity.
To such an extent, SQL Injection can provide an attacker with unauthorized access to sensitive data including, customer data, personally identifiable information (PII), trade secrets, intellectual property and other sensitive information. [30]

Django has the denial of SQL injection built into it's framework. SQL injection is a type of attack where a malicious user is able to execute arbitrary SQL code on a database. This can result in records being deleted or data leakage. By using Django's querysets, the resulting SQL will be properly escaped by the underlying database driver. However, Django also gives developers power to write raw queries or execute custom sql. These capabilities should be used sparingly and you should always be careful to properly escape any parameters that the user can control. [31]

Web Scraping

Python was the language used to scrape the data, “lxml” and “requests” were used together to store information and output it in the console as usable sql inserts.

```
import requests
from lxml import html
```

- antrim_page contains the website that is in question.
- antrim_tree stores the information in antrim_page in a readable format.
- antrim_clubs finds the location of the name of the clubs in the html using xpath.
- antrim_locations finds the location of the location of the clubs in the html using xpath.

```
antrim_page = requests.get('http://www.gaapitchlocator.net/provinces/ulster/antrim/#0/-60/-121')
antrim_tree = html.fromstring(antrim_page.content)
antrim_clubs = antrim_tree.xpath('//a[@title="show marker on map"]/text()')
antrim_locations = antrim_tree.xpath('//div[@class="lmm-listmarkers-panel-
icons"]//a[@title="Get directions"]//@href')
```

This is the code that needs to be run in the python console within the Django app. The for loop is executed adding the information to the database accordingly.

```
for index in range(len(antrim_clubs)):
    p = Club(name=antrim_clubs[index], location=antrim_locations[index], county='Antrim', province='Ulst
er')
    p.save()
```

This is the code for getting the readable code to insert into the “DBmanager.java”.

```
for index in range(len(antrim_clubs)):
    print("\"('32','"+ antrim_clubs[index] +"','N/A','N/A','" + antrim_clubs[index] + "','N/A'),\" +")
```

The output would be able to be put straight into the “DBmanager” java file, the output is shown in Fig 4.4.

```
"('1','All Saints','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.867705,-6.226931&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Antrim Centre of Excellence','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.736927,-6.230541&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Ardoyne Kickhams','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.685185,-5.997307&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Ballymoney','N/A','N/A','https://maps.google.co.uk/maps?daddr=55.067433,-6.520901&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Boucher Road Pitches','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.576092,-5.970855&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Cardinal O'Donnells','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.590510,-5.972614&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Carey Faughs Ballyvoy','N/A','N/A','https://maps.google.co.uk/maps?daddr=55.197996,-6.193843&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Casement Park (County Pitch)','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.573044,-5.984244&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Cherryvale Pitches','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.575146,-5.912061&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Cliftonville GAA','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.617040,-5.946876&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Clooney Gaels Hurling Club','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.823777,-6.387702&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Con Magees Glenravel','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.984611,-6.186435&t=m&layer=1&doflg=ptk&om=1','N/A')," +
"('1','Creggan Kickhams','N/A','N/A','https://maps.google.co.uk/maps?daddr=54.720713,-6.361288&t=m&layer=1&doflg=ptk&om=1','N/A')," +
```

Fig 4.4 output from python code extracting data

Problems during development

The main problem during development was the lack of time to develop the Manager side of the applications and the iOS application.

Another issue was that when developing the home page for each the mobile and web application, the addition of an interactive map was highly prioritised and a full month of development time went into trying to create a map that when clicking on the county on the map it would open and display the clubs which are in that county. The time was inevitably wasted as there was no breakthrough with that functionality and the the idea and the work had to be scraped and a new design had to take its place.

What it involved was to create a map in html, import an image, see Appendix A, which then would need to be mapped out with coordinates. For each county there were a different set of coordinates that drew out the shape of a county, for example Antrim would have a section on the map, that had its value as Antrim to be passed as a POST method in a form, but the information would not pass over. Another issue was that Django does not allow POST through “href”.

