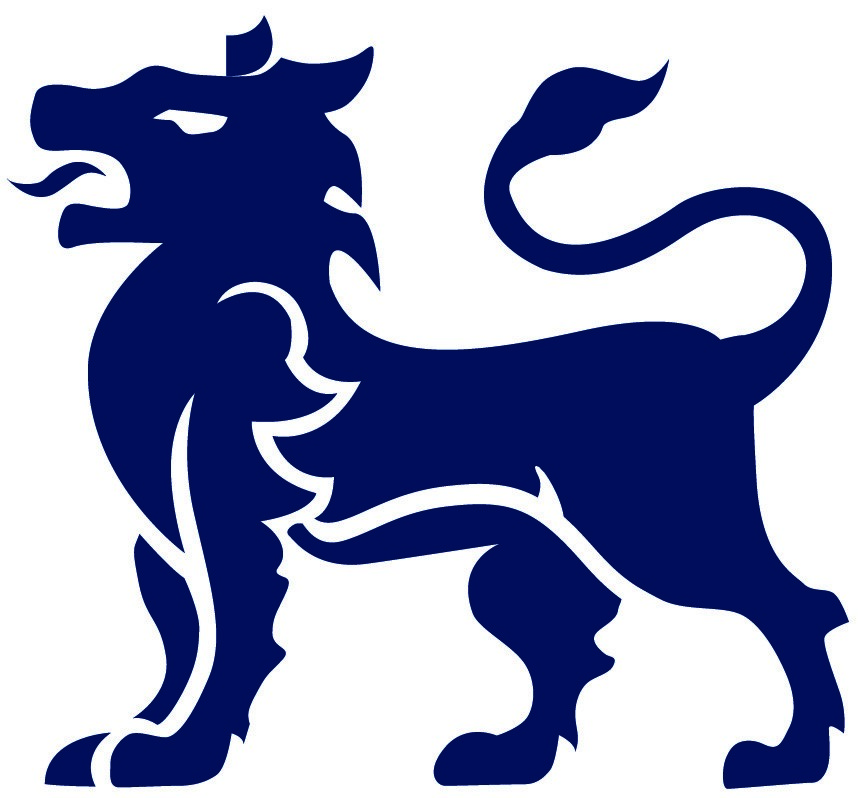
Student Mental Health Signposting App (My Mind)

Alexander Edward Davis



A report submitted as part of the requirements for the degree of BSc in Computer Science at the School of Computing and Digital Technology

Birmingham City University, Birmingham, UK

May 2017

Supervisor Dr. Yevgeniya Kovalchuk

Abstract

The Abstract of the report should be written here, it should provide a short summary of the work encompassing no more than one page.

# Acknowledgements

The Acknowledgements section may be used to thank your supervisor, family, research funding bodies, or any other applicable individuals or institutions.

# Declaration

****I confirm that the work contained in this BSc project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed: Date: 19th May 2017

Alexander Edward Davis

Table of Contents

[Abstract ii](#_Toc481799390)

[Acknowledgements iii](#_Toc481799391)

[Declaration iv](#_Toc481799392)

[Table of Contents v](#_Toc481799393)

[Introduction 1](#_Toc481799394)

[1.1 Background 1](#_Toc481799395)

[1.2 About this Thesis 1](#_Toc481799396)

[1.3 Chapter List 1](#_Toc481799397)

[1.4 Conclusion 2](#_Toc481799398)

[Using LATEX 3](#_Toc481799399)

[2.1 Structure of this Template 3](#_Toc481799400)

[2.2 Using Figures 3](#_Toc481799401)

[2.3 Referencing 5](#_Toc481799402)

[2.4 Citing Bibliographic References 5](#_Toc481799403)

[2.4.1 Compiling a BibTeX Database 6](#_Toc481799404)

[2.4.2 The Harvard Bibliography Style 6](#_Toc481799405)

[2.5 Inserting an Algorithm 6](#_Toc481799406)

[2.6 Table Creation 7](#_Toc481799407)

[2.7 Inserting Program Code Samples 7](#_Toc481799408)

[2.8 Working with Maths 8](#_Toc481799409)

[2.9 Using the Correct Quotes 9](#_Toc481799410)

[2.10 Start Compiling and Editing 9](#_Toc481799411)

[2.10.1 Required Software 10](#_Toc481799412)

[2.11 Conclusion 10](#_Toc481799413)

[Literature Review 11](#_Toc481799414)

[3.1 Introduction 11](#_Toc481799415)

[3.2 Key Topic Area 1 11](#_Toc481799416)

[3.3 Key Topic Area 2 11](#_Toc481799417)

[3.4 Key Topic Area 3 11](#_Toc481799418)

[3.5 Conclusions 11](#_Toc481799419)

[Background Research 11](#_Toc481799420)

[4.1 Conclusions 12](#_Toc481799421)

[Design 13](#_Toc481799422)

[5.1 Conclusions 13](#_Toc481799423)

[Implementation 14](#_Toc481799424)

[6.1 Conclusions 14](#_Toc481799425)

[Evaluation & Testing 15](#_Toc481799426)

[7.1 Conclusions 15](#_Toc481799427)

[Conclusion 16](#_Toc481799428)

[8.1 Conclusions 16](#_Toc481799429)

[8.2 Future Work 16](#_Toc481799430)

[Bibliography 17](#_Toc481799431)

[Appendix A - Project Specification 18](#_Toc481799432)

[A.1 Functional Requirements 18](#_Toc481799433)

[A.2 Non-Functional Requirements 18](#_Toc481799434)

[Appendix B - Project Management 19](#_Toc481799435)

[Appendix C – Presentation 20](#_Toc481799436)

[Appendix D – Source Code 21](#_Toc481799437)

[AppDelegate 21](#_Toc481799438)

[OnBoardingController 24](#_Toc481799439)

[Welcome Screens 26](#_Toc481799440)

[WelcomeScreenOne 26](#_Toc481799441)

[WelcomeScreenTwo 26](#_Toc481799442)

[WelcomeScreenThree 26](#_Toc481799443)

[Main Tab Views 27](#_Toc481799444)

[TabBarController 27](#_Toc481799445)

[MainViews 27](#_Toc481799446)

[Appendix E - Project Log 29](#_Toc481799447)

[Appendix F - UCEEL Copyright Waiver 30](#_Toc481799448)

**No table of figures entries found.**

**No table of figures entries found.**

Chapter 1

# Introduction

A short paragraph introducing the topic the chapter examines.

## 1.1 Background

A number of pages about the background of the project.

## 1.2 About this Thesis

This is the report of Alexander Edward Davis, submitted as part of the requirements for the degree of BSc in Computer Science at the School of Computing and Digital Technology, Birmingham City University, Birmingham, UK.

A number of paragraphs detailing the main expectations of this body of work.

## 1.3 Chapter List

Provide a list of all the chapters within the report and a brief summary of the content. Ensure each summary avoids having a repetitive structure such as starting with “This chapter deals with”.

Chapter 2 Using LATEX. This chapter deals with how to use the LATEX system.

Chapter 3 Literature Review. This chapter deals with *...*.

Chapter 4 Background Research. This chapter deals with *...*.

Chapter 5 Design. This chapter deals with *...*.

Chapter 6 Implementation. This chapter deals with *...*.

Chapter 7 Evaluation & Testing. This chapter deals with *...*.

Chapter 8 Conclusion. The conclusions of the report are presented.

## 1.4 Conclusion

A short conclusion summarising the chapter.

Chapter 2

# Using LATEX

By reading this chapter and comparing the generated output to the LATEX code, one should be able to produce all the necessary content expected of a typeset report. Instructions regarding software installation and compilation are to be found towards the end of the chapter. As with any source code, one should first compile, it to ensure it works. This document should compile with no errors if all the necessary packages are installed. A few warning messages relating to “Font style” (Arial) can be ignored.

