|  |  |  |
| --- | --- | --- |
| Quotr | | |
| Ronan Dillon | C12355251 | Fred Mtenzi |

Table of Contents

[1. Project statement 1](#_Toc437277908)

[2. What research has been done and what are the outputs? 1](#_Toc437277909)

[Background research 1](#_Toc437277910)

[Alternative existing solutions to the problem you are solving 3](#_Toc437277911)

[Technologies researched 4](#_Toc437277912)

[Framework 4](#_Toc437277913)

[Databases 6](#_Toc437277914)

[Server 6](#_Toc437277915)

[Other Relevant Tech Research 7](#_Toc437277916)

[Other relevant research done 7](#_Toc437277917)

[Resultant findings/requirements 10](#_Toc437277919)

[Bibliography (research sources) 10](#_Toc437277920)

[3. Analysis: Describe clearly what your solution will do 12](#_Toc437277921)

[4. Approach and Methodology 13](#_Toc437277922)

[5. Design 14](#_Toc437277923)

[Technical architecture diagram: 14](#_Toc437277924)

[Other design documents 15](#_Toc437277926)

[6. Prototyping and Development 16](#_Toc437277927)

[7. Testing 17](#_Toc437277928)

[8. Issues and risks 17](#_Toc437277929)

[9. Plan and future work 18](#_Toc437277930)

[10. Conclusions 18](#_Toc437277931)

# Project statement

The average person will spend nine years of their life in front of a television and with the increasing popularity of instant streaming sites like Netflix and Hulu more people are “binge watching” television shows.(A.C. Nielsen Co. 2015) The idea behind Quotr is that when a user is watching a television show and they have a question like “Who is that actor?”, “Who wrote this show”, “What is the name of this episode?” or even “What is the name of this show again?”, they can find all of those answers and more using the Quotr application. Quotr will take a short segment of the shows audio when the user presses the record button on the app. With this audio segment, Quotr will search for and return details about the show related to the segment.

# What research has been done and what are the outputs?

## Background research

The Internet Movie Database (IMDB) is one of the most popular online databases relating to movies, television shows and video games. It contains information about the cast, director, release date, plot synopses and other details relating to different titles. As of 24th of November the IMDB website was an Alexa Top 50 website. (Alexa top 500 global sites, November 2015) On the 17th of October the company saw its first signs of life, when the founder Col Needham published scripts which could allow people to search through lists of credits. Today, the website welcomes over one hundred million users a month. (IMDb History, 2015) IMDb contains over 3.5 million titles of movies and TV shows, this includes episodes. It has over 6.9 million personalities in its database, credits over 70 million and has 64 million registered users. (IMDb Database Statistics, November 2015)

A typical interaction with IMDB would be to search for a movie and receive details of the movie that was searched for. From there, a user can click on a cast member and go to their profile. If that cast member was an actor, details of all of their acting roles will be displayed along with other details about them. The user can then click into one of the actors movies and view the details of that particular movie and continue on from there. The user can view trivia and other fun facts about the person, movie or TV show they are viewing, along with viewing related titles which the user can click and view also.

July 10th, 2008, the world was first introduced to the Shazam mobile application, one of the first apps available on the AppStore but the company has been around for much longer than that. When the company first started operating the user would dial 2580 and call the number. They would then hold their phone up to the speaker where the song was playing and after thirty seconds the call would automatically hang up. The user would then receive a text message containing the song title and artist name. A user could pay a monthly fee for the usage of the 2580 number or they could pay per tag. (Cooley and Lim, 2006) Today, the user experience is much different, a user opens the Shazam application and on the home page of the application is the Shazam button. The user presses this button to tag the song they are currently listening to. Shazam uses audio fingerprinting to be able to identify music recorded by a user. A user records a short segment of a song which is fingerprinted by the app and compared to Shazam’s database to return the song the user is listening to. When the result is returned the user can see the song title, the artist, the album the song is from and a link is provided to purchase the song on iTunes or on the Google Play Store depending on whether the user is using iOS or Android. (SHAZAM COMPANY, 2015)

Shazam turns a song into a music fingerprint using a spectrogram, a time-frequency graph. Each point on the graph is the intensity of the frequency at certain points during the song. Shazam use these points of intensity to create a hash table with frequency as the key which works as a fingerprint. When Shazam receive a fingerprint it is searched through all matching songs, this will typical return multiple songs, after more fingerprints have been searched a pattern will emerge and Shazam will return the song it has determined is the one the user is looking for. If a distinct pattern is not evident from the finger prints Shazam will return that it can’t find what the user was looking for. Shazam has partnerships with the major music labels so that they can enter the newest songs into their database as soon as they are ready for release. (Jacobs, nd).

Automatic Content Recognition (ACR) is becoming more and more popular in recent years, one particular form of ACR which has gained some traction of late is Speech-To-Text. Speech-to-text has become common place in mobile phones with iPhones Siri and Androids Ok Google function users can enhance their experience by talking to their mobile phones. In Siri’s case, users can use the program to set timers, alarms, check the weather, find their locations and find the nearest restaurants. All the user needs to do is press record and ask the question or request and Siri will use speech-to-text to analyse the request. The user’s request will appear on the screen in text form and Siri’s reply will show up below it, Siri will also reply using text-to-speech. As mobile phones are small, most of the speech-to-text process is carried out on a server, so when a user speaks a request into their phone’s microphone, the data is sent to a central server where it accesses the software and databases. (Alexandra Ossola, August 15 2014)

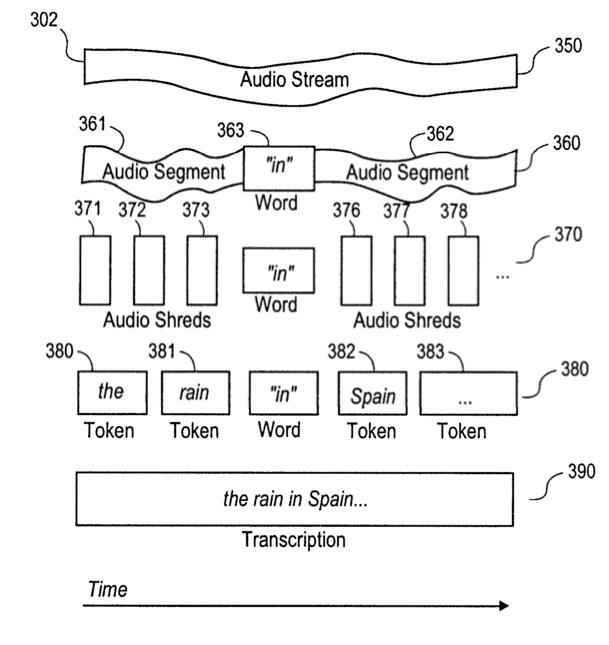


Figure 1 Exemplary Data Stream (Jon Jaroker May 27 2014)

NASA use speech to text on videos on their JPL Tube video sharing network, so that when a user is looking for a video they search for a sentence in the transcript and the returned results are videos with that sentence in them. When a user uploads a video to JPL Tube, a transcript is generated. This transcript is indexed so that it can be searched. When a user search returns a result midway through a video, the video will start at that point. This means a user doesn’t have to watch an hour of the video before getting to the part of the video they are interested in watching. A user can edit the transcript if the speech-to-text didn’t pick it up correctly. (Office of the Chief Knowledge Officer, NASA, 2014)

## Alternative existing solutions to the problem you are solving

## 

Recently, Shazam Entertainment have brought a new feature to their Shazam app. The new feature allows users to tag the television show they are watching, Shazam’s affective use of fingerprinting the app returns details of the show. It will let the users view the cast of the show and to view the latest tweets about the show. Shazam have partnerships with most of the national television channels in the United States allowing them to fingerprint the current shows that are airing on those channels. This means, that Shazam only returns a match if the show the user is watching is currently being broadcast. The application will not return any results if the show is not being broadcast on one of the channels that have a partnership with Shazam. Shazam don’t currently have any partnerships with television channels outside of the US so unless a user has access to US national television channels they will not be able to use this feature. (Ha, 2012)

Shazam also have a relatively new worldwide feature for advertisements. At Shazam Entertainment they have made some partnerships with companies. When an advertisement from one of these partnered companies is shown on television, a Shazam logo will show up tell viewers to tag this add using Shazam to find out more information on the product being advertised. Companies like Calvin Klein and Coca-Cola have been incentivising the use of Shazam. In the case of Calvin Klein customers can walk up to one of their branded podiums, take out the Shazam app and record the current song playing. In doing this simple interaction the customer can receive exclusive content for their mobile or in-store discounts and promotions. (O’Brien, 2011) Coca-Cola recently have an advertisement on television which was released in partnership with Shazam. A user could tag the ad getting themselves a free drink of Coca-Cola in the process. Other companies enter users into draws to win prizes, give away coupons or discounts.

Another mobile application with similar qualities to Quotr is Viggle. The Viggle app allows users to ‘check-in’ to what they are watching. A user collects points when they check-in to a show and then collect more points the longer they watch the show for. They also have in app quizzes about shows currently on air for users to win more points and keep them watching the show. Viggle has a marketplace where users can spend their points on gift cards, movies, Viggle merchandise or even a cruise. (Cunningham and Weinel, 2015)

Similar to Shazam’s television feature, the Viggle app is only available in the US and is also limited to certain channels. Unlike Shazam however, it allows users to check-in to shows that aired within the last seven days. (*Does Viggle work with recorded TV shows?*, 2015) This is a much better improvement to Shazam’s restriction of live television. It still doesn’t have the capability to match television shows which haven’t aired in the last seven days on their selected television shows.

IntoNow was a previous solution to the problem I am trying to solve with my project. IntoNow was started in 2010, twelve weeks after it was released it was bought by Yahoo in the region of $20-30 million. The idea of the application was similar to Viggle, a user tags a show they are watching and sends it to friends so they know what the user is watching. IntoNow, like Viggle and Shazam used audio fingerprinting to detect the TV show or movie that the user is watching. Unlike Viggle and Shazam however, IntoNow had a database which it was constantly adding to which would allow users to watch recorded TV shows and movies and still have the ability to check-in as watching them. (Siegler, 2011)

Unfortunately for IntoNow users, it did not last. In early 2014 Yahoo took IntoNow off the App Store and the Google Play Store. Users were sent emails to tell them that the app would stop working two months later. Although the application itself is gone, the auto content recognition (ACR) technology lives on in some other Yahoo applications like its smart TV app. (Lawler, 2014)

## Technologies researched

## Framework

I have chosen PhoneGap as the framework I will be using in my project. I have chosen it for myriad reasons, the principle one being the variety of devices supported by PhoneGap “PhoneGap supports seven mobile platforms (iOS, Android, HP WebOS, BlackBerry OS, Windows Phone 7, Symbian, Bada); apart from Web apps this is the highest number of different platforms” (Heitkötter, A. Majchrzak, and Hanschke, 2012). As I want to have my app running on multiple platforms this seemed the best for the job. PhoneGap allows a user to use standards-based Web technologies HTML5, JavaScript and CSS3 to create cross-platform mobile apps. (R.M.de Andrade et al., 2015) As I have worked extensively with HTML5, JavaScript and CSS development in a recent internship and also having worked with them the last three and a half years in college I would consider myself well versed. Figure 3 below shows the PhoneGap Architecture, PhoneGap uses native API calls to access mobile features like the camera, and the user’s geolocation. Therefore, using PhoneGap was optimal to using a different framework which I would have to learn a new language in from the start.

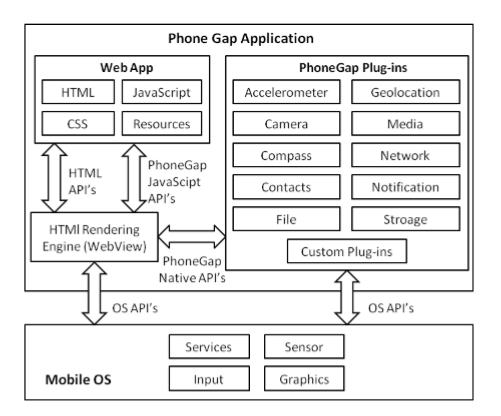


Figure 2 Phone Gap Architecture (Palmieri, Singh, and Cicchetti, 2012)

PhoneGap has a mobile application which allows users to connect to the server that their project is on. This allows the user to view the results of their changes as soon as they make them. This is a big plus for me as it will let me view my progress when I get to the development stage of the project. I have used PhoneGap recently while developing a prototype of my project. I found that using the mobile app while developing to be very efficient as it is much easier to spot when an error has been made.

Another option would be to use Android Developer to create a native Android application. This would limit the amount of platforms the application would be available to Android alone on which wouldn’t be ideal, but developing an Android application natively would provide a better look and performance. As most applications on the Google Play Store are made specifically for Android, users have an expectation when using an Android app. They have become accustom to the native platform which they would find easy to use as it is what they are familiar with. It is not just Android users, iOS users have a similar experience with applications available on the App Store. They have come to expect a style of application for iOS users. When using a hybrid application it is harder to satisfy users on all platforms. (Korf and Oksman, 2015) I have previous developed an Android application using Java, but I haven’t used the Swift language to create an application for iOS. As I would prefer if my application to be available on at least the Android and iOS app stores I would be required to learn Swift if I chose to create native applications for both platforms.

Although native applications have better performance, I chose to do a hybrid application using PhoneGap as I feel with the applications simplistic design, a user on any platform will be able to use it with little or no difficulty. Also with my knowledge of various JavaScript libraries like bootstrap and angular, I believe it would be best to use to make a more aesthetically pleasing application across all platforms.

## Databases

Amazon Web Services ecommerce platform, receives tens of millions of daily customers. This high level of demand means that Amazon require their services to be reliable, as any outages could cause a huge financial lose to both Amazon and their customers which could cause a loss of faith in Amazon’s services by customers. To combat this Amazon offers multiple database management alternatives, that are extremely reliable which makes hosting a database on Amazon Web Services advisable. (George and Mathew, nd) The cost of Amazon database management services is relatively low due to users only paying for what they use. Also, Amazon Web Services offers a free tier for the first twelve months, which gives users free usage to a limit for new users. (Amazon, nd)

Amazon Dynamo DB, the fast, fully managed noSQL database offered by Amazon Web Services. Dynamo Db provides fast and predictable performance. Data in a DynamoDB database are store on a solid state disks (SSD) and are automatically replicated across multiple Availability Zones in an AWS region. This provides the user with built-in high availability and data durability. (Varia and Mathew, 2014) However, as I want my project has the need for a relational database, DynamoDB was not a feasible option as it does not support joins which I will need from the database I use.

Amazon Relational Database Service (RDS) was another option I looked at which is also on Amazon Web Services. RDS offers a MySQL database engine, which I have used previously. RDS comes with Multi-Availability Zones which are automatic backups, these backups are put on a different Availability Zone than the one a user has their original copy of the database on. This is so in the unlikely event that the availability zone in which the user’s original database is on goes down, their backup will not be affected and the user will not lose any data. (Laurence, 2013)

I chose RDS to hold my database as it satisfies all of my needs in a database. In my three years studying computer science and databases I have become proficient in MySQL and as it is an option of use in RDS it is the database engine which I will use. Having a relational database as opposed to a NoSQL database as many of my queries will include the use of joins. With the backups provided using Multi-AZ, the data gathered will be secure in the case of the database going down.

## Server

For my server I will use an Elastic Compute Cloud (EC2) instance on Amazon Web Services. “*Amazon Web Services (AWS) is nowadays the leading public Infrastructureas-a-Service (IaaS) cloud provider in terms of number of users, allowing resources in their data centers to be rented on-demand through Elastic Compute Cloud (EC2) service.” (Expósito et al., 2013)* I chose EC2 due to its scalability so that I can increase or decrease the capacity as I need it. This means that as my application grows in popularity and usage I can increase the capacity to accommodate for the added users. With EC2 I can use this scalability so that I only pay for what I use. The EC2 instance will be located in a Virtual Private Cloud which provides security tools so that the instance is secure. (*Elastic compute cloud (EC2) cloud server & hosting – AWS*, 2010)

I will use an EC2 instance to run scripts, these scripts will include a Speech-to-text script and a script which will call the OMDb API to retrieve data of a movie or show. Although smart phones contain a Speech-to-text functionality I have chosen to run a Speech-to-text script on the server as I can be more or less guaranteed of the script running quickly with the reliability of AWS which can’t be guaranteed on a mobile as some have less processing power than others.

## Other Relevant Tech Research

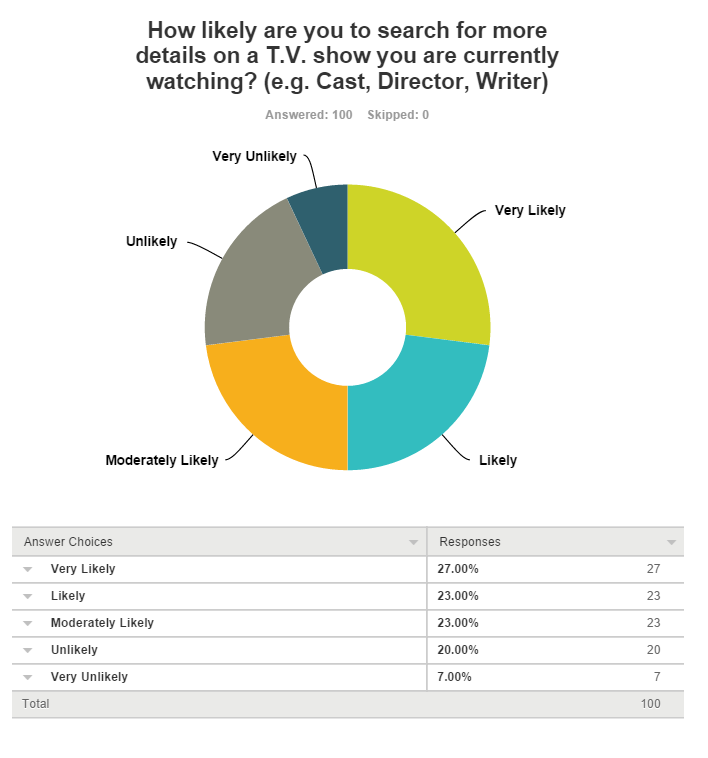
To retrieve data about a TV show or movie I will use the OMDb API, I have chosen this API as it gives me almost all of the data I need to display to users when a match is found. If it the result is a TV show OMDb can specify the show season and episode number to get more precise information about the particular episode that the user is watching. The API returns an object which can be used to separate different pieces of data about the show or movie. For example “data.title” will return the title of the show or movie. (Johann Schaible1 et al. July 15 2015) However, the API is lacking in some areas, it doesn’t return enough actors than I would have hoped. It also leaves out some trivia and other data which I will write a script to resolve myself.

## Other relevant research done

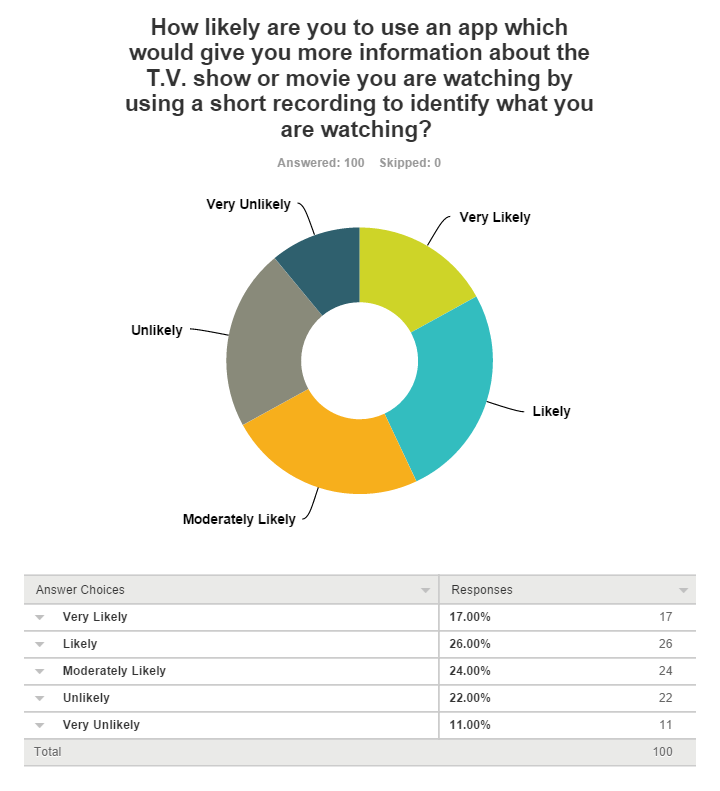
## 

I made a survey using SurveyMonkey and posted it on my Facebook page. The idea of the survey was to see how often people use similar solutions to the problem I am trying to solve and if they would use a mobile application with the same goals as my project.   
 The problem with this particular survey was that as I posted it to Facebook the main demographic is people aged 18-24. As I am in this age group myself, most of the people I am connected to on Facebook are also in that age group. The response rate for people in that age group was seventy-four percent, almost three quarters of the total responses.

SurveyMonkey only allows their users to view data on the first one hundred respondents so although I received over two hundred and fifty answers in total I can only view the first one hundred to respond. This leaves a small sample size.



The goal of my project is to show all of this information and more, so to see that half of the people surveyed are currently likely to use another solution is encouraging as I hope to make my project more accessible and easier to use than current solutions.



This is the basic idea or my project. To take a short recording and use that recording to find the details about the T.V. show or movie the user is watching. I would have preferred to have higher than forty-three percent in the Likely and Very Likely categories.

As my project will use Google to search for quotes in TV shows and movies, part of my research has involved watching television and taking down quotes from a show or movie to find the best algorithm to use when searching to retrieve the best result. I found that entering in a long quote with numerous sentences from multiple characters wouldn’t return the result on most occasions. I tried many other methods to come up with the result most frequently but I found splitting the quote into individual sentences would return a result in a lot of instances, with more correct returns when adding two sentences together.

From this research I concluded that the way in which the correct result can be returned on most occasions from being search is to split the text returned from the speech-to-text into single sentences. Each of these sentences will be searched and the first three results stored, this will be repeated for each sentence and when this is complete, sentences will be added together and researched with a maximum of three sentences per search. When all of the searching has been concluded the most common returned result will be chosen. From testing this method with myriad different TV shows and movies it has proven to be the most effective way of returning the correct result.

## Resultant findings/requirements

I have found that applications which are a similar solution to the problem I am trying to solve with my project use fingerprinting to identify content. To do this they use a hash table of frequencies during certain times of the content, and these hash tables are stored in a database. All of these alternative solutions all fail at recognizing content which has not been on American television channels recently. Some of the solutions can recognize content which has aired in the previous seven days others only allow for content which is currently on air. I emailed a few contacts that I could find Shazam Entertainment’s website to find out the reason for this but none of them have replied. A good guess as to why these applications don’t allow for content recognition of older content is due to the size of the database needed to store every TV show and movie’s audio fingerprints. This finding was a big influence on using speech-to-text over fingerprinting as the way to identify the content.

I discovered the OMDb API and thought it would give me all of the details I need for the movie or TV show that has been searched for, but I discovered that while it does give me a lot of the data I need I will have to write my own script which will access IMDb to get more information on the cast of the TV show or movie as the OMDb API only returns four actors per show or movie. As I plan to have more actors than that in the cast section of the results page I will be required to use the script to get more. To do this I will use a python script which will scrape data from the IMDb page, using an ID which I will retrieve from the OMDb API.

In order to use Amazon Web Services for the use of their database and elastic compute cloud I will be required to create an AWS account. In the early stages of development and releasing of the application it shouldn’t cost too much to have these running, but if the application becomes popular the rise in cost for the use of the services on AWS will rise also. This could mean that in order to keep the application from costing me I may have to add advertisements on the application to cover the amazon web services costs.

After all relevant research has been completed the requirements for the project are as follows. The PhoneGap framework which I will use to create the front-end of the project. Using the hybrid technology I will create an application across multiple platforms. Amazon RDS, the relational database system to store data in a scalable relational database. Amazon EC2, an elastic compute cloud instance will be used to run a server which will handle the scripts. The OMDb API which will be used to collect TV show and movie specific data which will be used in the application.

## Bibliography (research sources)

Alexa top 500 global sites (November 2015) Available at: http://www.alexa.com/topsites (Accessed: 24 November 2015).

Alexandra Ossola, (August 15, 2014) Ever Wondered: How does speech-to-text software work? » Scienceline. 2015. *Ever Wondered: How does speech-to-text software work? » Scienceline*. [ONLINE] Available at:<http://scienceline.org/2014/08/ever-wondered-how-does-speech-to-text-software-work/>. [Accessed 01 December 2015].

Amazon (nd) *Using the free tier - AWS billing and cost management*. Available at: http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/billing-free-tier.html (Accessed: 2 December 2015).

BLS American Time Use Survey, A.C. Nielsen Co., *Television Watching Statistics,* March 2015.

Cooley, B. and Lim, A. (2006) *Andrew Lim*. Available at: http://www.cnet.com/uk/news/shazam-aqa-the-answer-is-on-your-mobile/ (Accessed: 1 December 2015).

Cunningham, S. and Weinel, J. (2015) ‘Second screen comes to the silver screen: A technology feasibility study regarding mobile technologies in the cinema’, *2015 Internet Technologies and Applications (ITA)*, doi: 10.1109/itecha.2015.7317400.

*Does Viggle work with recorded TV shows?* (2015) Available at: http://support.viggle.com/customer/portal/articles/326526-does-viggle-work-with-recorded-tv-shows- (Accessed: 2 December 2015).

*Elastic compute cloud (EC2) cloud server & hosting – AWS* (2010) Available at: https://aws.amazon.com/ec2/ (Accessed: 1 December 2015).

Expósito, R. R., Taboada, G. L., Ramos, S., González-Domínguez, J., Touriño, J. and Doallo, R. (2013) ‘Analysis of I/O performance on an Amazon EC2 cluster compute and high I/O platform’, *Journal of Grid Computing*, 11(4), pp. 613–631. doi: 10.1007/s10723-013-9250-y.

George, K. and Mathew, T. (nd) *Big Database Stores A review on various big data datastores*. Available at: http://www.researchgate.net/profile/Koshy\_George2/publication/283079732\_Big\_Database\_Stores\_A\_review\_on\_various\_big\_data\_datastores/links/562dd92d08ae22b17034c87a.pdf (Accessed: 3 December 2015).

Ha, A. (2012) Shazam makes its big TV push, says App can now tag any show on any channel. Available at: http://techcrunch.com/2012/09/16/shazam-for-tv/ (Accessed: 2 December 2015).

Heitkötter, H., A. Majchrzak, T. and Hanschke, S. (2012) *COMPARING CROSS-PLATFORM DEVELOPMENT APPROACHES FOR MOBILE APPLICATIONS*. Available at: https://www.wi.uni-muenster.de/sites/default/files/public/department/pi/publications/heitkoetter/comparing-cross-platform-development-approaches-for-mobile-applications.pdf (Accessed: 23 November 2015).

IMDb Database Statistics (November 2015) Available at: http://www.imdb.com/stats (Accessed: 10 November 2015).

IMDb History (2015) Available at: <http://www.imdb.com/help/show_leaf?history> (Accessed: 10 November 2015).

Jacobs, B. (nd) How Shazam works to identify (nearly) every song you throw at it. Available at: http://gizmodo.com/5647458/how-shazam-works-to-identify-nearly-every-song-you-throw-at-it?sidebar\_promotions\_icons=testingoff&utm\_expid=66866090-67.e9PWeE2DSnKObFD7vNEoqg.1&utm\_referrer=https%3A%2F%2Fwww.google.ie%2F (Accessed: 24 November 2015).

Johann Schaible , Zeljko Carevic , Oliver Hopt , and Benjamin Zapilko (July 15, 2015) Utilizing the Open Movie Database API for Predicting the Review Class of Movies available at: http://ceur-ws.org/Vol-1365/paper12.pdf

Jon Jaroker (May 27 2014) Exemplary Data Stream, Patent US8738374 - System and method for the secure, real-time, high accuracy conversion of ... - Google Patents. 2015. *Patent US8738374 - System and method for the secure, real-time, high accuracy conversion of ... - Google Patents*. [ONLINE] Available at:<https://www.google.com/patents/US8738374>. [Accessed 03 December 2015].

Korf, M. and Oksman, E. (2015) Native, HTML5, or hybrid: Understanding your mobile application development options - developer.Force.Com. Available at: https://developer.salesforce.com/page/Native,\_HTML5,\_or\_Hybrid:\_Understanding\_Your\_Mobile\_Application\_Development\_Options (Accessed: 2 December 2015).

Laurence (2013) *Get email updates on the progress of my writing*. Available at: http://www.laurencegellert.com/2013/05/pros-and-cons-of-rds-vs-ec2-for-mysql-with-aws/ (Accessed: 3 December 2015).

Lawler, R. (2014) *Yahoo is shutting down IntoNow, nearly Three years after acquisition*. Available at: http://techcrunch.com/2014/01/28/to-every-social-tv-app-turn-turn-turn/ (Accessed: 10 November 2015).

Office of the Chief Knowledge Officer, NASA (2014) *JPL Knowledge Management NEWSLETTER*. Available at: http://km.nasa.gov/wp-content/uploads/sites/3/2014/04/JPL-KM-Newsletter-March-2014.pdf (Accessed: 24 November 2015).

Palmieri, M., Singh, I. and Cicchetti, A. (2012) ‘Comparison of cross-platform mobile development tools’, *2012 16th International Conference on Intelligence in Next Generation Networks*, . doi: 10.1109/icin.2012.6376023.

R.M.de Andrade, P., B.Albuquerque, A., F. Frota, O., V Silveira, R. and A. da Silva, F. (2015) ‘Cross platform App: A comparative study’, *International Journal of Computer Science and Information Technology*, 7(1), pp. 33–40. doi: 10.5121/ijcsit.2015.7104.

SHAZAM COMPANY(2015) Available at: http://www.shazam.com/company (Accessed: 23 November 2015).

Siegler, M. (2011) *Yahoo moves fast — real fast — to scoop up IntoNow for $20 – $30 Million*. Available at: http://techcrunch.com/2011/04/25/yahoo-intonow/ (Accessed: 10 November 2015).

Varia, J. and Mathew, S. (2014) *Overview of Amazon Web Services*. Available at: http://www.fronde.com/assets/Datasheets/AWS-Overview.pdf (Accessed: 3 December 2015).

# Analysis: Describe clearly what your solution will do

The goal of my project is to display details of the television show or movie that a user is watching. To do this I plan on using speech-to-text of a piece of audio recorded by the user. The user takes out their device and presses the record button. Speech-to-text is used to get all of the quotes from the show or movie. The reason why I have chosen Speech-to-text over fingerprinting the audio is due to feasibility. To use audio fingerprinting I would need to have a database full of every television show and movie whereas with Speech-to-text I can search the text to come up with the result.

For the searching of the show or movie I will use a google search and take the first three returned results. I will do this repeatedly for each sentence, I will also add sentences together with a maximum of three sentences and use them to search also. From these multiple searches the most common result will be taken. The most common result will be used as the resulting TV show or movie.

If a definitive result can’t be found the user will be shown an error message. The error message will tell the user that their search was unsuccessful and if they wish to retry to click the record button again. This will be the default page for when a search does not turn up with a result.

When a result is returned the OMDb API will be called which will return details about the show or movie. The details will be in the form of an object, the object will be used to fill in the details into a results page which will be displayed to the user. The API contains details about the show from a plot synopsis to the cast, it also has a link to the poster for the T.V. show or movie. The problem with this API is it does not give me all of the details I want so I will have to write a python script to get more detailed facts about the show or movie. For example the API only returns four actors in the object so in my script I will have to get more of the cast and some other important details in area the API is lacking. When a user gets a result they will be able to share it on social media websites like Facebook and Twitter.

I will use a database to enter the details of the show into so that searching will be faster in future. The database will be searched first to try find a match and if a match can’t be found it will use the API to get the details. In the database there will be an index for recommended shows. If a user searches for two different shows they will be indexed. Each show or movie will have a top five recommended shows or movies, this will be the shows and movies that most users who have searched the original show or movie have also searched for. The user will be able to click on the titles and view details about them.

The user will also be able to view the most popular TV shows and movies of the last week, month, year and of all time to see what other users have been searching for. The application will get this data from the database as it will store how many times each show has been searched for. This will show the trends of different TV shows and movies for the user and will also allow the user to view details on these popular shows.

# Approach and Methodology

I am approaching this project using the personal scrum method. I will have weekly sprints for a small scope on putting the most important work first. With weekly deadlines I will keep myself motivated to meet my weekly goals. If one of my goals hasn’t been met at my weekly sprint it will be moved up in priority. Each task will be given a timeframe of how long I should be spending on that task during the coming week. As I am working on that task I will note the amount of time spent on it and will write down the total time taken for that task at the start of the next sprint. The reason for this is so that I will be able to make better estimations of how much time should be allotted for tasks in the future when I see the actual time taken for previous tasks.

The reason why I have chosen the personal scrum method of development is so that I can keep on track of the tasks that need to be completed using prioritization. I can use sprints to make sure I am meeting my weekly targets and progress with the project at the level I want. I will be able to see if I am falling behind in some areas. With this information I will be able to determine whether or not I will have time to add in some extra features for my project. Although there will have to be a time allowance made for a weekly sprint, it will be worthwhile as it will keep the project on track and allow for more accurate future planning.

If I have time I plan to add some features allowing users to post the show they are watching onto Facebook, Twitter and some other social media websites. This will allow user’s friends see what they are watching and allow them to click into the app to view information about the show that their friend is watching. As this is not of main priority for the apps functionality if I don’t have time to implement it I will leave it out. The app will still function fully without this feature which is why it is expendable. Using the personal scrum method I plan to be able to a lot a time for this feature but if other tasks which take priority over it take longer than planned I will know that I won’t have enough time to implement this feature and can take it out.

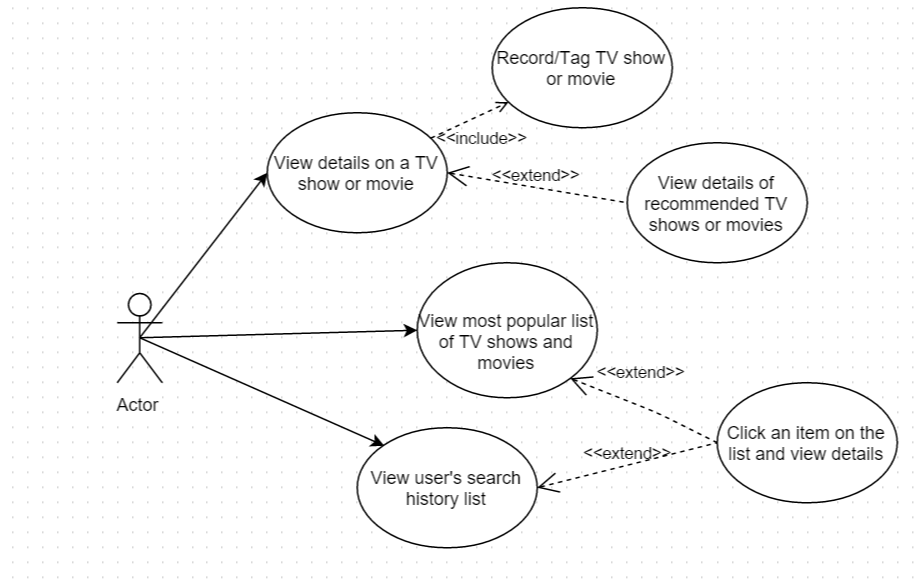
# Design

## Technical architecture diagram:

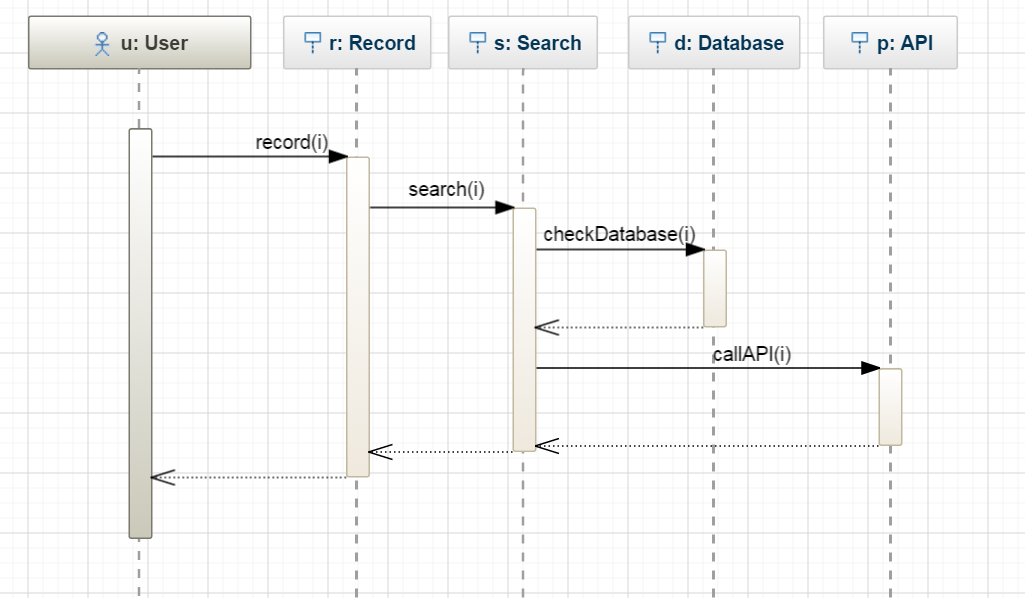
## 

## Other design documents

Use Case Diagram

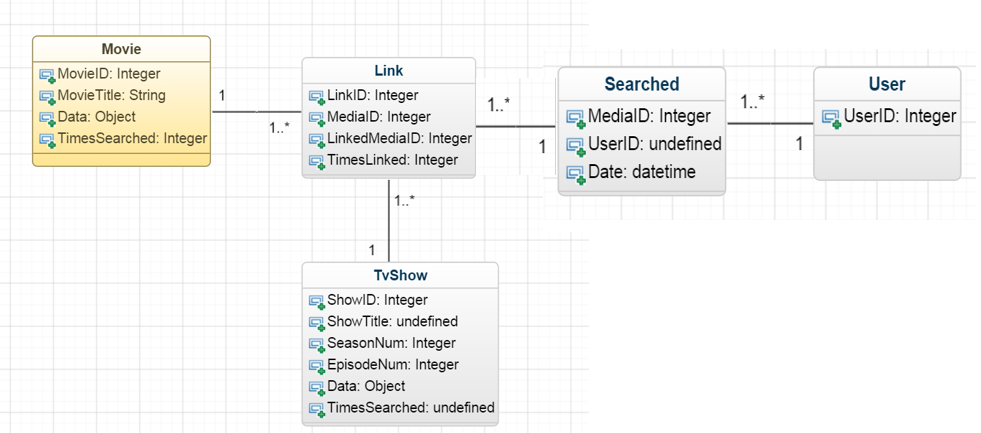


Sequence Diagram



The sequence diagram shown above shows the interaction of a user when they tag a show or movie. The user will press the record button, which will record the show or movie. After speech-to-text has been done on the recording the text will be searched to return a result. When a result is returned the database will be checked to see if data is already in the database for the result. If there is no data for the result, the OMDb API will be called to collect data on the show or movie.

Class Diagram



In the project a user can make many searches, these searches are time stamped so a user can view their most recent searches. If a user searches for the same show or movie that they have searched for before, the time stamp is updated. When a user searches for a show or movie, this show becomes linked with the other shows and movies that the user has also searched for. When a user searches for a show and receives a result, a recommended section will show up. In this recommended section will be movies and TV shows with the most common links to the show or movie that the user has searched for.

TV shows and movies will be separated in the database so that TV show can account for season and episode numbers. Both tables will have a TimesSearched variable, this variable will be used to create the list of most searched for which the user will be able to view on the application.

# Prototyping and Development

Currently, the prototype for the application calls the OMDb API with a set television show. The API returns an object with data concerning the show. As an AWS account is not currently set up JavaScript is used to call the API. There is also no database currently set up so the object is not stored in the database. The object returned from the API call is used to fill out a results page. The data is split into accordions which can be clicked on and opened to show the data about the show. At present the application doesn’t record audio and use speech-to-text to search for the show. The user goes onto the application to see the results page and view the data.

The next stage of development is to create a record button which will record the audio when a user presses it. After a set amount of time passes, the recording will stop and the audio will be sent to a server. On the server, the audio will undergo speech-to-text. After that I can begin searching using the text returned from the audio. When a result is returned from the search I will write a script to search the database for the result and if nothing is found, the API will be called.

# Testing

For the first stage of testing I intend to use QUnit to unit test my PhoneGap application. I will use QUnit to run unit tests on different functionalities of the application to check for bugs. When a new feature is added I will run a new unit test to try to reduce any bugs brought in by the new feature, while also running unit tests on existing features to make sure that no bugs have been uncovered when adding the new feature.

The reason why I have chosen unit testing over other forms of testing like integration testing is that unit testing is narrow in scope. It is easier to write unit tests as they only test small features and rather than test multiple features in one test. If one of my unit tests fail I will know which feature the failure is coming from and this knowledge will speed up the process of identifying the cause of the bug.

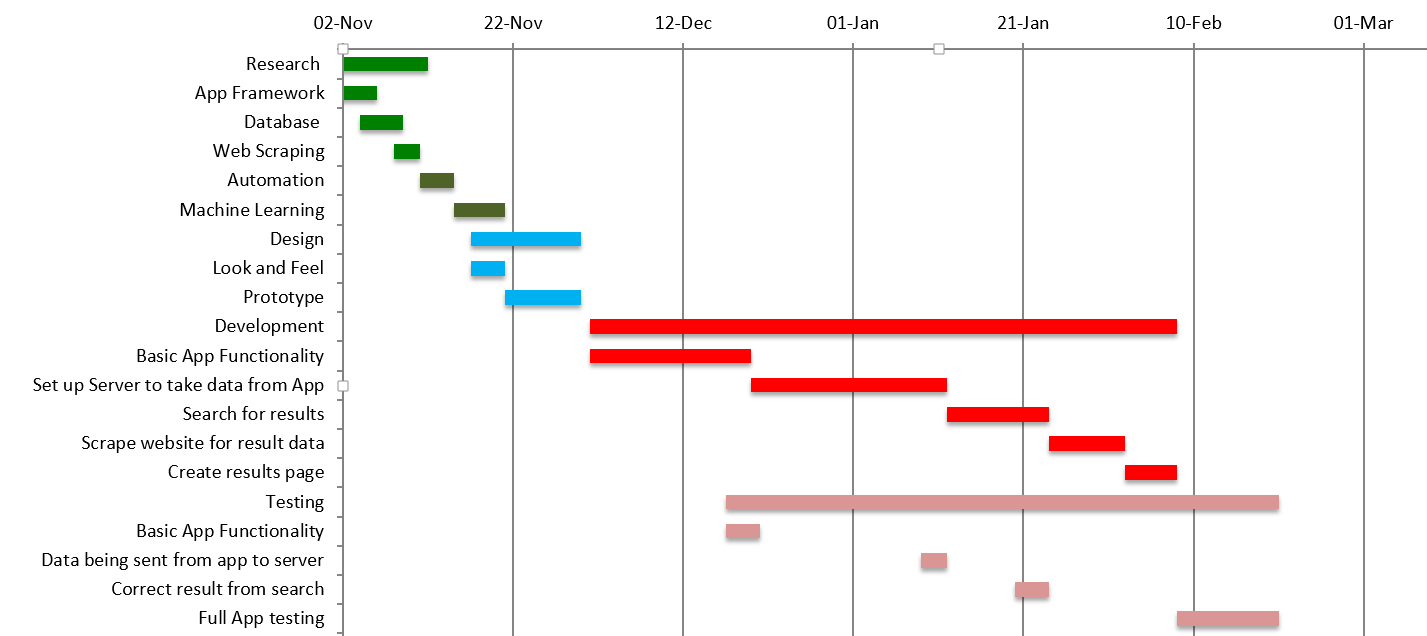
When I am satisfied that I have identified and fixed as many of the bugs that my unit tests can find I will release my apps to a few close friends and family, one of whom is my brother who works in system testing. Each of my chosen testers will be told in advance that if they come across a bug in the application to send an email to me to flag the problem in as much detail as they can. When a problem is flagged I will then work on fixing that bug, re-run my unit tests again to make sure all of them pass and release an updated version of the application to the testers.

# Issues and risks

The main issue with the project is the issue of background noise. Using automatic content recognition (ACR) with audio fingerprinting like Shazam does, the app would be able to work in a noisy room. Shazam is known to be capable of working with significant background noise as it just needs to find a pattern of frequency matches to get a matching song. As I have mentioned previously, using ACR fingerprinting is not feasible with this project due to the significant size of the database needed for finding audio fingerprints in every movie and TV show. Therefore I have chosen to use speech-to-text, this becomes a problem when other people are talking during the recording. If there is another source of speech while the user is recording the television source this could cause a disruption in the text that is created when the speech-to-text is done. This would mean the sentences could become jumbled up and nonsensical which could cause no result being returned or even a different show or movie.

As I can’t physically solve this problem I will have a warning on the application so that users will know that talking after they have hit the record button could alter the results significantly. When a result for a user’s recording can’t be found the user will see on the screen that no results were found and to make sure that sounds other than the TV are kept at a minimal during the recording.

# Plan and future work



The Gantt chart above shows my plan from the start of the project to the end of the project. I have passed the research and design stages and have moved on to the development stage. At the end of each stage of development I will test that each new feature is working correctly before a full application testing in mid-February.

I have started development on the basic application functionality, this is the user’s experience of the application and includes recording. It also includes a results page which takes data from an object and displays information about the show.

The next stage of development will be to set up server on Amazon Web Services and retrieve the audio recording from the application. Speech-to-text will be used on the recording to create a text script which will be used to find the name of the TV show or movie.

Once the development stage has been completed and all unit testing has been done to my satisfaction the application will be released to a small group of friends and family who will start to test the application. Through these tests I will have new sets of eyes looking at the application, able to identify issues that I may have overlooked.

# Conclusions

After some significant background research and research into similar solutions I have found that although there are some similar applications to mine but they lack in some areas. Viggle and Shazam both have the capability to identify shows and movies using audio fingerprinting but neither can use this technology to work on all TV show and movies outside of ones recently broadcast. The idea of my application is to allow for all shows and movies, no matter how long it was off the air. Also my application will be available across the world as I will not need to have partnerships with television channels like my competitors.

I have found research to be a very important tool when doing a large project like this. Things I took for granted like the framework I was going to use changed when I found better solutions. I was originally going to create a native android application because I had created native android applications in the past, but after some research I found creating a hybrid application would be better as I can reach a broader market with the application. Research of APIs provided some very useful feedback as this is how I found the OMDb API which I will be using rather than creating a script to collect the data.

After my research I believe that my project plausibility is solidified. I will be able to create the front-end of the application using PhoneGap and the back-end using an EC2 instance and a MySQL database on RDS. I will use APIs and write scripts in Python to collect data. Together, all of this is completely feasible and I see no reason why this should not work and I fully expect it to be completed in the time frame allowed.