The time spent trying to find ways to get this feature to work forced the project to move paths. The attempted code is shown below. The code was to work exactly like that in “clubs/county.html”.

```
<map name="map">
   <img src="{% static 'clubs/Ireland.png' %}" alt="map" usemap="#map"/>
      <map name="Map" id="County">
      <form method="POST" action="{% url 'clubs:listingcounties' %}" >
      {% csrf_token %}
      <input type="hidden" name="County" id= "County" value = "Donegal">
      <input type="submit" value="Submit">
      <area id= "County" title="Donegal" shape="poly" value = "Donegal" coords="236,143,254,140,260,
130,275,132,289,121,268,110,275,100,284,104,301,98,311,71,318,54,328,50,356,23,320,1,307,18,313
,50,294,65,297,54,303,45,302,38,298,34,293,18,284,23,276,25,265,28,252,36,239,36,223,57,229,68,2
22,72,237,78,225,83,217,87,213,93,206,96,199,99,194,105,191,108,191,111,197,115,207,116,217,117
,222,116,225,115,229,115,234,115,238,114,242,114,247,114,249,114,251,114,252,115,252,115,252,1
17,253,119,248,123,247,124,244,125,242,127,242,130,239,130,241,136,243,138,243,139" />
      </form>
         <area id= "County" title="Tyrone"       href="/clubs/listingcounties"   shape="poly" value = "Tyrone
"      coords="274,111,290,115,304,126,297,143,307,150,315,145,320,147,328,156,333,163,339,163,3
49,149,362,156,366,164,371,152,380,152,391,137,389,130,397,122,397,113,387,118,383,118,369,103
,357,103,360,95,359,87,349,89,330,78,318,70,312,86,306,98,289,107,278,104" />
         <area id= "County" title="Derry"       href="/clubs/listingcounties"   shape="poly" value = "Derry"
     coords="319,56,321,66,349,84,362,84,364,90,362,98,371,99,383,112,394,111,399,98,400,86,391,6
0,388,51,389,37,385,30,375,33,358,31,354,50,344,53,339,51,331,56" />
         <area id= "County" title="Armagh"       href="/clubs/listingcounties"   shape="poly" value = "Arma
gh"      coords="373,156,369,167,373,180,381,183,389,185,391,193,388,198,392,207,400,202,403,201
,408,203,412,201,414,197,418,197,413,180,412,165,408,156,411,149,418,146,416,139,408,140,401,1
37,392,137,385,152,383,156,422,140" />
         <area id= "County" title="Down"       href="/clubs/listingcounties"   shape="poly" value = "Down"
       coords="425,140,421,147,413,155,417,168,417,183,419,194,426,198,436,204,443,208,449,203,4
```

```
56,196,457,187,457,179,461,172,467,170,469,175,480,175,483,173,489,166,489,162,484,155,476,159
,480,150,475,132,474,124,487,128,490,142,486,152,493,160,496,150,496,143,497,138,495,130,492,1
23,491,119,483,110,477,111,468,113,462,117,459,125,458,129,455,132,448,135,445,138,438,141,436
,141,431,142,404,103" />
        <area id= "County" title="Antrim"      href="/clubs/listingcounties"   shape="poly" value = "Antrim"
    coords="406,101,392,29,401,21,417,22,426,26,434,25,438,41,437,47,447,51,446,59,448,66,450,7
1,457,74,458,75,459,78,466,85,469,86,470,92,471,97,467,102,455,110,454,114,455,119,456,122,452,
127,446,131,441,133,436,135,433,136,429,136,424,138,421,137,416,137,421,127,422,123,424,122,42
6,118,421,118,425,112,426,108,427,105,427,104,423,103,410,106" />
        <area id= "County" title="Monaghan"     href="/clubs/listingcounties"   shape="poly" value = "Mon
aghan"    coords="328,200,351,199,373,223,382,229,388,222,388,218,394,220,395,220,392,212,384,2
08,384,200,384,195,384,192,383,189,379,190,372,187,367,182,366,179,366,177,365,174,364,168,363
,165,361,162,357,159,355,158,355,158,353,157,352,157,351,157,347,159,346,164,340,166,340,171,3
44,179,344,183,334,185" />
        <area id= "County" title="Fermanagh"    href="/clubs/listingcounties"   shape="poly" value = "Fer
managh"    coords="290,125,299,129,293,139,294,147,301,152,305,155,311,154,317,151,319,153,322,
156,323,159,326,163,330,166,334,170,336,172,338,175,336,179,327,182,327,187,324,191,326,194,31
9,195,310,193,309,196,304,195,299,194,285,185,278,184,277,178,275,172,271,170,264,168,262,159,
255,153,251,150,251,148,252,145,261,140,264,137" />
        <area id= "County" title="Cavan"       href="/clubs/listingcounties"   shape="poly" value = "Cavan
"       coords="267,173,259,184,273,191,280,199,285,200,290,202,302,211,303,214,304,230,303,235,3
13,240,321,241,321,247,326,246,335,247,341,247,347,247,358,247,358,244,358,236,360,234,370,233
,377,231,367,227,364,219,361,216,350,208,347,206,327,202,320,199,314,196,302,200,287,189,278,1
85,271,181" />
    </map>
</map>
```

This wasted time really gave no additional time to build the other features, such as the post mechanism in the web app and there was no time to start development on the iOS app.