## 2.1 Structure of this Template

The file thesis.tex in the root of the directory (ThesisTemplate) is the main file of this template. This is the file that must be compiled to create the document. The thesis.tex document contains a lot of configuration settings. The only elements that require editing are details such as the title of the report, authors name and so forth. The only further addition to the file is to use the *\include* statement to include additional chapters in the report. One may also comment out the *\include* statements using the percentage sign (%) to develop the report on a chapter by chapter basis. The BibTeX database thesis.bib is also included within the root. All the actual content of the report is divided up into directories each with a .tex file containing the chapter content (Appendix C.1).

## 2.2 Using Figures

One can insert graphic elements using LATEX in a number of ways. Vector based imagery such as diagrams saved to the pdf format may use the *\includegraphics* command with the optional *viewport* attribute to specify a precise area of the graphic to be included.

Figures should also include a Caption and a Label for referencing.

When inserting a Figure (Figure 2.1) one uses the *\begin{figure}* and *\end{figure}* commands. The image presented is a vector graphic in the form of a pdf file. When working with such files it is usually necessary to include the optional *viewport* attribute to designate the specific area in which to focus. The first pair of coordinates (x & y) designate the pixel location of the lower left corner. The second pair identify the upper right hand corner. Modification of these coordinates allows one to focus in upon a particular area of interest within the vector image. The optional attribute [H] when beginning a Figure inserts the graphic element at the specified location. Other options such as [htb] (here, top, bottom) will place the graphic in the most suitable place that LATEX can find. This however can have a negative impact on memory allocation if a large number of images exist.

Bottom Left

Top Right

Figure 2.1: Vector Graphic - pdf of a PowerPoint Slide

One may use the *minipage* command when inserting two figures to span across the page. This allows for the subdividing of the page into a number of columns of specified width. Figure 2.2 & 2.3 demonstrate how one may zoom in / focus on a particular section of a graphic by altering the coordinates of the *viewport*.

Bottom Left

Figure 2.2: viewport = 20 20 600 440 Figure 2.3: viewport = 220 200 400 330

The example below (Figure 2.4) features a bitmap image. One can see that the extension for the image file isn’t specified, as this template is setup to automatically search for .jpg, .png, .gif and .pdf images. The size of the displayed image within the document can be varied by adjusting the height and width attributes. To rotate an image 90 degrees an optional attribute can be added, for example [width=.4\linewidth, angle=90].

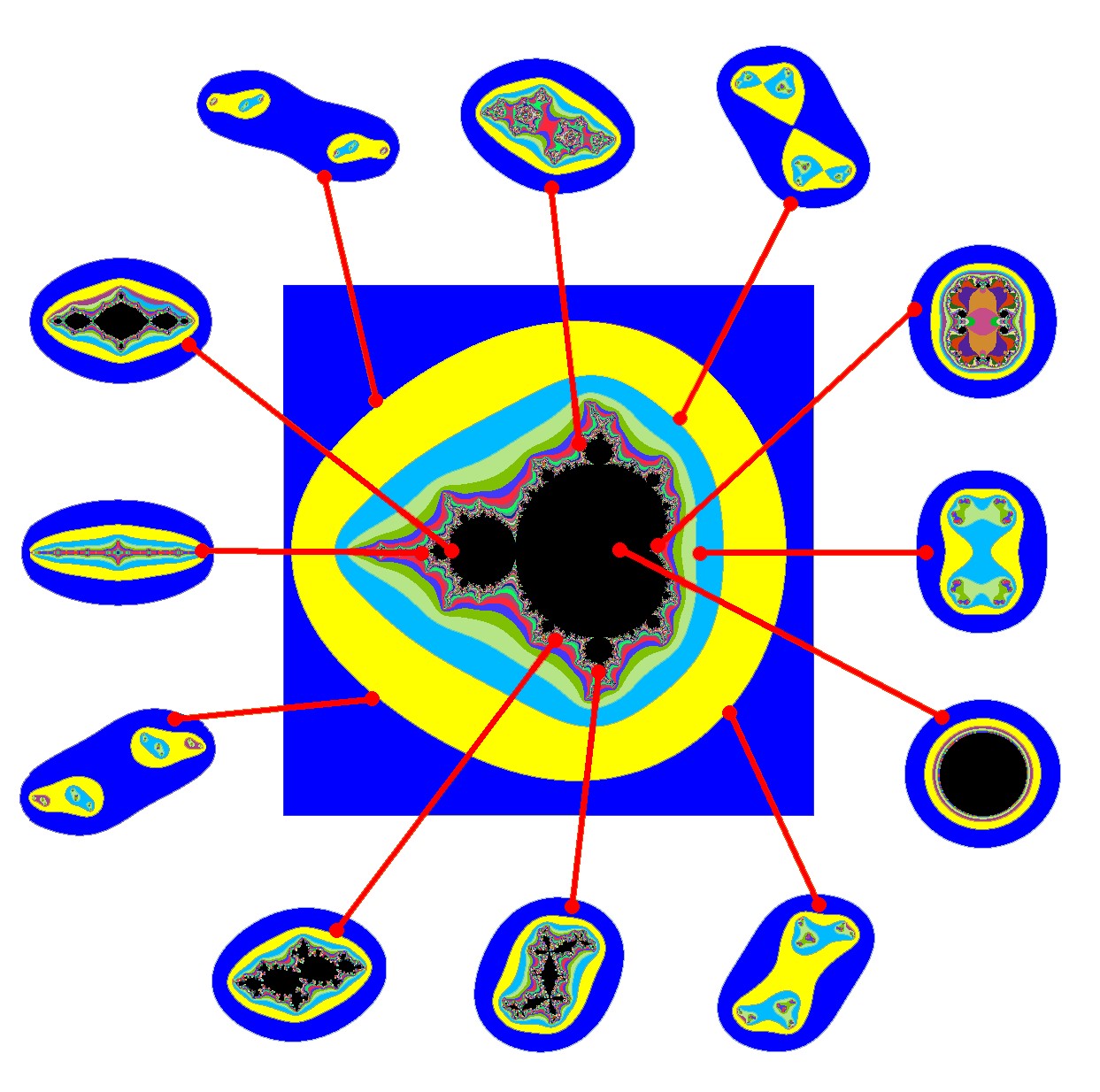


Figure 2.4: Caption for Bitmap Image Example

## 2.3 Referencing

To refer to another part of the document one must use a combination of the *\ref* and *\label* commands. The label is a unique identifier, therefore when working with large documents it helps to give references meaningful names. Examples of this includes prefixing Table references with *tab:*, figures with *fig:*, chapters with *ch:*. In very large documents in may also be useful to add an additional level of prefixing to represent the chapter the label is in. In this example chapter the tables and figures have the additional prefix of *using* to represent the *usingLatex* chapter. The tilde (∼) is used to ensure that a reference remains as a single object. All instances of *\ref* should be preceded with the tilde.

## 2.4 Citing Bibliographic References

Bibliographic references are stored in a database (.bib file extension), this contains a list of articles, proceedings, books, thesis and so forth. Each type of publication has a number of required fields such as a unique identifier, author and title. To cite a reference within the main body text one must use the *\cite* and *\citep* commands as in the following examples. Knuth (1973) for example is well known for his work on the Art of Computer Programming. The SETI@Home project (Berkeley University, 2016) is an example of a webpage citation. One can also work with articles (Russell, 1978), MSc Thesis (Shannon, 1940), PhD Thesis (Sutherland, 1963) or articles within conference proceedings (Ewald and Maas, 1978). Several other types of published work exist, but they are used to a lesser degree, see the Bath University Library Style guide (Ball, 2016).

Doolan (2016b) has provided a list of resources that may be of use to ones project in the form of links to searchable databases and LATEX tools. The previous citation was generated using the *\cite{bibEntryReference}* command.

