An iOS application for The University Times

Sally Pang
B.A.(Mod.) Business and Computing
Final Year Project April 2014
Supervisor: Dr. Mike Brady

School of Computer Science and Statistics
O'Reilly Institute, Trinity College, Dublin 2, Ireland
Abstract

The aim of this report is to detail the approach taken in the creation of an iOS application for The University Times, the only publication funded by the Trinity College Dublin Student Union.
DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

______________________________________________        ________________________
Name            Date

Name

Date
Acknowledgements

I would like to thank my supervisor Dr. Mike Brady for his continued support, advice and guidance. His experience was invaluable throughout the project. My family and friends for all your help and support and to everyone who downloaded the app. I would also like to extend my gratitude to The University Times and the Student’s Union for giving me the opportunity to create such an application.
Contents

Abstract ii

Acknowledgements iv

List of Figures v

1 Introduction 1
  1.1 Aims .................................................. 3
  1.2 Motivation ............................................. 3
  1.3 Readers Guide ......................................... 5

2 Background 7
  2.1 Rise of mobile ........................................ 8
  2.2 Leveraging print media ................................ 10
  2.3 Research .............................................. 11
  2.4 Chapter Summary ...................................... 12

3 Design 13
  3.1 Problem Overview .................................... 13
  3.2 Application Software Components ..................... 14
    3.2.1 Software Technologies ............................. 15
3.3 Application Hardware components ....................................... 16
3.4 User Interface Design .................................................... 16
  3.4.1 Research .............................................................. 17
  3.4.2 Designing each screen ............................................. 20
3.5 Chapter Summary ......................................................... 21

4 System Architecture ...................................................... 22
  4.1 TableView ............................................................... 23
    4.1.1 Table View Cell .................................................. 23
    4.1.2 Custom Cell ...................................................... 24
    4.1.3 Displaying in the cell ......................................... 25
  4.2 Web View ............................................................... 25
  4.3 What’s happening? .................................................... 25
    4.3.1 What the user sees ............................................. 25
    4.3.2 What’s happening in the background? ....................... 26
  4.4 Alternating from view to view ..................................... 26
  4.5 Chapter Summary .................................................... 27

5 Technical Implementation ............................................... 29
  5.1 Parsing ................................................................. 29
    5.1.1 Parsing method .................................................. 30
  5.2 Two APIs .............................................................. 32
    5.2.1 Original JSON API ............................................. 32
    5.2.2 New JSON API .................................................. 35
    5.2.3 Handling pages .................................................. 37
  5.3 Displaying a list of articles — Article List View Controller .. 38
    5.3.1 Custom Cell ...................................................... 38
    5.3.2 Custom Height ................................................... 39
5.3.3 Retrieving and storing the data ........................................ 40
5.3.4 Number of rows ....................................................... 42
5.3.5 Separating the Title .................................................. 42
5.3.6 Formatting the text .................................................... 43
5.3.7 The Image .............................................................. 44
5.3.8 Result ................................................................. 45
5.3.9 Passing the data ....................................................... 45

5.4 Displaying a list of categories and options — Sections List View
   Controller ................................................................. 47
   5.4.1 Custom Cell ......................................................... 47
   5.4.2 Passing the data ................................................... 50

5.5 Displaying an article — Content View Controller ................... 51
   5.5.1 Components of an article ....................................... 51
   5.5.2 Image .............................................................. 52
   5.5.3 Title ............................................................... 52
   5.5.4 Date ............................................................... 52
   5.5.5 Main Text ......................................................... 53
   5.5.6 Creating the Large HTML String ............................... 55

5.6 Chapter Summary ....................................................... 57

6 Added Features ............................................................ 58
   6.1 Social Integration ..................................................... 59
   6.2 Advertisements ....................................................... 60
   6.3 Future work .......................................................... 62
   6.3.1 Testing ............................................................ 62
   6.3.2 Reading articles offline ....................................... 62
   6.3.3 Memory Leaks ..................................................... 62
6.3.4 iPad ........................................ 63
6.3.5 Paywall ..................................... 63
6.3.6 Source Code ................................. 64
6.4 Chapter Summary ............................. 65

7 Evaluation ..................................... 66
  7.1 Project Successes .......................... 66
  7.2 Difficulties ................................. 67

8 Conclusion .................................... 69

Bibliography .................................... 71
# List of Figures

2.1 Rise of smartphones in recent years [6] .............................................. 8
2.2 Growth in mobile sales [6] ................................................................. 10

3.1 The New York Times (L), The Irish Independent (C), The Irish Times(R) .......................................................... 17
3.2 The New York Times .............................................................................. 18
3.3 The Irish Times ...................................................................................... 19
3.4 The Irish Independent ........................................................................... 19
3.5 Layout used in this project .................................................................... 20

4.1 UITableViewCell [15] .............................................................................. 24
4.2 System architecture ................................................................................ 26
4.3 prepareForSegue method [16] ............................................................... 27

5.1 JSON example ....................................................................................... 31
5.2 Old JSON feed ....................................................................................... 33
5.3 Mock website [20] .................................................................................. 35
5.4 AppFog account ..................................................................................... 36
5.5 JSON API plug-in [21] .......................................................................... 36
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6</td>
<td>New JSON feed</td>
<td>37</td>
</tr>
<tr>
<td>5.7</td>
<td>Custom cell</td>
<td>39</td>
</tr>
<tr>
<td>5.8</td>
<td>Custom cell height depending on content</td>
<td>40</td>
</tr>
<tr>
<td>5.9</td>
<td>JSON feed where ‘posts’ variable needs to be accessed</td>
<td>41</td>
</tr>
<tr>
<td>5.10</td>
<td>The six arrays created and where the values were stored in each array</td>
<td>41</td>
</tr>
<tr>
<td>5.11</td>
<td>Method to separate ‘headline’ and ‘subtitle’</td>
<td>43</td>
</tr>
<tr>
<td>5.12</td>
<td>End result: What the user sees</td>
<td>44</td>
</tr>
<tr>
<td>5.13</td>
<td>Checking, storing and displaying the image</td>
<td>44</td>
</tr>
<tr>
<td>5.14</td>
<td>End result: What the user sees</td>
<td>45</td>
</tr>
<tr>
<td>5.15</td>
<td><code>prepareForSegue</code> method and the variables to be passed into the (Content View Controller)</td>
<td>46</td>
</tr>
<tr>
<td>5.16</td>
<td>Custom category cell</td>
<td>47</td>
</tr>
<tr>
<td>5.17</td>
<td>Category Object (L), Custom Cell (R)</td>
<td>48</td>
</tr>
<tr>
<td>5.18</td>
<td>A list of categories and choices available to the user</td>
<td>49</td>
</tr>
<tr>
<td>5.19</td>
<td>Alert view for ‘Feedback’ and ‘About Us’ sections</td>
<td>49</td>
</tr>
<tr>
<td>5.20</td>
<td>Link to Email</td>
<td>50</td>
</tr>
<tr>
<td>5.21</td>
<td><code>prepareForSegue</code> method</td>
<td>51</td>
</tr>
<tr>
<td>5.22</td>
<td>Method used to identify an image</td>
<td>54</td>
</tr>
<tr>
<td>5.23</td>
<td>Method used to resize an image by keeping its aspect ratio</td>
<td>55</td>
</tr>
<tr>
<td>5.24</td>
<td>How the strings are concatenated together</td>
<td>56</td>
</tr>
<tr>
<td>5.25</td>
<td>What the user sees when an article is accessed</td>
<td>56</td>
</tr>
<tr>
<td>6.1</td>
<td>Social networking accounts in device settings</td>
<td>60</td>
</tr>
<tr>
<td>6.2</td>
<td>Article List View Controller with ad banner</td>
<td>61</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This report will detail the creation of an iOS application for The University Times, one of Trinity Colleges student newspapers. The report will examine each stage of the process, from the market analysis completed to the future body of work. This project followed the software development life cycle, a structure used in software creation. This chapter will begin with an introduction to the project followed by a readers guide.

Print media has undergone transformative change in recent years. New technologies have brought about disruptive change for the industry, introducing new channels of distribution as well as increasing interaction with their readers.[1] News outlets have been forced to adopt new technologies or face being left behind. In the context of this change, the aim of this project was to analyse, design and implement the creation of an iOS application for The University Times. The World Wide Web had a huge impact on print media upon its adoption. It added a distribution channel that allowed for instant publishing and access by anyone, anywhere. This decreased the restrictions
Chapter 1. Introduction

the physical print schedule placed on publishing. Social media allowed for an extension of this digital distribution channel, increasing the interaction between readers and media through commenting and sharing. The adoption of mobile technologies through smartphones and tablets has again marked a huge change in the industry. It has been said that the switch to mobile technologies has changed the way people consume media even more than the switch from print to digital did. This quote from the Managing Director of the Financial Times, Robert Grimshaw, illustrates the impact that mobile technology has had on the industry:

“My best guess is that mobile will be the lead channel for delivery within three to four years... I think the switch to mobile is bigger in magnitude than the switch from print to desktop, in terms of what it means for the way people consume content, and its happening faster”[2]

The evolution of the app ecosystem has seen newspapers build and optimise their own applications to create the best possible user experience. A presence on the app ecosystem in the form of both Android and iOS has become a core competency for newspapers. Many innovative publications are prioritising digital content over physical print. The Guardian and the Financial Times have implemented digital-first strategies.[3][4] The aim of this project was to bring The University Times in line with industry standards through the introduction of a mobile application. This report will detail every aspect of the project including the background, technologies, design, implementation and evaluation of the finished product. It will also examine future work that may be implemented on the product. Software is continually evaluated and
updated when deemed necessary. As such, there is a body of work that could be implemented on this app pending further evaluation and research as will be discussed later in this report. This chapter will set out the aims and motivations of the project before providing a readers guide to the report.

1.1 Aims

The primary aim of this project was to create a professional standard newspaper application for The University Times for iOS devices. This was benchmarked against publications with an established presence on the mobile ecosystem, as will be seen in the design section. This project targeted iPhone and iPod Touch devices.

The project had three high-level objectives:

1. Display a list of articles to the user.
2. Allow the user to view a list of articles under a category of choice.
3. Allow the user to read an entire article.

1.2 Motivation

This project was inspired by the changes that the adoption of mobile technologies have had on print media. The benefits of having a mobile application have made them core competencies for publications.

The addition of a mobile application to the publishing of a newspaper allows the publication to create an optimised environment for the user as
well as benefiting from increased revenue opportunities through subscription paywalls or mobile advertising. It allows users to instantly access articles anywhere in the world on mobile devices. The University Times current print schedule is tri-weekly, however articles are published digitally almost daily. These articles were previously only available on the website. There was a clear need for a mobile presence given the increased significance of mobile technologies and the adoption by all major newspapers of mobile apps. Having a dedicated mobile app for publications has become a core competency for competing in the print media market. Given the decreasing significance of browsers[5] and the growth of apps on mobile technologies it was clear a dedicated app for the publication could increase readership and access. Mobile consumers who wish to access publications expect a distribution channel designed and optimised for their devices as browsers cannot offer the same standards.

The University Times is one of the primary sources of news for Trinity College Dublin. The college has combined staff and student population of roughly 20,000 as well as tens of thousands of alumni. The university system within Ireland is increasingly relying on alumni donations for funding so universities are trying to improve links with alumni worldwide. Print media produced from within universities is one of the most accessible links alumni can make to the community. This added distribution channel could act an important connection to alumni around the world for Trinity College. This project was innovative as, to the author’s knowledge, the first dedicated student newspaper app of its kind in Ireland.
1.3 Readers Guide

Chapter 2 — Background
This chapter examines the impact mobile technology has had on print media and the implications of this for this project.

Chapter 3 — Design
This chapter deals with the design process involved in creating the application. It begins by identifying the high level functions necessary for the product to meet the consumers needs. It then details the hardware and software stack used in designing the User Interface (UI) to fit these specifications. It also examines some established iOS newspaper apps that were used to benchmark the design.

Chapter 4 — System Architecture
This chapter incorporates the designs in Chapter 3 to create the system architecture. This chapter explains the different components that make up the system architecture and how each screen is linked within the architecture.

Chapter 5 — Technical Implementation
This chapter incorporates the design and system architecture in the overall technical implementation of the project. This chapter consists of four implementation sections.

Section 1 Explains the parsing method used to retrieve the content from the server to the application.

Section 2 Illustrates how the content retrieved from Section One displays in
the article list.

Section 3 Illustrates how to display the category list and the about us and feedback section.

Section 4 Illustrates how a full article is displayed.

Chapter 6 — Added Features
Building on Chapter 5 this chapter incorporates additional features into the project. This includes social media integration and advertisements. It also examines future work that may be carried out as well as the possibility of licensing the source code.

Chapter 7 — Evaluation
Chapter 7 will give an overall evaluation of the project.

Chapter 8 — Conclusion
Chapter 2

Background

The University Times is one of Trinity’s student newspapers. It was founded in 2009. It replaced its predecessor the University Record. It is the only student newspaper funded by Trinitys Student Union. The newspaper has won many awards from “Irish Student Newspaper of the Year” to “Web Design of the Year”. It prints tri-weekly and published digitally on a WordPress website. It makes over 12,000 social media impressions, between Facebook and Twitter, with each post.

The rise of mobile technologies in the world has been very swift. Since the iPhone was launched in 2007 there has been an explosion in the use of smartphones (See Figure 2.1). The growing use of tablets has accelerated this affect. This has had a huge impact on many industries. This chapter will examine first the rise of mobile technology and then the effects this has had on print media. The implications for The University Times of this change will also be discussed. At the end of this analysis of the industry, we will see why a mobile presence was needed for The University Times and why iOS
was chosen.

## 2.1 Rise of mobile

The rise of mobile technologies such as smartphones, tablets and their associated software has marked a huge shift in the way society interacts with technology. This is most pronounced in the developed world, but the developing world is beginning to leapfrog the use of landlines and desktop computers to smartphones and mobile telephones. The evolution of the app ecosystems has affected many industries. Figure 2.1 shows the rise of smartphones in recent years.

![Graph showing the rise of smartphones in recent years](image)

**Figure 2.1: Rise of smartphones in recent years [6]**

Media must make their content available online to any potential customer.
worldwide, whether it be free or behind a pay wall. This involves optimising the content to all devices used in accessing it. This allows the publication to curate the best possible user experience and maximise advertising revenue where applicable. The New York Times was one of the first newspapers to create a mobile application in 2008. All major newspapers worldwide have followed suit in order to remain relevant and accessible. The New York Times has identified mobile apps as a core part of their future strategy. In 2014 they begun unbundling their core product with the launch of NYT Now, a subscription app for news updates. They will be releasing two further apps this year as part of this strategy. Social media too has impacted on print medias online presence as publications use posts to drive traffic. The adoption of mobile technologies has been the biggest shift since the world went digital in recent years. The number of PC sales have levelled off and begun to dip with smartphone and tablet sales eating their market (See Figure 2.2).
2.2 Leveraging print media

There are now four methods of distribution that print media use. They are physical, social, web and mobile. The University Times is currently missing this mobile platform in its distribution channels. As we have seen, mobile is the fastest growing distribution platform and may soon be the largest. With the paper only printing tri-weekly they are already limiting their readership. By launching the newspaper on a mobile platform, they are allowing for increased convenience and accessibility to their current and potential readers.

Following this analysis of the impact of mobile technology on print media this information was gathered and applied to the research and design of this project.
Chapter 2. Background

This report will be focusing on the impact of mobile technology and print media and in particular newspapers. The Guardian was one of the world's first newspapers to recognise the challenges and opportunities appearing as a result of this shift in technology.

This quote from the editor Alan Rusbridger illustrates two of the most important changes in print media:

"Were no longer a once-a-day text medium for a predominantly domestic audience. Increasingly around the clock we use a combination of media in telling stories, and in commentary, to millions of users around the globe."[8, pg. 106]

Content publication is no longer limited to the physical print schedule and the audience is no longer confined to those with physical access to the publication. Access to mobile Internet has increased these effects.

2.3 Research

When it was decided that a mobile distribution platform for The University Times was necessary, a decision had to be made regarding which operating system it would be built for. Market analysis was completed which found that iOS was the most popular mobile operating system among Trinity College students connected to the TCD Wi-Fi network. It had 6454 iOS devices (iPhone, iPad, iPod Touch) connected to the network, compared to the 3538 Android and 67 Windows OS devices.[9] The decision to create the application for iOS was made having completed this research. iOS newspaper applications have the choice of either being placed in the App Store as a native stand alone
app or in the Newsstand. Alternatively they can optimise their website for mobile browsers. A 2013 report by the US analytics company Flurry showed that upwards of 80% of time spent on smartphones is within apps and not in web browsers. In addition to this all major newspapers have created a standalone presence within the app ecosystem either within the Newsstand or as a native app.[5]

Having researched the options it was decided to place the app in the App Store as a Native app and not in the Newsstand. This was due to the reason that Newsstand offers subscriptions for their newspapers however The University Times does not offer subscriptions to readers online. Whilst a subscription service would generate revenue and offer exclusivity, the paper would not benefit from this, as the majority of its readers are students who would most likely not pay for such a service.

2.4 Chapter Summary

Having examined the increasing significance of mobile technologies in today’s world and the effects this has had on print media, the next task was to design an application that was optimised for users and was of industry standards. This analysis showed that publications must maintain some mobile presence in order to compete. This chapter finally looked at why the iOS operating system was chosen and where it was placed within this ecosystem.
Chapter 3

Design

Software technologies and hardware components used throughout the project are discussed in this chapter, whilst briefly explaining why each was chosen and how they were utilised. The User Interface (UI) design will also be discussed in this chapter.

3.1 Problem Overview

The overall aim of this project was to provide readers of The University Times with an additional channel of distribution to access the publications content. Developing the app for iOS devices allowed the project to utilise the various features iOS development offers and customise it for an optimal user experience.

One overall objective for this project was to be able to retrieve articles from the server and display these on iOS devices.

The general overview of the approach taken in completing the project will
be explored here.

At the highest level the application performs the following:

1. Parses article data from the server to the application.

2. Displays this data to the user in the form of an article list.

3. Allow the user to filter articles by categories of interest.

4. Allow the user to read an entire article.

5. Allow the user to post articles through social media sites.

6. Allow users to read a short summary about the app, the newspaper and the developer.

7. Allow the user to send feedback directly to the developer.

8. Display advertisements to generate revenue.

These steps were then broken down into multiple sub steps, which will be explained throughout the report.

### 3.2 Application Software Components

**iOS SDK**

iOS SDK is a third party developer development kit released by Apple in 2008. The SDK includes a debugger, software libraries and an emulator to test applications.[10]
3.2.1 Software Technologies

Below is the list of software technologies employed throughout the project.

- **Objective-C**
  Objective-C is the programming language used to write iOS applications. This project also used the Cocoa Touch Library. Cocoa Touch is a collection of Objective-C classes.\[11, p.28\] Objective-C is an object orientated programming language similar to C++, Java and C#.

- **JavaScript Object Notation — JSON**
  JavaScript Object Notation (JSON) transmits data from a server to a web application. This was used in this project to retrieve the data from the server to the application.\[11, p.520\] JSON is like XML, which is a human-readable data interchange format. However JSON is a lot more concise so it takes up less memory and is faster to transfer across networks. XML is becoming less popular, with JSON becoming the new industry standard. A RESTful JSON API was used to retrieve and manipulate WordPress content using HTTP requests. The API allowed for three main goals:

  1. Provide a simple, consistent external interface.
  2. Create a stable, understandable internal implementation.
  3. Enable new types of extensions for WordPress.

- **WordPress**
Wordpress is an open source tool which can be used to make blogs or websites. It is built on PHP and MySQL and runs on a web hosting service.

- **HTML**
  HyperText Markup Language (HTML) is the markup language used in web development, which allows web pages to display in web browsers.[12]

- **GitHub**
  GitHub uses web-hosting services for software development projects using git revision control. A GitHub private repository was created to allow revision control on the project.[13]

### 3.3 Application Hardware components

The hardware components necessary were:

- **iOS device — iPhone/iPod Touch**
  These physical devices were used to test the application. They allowed for a physical prototype. 3.5 and 4-inch devices were used for testing.

### 3.4 User Interface Design

The User Interface (UI) was one of the most important aspects to the app. If the UI was unappealing users would reject the application and once rejected it would very hard to draw the user back in. As with any software project an understanding of human user interface interaction is vital, along with
researching other players in this market to understand industry standards and what features have been successfully implemented.

3.4.1 Research

To design the UI, research into other iOS versions of newspaper applications was necessary. Below are screenshots of three well known newspaper iOS applications (See Figure 3.1).

1. The New York Times

2. The Irish Independent

3. The Irish Times

Figure 3.1: The New York Times (L), The Irish Independent (C), The Irish Times (R)
Chapter 3. Design

It was decided with the client that The New York Times simple and clear layout would be the favoured design to benchmark against. The alternative designs looked at were less attractive due to the smaller article cells squeezed into their tableviews.

When an article is accessed, all three newspapers display the content in a similar fashion, which includes a title, an image, an author and body of the text, which may or may not contain images. To choose a category of interest, both The New York Times and The Irish Times display this in a side bar whereas the Irish Independent displays this in a tab view in a tool bar (See Figures 3.2, 3.3, 3.4).

![Figure 3.2: The New York Times](image)

Here the various categories are displayed in a side bar. The content of an article consists of a large image, title, author and main text.
Chapter 3. Design

Figure 3.3: The Irish Times
Here the list of categories is displayed in a side bar. The content of an article consists of a large image, title, author and main text.

Figure 3.4: The Irish Independent
Here the list of categories is displayed in a tab bar at the bottom of the screen. The content of an article consists of a large image, title, author and main text.
Having examined these other newspaper applications, it was necessary to decide what the user would see when the application was launched. This project took the New York Times as an example of what the end prototype should look like along with its usability. All the publications examined used similar design components for the article pages. These were taken to be industry standards and applied to the project.

3.4.2 Designing each screen

Whilst using the New York Times as a template the UI was sketched to mimic theirs. Having sketched by hand, the storyboard feature in Xcode was then used to give a more realistic view of the features in each screen that the user will see (See Figure 3.5).

Figure 3.5: Layout used in this project
3.5 Chapter Summary

This chapter dealt with the design process involved in creating the application. It began by identifying the high level functions necessary for the product to meet the consumers needs. It then detailed the hardware and software stack used in designing the User Interface (UI) to fit these specifications. It also examined some established iOS newspaper apps that were used to benchmark the design. The next chapter takes this design and applies it to create the system architecture upon which the back-end can be implemented.
Chapter 4

System Architecture

The User Interface (UI) designs in Chapter 3 were used to design each screen the user sees. These will be referred to as view controllers. “A UIViewController instance specialises in controlling a single screen within an application”.[14, p.145]

These view controllers are a vital link between an apps data and its visual appearance. When an iOS app displays a user interface, a view controller or a group of view controllers coordinating with each other manage the displayed content. Therefore, view controllers provide the skeletal framework on which apps are built.[14, p.145]

Three view controllers were created:

1. Article List View Controller
   This was created to display a list of articles.

2. Sections List View Controller
   This was created to display a list of categories.
3. Content View Controller

This was created to allow the full content of an article to be read.

The UITableView layout was used in both the Sections View Controller and the Article List View Controller. This was to allow data to be displayed in a table view form with different text in each cell. A UIWebView was used in the Content View Controller to display an entire article.

4.1 Table View

A UITableView was used to display a single column of data with a variable number of rows.[14, p.187] This table view is the first view that the user sees once the application is launched. The UITableView is used in conjunction with the UITableViewController and UITableViewCell in order to display a list of articles similar to that of the contacts page on an iOS device.

4.1.1 Table View Cell

Each row in a UITableView is a UITableViewCell where data can be displayed. The UITableViewCell is a subclass of UIView that adds certain properties and functionalities to a UIView.[14, p.198] Each cell contains one subview where data can be displayed. A UITableViewCell contains a basic layout which comprises of an image on the left, text and detailed text label in the centre and an accessory button on the right (See Figure 4.1).
Chapter 4. System Architecture

The table view consists of a number of cells. These can either be static or prototype cells. Static cells allow the user to set the number of cells they need and its contents. Prototype cells allow for dynamic constantly changing data to be stored in the cells, including the amount of cells to be displayed. In the case of this project, prototype cells were used as the data being displayed was constantly changing. A custom subclass of `UITableViewCell` was implemented in both the Sections List View Controller and the Article List View Controller where prototype cells are used.

### 4.1.2 Custom Cell

A subclass of `UITableViewCell` was used to allow for customisation and control of the project. Two custom cells were created for this project. One cell was in the Sections List View Controller to display various categories and options and another cell was in the Article List View Controller to display a list of articles.
4.1.3 Displaying in the cell

To display text in each cell the default method

```objective-c
cellForRowAtIndexPath:(NSIndexPath *)indexPath
```

was used.

4.2 Web View

A `UIWebView` allows the rendering of web content.[14, p.456] Safari also uses this to display and render web content. A `UIWebView` was used in the Article List View Controller and the Content View Controller. This was to allow the JSON being retrieved to keep its HTML characteristics and render them into the web view.

4.3 What’s happening?

When the app is launched what the users sees is different to what is happening in the background. This section will briefly explain what happens in both the front and back end.

4.3.1 What the user sees

When the user first launches the application they are greeted with a launch image and then the list of the most recently added articles. From here the user can either select a category in which they would like to view a list of articles or select an article already displayed in the list.
4.3.2 What’s happening in the background?

What the user sees in the front is quite different to what is happening in the background of the application. The first view controller the user sees is the **Article List View Controller** (list of articles), in the background however the **Sections View Controller** (category list) is instantiated first. An **appDelegate** provided allows the app to do this. The reason behind doing so is to let list of categories load first, making the program more efficient. This is similar to The New York Times, The Irish Times and The Irish Independent.

![Figure 4.2: System architecture](image)

4.4 Alternating from view to view

As previously mentioned the user can switch from view to view (screen to screen) displaying different content on each screen. This can be done by either connecting the screens via the storyboard feature or by the **prepareForSegue** method.
This project used the `prepareForSegue` method instead of the storyboard feature, as it allowed data to be passed from one view controller to the next. The storyboard feature doesn’t allow this. The `prepareForSegue` method was used, enabling the application to pass data from one view to the next. In the case of this project, various characteristics of an article need to be passed from the Article List View Controller to the Content View Controller. Items such as the article name, date, images and main text need to be passed through to display in the next view (Content View Controller). When the user selects a category this needs to be brought forward to the Article List View Controller to allow the app to know which articles to display. The `prepareForSegue` method ultimately tells the view controllers what data they are sending and which view controller to send this data to.

\[
- \text{(void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender}
\]

Figure 4.3: `prepareForSegue` method [16]

### 4.5 Chapter Summary

Upon completion of the UI design, the system architecture was created to house the UI and allow for navigation within the app. The view controllers acted as each screen view as seen by the user. The `prepareForSegue` method was used to allow interaction between the view controllers. The system architecture acted as the app’s foundation, allowing for the integration of the UI design and the technical implementation. The next chapter involves
integrating the content drawn from the server into the system architecture.
Chapter 5

Technical Implementation

This chapter will illustrate how the design phases were used in conjunction with the system architecture to produce a working prototype. Parsing was one of the most vital elements to this project as it retrieved the data from the server to the application. This chapter is divided into four sections, the first of which explains how parsing was done. The second section illustrates how the lists of articles was displayed and used the parsing method from Section One to render the data for the entire application. This was then used to pass data into the view controllers as will be explained in Sections Three and Four.

5.1 Parsing

Parsing is the division of statements into different parts. It is the process of retrieving the content from the server and displaying it to the user. In this project a JSON API was used to fetch the data. The API sends HTTP requests to the web server which responds with the data requested. After
Chapter 5. Technical Implementation

this parsing could then begin. The current website is a WordPress site with this API installed. This chapter explains the structure and methods used to parse JSON data from the server to the application. The manipulation of this data to fit the design specifications will be explained in Chapter 6. The implementation phase of the project was the most time consuming and the section in which the largest issues arose.

5.1.1 Parsing method

In order to parse, a URL containing the JSON data was necessary. This URL was then stored as a string, as follows:

1. Request the data (URL) and store in a string.

2. Convert the string into an **NSData** object.

   **NSData** is a buffer that Objective-C provides in order to create, maintain and destroy these objects. An **NSData** instance holds some number of bytes of binary data and in this instance was used to store the URL data. [14, p. 289]

3. Parse the data object using **NSJSONSerialization** without options.

   In iOS the **NSJSONSerialization** is used to deal with JSON. This class can convert a large amount of JSON data and turn it into objects and vice versa. This **NSJSONSerialization** class takes the data and returns either a **NSArray** or a **NSDictionary** and within these arrays and dictionaries are strings, numbers or more dictionaries and arrays. [14, p. 521]
Figure 5.1: JSON example

```json
{
    "status": "ok",
    "count": 10,
    "count_total": 2572,
    "pages": 258,
    "posts": [
        {
            "id": 25025,
            "type": "post",
            "slug": "student-protests-over-nuig-same-sex-referendum-leads-to-gardai-involvement",
            "url": "http://\//universitytimes.ie/?p=25025",
            "status": "publish",
            "title": " Student Protests over NUIG Same-Sex Referendum Leads to Gardai Involvement <small class=\"subtitle\">Polling took place throughout yesterday and resulted in a pro-same-sex marriage stance, with 2,214 votes in favour and 114 votes in opposition.<\/small>\",
            "date": "2014-03-14 14:39:31",
            "modified": "2014-03-14 14:39:31",
            "categories": [
                {
                    "id": 9,
                    "slug": "newsfeed",
                    "title": "NewsFeed",
                    "description": "",
                    "parent": 0,
                    "post_count": 726
                }
            ]
        }
    ]
}
```
5.2 Two APIs

This project used two JSON APIs, the original and the current API. Issues with the original API resulted in a new API needing to be installed. Both these APIs are explained in the next two subsections.

5.2.1 Original JSON API

Previously, each article and its various characteristics were stored under its own article number. These articles numbers were then stored in a directory numbered from the oldest article to the newest. In order to get each article and its contents, the whole directory needed to be passed in as an array and then flipped back to front, in order to get the newest article first. Once this was completed each articles number was accessed in order to get the contents of each article. With articles dating back to 1970, loading and flipping the array needed to be done in a speedy manner. To accommodate this, asynchronous loading was used to push the data out as the content was being loaded. This allowed the user to see the articles as they were being loaded in the background.

The JSON structure for each article was also limited to only ‘body’, ‘heading’, ‘image URL’, ‘thumbnail’ and ‘id’ (See Figure 5.2). This was an issue as there were no subtitles to the articles and instead it would need to be taken from the actual content of each article. Dates of articles were also not present in the JSON feed. As the API was plugged into a WordPress website the feed being retrieved was expected to be in HTML form. Although the feed met the basic requirements, it was inconsistent. It
contained characters not usually seen in HTML such as \r and \n form. These often flowed onto the screen. In some articles the body of the text included an image reference, which was often empty.

```json
{
    "Body": "<strong>Anna Nichols</strong> | Contributing Writer

Wednesday 5th February saw the Society for International Affairs and Diplomacy host one of its most ambitious and prevalent events to date when it held a panel discussion in the Innovation Academy concerning the political upheaval in Bahrain. Unrest began in 2011, and has particularly affected the medical profession, but events have received little media. The guest speakers were Irish doctor and leader of a recent Irish humanitarian delegation to Bahrain Dr Damien McCormack, independent human rights activist Tara O’Grady, deputy director of Frontline Defenders Andrew Anderson and Bahraini doctor Dr Fatima Hajjim.

Although each speaker addressed a different aspect of the situation, all four guests acknowledged the importance of the event and its capacity to raise awareness of the abuses that continue to be carried out by the Bahraini regime, and to provide a forum for discussion where thoughts, questions and first hand accounts of the events could be exchanged.

Dr McCormack was the first to speak, focusing on the links between the Irish medical community and the situation in Bahrain,

"ID": "24310",
"Heading": "Silence is A War Crime"
}
```

Figure 5.2: Old JSON feed
Two considerations arose with these issues, either replace the API or salvage the feed. The client was reluctant to replace the feed so an attempt was made to salvage it.

Stripping the ‘body’ of the text of all HTML and its various other characters was necessary in order to leave the plain text. Stripping out the image was also needed as the image URLs were null. The attempt to do this was unsuccessful as it was difficult to identify the other various characters. The app was not performing optimally and it was not the high quality and standard that was agreed. These issues were brought back to the client who also agreed the API wasn’t performing as well as it should. Although in agreement, the client was still reluctant to install a new plug-in as it would take time to implement and test a new API. The client proposed that the articles could be shown in the web view by using the articles own URL. This was a valid way of completing the project however it was not the most efficient way of doing so. This method would be no different to reading articles in a web browser like Safari, which was one of the reasons the client wanted to create a native application. When the newspaper is accessed on a mobile device using a web browser the toolbar floats on the screen, scrolling as the user also scrolls. This is a suboptimal design that does not make full use of the space available. This was explained and shown to the client yet they were still reluctant to implement a new API.

As a result of this the timeline was pushed back. It was necessary to display to the client that a new JSON API was vital and could be installed easily providing a consistent reconstructed feed.
5.2.2 New JSON API

To show the client a new API was imperative, a duplicate University Times website was created to mimic the current site (See Figure 5.3). The idea behind this was to implement a new JSON API and test the feed coming out to see if an API was the solution (See Figure 5.5). An AppFog\[17\] account was created to host the duplicate site (See Figure 5.4). A WordPress\[18\] account was also created to post the articles (See Figure 5.3). Within this WordPress site a RESTful JSON\[19\] API was installed.

Figure 5.3: Mock website \[20\]
With the new API installed, the feed retrieved was able to:

1. Display content in pure HTML form.

2. Increase the characteristics available within the article such as title, subtitle, images of different sizes, date and time, categories and page counts.

This new feed also allowed the program to run at a much higher speed, as the articles were stored newest first (See Figure 5.6).
5.2.3 Handling pages

The new API could retrieve over 250 pages however only the first page was shown. http://universitytimes.ie/?json=id&page=1 was the URL used to retrieve the feed. Here the count is set to ‘1’ which was why only the first page displayed. To combat this issue the page number in the URL is set to ‘i’. http://universitytimes.ie/?json=id&page=i

This allowed a for loop to iterate around and increase the page count each time or for as long as the program allowed it to. With ten articles in each page it was important to set the article list to a reasonable number as to not have a never ending scroll. In the case of this project the page count was set
to three. This meant thirty articles would display in each category, provided there are thirty articles to display.

5.3 Displaying a list of articles — Article List View Controller

This section explained how the design, parsing and system architecture were combined to display lists of articles. This view controller used a UITableView that used UITableViewCell where a custom cell was created. The next section will illustrate how the data is stored and parsed to display the article list in the custom cell.

5.3.1 Custom Cell

In this view controller the custom cell comprised of a UIWebView and a UIImage. The UIWebView rendered the web content allowing for the display of HTML. This allowed the program to keep the articles original format and font style as explained in Chapter 4. Subclassing allowed for a UIImage to be placed on the right hand side of the cell rather than the left (see Figure 5.7).
5.3.2 Custom Height

When the custom cell is placed in the table view, the height of the table is usually static. With headlines and subtitles being different length this app took the text that was displayed in the cell and adjusted the cell height according to the corresponding text length (See Figure 5.8).
5.3.3 Retrieving and storing the data

Figure 5.9 shows the ‘posts’ variable, which stores many of the values including the title and thumbnail that are needed for this program. This meant that all the variables in ‘posts’ needed to be stored in order to access the two values. valueForKey was used to get the data from the dictionaries and arrays (See Figure 5.10). Three arrays were created to store these characteristics (posts [], article [], articleImages []) and another three arrays were created to copy them (contentArray [], artIMG [], articleTitles []). This was used to combat count issues that arose when looping through the pages of the API.
Figure 5.9: JSON feed where ‘posts’ variable needs to be accessed

```json

"posts": [
  {
    "id": 25034,
    "type": "post",
    "slug": "tobacco-free-trinity-consultation-continues-despite-referendum-result",
    "url": "http://www.universitytimes.ie/?p=25034",
    "posted": "publish",
    "content": "Tobacco Free Trinity Consultation Continues Despite Referendum Result was not determinative of College policy."
  },
  {
    "id": 25034,
    "type": "post",
    "slug": "tobacco-free-trinity-consultation-continues-despite-referendum-result",
    "url": "http://www.universitytimes.ie/?p=25034",
    "posted": "publish",
    "content": "Tobacco Free Trinity Consultation Continues Despite Referendum Result was not determinative of College policy."
  },
  {
    "id": 25034,
    "type": "post",
    "slug": "tobacco-free-trinity-consultation-continues-despite-referendum-result",
    "url": "http://www.universitytimes.ie/?p=25034",
    "posted": "publish",
    "content": "Tobacco Free Trinity Consultation Continues Despite Referendum Result was not determinative of College policy."
  },
  {
    "id": 25034,
    "type": "post",
    "slug": "tobacco-free-trinity-consultation-continues-despite-referendum-result",
    "url": "http://www.universitytimes.ie/?p=25034",
    "posted": "publish",
    "content": "Tobacco Free Trinity Consultation Continues Despite Referendum Result was not determinative of College policy."
  },
  {
    "id": 25034,
    "type": "post",
    "slug": "tobacco-free-trinity-consultation-continues-despite-referendum-result",
    "url": "http://www.universitytimes.ie/?p=25034",
    "posted": "publish",
    "content": "Tobacco Free Trinity Consultation Continues Despite Referendum Result was not determinative of College policy."
  }
]
```

Figure 5.10: The six arrays created and where the values were stored in each array

```javascript
posts = [jsonDict valueForKey: @"posts" ];
article = [posts valueForKey: @"title" ];
articleImages = [posts valueForKey: @"thumbnail" ];

[contentArray addObjectsFromArray: posts ];
[artIMG addObjectsFromArray: articleImages ];
[articleTitles addObjectsFromArray: article ];
```
5.3.4 Number of rows

As the number of rows differ depending on whether a category contains the full 30 articles, it was not possible to set the number of rows to a static figure. Instead it was set to the count of the array articleTitle[]. Alternatively the count could have been set to the count of contentArray[] or artIM[]. However posts[], article[] and articleImages[] may not be have been used as the figures double every time the program iterated over the pages.

5.3.5 Separating the Title

When the program takes the ‘title’ from the JSON feed the ‘title’ consists of both the headline and subtitle. These two characteristics needed to be separated to have the headline stand out and be larger than the subtitle. To do so the NSScanner method was used to scan the string up until the first < character, store this in a large string and then store the next < character in a small string. This was done as HTML ends and starts with <> labels (See Figure 5.11). This maintained the HTML characteristics when displaying the text in the web view. Once the strings were separated, formatting could then commence.
5.3.6 Formatting the text

<font face='TimesNewRomanPS-BoldMT' size='4.8'> was added to the front of the title to make the text large and bold allowing it to stand out while also complying with HTML formatting standards.

<br />
<font face='TimesNewRomanPSMT' size='3'> was added to the front of the subtitle, including a line break, so as to push the subtitle onto the next line.

Once formatting was complete the strings were then concatenated together to make one large HTML string and then display in the web view (See Figure 5.12).
5.3.7 The Image

Displaying an image was similar to displaying a title and subtitle. There were not always images in articles so this was tested. Figure 5.13 checked if there was any thumbnails present and if so then the `NSData` retrieved the image objects within the URL, which then got converted and stored as an image (See Figure 5.13).

```objective-c
if (thumbnailURL == (id)[NSNull null]) {
} else {
    NSURL *url = [NSURL URLWithString:thumbnailURL];
    NSData *data2 = [[NSData alloc] initWithContentsOfURL:url];
    UIImage *tmpImage = [[UIImage alloc] initWithData:data2];

    cell.customImageView.image = tmpImage;
}
```

Figure 5.13: Checking, storing and displaying the image
5.3.8 Result

Once the title was separated and formatted and the image (if any) was retrieved and stored successfully, the program was then able to run these two elements together at the same time. Figure 5.14 shows a list of articles that the user would see.

![The University Times](image)

**24-Hour Ushers**
*Extension Delayed Until Summer 2014*
Work on this extension was originally planned to begin in January of this year.

**Body Image Panel**
The University Times teamed up with Q-Soc and TCSU to host a panel discussion about body image in the media.

**Moschino – Fast Fashion**
Megan Bums takes a look at Jeremy Scott's first collection for Moschino

Figure 5.14: End result: What the user sees

5.3.9 Passing the data

Chapter 4 explained the use of the `prepareForSegue` method. If the user selected an article from the list, this method passed the data the program deemed necessary into a selected view controller, in this case the (Content View Controller). The data brought forward contained the values for the image above the article, the plain title of the article, the date and the article
text (See Figure 5.15). The width of the image was also retrieved here and was used in the next view controller to help handle and resize images within an article. This will be further explained in Section Four. The title in this case was resized as a header in size 5 (h5) and the date was formatted to the paragraph size.

Another element that was taken was the URL of the article. This was also passed forward for posting to Facebook and Twitter, which will be explained in Chapter 6.

- (void)prepareForSegue: (UIStoryboardSegue *)segue sender:(id)sender
{
    if ([segue identifier] isEqualToString: @"PushContent") {
        ContentViewController * CV = segue.destinationViewController;

        NSString * content =[[contentArray objectAtIndex:index] valueForKey:@"content" ];
        images = [[contentArray valueForKey:@"thumbnail_images"] valueForKey:@"medium" ];
        CV.MediumIMGURL = [[images objectForKey:index] valueForKey:@"url" ];
        NSNumber * width=[[images objectForKey:index] valueForKey:@"width" ];
        CV.contents=content;
        CV.articleURL =[[contentArray objectAtIndex:index] valueForKey:@"url" ];
        CV.IMGURL = [NSURL URLWithString:CV.MediumIMGURL];
        CV.WIDTH = [width integerValue];
        NSString * plainTitle = [[contentArray objectAtIndex:index] objectForKey:@"title_plain" ];
        NSString * header=[@"<h5>" stringByAppendingString:plainTitle];
        CV.plain=[header stringByAppendingString:@"</h5>" ];
        NSString * modDate =[[contentArray objectAtIndex:index] objectForKey:@"date" ];
        NSString * headerDate=[@"<p>" stringByAppendingString:modDate];
        CV.date=[headerDate stringByAppendingString:@"</p>" ];
    }
}

Figure 5.15: prepareForSegue method and the variables to be passed into the (Content View Controller)
5.4 Displaying a list of categories and options

--- Sections List View Controller

This section explains how the custom cell was used to display the category list and options. One of the project objectives was to allow the user to view a list of articles under a category of choice. Once a selection had been made the list of articles displayed in the Article List View Controller, as explained in Section Two of this chapter. The user is able to move between these two view controllers to see various articles under different categories. ‘About Us’ and ‘Feedback’ sections were also introduced which come in the form of alert views, which will be explained.

5.4.1 Custom Cell

A custom cell was created to display the list of categories and icons along with the ‘About Us’ and ‘Feedback’ sections. The custom cell consisted of a UIImage on the left and a text label on the right (See Figure 5.16). This was similar to the default cell but it made any future customisation easier.

![Custom category cell](image)

Figure 5.16: Custom category cell

The list of categories was coded in manually rather than being fetched through the JSON feed. This was decided with the client as the articles date
back to the 1970s with many categories not being used. This allowed for greater efficiency within the application. The categories include NewsFeed, InFocus, Opinion, Societies, Sport, Magazine and Review. ‘About Us and ‘Feedback sections were also added (See Figure 5.18). These elements were added as NSObjects in a NSMutableArray, categoryName under the string named catName. Icons stored in images were also stored in the categoryName[] array as catImage in the categoryObject class (See Figure 5.17) within the Content View Controller they were stored in a mutable array named categoryItems. The count of this array was used when determining the amount of rows in the table. In the case of the ‘About Us’ section, the alert view displays a piece of text about The University Times and this application (See Figure 5.19). If the user chooses the send feedback option an alert view pops up which has a button to send feedback (See Figure 5.19). When the button is pressed the user is redirected to a ‘Compose an Email’ (See Figure 5.20) page. The developer e-mail and subject have been pre-filled with the user only having to enter the feedback.
Figure 5.18: A list of categories and choices available to the user

Figure 5.19: Alert view for ‘Feedback’ and ‘About Us’ sections
5.4.2 Passing the data

When a category was selected the selection was passed forward into the Article List View Controller where the appropriate list of articles was displayed. The Article List View Controller is pushed forward as it is set as the Destination View Controller. As explained in Chapter 4 the prepareForSegue method was used to do so. Passing the data forward allowed the Article List View Controller the appropriate list of articles under the category selected by the user.
Chapter 5. Technical Implementation

5.5 Displaying an article — Content View Controller

The `UIView` in this view controller contained a `UIWebView` to render the web content retrieved. The `prepareForSegue` method in Section Two passed variables associated with the selected article into this view controller. An article consisted of an image, title, date, author and main text. This section will explain the steps taken in displaying each variable in the web view.

5.5.1 Components of an article

The image, title and date did not require extensive formatting or resizing as it was already done. The main text contained both the author and body of the article and needed to be scanned for images. This allowed the images to be resized to ensure the article and its contents stay within the scope of the device screen. Once this was completed each element stated was then concatenated together to make one large HTML string which displayed in the web view. Each of the steps will be explained separately throughout this
chapter. The content of the article can be broken down into four elements in the web view.

1. An image
2. Title
3. Date
4. Author and main text of the article

5.5.2 Image

The image retrieved was stored as a string, this needed to be converted into HTML as a UIWebView was used. ‘<img src=’ was added to the beginning of the string to comply with HTML standards. To ensure the image fit the screen of the device a medium thumbnail sized image was chosen.

5.5.3 Title

Having retrieved the title from the previous view controller here the text was formatted to ostrich-black-webfont and of a header size 3. These were placed in HTML labels to allow the web view to render the content.

5.5.4 Date

The date was left the same with only the font changing as specified above.
5.5.5 Main Text

The main text of an article was stored as one large string. Embedded in the main text are images that were displayed within the article. These images when examined have the same sizing as they would when displayed on the WordPress site. As the screen size on iOS device is significantly smaller than that of a PC these images were resized accordingly. The text was not resized as it follows the HTML within the article.

In the Article List View Controller, the width of a medium sized image was stored and passed into this view controller to aid the resizing of an image within an article. The width stored was used with the height varying as to keep the aspect ratio. The content of an article was first scanned for image references. The reference ‘<img src=’ was used in order to do so. The offset of the images width and height in the HTML string was calculated so the range position of the image matched that of HTML string. This enabled the program to alter the size after the string was modified. This was done by nesting an if statement within a for loop.

The NSRegularExpression was invoked every time an image matched a portion of the reference.[22] The main text was then scanned for image sources to ensure all images were picked up. Case sensitivity was turned off to ensure this.

NSTextCheckingResult was used to describe items located by text checking. Each of these objects represent an occurrence of requested textual content that was found during the analysis of a block of text.[23]
Figure 5.22: Method used to identify an image

(void)replaceCharactersInRange:(NSRange)aRange withString:
(NSString *)aString

This statement resized the width and height of the image appropriately. Once the width and height were found, the method replaces the unwanted characters in the string with the necessary characters.
5.5.6 Creating the Large HTML String

If there are no images in the main text of the article or the image(s) has already been correctly proportioned, then the program skipped to this stage. Taking all the areas discussed above, the large HTML string was then concatenated to give the full article to the user. Figure 5.24 illustrates how this was done with Figure 5.25 showing what the user sees.
Chapter 5. Technical Implementation

Figure 5.24: How the strings are concatenated together

```objective-c
NSString *image = [NSstring stringByAppendingString: MediumIMGURL];
NSString *imgAndHTML = [NSstring stringByAppendingString: '<img src="" style=""/>'];
NSString *titleAndHTML = [NSstring stringByAppendingString: title];
NSString *imgAndTitle = [NSstring stringByAppendingString: title];
NSString *imgTitleAndHTML = [NSstring stringByAppendingString: titleAndHTML];
NSString *full = [NSstring stringByAppendingString: modifiedHTML];
WebView loadHTMLString: [NSstring stringWithFormat: "%<div align=""%>", full]; baseURL:nil;
```

Figure 5.25: What the user sees when an article is accessed.

(The colours represent the different elements put together to display in the web view.

Blue - Large image above title

Green - Title

Orange - Date

Dark Red - Main text of the article

Yellow - Image within the article)
5.6 Chapter Summary

When the system architecture was successfully created, the next task consisted of parsing the content from the server to the app and housing it within the system architecture. The parsing was initially unsuccessful due to the original inconsistent JSON feed. When this problem was identified and solved by the use of the new API, the content within the app could then be customised. The JSON feed was parsed into the Article List View Controller where it was rendered and passed into the other view controllers. Industry standard sections, ‘About Us’ and ‘Feedback’, were added to the Sections List View Controller. Feedback is extremely valuable to developers as it allows them to interact directly with the apps users and prioritise future work. Each view controller was customised upon the successful rendering in order to create an attractive page for the user. This chapter detailed the creation of a functional prototype including all the clients basic specifications. This marked the completion of the app’s core components as set out at the start of the project. The next chapter will go on to examine the additional features that were added.
Chapter 6

Added Features

Once the core objectives were completed with the creation of a functional prototype, extra features could be added into the application. The two features implemented first into this app were social media integration and advertisements. There are additional features that may be introduced in future updates that will also discussed. The source code created in this project holds the potential to be used on other projects which will be examined finally. As mentioned in the background to this project, social media has become an important driver of traffic for print media. It was decided early that the ability to share articles on social media was one of the most desired functionalities beyond the core objectives. Advertisements are implemented in many apps due to their revenue generating ability. It was incorporated into this project to offer the client an alternative revenue source. This chapter will illustrate how these features were integrated into this project.
6.1 Social Integration

Apple provides a Social Framework that allows developers integrate their apps to supported social networking services. The supported social network services included in this build were Facebook and Twitter. This framework provides a template for creating HTTP requests. It also provides a generalised interface for posting requests on behalf of the user.[11, p.363]

This framework allows users to:

1. Create a network session.
2. Get the activity feed for a user.
3. Make a new post.
4. Set properties on a post, add attachments, etc.
5. Publish a post to an activity feed.

Posts made to either social networking site are only posted to the account logged in at the settings level of the device (see Figure 6.1) and not in standalone apps. The icons for both Facebook and Twitter were placed in a toolbar. This toolbar is only visible within the Content View Controller where a full article can be read. This toolbar is static, allowing the user to post to either social networking site as they read the article. If a user is not signed into these networks at the device level these buttons will be inactive and non-clickable.
6.2 Advertisements

When conducting the market analysis on this project it became clear that advertising was a popular means of monetization within the industry. Advertisements are common in free apps. After the core objectives were completed, it was decided ads would be implemented. Further monetization routes will be examined in the future, such as the possibility of a paywall. To implement ads the iAd Framework supplied by Apple was used. Within this application ads are only displayed in the Article List View Controller (Chapter 5). The ad is placed at the bottom of the screen and once clicked the entire ad is displayed. The ad exits only when the user chooses to exit the ad.
There is a significant body of work that may be implemented into this app in the future. These will be assessed and prioritised for future releases in conjunction with the strategic plan for the publication. A number of these are discussed in the following section.
6.3 Future work

6.3.1 Testing

This app was tested throughout the implementation process using both 3.5 and 4-inch iOS devices. There were many inconsistencies with sizing and optimisation that became evident during testing on these two devices that would not have been noticed if only one sized device was used. Rigorous testing such as unit testing was not incorporated. With more time this type of testing should be looked into as rigorous testing will show if and where the app has errors or needs changing.

6.3.2 Reading articles offline

Many newspapers applications offer users the ability to read articles offline. This requires further research but could technically be implemented. Incorporating a cache to store the data will be researched. A common form of this is the ability of readers to Favourite articles they wish to access offline and these are cached.

6.3.3 Memory Leaks

Memory leaks are most likely present in this application. These are blocks of allocated memory that the program no longer references. Leaks waste space by filling up pages of memory with inaccessible data and waste time due to extra paging activity. Leaked memory eventually forces the system to allocate additional virtual memory pages for the application, the allocation of which
could have been avoided by reclaiming the leaked memory.\[24\] Testing and removing these leaks will allow for a more efficient and faster app.

### 6.3.4 iPad

Currently the app is only available to iPhone or iTouch devices. With tablet sales increasing, optimising the application for iPads would increase the publications accessibility for users. The code for optimising the project for iPads would be similar to the code currently in place. There are issues regarding the layout of the screen given the extra space that must be analysed. This would not consist of a huge body of work and could potentially be very beneficial.

### 6.3.5 Paywall

Many newspapers use paywalls to limit their web content to users. This may include allowing the user to access a limited number of articles free within a given time period. Two types of paywalls exist, hard and soft. Hard paywalls limit the content available to the user significantly leaving minimal to no content. Soft paywalls however allow users to view content without a subscription, but they are limited to a specific number within a given time period.

Given that The University Times is a free publication the implementation of a hard paywall is unlikely. A soft paywall could be implemented allowing users to access extraiality such as catalogued articles and the ability to save their favourite articles for offline reading as discussed earlier. Although this idea sounds simple in terms of generating revenue, further research needs
to be done to assess whether it would be the correct strategic move for the paper or not.

6.3.6 Source Code

The source code created has individual value in that it has the potential to be licensed to other publications using WordPress websites for publishing. Similar mobile applications could be easily constructed by creating a template based on the code with the opportunity for further customisation. The value of this code is evident based upon the research completed during this project displaying the marked trend toward mobile technology and the need for publications to have a presence on this ecosystem.
6.4 Chapter Summary

The previous chapters detailed the design and creation of a functional app in line with the client’s specifications. Chapter 6 involved the implementation of two additional features that were added outside of these specifications. These features added benefits to the client in the form of increased exposure through linking with social networks and an additional form of income through in-app advertising. The chapter concluded with an analysis of future work that may be implemented in future updates. The next chapter is an evaluation of the success of the project.
Chapter 7

Evaluation

The purpose of this chapter is to review the project as a whole. The successes of the project and difficulties encountered are evaluated.

7.1 Project Successes

The project can be considered a success as a result of completing all of the core objectives set out in October 2013, as well as successfully implementing additional features. The app was completed on time despite several setbacks encountered throughout the project. On March 12th 2014, the application was submitted to the app store for approval. Promotional images and screenshots were also uploaded to accompany the source code. Apps uploading to the Apple App Store undergo testing from Apple to ensure their quality and safety. As of 21st March 2014 the app was successfully approved and uploaded for sale to the App Store.

Overall the project was extremely rewarding from both a Computer Science
and business perspective. Each stage of the project resulted in valuable experience. Given the ubiquity of software in the world today, and in particular apps, the opportunity to implement a project from start to finish was extremely illuminating. In software development, programs are iterated upon to improve and update as time passes. The upload to the app store by no means marked the completion of the product, as mentioned in the previous chapter there is a body of future work that will be assessed for suitability for future updates.

The first update of the app was approved and was uploaded to the App Store on the 28th March 2014. This new version fixed a few bugs including the URL given to the JSON. 3.5-inch devices prior to the new update were not able to post to Facebook or Twitter given sizing issues, they are now able to do so. The About Us and Feedback section has been implemented for both 3.5 and 4-inch iOS devices. A new cleaner launch image was also uploaded with the old image taken due to a blurry graphic.

### 7.2 Difficulties

As with all projects difficulties arose, some more significant than others. The issues with the JSON feed were the most difficult and time consuming issues faced during the completion of this project. Without a coherent feed the app would not be consistent. Although the old feed could have been used the application itself wouldnt be of high quality nor would it have met the project objectives. This issue pushed back the timeline of the project increasing the pressure to complete the rest of the work within a constrained time period.
As this project was completed with no prior knowledge of Objective-C the initial construction of the systems architecture was extremely inefficient due to the presence of garbage code. This resulted in loading times that were not acceptable. This was redesigned and rebuilt creating more efficient version during the halted period of development as a result of the broken JSON feed, which solved these issues.

Many inconsistencies arose during the implementation process as a result of having to build the application for two screen sizes. The options to share articles to Facebook and Twitter were unavailable in the first version of the app uploaded to the App Store.
Chapter 8

Conclusion

This report detailed the implementation of an iOS application for The University Times. From design to publishing professional standards were applied to this project. The product was accepted to the Apple App Store on the 21st of March 2014. The project has yielded an application exceeding the core objectives that were set out in October 2013, however there are additional functionalities that have not been implemented due to time constraints. The suitability and feasibility of these functionalities will be explored before being added to future iterations of the software.

To the authors knowledge this is the first iOS application of its kind for a student publication in Ireland. In combination with a project creating an Android app for the publication, these extra distribution and interaction channels mark The University Times as an innovative publication. Given the increasing importance of mobile technology in the print media industry this project will be a valuable distribution platform for the publication and college. The publication of these apps can be promoted as an accessible means
for alumni to keep in contact with the college in a time where there is an increasing reliance on strengthening these links to bridge funding gaps.

This project was valuable not just from a Computer Science perspective but also resulted in an increased appreciation of the disruptive effects technology is having on many industries, and in particular print media. Organisations must adapt and compete on new platforms. It is now a core competency for print media publications to have an optimised presence on the mobile ecosystem. Not alone is a presence on these platforms imperative but customisation and optimisation can offer rewarding user experiences making them more likely to return, and thus increasing readership.

While the product uploaded to the App Store exceeded the clients specifications, there are several features that can be implemented in the future to improve the app. The App Store is designed to allow developers to iterate their product to create the best possible user experience based on feedback and improvements in technology.

Overall, the project was an extremely rewarding one in many respects. Many different skills were developed throughout. The combination of Computer Science and business knowledge gained in college to solve a business problem from start to finish was particularly satisfying. The completion of the project using many technologies and languages without previous knowledge was also an achievement. The build and test process that was taken throughout this project in retrospect was a suitable structure to take given the lack of experience with many of these technologies and languages.
Bibliography


[9] Figures received from ISServices on February 17th 2014.