The set back in which the data on the “Digital ocean” server was hacked into really knocked the project down a couple of notches also. This was mainly down to not researching the security of a live server compared to a localhost for testing.

Testing

Usability Testing

To test the usability of the system, some players and spectators were used who had games in the same week, some testers who had no knowledge of GAA and would not be using it go to any matches. The testers would go through the use case of finding a club using the android app in real scenarios and, to just test the app to find any implications on the flow. This provided some great feedback on how the system could be improved and how the app looked.

Some of the feedback included:

- Adding a “prettier” design, more colour etc..
- Descriptions on each page to tell the user what to do, what the purpose of has.
- Some faults with club information being displayed overlapping of information.
- Limit user interaction to make the app smooth, which makes for faster navigation.

The use case that was tested in the android app was:

⇒ Search for club → find club from search results → select club and navigate.
⇒ Select specific county → find club in county list → select club and navigate.

They were very little issues with the finished product for the android application. The exclusive of the manager section aided the success of testing due to having a simple app rather than having more chances to go wrong and having a slightly more advanced application, in hind sight the addition of the Manager/Player section would definitely have improved the attraction to the app for the users. The application is now live on the android play store with 50+ downloads and growing.

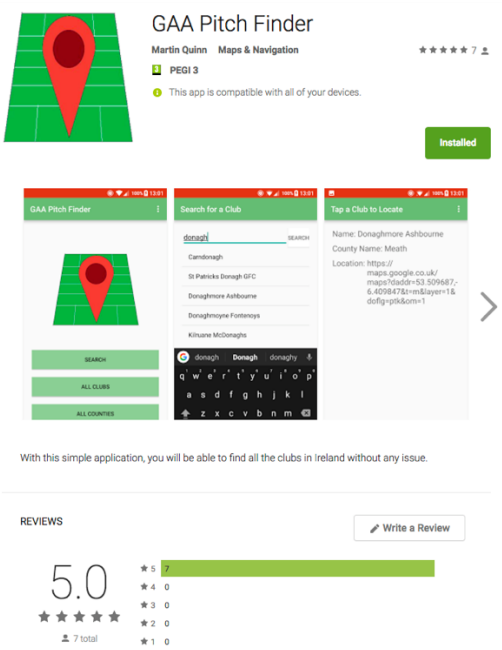


Fig 5.1 GAA Pitch Finder in the app store [32]

How the project changed

The Project changed quite substantially from what was initially proposed. The proposed project was to have 3 applications, one Android application, one iOS application, and one Web application. These 3 applications all feeding off the one database stored on a Digital ocean server, in a MongoDB database. This was not to be.

The issues faced during the project forced the applications to be only live on one and developed on another but as for the iOS application, it was not started or developed, a lot of time went into learning swift and in the future, the knowledge gained through researching and learning this language will be put to good use.

The features that were removed were as follows:

- Manager / Player section in android application
- Manager / Player section in web application
- Full iOS not developed

Although the project was not entirely completed the way in which it is designed allowed for a safe exclusion of a module i.e. the Manager section. The applications work perfectly well when searching for clubs and the main purpose of the project was to get this complete.

If the project was to be repeated from scratch, the main objective would be to get the backend working immediately and address the security issues with great care and assure the data will be safe to continue on to the design and development.

Future work

There are many areas of this project that can and will be improved with additional features and functionality. Some of which were initially intended for this project. Some examples include:

1. The iOS version of the application.
2. Full functionality of the Manager section of each app.
3. The ability to view results and fixtures of the user's club of choice.

Conclusion

On reflection, I am happy with the developed project. Not every proposed feature is implemented or developed. However, the main part of the project was developed fully and live on the Google play store. I think there is plenty more room for development in this project and certainly more features that allow the user to experience a more pleasant and enjoyable time whilst enjoying playing and watching GAA matches. It was very satisfying having people comment on the application, saying how it helped them. Feedback has since been received since the publishing by many players praising the work done, which is very rewarding and exactly what was intended when beginning the project, the main objective was to better the experience of any follower of the GAA.

In closing, I have learned a lot of valuable skills in the course of this project and plan to use these skills to better the approach to future projects, and of course develop this project further.