### 2.4.1 Compiling a BibTeX Database

Having initially compiled the document using pdfLatex a number of helper files are created that aid in referencing and citations. One must the compile the bibtex database, followed by an additional two compiles using pdfLatex. Citing additional bibliographic references within the body of the document being produced will require the recompile of the bibtex database. In the case that the bibtex reference of a cited article cannot be found one will see a question mark (?) instead of the proper citation and a compile warning.

### 2.4.2 The Harvard Bibliography Style

The bibliographic references are laid out using the Harvard referencing style. This style is generally more applicable to humanities rather than the sciences. One would typically expect to see the use of a numbered reference style in the sciences - such as IEEE or Vancouver. In LATEX the Harvard style is depreciated. As such alternative solutions are to use the *natbib* or *biblatex* packages, these however yield references of an author, name format, but don’t quite conform to a Harvard style.

Fortunately a Harvard style has been developed at the University of Bath (Ball, 2016). It has been included in this template in the form of a file called *bath.bst*, dated 24 October 2016. The style file can be downloaded as a zip file from the University of Bath Library webpage dedicated to referencing and plagiarism. The zip file includes very useful package documentation, with numerous bibtex examples alongside the resultant output.

## 2.5 Inserting an Algorithm

The *algorithm2e* environment (Fiorio, 2016) may be used to generate algorithms (Algorithm 1). If no algorithms are used within the document then comment out *\listofalgorithms* in the file thesis.tex to remove the list of algorithms page.

Algorithm 1:

A Sample Algorithm

while

*(*

*RANK*

*<*

*COMPSIZE)*

do

if

*(*

*RANK == MASTER*

*)*

then

generate random value ;

for

*(*

*each item K*

*)*

do

get result ;

end

end

end

## 2.6 Table Creation

The data in Table 2.1 displays three columns of left-aligned and one column of justified data. The cell contents can be aligned to the left (l), right (r), center (c) or (p). Vertical bars may sometimes be seen in tables but these generally look unprofessional. The Booktabs package (Fear, 2016) allows for the creation of professional looking tables as shown in the example. The use of *\toprule*, *\midrule* and *\bottomrule* commands provided by the package allow for rules of varying thickness and spacing. Data elements (cells / columns) within a table are divided up using the ampersand (&). To complete a row one must end with a double backslash (\\). Tables as with figures need a caption and a label.

The WinShell editor has a GUI based utility to aid in the creation of the tabular data.

Table 2.1: Table Caption

|  |  |  |  |
| --- | --- | --- | --- |
| Heading 1 | Heading 2 | Heading 3 | Heading 4 |
| Cell A1 | Cell B2 | Cell C3 | Cell D4 |
| Cell E1 | Cell F2 | Cell G3 | Justified text with a defined column size of 5cm |

## 2.7 Inserting Program Code Samples

To insert small segments of program code (Listing 2.1) that detail how interesting algorithms and so forth are implemented use the *lstlisting* command. Inclusion of program code again requires a Caption and Label. Sample code from external files may also be included (Listing 2.2), by supplying the relative path to the source file.

|  |
| --- |
| if(rndVal==0){ if(opType > 2){  //do something  }  } |

1

2

3

4

5

Listing 2.1: Sample Program Code Listing

|  |
| --- |
| public void someInterestingMethod( int x ) {  try {  DataInputStream in = new DataInputStream(req.getInputStream());  HttpSession session = req.getSession(true);  //Read Input Data in.close();  } catch ( Exception err ) { } outputStream.close();  } |

1

2

3

4

5

6

7

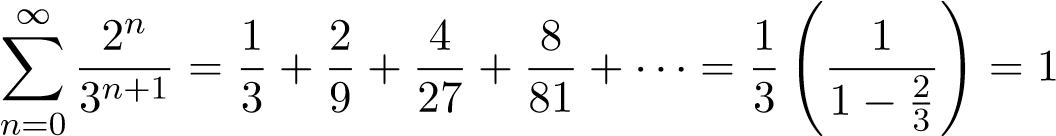
8

9

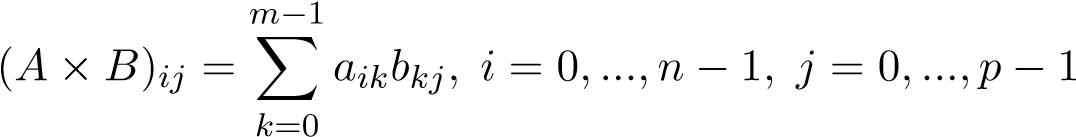
Listing 2.2: The Caption for the Code Listing

## 2.8 Working with Maths

The power of LATEX in typesetting mathematical formula is one of its key strengths. The mathematical definition of the “Cantor set” is a good example of this in action encompassed within an *\equation* environment.

 (2.1)

The previous equation demonstrates the use of sigma, fractions, large brackets, power, and dots. The function that defines the MSet *Zn*+1 = *Zn*2+*C* is a simpler example of math in use within the body text of a paragraph. Matrix Multiplication is typically regarded as an *O*(*n*3) operation. One may use the *equation* environment for more complex mathematical formula that should standout. For example the product *C* of two matrices *A* ∈ *Mn,m*(*R*) and *B* ∈ *Mm,p*(*R*) may be defined as

*.* (2.2)

The sizes of the matrices must satisfy (*n* × *m*)(*m* × *p*) = (*n* × *p*). Matrix multiplication is an associative process thereby *a* · (*b* · *c*) = (*a* · *b*) · *c*. Essentially to find the value of a particular cell *Ci,j* it is necessary to multiply row *i* of the matrix *A* with column *j* of matrix *B* summing all the multiplications.

## 2.9 Using the Correct Quotes

To surround a piece of text with double quotes one must place two single quotes on either side of the text. The double quote on the left is created using two left quotes (‘) this is located just above the *tab* key on the keyboard. The right hand double quote is created using two right hand quotes via (’) located just above and to the left of the right shift key. A properly formatted quotation should look like “This is a quotation”. Notice how the direction of the quotes are opposite to one another. A larger example using the *\Huge* font size option, is presented below so the quotes can be clearly seen.

“This is a quotation”

## 2.10 Start Compiling and Editing

1. Read the instructions in this file thesisTemplate.pdf, compare the code in the file usingLatex.tex to that of the resultant output of the Using LATEX chapter.
2. Download and install the necessary software - back-end LATEX system and front-end text editor.
3. Compile the files thesis.tex and thesis.bib to generate the file thesis.pdf so that it is identical to file thesisTemplate.pdf.
   1. To achieve this compile with pdfLaTeX, then run BibTeX and compile with pdfLaTeX two further times.
   2. It is only necessary to run BibTeX when new citations are added in the thesis document.
   3. Two compiles of pdfLaTeX is sufficient to allow for correct references to be established.
   4. If one is simply adding additional content (text / figures / tables) then a single compile of pdfLaTeX is sufficient for testing purposes.
4. Note that quite a number of additional files may now be seen in the root and subfolders of the (ThesisTemplate) directory.
5. With a successful compile achieved, start reading through the LATEX documents (.tex) to see how the various elements are assembled.
6. Begin editing the document starting with thesis.tex by entering elements such as Thesis Title and Author.
7. When it is necessary to start inserting figures / tables and so forth, copy and paste the LATEX code provided and edit as necessary.

### 2.10.1 Required Software

The implementation of LATEX typically used is MikTeX ([http://miktex.org)](http://miktex.org). It is typically best to install the complete MiKTeX system. A complete system comprises of around 40K files. A minimal install will necessitate the installation of packages as needed. One can individually download and integrate packages into a MikTeX system using the Package Manager (Admin) application. Initially a small installer application must be downloaded and executed. This in-turn downloads the most recent implementation of the MikTeX system. Run the installer again and select the directory of the downloaded package. The MacTeX distribution ([http://www.tug.org/mactex/)](http://www.tug.org/mactex/) is a useful alternative for Mac users.

The other necessary element is an editor. TeXnicCenter is a free download available at [(http://sourceforge.net/projects/texniccenter/)](http://sourceforge.net/projects/texniccenter/). An alternative is WinEdt a shareware ASCII editor ([http://www.winedt.com)](http://www.winedt.com). WinEdt can be freely used for a 30 day period, after which one will periodically receive reminders to register the product. Another option for Windows users is WinShell ([http://www.winshell.de)](http://www.winshell.de). An advantage of WinShell is its in-built BibTeX GUI editor. It also features a useful Table Wizard. For Mac users TexShop ([http://pages.uoregon.edu/koch/texshop/)](http://pages.uoregon.edu/koch/texshop/) is a popular option, providing a very easy to use editor. Note the website links in this section were created using the *\url* command and included for ease of reference. One should ideally include such references in the Bibliography.

## 2.11 Conclusion

Having read this chapter and compared the .tex source to the generated output, one should have all the knowledge necessary to typeset a well formatted and elegant project report. One needs to become familiar with a number of commands, but once mastered it should greatly speedup the report writing process. By using LATEX to typeset a document one is removed from a myriad of issues in relation to formatting allowing one to concentrate on the most important task, that of content creation.

Chapter 3

# Literature Review

This chapter provides a comprehensive review of the most relevant literature in the field showing how it feeds into the product being delivered. IEEExplore, Google Scholar and the ACM Digital Library are examples of useful data sources that may be queried as part of the review process. Doolan (2016b) provides links to these and other resources.

## 3.1 Introduction

## 3.2 Key Topic Area 1

## 3.3 Key Topic Area 2

## 3.4 Key Topic Area 3

## 3.5 Conclusions

The main conclusions of the chapter.

Chapter 4

# Background Research

Include excluded intervoews etc PICO process

This chapter provides some background research on the project and examines some previous work.

## 4.1 Conclusions

The main conclusions for this chapter.

Chapter 5

# Design

This chapter examines the design of the project.

## 5.1 Conclusions

The main conclusions for this chapter.

Chapter 6

# Implementation

This chapter examines the implementation of the project.

## 6.1 Conclusions

The main conclusions for this chapter.

Chapter 7

# Evaluation & Testing

This chapter evaluates the overall project and provides results of tests carried out.

## 7.1 Conclusions

The main conclusions for this chapter.

Chapter 8

# Conclusion

This chapter summarises the main positive outcomes and conclusions resulting from this body of work. One can explore the overall journey, problems encountered and solutions found. Of key importance is the “Future Work” section highlighting how the product may be further developed with new and improved features, futher resources and time.

## 8.1 Conclusions

The main conclusions that may be drawn from the body of work.

## 8.2 Future Work

Further development that could be carried out in the future.

# Bibliography

# Appendix A - Project Specification

Summary of the project outline.

## A.1 Functional Requirements

some text here

## A.2 Non-Functional Requirements

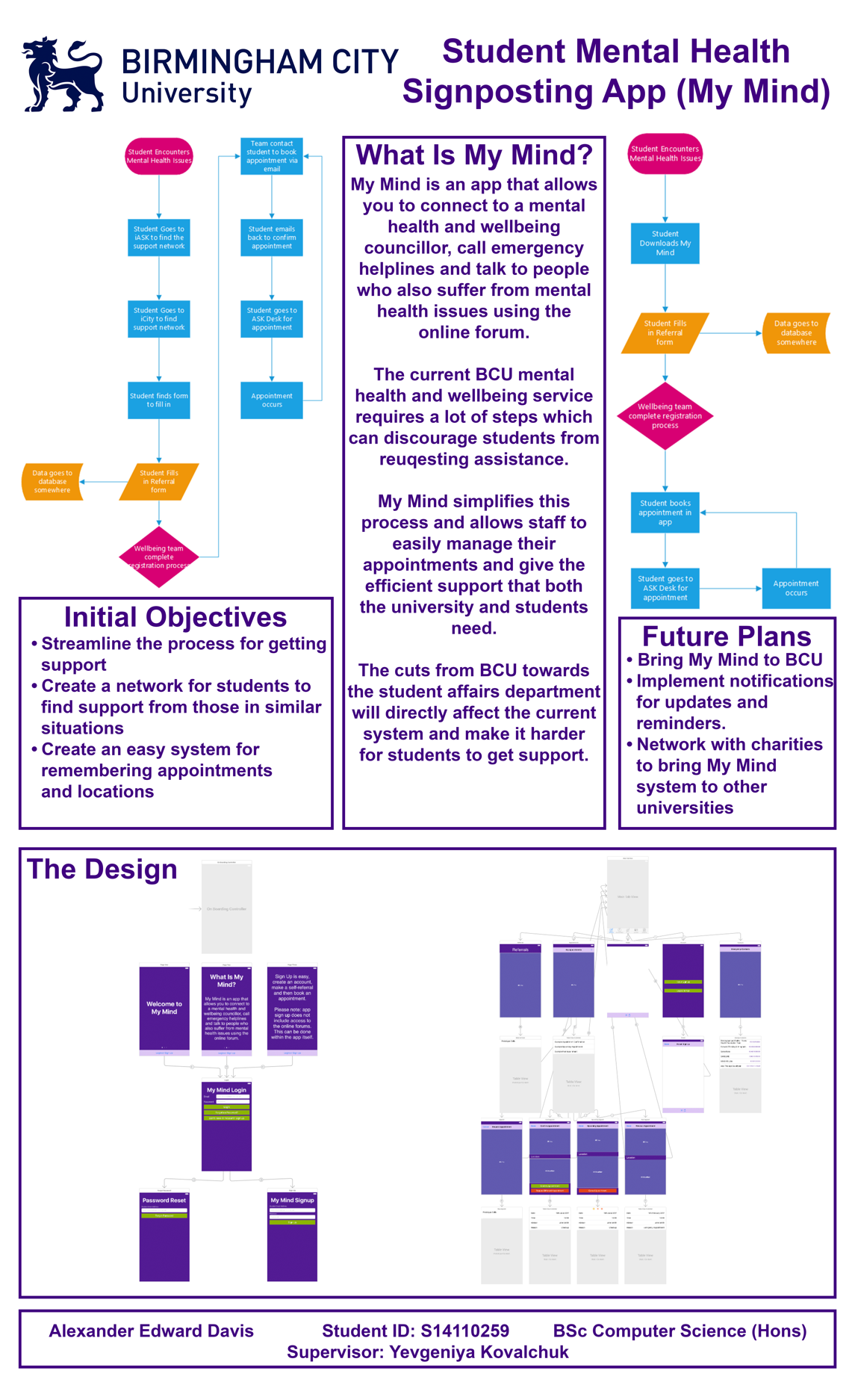
some text here

# Appendix B - Project Management

Discussion on how the project was managed. What things impacted the success of the project. How does the continually revised versions of the project plan compare to the initial draft developed at the start of the project. Did everything run according to schedule. What impact did elements such as exams & coursework have on the project.

Professor Randy Pausch is well known for his contribution to the Alice 3D environment as well as lectures on Time Management and Fulfilling your Childhood Dreams. Both of these lectures are well worth watching, as such Doolan (2015) outlines the main elements of both lectures, the post features a number of links allowing one to jump to particular segments of interest in the video lectures. Another post Doolan (2016a) highlights the key elements of success in the form of Goals, Confidence and Time Management. The post once again summarizes and provides links to the key elements of a video interview with Arnold Schwarzenegger.

# Appendix C – Presentation



# Appendix D – Source Code

## AppDelegate

//

//  AppDelegate.swift

//  My Mind

//

//  Created by Alexander Davis on 09/04/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import UIKit

import Firebase

import UserNotifications

import NotificationCenter

import FirebaseMessaging

import FirebaseDatabase

@UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    let gcmMessageIDKey = "gcm.message\_id"

    func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

        // Use Firebase library to configure APIs

        FIRApp.configure()

        FIRDatabase.database().persistenceEnabled = true

        //If User Is Signed In Direct to Main App Section

        let currentUser = FIRAuth.auth()?.currentUser

        let mainStoryboard = UIStoryboard(name: "Main", bundle: nil)

        if currentUser != nil

        {

            self.window?.rootViewController = mainStoryboard.instantiateViewController(withIdentifier: "MainController")

        }

        else

        {

            self.window?.rootViewController = mainStoryboard.instantiateViewController(withIdentifier: "OnBoard")

        }

        //Turn On Notifications for App

        if #available(iOS 10.0, \*) {

            // For iOS 10 display notification (sent via APNS)

            UNUserNotificationCenter.current().delegate = self

            let authOptions: UNAuthorizationOptions = [.alert, .badge, .sound]

            UNUserNotificationCenter.current().requestAuthorization(

                options: authOptions,

                completionHandler: {\_, \_ in })

            // For iOS 10 data message (sent via FCM)

            FIRMessaging.messaging().remoteMessageDelegate = self as? FIRMessagingDelegate

        } else {

            let settings: UIUserNotificationSettings =

                UIUserNotificationSettings(types: [.alert, .badge, .sound], categories: nil)

            application.registerUserNotificationSettings(settings)

        }

        application.registerForRemoteNotifications()

        return true

    }

    func applicationWillResignActive(\_ application: UIApplication) {

        // Sent when the application is about to move from active to inactive state. This can occur for certain types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.

        // Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.

    }

    func applicationDidEnterBackground(\_ application: UIApplication) {

        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.

        // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

    }

    func applicationWillEnterForeground(\_ application: UIApplication) {

        // Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.

    }

    func applicationDidBecomeActive(\_ application: UIApplication) {

        // Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

    }

    func applicationWillTerminate(\_ application: UIApplication) {

        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.

    }

    func application(\_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable: Any]) {

        // If you are receiving a notification message while your app is in the background,

        // this callback will not be fired till the user taps on the notification launching the application.

        // TODO: Handle data of notification

        // Print message ID.

        if let messageID = userInfo[gcmMessageIDKey] {

            print("Message ID: \(messageID)")

        }

        // Print full message.

        print(userInfo)

    }

    func application(\_ application: UIApplication, didReceiveRemoteNotification userInfo: [AnyHashable: Any],

                     fetchCompletionHandler completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {

        // If you are receiving a notification message while your app is in the background,

        // this callback will not be fired till the user taps on the notification launching the application.

        // TODO: Handle data of notification

        // Print message ID.

        if let messageID = userInfo[gcmMessageIDKey] {

            print("Message ID: \(messageID)")

        }

        // Print full message.

        print(userInfo)

        completionHandler(UIBackgroundFetchResult.newData)

    }

}

// [START ios\_10\_message\_handling]

@available(iOS 10, \*)

extension AppDelegate : UNUserNotificationCenterDelegate {

    // Receive displayed notifications for iOS 10 devices.

    func userNotificationCenter(\_ center: UNUserNotificationCenter,

                                willPresent notification: UNNotification,

                                withCompletionHandler completionHandler: @escaping (UNNotificationPresentationOptions) -> Void) {

        let userInfo = notification.request.content.userInfo

        // Print message ID.

        if let messageID = userInfo[gcmMessageIDKey] {

            print("Message ID: \(messageID)")

        }

        // Print full message.

        print(userInfo)

        // Change this to your preferred presentation option

        completionHandler([])

    }

    func userNotificationCenter(\_ center: UNUserNotificationCenter,

                                didReceive response: UNNotificationResponse,

                                withCompletionHandler completionHandler: @escaping () -> Void) {

        let userInfo = response.notification.request.content.userInfo

        // Print message ID.

        if let messageID = userInfo[gcmMessageIDKey] {

            print("Message ID: \(messageID)")

        }

        // Print full message.

        print(userInfo)

        completionHandler()

    }

}

// [END ios\_10\_message\_handling]

// [START ios\_10\_data\_message\_handling]

extension AppDelegate : FIRMessagingDelegate {

    // Receive data message on iOS 10 devices while app is in the foreground.

    func applicationReceivedRemoteMessage(\_ remoteMessage: FIRMessagingRemoteMessage) {

        print(remoteMessage.appData)

    }

}

// [END ios\_10\_data\_message\_handling]

## OnBoardingController

//

//  OnBoardingController.swift

//  My Mind

//

//  Created by Alexander Davis on 30/03/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

class OnBoardingController : UIPageViewController {

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    override func viewDidLoad() {

        // Set the dataSource and delegate in code.

        // I can't figure out how to do this in the Storyboard!

        dataSource = self

        delegate = self

        // This is the starting point.  Start with step zero.

        setViewControllers([getPageOne()], direction: .forward, animated: false, completion: nil)

    }

    func getPageOne() -> PageOne {

        return storyboard!.instantiateViewController(withIdentifier: "WelcomeScreenOne") as! PageOne

    }

    func getPageTwo() -> PageTwo {

        return storyboard!.instantiateViewController(withIdentifier: "WelcomeScreenTwo") as! PageTwo

    }

    func getPageThree() -> PageThree {

        return storyboard!.instantiateViewController(withIdentifier: "WelcomeScreenThree") as! PageThree

    }

}

// MARK: - UIPageViewControllerDataSource methods

extension OnBoardingController : UIPageViewControllerDataSource {

    func pageViewController(\_ pageViewController: UIPageViewController, viewControllerBefore viewController: UIViewController) -> UIViewController? {

        if viewController.isKind(of: PageTwo.self) {

            // 2 -> 1

            return getPageTwo()

        } else if viewController.isKind(of: PageTwo.self) {

            // 1 -> 0

            return getPageOne()

        } else {

            // 0 -> end of the road

            return nil

        }

    }

    func pageViewController(\_ pageViewController: UIPageViewController, viewControllerAfter viewController: UIViewController) -> UIViewController? {

        if viewController.isKind(of: PageOne.self) {

            // 0 -> 1

            return getPageTwo()

        } else if viewController.isKind(of: PageTwo.self) {

            // 1 -> 2

            return getPageThree()

        } else {

            // 2 -> end of the road

            return nil

        }

    }

    // This only gets called once, when setViewControllers is called

    func presentationIndex(for pageViewController: UIPageViewController) -> Int {

        return 0

    }

}

// MARK: - UIPageViewControllerDelegate methods

extension OnBoardingController : UIPageViewControllerDelegate {

}

## Welcome Screens

### WelcomeScreenOne

//

//  WelcomeScreenOne.swift

//  My Mind

//

//  Created by Alexander Davis on 30/03/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

class PageOne : UIViewController {

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

### WelcomeScreenTwo

//

//  WelcomeScreenTwo.swift

//  My Mind

//

//  Created by Alexander Davis on 30/03/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

class PageTwo : UIViewController {

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

### WelcomeScreenThree

//

//  WelcomeScreenThree.swift

//  My Mind

//

//  Created by Alexander Davis on 30/03/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

class PageThree : UIViewController {

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

## Login Section and User Information

### UserViews

//

//  LoginView.swift

//  My Mind

//

//  Created by Alexander Davis on 31/03/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

import Firebase

import FirebaseAuth

class LoginView: UIViewController {

    @IBOutlet weak var Emailtbx: UITextField!

    @IBOutlet weak var Passwordtbx: UITextField!

    @IBAction func Loginbtn(\_ sender: UIButton) {

        if self.Emailtbx.text == "" || self.Passwordtbx.text == "" {

            //Alert to tell the user that there was an error because they didn't fill anything in the textfields because they didn't fill anything in

            let alertController = UIAlertController(title: "Error", message: "Please enter an email and password.", preferredStyle: .alert)

            let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

            alertController.addAction(defaultAction)

            self.present(alertController, animated: true, completion: nil)

        } else {

            FIRAuth.auth()?.signIn(withEmail: self.Emailtbx.text!, password: self.Passwordtbx.text!) { (user, error) in

                if error == nil {

                    //Print into the console if successfully logged in

                    print("You have successfully logged in")

                    //Go to the MainController if the login is sucessful

                    let alertController = UIAlertController(title: "Welcome", message: "You have successfully logged in", preferredStyle: .alert)

                    let vc = self.storyboard?.instantiateViewController(withIdentifier: "MainController")

                    let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in self.present(vc!, animated: true, completion: nil);})

                    alertController.addAction(defaultAction)

                    self.present(alertController, animated: true, completion: nil)

                } else {

                    //Tells the user that there is an error and then gets firebase to tell them the error

                    let alertController = UIAlertController(title: "Error", message: error?.localizedDescription, preferredStyle: .alert)

                    let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

                    alertController.addAction(defaultAction)

                    self.present(alertController, animated: true, completion: nil)

                }

            }

        }

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

 }

class SignUp : UIViewController {

    @IBOutlet weak var Emailtbx: UITextField!

    @IBOutlet weak var Passwordtbx: UITextField!

    @IBAction func createAccountbtn(\_ sender: UIButton) {

        if Emailtbx.text == "" {

            let alertController = UIAlertController(title: "Error", message: "Please enter your email and password", preferredStyle: .alert)

            let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

            alertController.addAction(defaultAction)

            present(alertController, animated: true, completion: nil)

        } else {

            FIRAuth.auth()?.createUser(withEmail: Emailtbx.text!, password: Passwordtbx.text!) { (user, error) in

                if error == nil {

                    let alertController = UIAlertController(title: "Sign Up Complete", message: "You have Successfully Signed Up", preferredStyle: .alert)

                    let vc = self.storyboard?.instantiateViewController(withIdentifier: "Login")

                    let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in self.present(vc!, animated: true, completion: nil);})

                    alertController.addAction(defaultAction)

                    self.present(alertController, animated: true, completion: nil)

                } else {

                    let alertController = UIAlertController(title: "Error", message: error?.localizedDescription, preferredStyle: .alert)

                    let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

                    alertController.addAction(defaultAction)

                    self.present(alertController, animated: true, completion: nil)

                }

            }

        }

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

class ForgotPass : UIViewController {

    @IBOutlet weak var Emailtbx: UITextField!

    @IBAction func Resetbtn(\_ sender: UIButton) {

        if self.Emailtbx.text == "" {

            let alertController = UIAlertController(title: "Oops!", message: "Please enter an email.", preferredStyle: .alert)

            let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: nil)

            alertController.addAction(defaultAction)

            present(alertController, animated: true, completion: nil)

        } else {

            FIRAuth.auth()?.sendPasswordReset(withEmail: self.Emailtbx.text!, completion: { (error) in

                var title = ""

                var message = ""

                if error != nil {

                    title = "Error!"

                    message = (error?.localizedDescription)!

                } else {

                    title = "Success!"

                    message = "Password reset email sent."

                    self.Emailtbx.text = ""

                }

                let alertController = UIAlertController(title: title, message: message, preferredStyle: .alert)

                let vc = self.storyboard?.instantiateViewController(withIdentifier: "Login")

                let defaultAction = UIAlertAction(title: "OK", style: .cancel, handler: { action in self.present(vc!, animated: true, completion: nil);})

                alertController.addAction(defaultAction)

                self.present(alertController, animated: true, completion: nil)

            })

        }

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

class Logout : UIViewController {

    @IBAction func Logoutbtn(\_ sender: UIButton) {

        if FIRAuth.auth()?.currentUser != nil {

            do {

                try FIRAuth.auth()?.signOut()

                let vc = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "OnBoard")

                present(vc, animated: true, completion: nil)

            } catch let error as NSError {

                print(error.localizedDescription)

            }

        }

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

## Main Tab Views

### TabBarController

//

//  TabBarController.swift

//  My Mind

//

//  Created by Alexander Davis on 05/04/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

class MainTabView: UITabBarController{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

### MainViews

//

//  MainViews.swift

//  My Mind

//

//  Created by Alexander Davis on 09/04/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

import Firebase

import FirebaseDatabase

class Referrals: UIViewController{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

class ReferralConf: UIViewController{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

class Appoint: UIViewController{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

class Contact: UIViewController{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

}

class EmergeContacts: UITableViewController{

    @IBAction func TrustCallBtn(\_ sender: UIButton) {

        if let url = URL(string: "telprompt://01213010000")

        {

            UIApplication.shared.open(url, options: [:], completionHandler: nil)

        }

    }

    @IBAction func FwdThinkingbtn(\_ sender: UIButton) {

        if let url = URL(string: "telprompt://03003000099")

        {

            UIApplication.shared.open(url, options: [:], completionHandler: nil)

        }

    }

    @IBAction func Samaritansbtn(\_ sender: UIButton) {

        if let url = URL(string: "telprompt://08457909090")

        {

            UIApplication.shared.open(url, options: [:], completionHandler: nil)

        }

    }

    @IBAction func Sanebtn(\_ sender: UIButton) {

        if let url = URL(string: "telprompt://08457678000")

        {

            UIApplication.shared.open(url, options: [:], completionHandler: nil)

        }

    }

    @IBAction func Mindbtn(\_ sender: UIButton) {

        if let url = URL(string: "telprompt://0300123393")

        {

            UIApplication.shared.open(url, options: [:], completionHandler: nil)

        }

    }

    @IBAction func Alexsbtn(\_ sender: UIButton) {

        if let url = URL(string: "telprompt://07871778000")

        {

            UIApplication.shared.open(url, options: [:], completionHandler: nil)

        }

    }

}

### FormViews

//

//  FormViews.swift

//  My Mind

//

//  Created by Alexander Davis on 11/04/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

import SwiftForms

import MessageUI

import MapKit

class ReferralForm: FormViewController, MFMailComposeViewControllerDelegate{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    struct Static {

        static let nameTag = "name"

        static let passwordTag = "password"

        static let lastNameTag = "lastName"

        static let addressTag = "address"

        static let IDTag = "idnumber"

        static let GenderTag = "gender"

        static let personalEmailTag = "personalemail"

        static let uniEmailTag = "universityemail"

        static let CourseTag = "course"

        static let phoneTag = "phone"

        static let Voicemail = "voicemail"

        static let DSATag = "disabledstudent"

        static let FacultyTag = "faculty"

        static let birthday = "birthday"

        static let PastSupportTag = "pastsupport"

        static let CurrentSupportTag = "currentsupport"

        static let subjectTag = "subject"

        static let timescaleTag = "timescale"

        static let button = "button"

    }

    required init(coder aDecoder: NSCoder) {

        super.init(coder: aDecoder)

        self.loadForm()

    }

    override func viewDidLoad() {

        super.viewDidLoad()

        //self.navigationItem.rightBarButtonItem = UIBarButtonItem(title: "Submit", style: .plain, target: self, action: #selector(ReferralForm.submit(\_:)))

    }

    // MARK: Actions

    func submit() {

        let data = self.form.formValues().description

        let alertController = UIAlertController(title: "Form Completed", message: "This form has sucessfully been submitted", preferredStyle: .alert)

        let vc = self.storyboard?.instantiateViewController(withIdentifier: "MainController")

        let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in SendRef(input: data);self.tabBarController!.tabBar.items?[0].isEnabled = false;self.present(vc!, animated: true, completion: nil);})

        alertController.addAction(defaultAction)

        self.present(alertController, animated: true, completion: nil)

    }

    // MARK: Private interface

    fileprivate func loadForm() {

        let form = FormDescriptor(title: "Referral Form")

        let section1 = FormSectionDescriptor(headerTitle: nil, footerTitle: nil)

        var row = FormRowDescriptor(tag: Static.button, type: .button, title: "Send Form")

        row.configuration.button.didSelectClosure = { \_ in

            self.view.endEditing(true)

            self.submit()

        }

        section1.rows.append(row)

        let section2 = FormSectionDescriptor(headerTitle: "About Yourself", footerTitle: nil)

        row = FormRowDescriptor(tag: Static.nameTag, type: .name, title: "First Name")

        row.configuration.cell.appearance = ["textField.placeholder" : "First Name" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.lastNameTag, type: .name, title: "Last Name")

        row.configuration.cell.appearance = ["textField.placeholder" : "Last name" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.IDTag, type: .number, title: "Student ID")

        row.configuration.cell.appearance = ["textField.placeholder" : "12345678" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.birthday, type: .date, title: "Date of Birth")

        row.configuration.cell.showsInputToolbar = true

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.uniEmailTag, type: .email, title: "University Email")

        row.configuration.cell.appearance = ["textField.placeholder" : "john.smith@mail.bcu.ac.uk" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.personalEmailTag, type: .email, title: "Alternative Email")

        row.configuration.cell.appearance = ["textField.placeholder" : "john.smith@outlook.com" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.phoneTag, type: .phone, title: "Phone")

        row.configuration.cell.appearance = ["textField.placeholder" : "Mobile Number" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.Voicemail, type: .booleanSwitch, title: "Can we leave a voicemail message?")

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.GenderTag, type: .picker, title: "Gender")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["F", "M", "O", "U"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "F":

                return "Female"

            case "M":

                return "Male"

            case "O":

                return "Other"

            case "U":

                return "I'd rather not to say"

            default:

                return ""

            }

        }

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.addressTag, type: .multilineText, title: "Address")

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.DSATag, type: .booleanSwitch, title: "Are you a disabled student (eligible for DSA)?")

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.CourseTag, type: .text, title: "Course")

        row.configuration.cell.appearance = ["textField.placeholder" : "BSc Computer Science (Hons)" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.FacultyTag, type: .picker, title: "Faculty")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["CEBE", "HELS", "ADM", "BLSS"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "CEBE":

                return "Computing, Engineering and The Built Environment"

            case "HELS":

                return "Health, Education and Life Sciences"

            case "ADM":

                return "Arts, Design and Media"

            case "BLSS":

                return "Business, Law and Social Sciences"

            default:

                return ""

            }

        }

        section2.rows.append(row)

        let section3 = FormSectionDescriptor(headerTitle: "What you would like to talk about.", footerTitle: nil)

        row = FormRowDescriptor(tag: Static.subjectTag, type: .multilineText, title: "")

        section3.rows.append(row)

        row = FormRowDescriptor(tag: Static.timescaleTag, type: .picker, title: "How long have you been concerned?")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["NL", "ST", "QL", "VL"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "NL":

                return "Not Long (within 1 month)"

            case "ST":

                return "Some Time (1-3 months)"

            case "QL":

                return "Quite Some Time (3-6 months)"

            case "VL":

                return "A Long Time (6+ months)"

            default:

                return ""

            }

        }

        section3.rows.append(row)

        row = FormRowDescriptor(tag: Static.CurrentSupportTag, type: .multipleSelector, title: "Are you currently receiving support?")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["No", "CBT", "CPN", "CP", "GP", "MHWA", "Psyi", "Psyo", "SW", "O"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "No":

                return "No"

            case "CBT":

                return "CBT Practitioner (e.g. Healthy Minds)"

            case "CPN":

                return "Community Psychiatric Nurse (CPN)"

            case "CP":

                return "Counsellor/Psychotherapist"

            case "GP":

                return "GP"

            case "MHWA":

                return "Mental Health/Wellbeing Adviser"

            case "Psyi":

                return "Psychiatrist"

            case "Psyo":

                return "Psychologist"

            case "SW":

                return "Social Worker"

            case "O":

                return "Other"

            default:

                return ""

            }

        }

        section3.rows.append(row)

        row = FormRowDescriptor(tag: Static.PastSupportTag, type: .multipleSelector, title: "Have you received support in the past?")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["No", "CBT", "CPN", "CP", "GP", "MHWA", "Psyi", "Psyo", "SW", "O"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "No":

                return "No"

            case "CBT":

                return "CBT Practitioner (e.g. Healthy Minds)"

            case "CPN":

                return "Community Psychiatric Nurse (CPN)"

            case "CP":

                return "Counsellor/Psychotherapist"

            case "GP":

                return "GP"

            case "MHWA":

                return "Mental Health/Wellbeing Adviser"

            case "Psyi":

                return "Psychiatrist"

            case "Psyo":

                return "Psychologist"

            case "SW":

                return "Social Worker"

            case "O":

                return "Other"

            default:

                return ""

            }

        }

        section3.rows.append(row)

        form.sections = [section1, section2, section3]

        self.form = form

    }

}

class ReqAppoint: FormViewController, MFMailComposeViewControllerDelegate{

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    struct Static {

        static let date = "date"

        static let time = "time"

        static let location = "location"

        static let phoneTag = "phone"

        static let adivsor = "advisor"

        static let reason = "reason"

        static let button = "button"

    }

    required init(coder aDecoder: NSCoder) {

        super.init(coder: aDecoder)

        self.loadForm()

    }

    override func viewDidLoad() {

        super.viewDidLoad()

        //self.navigationItem.rightBarButtonItem = UIBarButtonItem(title: "Submit", style: .plain, target: self, action: #selector(ReferralForm.submit(\_:)))

    }

    // MARK: Actions

    func submit() {

        let data = self.form.formValues().description

        let alertController = UIAlertController(title: "Request Sent", message: "Your Request Has Been Sent", preferredStyle: .alert)

        let vc = self.storyboard?.instantiateViewController(withIdentifier: "MainController")

        let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in SendRef(input: data);self.tabBarController!.tabBar.items?[0].isEnabled = false;self.present(vc!, animated: true, completion: nil);})

        alertController.addAction(defaultAction)

        self.present(alertController, animated: true, completion: nil)

    }

    // MARK: Private interface

    fileprivate func loadForm() {

        let form = FormDescriptor(title: "Referral Form")

        let section1 = FormSectionDescriptor(headerTitle: nil, footerTitle: nil)

        var row = FormRowDescriptor(tag: Static.button, type: .button, title: "Send Form")

        row.configuration.button.didSelectClosure = { \_ in

            self.view.endEditing(true)

            self.submit()

        }

        section1.rows.append(row)

        let section2 = FormSectionDescriptor(headerTitle: nil, footerTitle: nil)

        row = FormRowDescriptor(tag: Static.date, type: .date, title: "Date Requested")

        row.configuration.cell.showsInputToolbar = true

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.time, type: .time, title: "Time Requested")

        row.configuration.cell.showsInputToolbar = true

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.location, type: .picker, title: "Location")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["Centre", "North", "South", "Cons", "Margaret", "Jew","Bour"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "Centre":

                return "City Centre (Curzon Building)"

            case "North":

                return "City North (Baker Building)"

            case "South":

                return "City South (Seacole Building)"

            case "Cons":

                return "The Birmingham Conservatoire"

            case "Margaret":

                return "Margaret Street"

            case "Jew":

                return "Victoria Street"

            case "Bour":

                return "Bournville Campus"

            default:

                return ""

            }

        }

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.phoneTag, type: .phone, title: "Phone")

        row.configuration.cell.appearance = ["textField.placeholder" : "Mobile Number" as AnyObject, "textField.textAlignment" : NSTextAlignment.right.rawValue as AnyObject]

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.adivsor, type: .picker, title: "Preferred Advisor")

        row.configuration.cell.showsInputToolbar = true

        row.configuration.selection.options = (["JS", "AD", "TM", "U"] as [String]) as [AnyObject]

        row.configuration.selection.optionTitleClosure = { value in

            guard let option = value as? String else { return "" }

            switch option {

            case "JS":

                return "John Smith"

            case "AD":

                return "Alexander Davis"

            case "TM":

                return "Tim Minchin"

            case "U":

                return "I Don't Mind"

            default:

                return ""

            }

        }

        section2.rows.append(row)

        row = FormRowDescriptor(tag: Static.reason, type: .multilineText, title: "Reason for Appointment")

        section2.rows.append(row)

        form.sections = [section1, section2]

        self.form = form

    }

}

class ConfAppoint: UIViewController{

    @IBOutlet var ConfirmMap: MKMapView!

    override func viewDidLoad() {

        let initalLocation = CLLocation(latitude: 52.483358, longitude: -1.883024)

        centerMapOnLocation(location: initalLocation)

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    let regionRadius: CLLocationDistance = 1000

    func centerMapOnLocation(location: CLLocation) {

        let coordinateRegion = MKCoordinateRegionMakeWithDistance(location.coordinate,

                                                                  regionRadius \* 2.0, regionRadius \* 2.0)

        ConfirmMap.setRegion(coordinateRegion, animated: true)

    }

    @IBAction func Confirm(\_ sender: UIButton) {

        let alertController = UIAlertController(title: "Appointment Confirmed", message: "Your appointment has been confirmed.", preferredStyle: .alert)

        let vc = self.storyboard?.instantiateViewController(withIdentifier: "MainController")

        let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in self.present(vc!, animated: true, completion: nil);})

        alertController.addAction(defaultAction)

        self.present(alertController, animated: true, completion: nil)

    }

    @IBAction func RequestAptbtn(\_ sender: UIButton) {

        let alertController = UIAlertController(title: "New Request Made", message: "A new appointment request has been made for you.", preferredStyle: .alert)

        let vc = self.storyboard?.instantiateViewController(withIdentifier: "MainController")

        let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in self.present(vc!, animated: true, completion: nil);})

        alertController.addAction(defaultAction)

        self.present(alertController, animated: true, completion: nil)

    }

}

class UpcomingAppoint: UIViewController{

    @IBOutlet var UpcomingMap: MKMapView!

    override func viewDidLoad() {

        let initalLocation = CLLocation(latitude: 52.483358, longitude: -1.883024)

        centerMapOnLocation(location: initalLocation)

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    let regionRadius: CLLocationDistance = 1000

    func centerMapOnLocation(location: CLLocation) {

        let coordinateRegion = MKCoordinateRegionMakeWithDistance(location.coordinate,

                                                                  regionRadius \* 2.0, regionRadius \* 2.0)

        UpcomingMap.setRegion(coordinateRegion, animated: true)

    }

    @IBAction func Cancelbtn(\_ sender: UIButton) {

        let alertController = UIAlertController(title: "Appointment Cancelled", message: "Your appointment has been cancelled.", preferredStyle: .alert)

        let vc = self.storyboard?.instantiateViewController(withIdentifier: "MainController")

        let defaultAction = UIAlertAction(title: "OK", style: .default, handler: { action in self.present(vc!, animated: true, completion: nil);})

        alertController.addAction(defaultAction)

        self.present(alertController, animated: true, completion: nil)

    }

}

class PastAppoint: UIViewController{

    override func viewDidLoad() {

        let initalLocation = CLLocation(latitude: 52.483358, longitude: -1.883024)

        centerMapOnLocation(location: initalLocation)

    }

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    @IBOutlet var PastMap: MKMapView!

    let regionRadius: CLLocationDistance = 1000

    func centerMapOnLocation(location: CLLocation) {

        let coordinateRegion = MKCoordinateRegionMakeWithDistance(location.coordinate,

                                                                  regionRadius \* 2.0, regionRadius \* 2.0)

        PastMap.setRegion(coordinateRegion, animated: true)

    }

}

### ForumView

//

//  ForumView.swift

//  My Mind

//

//  Created by Alexander Davis on 31/03/2017.

//  Copyright © 2017 Alexander Davis. All rights reserved.

//

import Foundation

import UIKit

class ForumViewController: UIViewController {

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    @IBOutlet var webView: UIWebView!

    override func viewDidLoad() {

        super.viewDidLoad()

        // Do any additional setup after loading the view, typically from a nib.

        let url = URL (string: "https://www.mymindforum.com");

        let requestObj = URLRequest(url: url!);

        webView.loadRequest(requestObj);

    }

    override func didReceiveMemoryWarning() {

        super.didReceiveMemoryWarning()

        // Dispose of any resources that can be recreated.

    }

}

class ForumSignUpViewController: UIViewController {

    override var preferredStatusBarStyle: UIStatusBarStyle {

        return .lightContent

    }

    @IBOutlet weak var ForumSignUp: UIWebView!

    override func viewDidLoad() {

        super.viewDidLoad()

        // Do any additional setup after loading the view, typically from a nib.

        let url = URL (string: "https://www.mymindforum.com/ucp.php?mode=register");

        let requestObj = URLRequest(url: url!);

        ForumSignUp.loadRequest(requestObj);

    }

    override func didReceiveMemoryWarning() {

        super.didReceiveMemoryWarning()

        // Dispose of any resources that can be recreated.

    }

}

# Appendix E - Project Log

The following is a weekly summary of the work carried during the development of this body of work. It covers tasks that were completed, tutorials that were worked through, articles that were read and summaries of discussions held with the project supervisor and other third parties including other students and staff within Birmingham City University.

Week Beginning: Monday 26/09/2016

First week working on the project. Had a meeting with supervisor and discussed some of the issues related to the project. The first deliverable is due for the end of next week (project outline & ethics form).

* Downloaded and Installed LATEX (MikTeX full install) & Winshell.
* Started to get to grips with the LATEX system by making simple modifications to the template and editing the project log.
* Developed a Mind Map to clarify understanding of project elements.
* Prepared an initial draft of project plan in the form of a Gantt chart.
* Prepared and revised draft of project proposal & filled in ethics form.
* Downloaded and read half a dozen BSc Reports to see the general format and expected content.
* Started learning how to use some API’s needed for the project.

# Temp new logoAppendix F - UCEEL Copyright Waiver

**Student Name: \_Alexander Edward Davis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Project/Thesis Title \_Student Mental Health Signposting App (My Mind) \_\_\_\_\_**

**Course: \_BSc Computer Science with Honours\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student Agreement**

1. I confirm that Birmingham City University can electronically archive and make accessible the project / thesis described above via the UCEEL Electronic Library system. I retain all other ownership rights to the copyright of the document / project work described above.

2. I confirm the above project / thesis is a true and unaltered representation of the project / thesis as submitted to Birmingham City University course tutors and examiners.

3. I confirm that the above project / thesis material copied from a source (e.g. a book) where ownership of the copyright does not belong to myself.

If the **project / thesis includes** such materialplease supply the following details:

Please see references section within report.

**I have not** obtained and attached a written permission statement from the owner(s) of each third party copyrighted matter included in my project / thesis

**Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Print Name: \_ Alexander Edward Davis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date: \_19th May 2017­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**N.B. If you are at any time in consultation with a publisher regarding this work you will need to declare the copy held on UCEEL. Some publishers may regard the UCEEL copy as constituting prior publication. The copy can be removed from UCEEL if it becomes an obstacle to future commercial publication.**

**Official Use only**

ORION unique number: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Date added to the system: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**