Bibliography

[1] About the GAA – [Online] Available: http://www.gaa.ie/the-gaa/about-the-gaa/ [Accessed 28-10-2016]

[2] Exhibitions of Croke Park museum – [Online] Available: https://crokepark.ie/gaa-museum-tours/gaa-museum/exhibitions [Accessed 28-10-2016]

[3] Sport in Ireland – [Online] Available: http://www.ireland.com/what-is-available/sports/ [Accessed 28-10-2016]

[4] Stress Management – [Online] Available: http://www.avogel.ca/en/health/stress-anxiety-low-mood/stress/ [Accessed 01-11-2016]

[5] Being late is the number one cause of stress – [Online] Available: http://www.irishmirror.ie/lifestyle/health/being-late-number-one-cause-9444171 [Accessed 01-11-2016]

[6] 5 things that happen to your body when stressed – [Online] Available: https://www.bustle.com/articles/118470-5-things-that-happen-to-your-body-when-youre-stressed-about-being-late [Accessed 01-11-2016]

[7] GAA Survey – [Online] Available: https://goo.gl/forms/w2hrcbXYc6UDwSdq2 [Accessed 29-11-2016]

[8] 10 ease-of-use considerations – [Online] Available: http://www.techrepublic.com/blog/10-things/10-ease-of-use-considerations-for-building-topnotch-apps/ [Accessed 03-11-2016]

[9] The five personalities of tech frustration and how to handle them – [Online] Available: http://www.hrdive.com/news/the-five-personalities-of-tech-frustration-and-how-to-handle-them/414393/ [Accessed 12-11-2016]

[10] Time-management Skills – [Online] Available: http://www.coachingpositiveperformance.com/17-essential-time-management-skills/ [Accessed 20-12-2016]

[11] GAA Pitch Locator – [Online] Available: http://www.gaapitchlocator.net/ [Accessed 03-11-2016]

[12] About DigitalOcean. [Online]. Available: https://www.digitalocean.com/company/press/ [accessed 22-10-2016]

[13] Do What You Could Never Do Before. (n.d.). [Online] Available: https://www.mongodb.com/what-is-mongodb [accessed 22-10-2016].

[14] The Top 7 Free and Open Source Database Software Solutions [Online]. Available: http://blog.capterra.com/free-database-software/ [accessed 22-10-2016].

[15] PHPMYSQL. What is MySQL [Online]. Available: http://www.msinfosolutions.net/training-at-msinfosolutions/programming-languages/php-and-mysql/top-my-sql-php-training-institute-ameerpet-ms-info-solutions-hyderabad.php . [accessed 22-10-2016].

[16] PostgreSQL, About. [Online]. Available: https://www.postgresql.org/about/ [accessed 04-11-2016].

[17] SQLite, About. [Online]. Available: https://sqlite.org/about.html [accessed 22-10-2016].

[18] Python, What is Python. [Online]. Available: https://www.python.org/doc/essays/blurb/ [accessed 22-10-2016].

[19] Why Django. [Online]. Available: https://www.djangoproject.com/start/overview/ [accessed 04-11-2016].

[20] Android, the world's most popular mobile platform. [Online]. Available: https://developer.android.com/about/android.html [accessed 26-09-2016].

[21] J2ObjC. What J2ObjC is. [Online]. Available: http://j2objc.org [accessed 04-11-2016].
[22] Programming with Objective-C. At a Glance [Online]. Available: https://developer.apple.com/library/content/documentation/cocoa/conceptual/programmingwithobjectiveC/Introduction.html#//apple_ref/doc/uid/TP40011210-CH1-SW1 [accessed 04-11-2016].
[23] About Swift. [Online]. Available: https://swift.org/about/#swiftorg-and-open-source [accessed 04-11-2016].
[24] What is JavaScript really? [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics [accessed 05-11-2016]
[25] An Empirical Framework For Learning. [Online]. Available: http://scrummethodology.com/ [accessed 22-10-2016]
[26] Using a Three-Tier Architecture Model. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ms685068(v=vs.85).aspx [accessed 22-10-2016]
[27] MVC. [Online]. Available: https://basicsofwebdevelopment.files.wordpress.com/2015/01/mvc_role_diagram.png [accessed 02-02-2017]
[28] The Model-View-Controller Design Pattern. [Online]. Available: http://djangobook.com/model-view-controller-design-pattern/ [accessed 02-02-2017]
[29] What is Bootstrap. [Online]. Available: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp [accessed 06-02-2017]
[30] SQL Injection. [Online]. Available: https://www.acunetix.com/websitesecurity/sql-injection/ [accessed 03-03-2017]
[31] Security in Django. [Online]. Available: https://docs.djangoproject.com/en/1.10/topics/security/ [accessed 04-03-2017]
[32] GAA Pitch Finder. [Online]. Available: https://play.google.com/store/apps/details?id=martin.quinn.gaapitchfinder [04-04-2017]

Appendix

Appendix A: