UWE Wellbeing & Drug Abuse Application

Tomas Jaggard

19016621

UXCFXK-30-3

Digital Systems Project



# Abstract

Bristol has an increasingly high drug problem, being one of the capitals for drug abuse in Europe in 2018. With substance abuse increasing year-on-year, access to support and resources to aid the most vulnerable is crucial.

Covered in this report, the proposed application will aim to help UWE residents in Bristol, providing free and accessible wellbeing support while also aiming to provide tools and resources for those affected by drug-abuse, helping to reduce casualties. Prior to creating this application, research was conducted into drug abuse, mental wellbeing and how both impact education, which only furthered the necessity for a piece of software as mentioned.

# Table of Contents

[Abstract 1](#_Toc109301035)

[Table of Contents 2](#_Toc109301036)

[Table of Figures 5](#_Toc109301037)

[Table of Tables 6](#_Toc109301038)

[1 Introduction 7](#_Toc109301039)

[1.1 Chapter Introduction 7](#_Toc109301040)

[1.2 Scope 7](#_Toc109301041)

[1.3 Real world application 7](#_Toc109301042)

[1.4 Aims and objectives 7](#_Toc109301043)

[1.5 Research Questions 8](#_Toc109301044)

[1.6 Report Outline 8](#_Toc109301045)

[2 Literature Review 8](#_Toc109301046)

[2.1 Chapter Introduction 8](#_Toc109301047)

[2.2 Bristol’s devastating history with drug abuse 9](#_Toc109301048)

[2.3 How psychological factors affect education and productivity 9](#_Toc109301049)

[2.4 The importance of educational resources to tackle drug abuse 9](#_Toc109301050)

[2.5 The future of mobile technology in education 10](#_Toc109301051)

[2.6 API Systems 10](#_Toc109301052)

[2.7 Conclusion 10](#_Toc109301053)

[3 Requirements 10](#_Toc109301054)

[3.1 Chapter Introduction 10](#_Toc109301055)

[3.2 Methodology 10](#_Toc109301056)

[3.3 MoSCoW 11](#_Toc109301057)

[3.4 Functional Requirements 11](#_Toc109301058)

[Requirement ID 11](#_Toc109301059)

[Description 11](#_Toc109301060)

[Priority 11](#_Toc109301061)

[3.5 Non-Functional Requirements 11](#_Toc109301062)

[4 Methodology 12](#_Toc109301063)

[4.1 Agile Methodology 12](#_Toc109301064)

[4.2 Gannt Chart 13](#_Toc109301065)

[5 Design 13](#_Toc109301066)

[5.1 Tool Overview 13](#_Toc109301067)

[5.2 Use Case Diagram 14](#_Toc109301068)

[5.4 Flow Diagrams 16](#_Toc109301069)

[5.6 Wireframes 17](#_Toc109301070)

[5.7 Testing Strategy 18](#_Toc109301071)

[6 Implementation 19](#_Toc109301072)

[6.1 Sprint 1 19](#_Toc109301073)

[6.2 Introduction 19](#_Toc109301074)

[6.3 Development 19](#_Toc109301075)

[6.3.1 Creating the project workspace 19](#_Toc109301076)

[6.3.2 Creating the TabView view controller 20](#_Toc109301077)

[6.3.3 Creating HelpView 21](#_Toc109301078)

[6.3.4 Creating MapView 22](#_Toc109301079)

[6.3.5 Accessibility Features 22](#_Toc109301080)

[6.3.5 Installing CocoaPods dependency manager and Google Maps Frameworks 23](#_Toc109301081)

[6.4 Testing 23](#_Toc109301082)

[6.5 Sprint 2 24](#_Toc109301083)

[6.6 Introduction 24](#_Toc109301084)

[6.7 Development 25](#_Toc109301085)

[6.7.1 Updating & Fixing accessibility features 25](#_Toc109301086)

[6.7.2 Implementing auto-generated views for Contacts&Resources View 25](#_Toc109301087)

[6.7.3 Implementing external links & call prompts 28](#_Toc109301088)

[6.8 Testing 29](#_Toc109301089)

[6.9 Sprint 3 30](#_Toc109301090)

[6.10 Introduction 30](#_Toc109301091)

[6.11 Development 30](#_Toc109301092)

[6.11.1 Setting up Google Maps API Key 30](#_Toc109301093)

[Creating the Map 31](#_Toc109301094)

[6.11.1 Creating ViewControllerBridge to enable Swift Viewcontroller functionality 32](#_Toc109301095)

[6.11.2 Implementing location tracking 32](#_Toc109301096)

[6.12 Testing 34](#_Toc109301097)

[6.13 Sprint 4 34](#_Toc109301098)

[6.14 Introduction 35](#_Toc109301099)

[6.15 Development 35](#_Toc109301100)

[6.15.1 Setting up the heatmap overlay 35](#_Toc109301101)

[6.15.2 Making Crime API call 35](#_Toc109301102)

[6.15.3 Constructing the heatmap array data 38](#_Toc109301103)

[6.15.4 Generating the heatmap 38](#_Toc109301104)

[6.16 Testing 39](#_Toc109301105)

[6.17 Sprint 5 39](#_Toc109301106)

[6.18 Introduction 39](#_Toc109301107)

[6.19 Development 39](#_Toc109301108)

[6.19.1 Creating the Drug combination tool UI 39](#_Toc109301109)

[6.19.2 Creating the Drug combination tool functionality 41](#_Toc109301110)

[6.19.4 Adding visual assets to the application 42](#_Toc109301111)

[Testing 44](#_Toc109301112)

[6.20 Acceptance Testing 44](#_Toc109301113)

[6.21 Requirements Review 47](#_Toc109301114)

[6.21.1 Functional Requirements 47](#_Toc109301115)

[6.21.2 Non Functional Requirements 47](#_Toc109301116)

[7 Project Evaluation 47](#_Toc109301117)

[7.1 Research 47](#_Toc109301118)

[7.2 Requirements 48](#_Toc109301119)

[7.3 Methodology 48](#_Toc109301120)

[7.4 Design 48](#_Toc109301121)

[7.5 Implementation 48](#_Toc109301122)

[7.6 Testing 48](#_Toc109301123)

[7.7 Future Improvements 49](#_Toc109301124)

[8 Conclusion 49](#_Toc109301125)

[Table of Abbreviations 50](#_Toc109301126)

[Bibliography 51](#_Toc109301127)

# Table of Figures

[Figure 1: Gantt Chart 13](#_Toc109301128)

[Figure 2: Use Case diagram 15](#_Toc109301129)

[Figure 3: Sequence diagram describing the Help page interactions 15](#_Toc109301130)

[Figure 4: Sequence diagram describing the Map page interactions 16](#_Toc109301131)

[Figure 5: User Flow diagram describing navigation and interacting within the application. 16](#_Toc109301132)

[Figure 6: Administrative Flow diagram describing the modification process of view data 17](#_Toc109301133)

[Figure 7: Wireframes for example views 18](#_Toc109301134)

[Figure 8: Project Setup 20](#_Toc109301135)

[Figure 9: ContentView(Main) View 20](#_Toc109301136)

[Figure 10: App Header 21](#_Toc109301137)

[Figure 11: Navigation Links 21](#_Toc109301138)

[Figure 12: MapView View 22](#_Toc109301139)

[Figure 13: Light/Dark mode button 22](#_Toc109301140)

[Figure 14: Lightbulb toggle button (light mode) 22](#_Toc109301141)

[Figure 15: Lightbulb toggle button (dark mode) 23](#_Toc109301142)

[Figure 16: Podfile 23](#_Toc109301143)

[Figure 17: HelpView view implementing new theme controller 25](#_Toc109301144)

[Figure 18: Header text controlled by new theme controller functionality 25](#_Toc109301145)

[Figure 19: Contact struct 26](#_Toc109301146)

[Figure 20: Contact details inside array 26](#_Toc109301147)

[Figure 21: ContactRow View 26](#_Toc109301148)

[Figure 22: Contacts UI 27](#_Toc109301149)

[Figure 23: ContactsView View 27](#_Toc109301150)

[Figure 24: ContactsDetailView View 28](#_Toc109301151)

[Figure 25: Inequality operators 28](#_Toc109301152)

[Figure 26: Implementation of Links 29](#_Toc109301153)

[Figure 27: API Key Setup 31](#_Toc109301154)

[Figure 28: Partially censored API key 31](#_Toc109301155)

[Figure 29: AppDelegate class 31](#_Toc109301156)

[Figure 30: loadView function 32](#_Toc109301157)

[Figure 31: MapViewControllerBridge struct 32](#_Toc109301158)

[Figure 32: viewDidLoad function 33](#_Toc109301159)

[Figure 33: locationManager function 33](#_Toc109301160)

[Figure 34: locationManager Error Handler 33](#_Toc109301161)

[Figure 35: LocationManager authorization switch statement 34](#_Toc109301162)

[Figure 36: Info.Plist Permission Description 34](#_Toc109301163)

[Figure 37: setUpHeatMap function 35](#_Toc109301164)

[Figure 38: heatmap-generated marker 35](#_Toc109301165)

[Figure 39: Search radius border (Visual representation) 36](#_Toc109301166)

[Figure 40: Crime struct 37](#_Toc109301167)

[Figure 41: Assignment of variables in getCrimes function 38](#_Toc109301168)

[Figure 42: Retrieving JSON data from URL 38](#_Toc109301169)

[Figure 43: Drug combination tool UI 40](#_Toc109301170)

[Figure 44: returnCombo function 41](#_Toc109301171)

[Figure 45: Adding returnCombo function call button 42](#_Toc109301172)

[Figure 46: Application logo 42](#_Toc109301173)

[Figure 47: Photoshop asset layers 43](#_Toc109301174)

[Figure 48: UI View with assets added 43](#_Toc109301175)

# Table of Tables

[Table 1: Functional Requirements list 11](#_Toc109301176)

[Table 2: Non-Functional Requirements list 12](#_Toc109301177)

[Table 3: Example Test table 19](#_Toc109301178)

[Table 4: Example UAT table 19](#_Toc109301179)

[Table 5: Sprint 1 tests 24](#_Toc109301180)

[Table 6: Sprint 2 tests 30](#_Toc109301181)

[Table 7: Sprint 3 tests 34](#_Toc109301182)

[Table 8: Sprint 4 tests 39](#_Toc109301183)

[Table 9: Sprint 5 tests 44](#_Toc109301184)

[Table 10: User Acceptance Testing 47](#_Toc109301185)

[Table 11: Functional Requirements Review 47](#_Toc109301186)

[Table 12: Non-functional Requirements Review 47](#_Toc109301187)

# 1 Introduction

## 1.1 Chapter Introduction

This chapter describes the project overview, discussing the scope of the project and its real-world problem that will be tackled in this report.

## 1.2 Scope

The project will be an application developed to aid UWE university residents, providing wellbeing support, tools and resources to help in navigating Bristol.

The application will offer an accessible means of discovering resources such as contacting the university’s 24-hour support team, campus security for emergencies and services where you can book appointments to speak to a specialist in your selected issue. There will be resources for daily life in Bristol and on campus, with useful tools and links to provide additional support. Another resource included will be a tool capable of advising users on the dangers of drug combinations to help prevent abuse. Furthermore, there will be the inclusion of a heatmap to identify key areas of drug crime in Bristol, providing a visual representation of areas of high risk which could be dangerous to vulnerable residents.

The application will be branded as a UWE resource, with a UI consistent with applications on the App Store which follow the Apple Human Interface Guidelines.

## 1.3 Real world application

Bristol is renowned by many as a hotspot for drug abuse, with the city being described as “the epicentre of ketamine use in the UK” (Crane, 2018), affecting a large portion of students and their wellbeing. From researching, many of those suffering from addiction and mental health issues lack the access to resources to support and assist in their recovery, leading to a further decline in their health. The creation of mobile applications which aide in providing on-the-go support has increased as technology has advanced, with higher demand from consumers for quick and accessible health services. The application developed aims to identify key areas of issue in Bristol and provide wellbeing support to residents who may be suffering from addiction and mental health issues.

## 1.4 Aims and objectives

1. Research drug abuse and its impact on mental health in Bristol.
2. Identify key resources already available to residents and build upon them in an environment that is easily accessible and straightforward to access.
3. Identify the most appropriate method of accessing the aforementioned resources
4. Provide software to provide on-the-go wellbeing support for UWE residents
5. Utilise UK crime APIs to develop a visual tool for identifying at-risk areas for drug-related issues
6. Create a report covering the research, design, development, testing and implementation of the software.
7. Critically evaluate the development of the software
8. Discuss future development of the software and its functionalities

## 1.5 Research Questions

Research questions were developed by evaluating what knowledge would be required to develop the application

Questions:

1. How does drug crime in Bristol impact UWE students?
2. Why is wellbeing support important to individuals?
3. How does providing wellbeing support affect the ability to produce work?
4. How can support be provided to individuals in vulnerable positions?
5. How can support be effectively provided among a range of users with varying capabilities?
6. What systems can be implemented to monitor and display at-risk areas of drug abuse?
7. Does drug education work?
8. How does interactive learning aide drug-use prevention?

## 1.6 Report Outline

**Chapter 2** provides a critical literature review of research related to the software topic, Bristol and its historical relation with drug abuse and research in software and systems applied to the applications development.

**Chapter 3** discusses the requirements of the application, including both functional and non-functional requirements.

**Chapter 4** describes the methodology and Gannt chart used to manage time effectively whilst developing different functionalities inside the application.

**Chapter 5** describes the design process for the application, highlighting key tools used, multiple diagrams to display the flow of the application and wireframes and composites to exhibit how the application UI will be designed to be user-friendly.

**Chapter 6** provides an overview of the creation and implementation of the application, with detailed description of core components and functionalities.

**Chapter 7** provides a reflection of the project outcome, describing any hardships faced throughout the previous chapters.

**Chapter 8** contains the summary of the project, discussing what has been achieved, and what could be improved upon in future iterations.

# 2 Literature Review

## 2.1 Chapter Introduction

This chapter describes the problem addressed through this project and provides research into Bristol and its association and history with drug abuse. It offers insight into the systems provided by the project and relevant examples. Additionally, research into the necessity of educational resources to tackle drug abuse and addiction have been covered, discussing how the application can benefit those affected. Furthermore, it includes exploration of the importance of APIs in computer science and how they are often implemented in computer systems.

### 2.2 Bristol’s devastating history with drug abuse

The South-West of England has, for the longest time had a strong history with drug abuse, with approximately 11.7% of people in the region having reported using illegal drugs in the last year in comparison to the average of 9.4% in all other regions in England and Wales (Torrance, J et al. 2018). A 2018 study (European Monitoring Centre for Drugs and Drug addiction, 2018) of over 70 European cities indicated that “Bristol consumes more cocaine per capita than any other city” which participated in the sample. When considering that illegal drug abuse is often underreported (Torrance, J et al. 2018), to be the number one consumer of cocaine in Europe is incredibly alarming.

In 2017, the UK Office for National Statistics released figures stating that Bristol had an increase of 43% in drug-related deaths from 2014-2016 in comparison to 2011-2013 (Office of National Statistics, 2022). Over the next 4 years, new data was released that highlighted a further increase since then of 38% (Office of National Statistics, 2018-2020). This shocking increase undoubtedly continued to get worse year-by-year as the deputy mayor for communities (Asher Craig) admitted that the city did not have a strategy to tackle drug crime in Bristol however acknowledged the necessity for one (Cameron, A. 2019).

The application developed will offer immediate, hand-picked resources available to all who need them to aid in reducing drug deaths in the Bristol area. The app will provide options for crisis response and safeguarding for UWE residents, as well as wellbeing support for individuals impacted by drug abuse who may need support. These resources allow vulnerable individuals to find support quickly and easily through the accessible contacts page. It will also provide insight for both average users and potentially individuals in positions of power to visualise areas of issue. This visualisation of data makes it abundantly clear where resources need to be allocated in order to reduce the overall consequences of drug abuse.

### 2.3 How psychological factors affect education and productivity

Wellbeing and mental health play a large factor in one’s ability to learn, with it being noted by Cornaglia et al.(2015) that mental health problems reduce the amount of schooling individuals partake in and their overall productivity, “which may have lifelong consequences for employment, income and other outcomes”(Cornaglia, 2015). Historically mental health and wellbeing has been associated with impacting willingness to learn, with reduced numbers of adolescents pursuing further education as a result (OECD, 2013)(Bradshaw and Richardson, 2009). To help reduce mental strain and improve wellbeing in UWE residents, the application aims to provide services, resources, and points of contact to speak about mental wellbeing which has been shown to widely benefit schooling (Santor, D. 2009).

### 2.4 The importance of educational resources to tackle drug abuse

Studies conducted have shown that effective education programmes often have combined elements of both focused and generic components. It was found that programmes that include interactive elements showed greater effects on learning that non-interactive programmes (Black and Wiliam, 1998). This highlights the benefits of educating individuals through personalised interactive services and how providing material relevant to student’s personal situations can have a large positive impact.

Another study into the effectiveness of Drug Abuse Resistance Education (DARE), which involves providing a drug intervention curriculum to educate students, further discusses how interactive programs that employed participatory teaching strategies had higher effect sizes than traditional methods of education as previously discussed (Clayton, R. 1996). The UK government also acknowledges the benefit of well-funded services which have been said to “cut crime, improve health, and can support individuals and families on the road to recovery” (Public Health England, 2018).

The application developed in the project will provide tools and resources to encourage learning such as a substance interaction checker that allows the user to select from multiple different substances and combine them to see their interaction with the human body, providing a personalised experience catered to them.

### 2.5 The future of mobile technology in education

Development of applications for mobile devices in education has become increasingly popular, with the demand for accessible learning constantly increasing as a result of the Covid-19 pandemic (Wood, J. 2022). Online education has been shown to improve material retention by 25-60% when compared to in-person education due to reduced time-to-learn than traditional learning (Li and Lalani, 2020), and many higher education organizations are using mobile learning to provide flexibility and accessibility in learning (Ally and Prieto-Bláquez. 2014). Considering the aforementioned details, the mobile application developed will act as an accessible form of resources and tools to a wide range of users with varying capabilities, whilst providing an array of features to educate the user on-the-go.

### 2.6 API Systems

APIs or Application Programming Interfaces are software intermediaries that enable communication between two applications. They are needed to connect applications and are designed to perform a specific function built around sharing data and executing processes. A publication by Verborgh et al.(2013) describes the frequency of RESTful architectural styles in practice that often make use of standard non-specific content types such as text/xml or application/JSON. The application being developed will make use of such styles by retrieving JSON data directly from the GET call and formatting the data necessary to complete the designated task.

### 2.7 Conclusion

After conducting the above research, it has only further confirmed the necessity for an application as specified in the project scope. The creation of this application aims to reduce critical drug abuse and provide support for UWE residents who may be struggling with their mental wellbeing.

# 3 Requirements

## 3.1 Chapter Introduction

This chapter describes the project overview, discussing the scope of the project and its real-world problem that will be tackled in this report.

## 3.2 Methodology

The methodology chosen for this application was agile as the research, planning and development of requirements/deliverables were identified in detail prior to the design and development process, meaning the targets for each stage have already been outlined. The clear structure and short-term development deadlines provided by a agile methodology helped to keep to time constraints. This method also increased the ability to innovate, iterating on previously developed solutions to provide the optimum solution.

## 3.3 MoSCoW

When developing the functional and non-functional requirements (FR and NFR respectively), the MoSCoW approach was utilised. Originating from the “Dynamic Software Development Method” (Waters, 2009), MoSCoW uses the following categories to manage requirements:

**M** – Must have: Necessary deliverables that are mandatory for completion.

**S** – Should have: Deliverables that are not vital, but should be implemented due to their added value.

**C** – Could have: Deliverables that would be beneficial to add, but would lose minimal value if not implemented

**W** – Will not have: Deliverables which are not a priority for the development process

## 3.4 Functional Requirements

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | Priority |
| FR1 | Access crime data from an external API | M |
| FR2 | Display crime data visually via a custom heatmap | M |
| FR3 | Provide the user with contact information for wellbeing services | M |
| FR4 | Allow the user to initiate a phone call from inside the application | M |
| FR5 | Provide informational resources to assist the user | M |
| FR6 | Forward the user to appropriate web forms to create appointments | S |
| FR7 | Provide tools to support vulnerable individuals and prevent abuse | S |
| FR8 | Include accessibility options | S |
| FR9 | Implement GPS systems for the crime map | C |
| FR10 | Implement notifications for scheduled appointments | C |
| FR11 | Provide out-of-app tracking to notify the user when entering crime hotspots | W |

Table : Functional Requirements list

## 3.5 Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| Requirement ID | Description | Priority |
| NFR1 | Usability – The application is user-friendly and functions are clearly identifiable. Navigation is kept simple and it has recognisable UWE branding | M |
| NFR2 | Security – User confidentiality is maintained, to ensure the user feels safe using the application. | M |
| NFR3 | Regulatory – The application complies to GDPR standards | M |
| NFR4 | Compatibility – The application can be used across a range of devices. | M |
| NFR5 | Reliability – The downtime for the application is kept to the minimum. | M |
| NFR6 | Performance – The application performs smoothly without issue, with an input response of under 1ms. | S |
| NFR7 | Accessibility- The application is inclusive and can be used by a wide range of users with varying capabilities | S |
| NFR8 | Scalability – The application can be built upon to allow for further implementations of features. | C |
| NFR9 | Maintainability – The application can be managed and fixed easily. | C |
| NFR10 | Capacity – The storage needed to download the application is kept to the minimum. | W |

Table : Non-Functional Requirements list

# 4 Methodology

The phases of development for the application were very clear, with multiple different writing and design stages, prior to the creation of the application. In order to provide clarity and conciseness surrounding the tasks to be completed in these stages, I have adopted the methodology described below.

## 4.1 Agile Methodology

For this project the chosen methodology was Agile, decidedly so due to its benefit of encouraging consistent reports and visibility over completed tasks, which was of the utmost importance to as the starting point of this project was considerably later than preferred. It also helps to reduce risk, as the product can be analysed per iteration, ensuring functionalities work as intended before continuing onwards. Another benefit found in Agile development is the flexibility in development, allowing for continued improvements to be made where necessary without having to start from phase one.

## 4.2 Gannt Chart

A Gannt chart has been provided which was created to assist in the development cycle of this project and offers a visual representation of the project timeline. Time management using the Agile methodology allows for weightings to be calculated, giving an estimation of how much time will be needed to be spent on each task.

Chart, timeline

Description automatically generated

Figure : Gantt Chart

# 5 Design

## 5.1 Tool Overview

The listed resources below were used in the design and development of the application:

**Astah UML**

All use case and sequence diagrams developed for this project were created using Astahs UML diagram generator.

**CocoaPods**

A Ruby Gem software package, Cocoapods was used as the dependency manager for the application. This provides a simple method of installing and managing external libraries in Swift.

**Diagrams.net**

Draw.io was used to design flow chart diagrams and provides a simple drag & drop environment for planning application flows.

**Google Maps API&SDK**

Google Maps API was used to generate the map & heatmap overlay for identifying drug crime hotspots in Bristol. This was due to the wide range of functionalities offered in the libraries included, as well as providing a recognisable UI for usability and aesthetic purposes.

**Mockflow**

Wireframes and composites created during the design phase were generated using Mockflows innovative drag-and-drop system, allowing for detailed and visually appealing designs.

**Photoshop**

Photoshop was used to design and edit images inside of the application using a variety of tools offered.

**TeamGantt**

TeamGantt was used to develop Gantt charts for accurate planning of the agile development cycle, providing simplistic and accessible charts when working from university and home.

**Xcode**

Xcode was the environment used to develop the application. This decision was made due to iOS being the platform the application was developed for, and Xcode being the most popular development environment for developing applications on iOS due to its direct integration with Apple and its development libraries.

## 

## 5.2 Use Case Diagram

Diagram

Description automatically generated

Figure : Use Case diagram

5.3 Sequence Diagrams

Diagram, engineering drawing

Description automatically generated

Figure : Sequence diagram describing the Help page interactions

Diagram

Description automatically generated

Figure : Sequence diagram describing the Map page interactions

## 5.4 Flow Diagrams

Diagram

Description automatically generated

Figure : User Flow diagram describing navigation and interacting within the application.

Diagram

Description automatically generated

Figure : Administrative Flow diagram describing the modification process of view data

## 5.6 Wireframes

The first row of Figure 7 displays the Help & Info tab views wireframes, representing the basis of the UI presentation of the application subsection. The HelpView() will act as the hub for information for students, and allow access to contacts and resources to aid them whilst at UWE. All information & future information can be stored in arrays, which are designed to automatically generate new views as seen in ContactTab1(). The second row of Figure 7 displays the Map View tab view wireframe, representing the ideal UI presentation of the Google Maps API & Heatmap. The MapView() will call upon the police data API to collate the locations of drug incidents in the past month to present an accurate depiction of areas of high risk. This is automatically updated upon loading the application and will be controlled using the users mobile phones internal calendar.

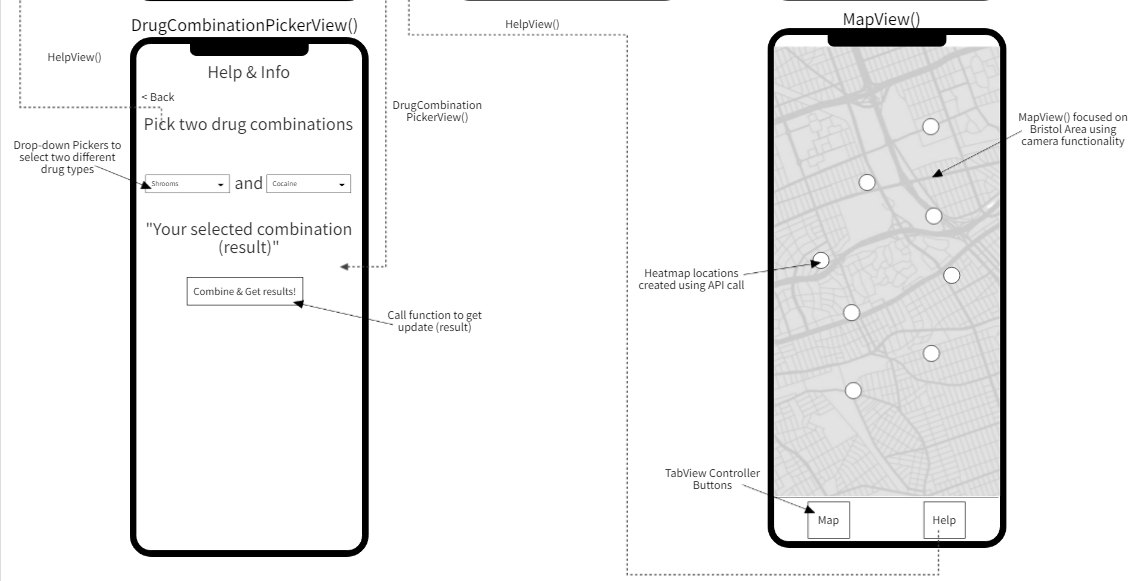
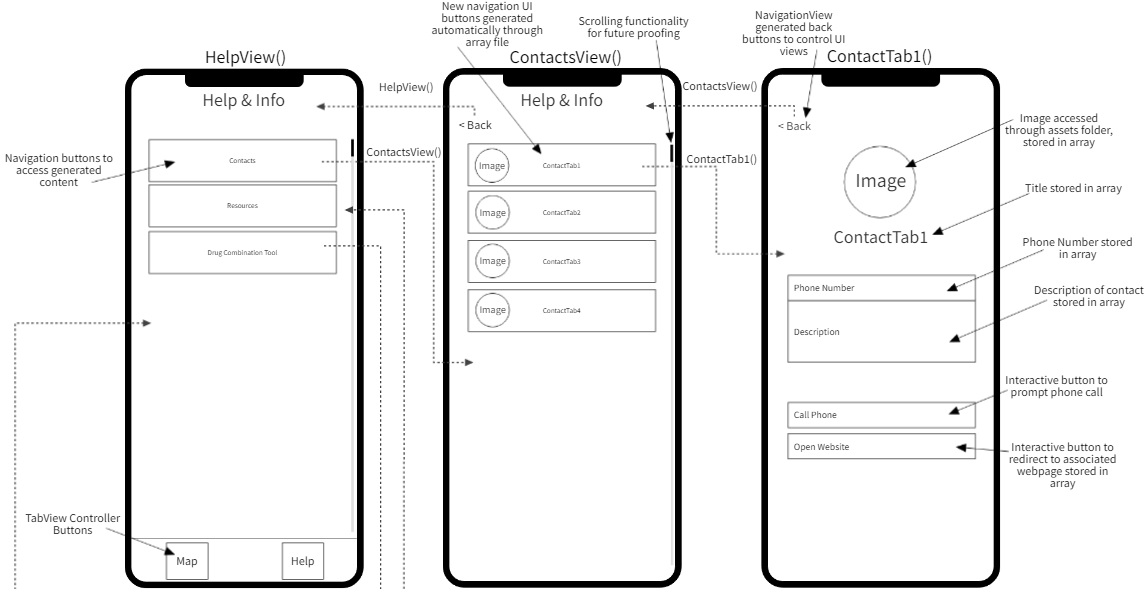


Figure : Wireframes for example views

*Row One: HelpView(), ContactsView() and ContactTab1(). Row Two: DrugCombinationPickerView(), Mapview().*

## 

## 5.7 Testing Strategy

After the completion of each sprint, unit testing of the application will be conducted to identify and resolve any issues. As an example of this, navigation between different viewports would be presented in the format below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Outcome | Actual Outcome | Pass/Fail |
| 1.0 | Swap to Maps viewport | Redirected to Maps viewport | Redirected to Maps viewport | Pass |
| 1.1 | Swap to Resources viewport | Redirected to Resources viewport | Viewport does not swap | Fail |

Table : Example Test table

At the final stages of development, user acceptance testing will be conducted to ensure the application is ready for delivery. This will be more in-depth testing, identifying issues and ensuring all functional and non-functional requirements have been completed. An example of this can be found below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UAT ID | Test Description | Acceptance Test | Prerequisites | Pass/Fail |
| 1.0 | Access University Wellbeing & Support | **Method:**  1. Open application  2. Swap to “Help” viewport  3. Open “Resources” navigation link  **Expected Outcome:**  1. User is presented with an automatically generated list of resources based on data stored inside an array | N/A | Pass |

Table : Example UAT table

# 6 Implementation

## 6.1 Sprint 1

## 6.2 Introduction

Sprint 1 describes the creation of the project workspace, initial front-end development of the application i.e the implementation of the map & resources views and the navigation buttons between each viewport. This also includes the setup/implementation of the Google Maps SDK and additional accessibility features.

## 6.3 Development

### 6.3.1 Creating the project workspace

Using Xcode14, a SwiftUI project was created using the bundle identifier “com.jaggard.tomas.DSPApp”. A bundle Identifier allows the project application to be uniquely identified in Apples ecosystem, which will be used later to specify external access to the application. The configuration of this can be seen in Figure 8.

A screenshot of a computer

Description automatically generated

Figure : Project Setup

### 6.3.2 Creating the TabView view controller

In this sprint, there were many considerations made as to how to streamline navigation inside of the application to provide a simple and effective user experience. In order to achieve this, the TabView structure was implemented, providing a footer bar to contain interactive buttons to navigate between separate views. As showcased in figure 9, all views will be contained within the tabview and handled by ContentView which acts as the main view for the application. Two tabItems were included, with a title and associated icons to represent the associated view and its functions.

Text

Description automatically generated

Figure : ContentView(Main) View

### 6.3.3 Creating HelpView

The creation of the UI for the landing pages for both the help views and map view were developed using the “stack” structures available in the swift programming language, allowing for the application UI to automatically scale with different device screen sizes. Using the stack structures, a header was generated of which included the title of the view with the addition of the UWE logo alongside it, both customised using instance methods. Figure 10 show the initial implementation of this header functionality inside of the view.

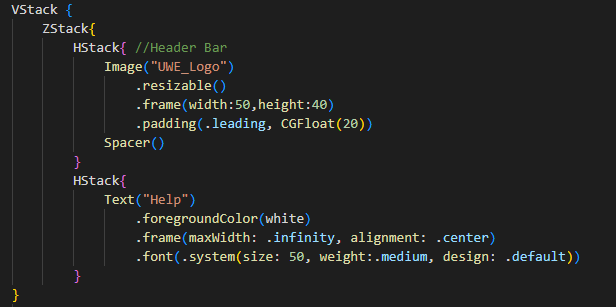


Figure : App Header

Inside the contents of the view, a NavigationView was implemented to traverse the views which will be implemented in later sprints. Using NavigationLinks, buttons were generated which on-press, bring the user to a new view specified in the “destination” of the function. This method of controlling views is ideal as it conforms to the Apple Human Interface Guidelines of iOS app development required to post applications to the App Store. The structure of this can be seen in Figure 11.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Navigation Links

### 6.3.4 Creating MapView

As MapView will eventually be used to display the Google Map with heatmap data, this does not require any more setup than initialising the view as seen in Figure 12.

A screenshot of a computer

Description automatically generated with low confidence

Figure : MapView View

### 6.3.5 Accessibility Features

The next section of this sprint focused on developing additional functionalities for accessibility purposes. This is of importance to include as the application should be inclusive to a range of different users of varying ability.

A bool-based light/dark mode button was implemented to swap between themes in the application. This decision was made to assist those with visual impairments and those who may suffer from photophobia. As seen in Figure 13, a lightbulb was added to the header of the application to control this functionality.

Text

Description automatically generated

Figure : Light/Dark mode button

To enable this functionality, a new enum was created with the contents of “isDark”, which is cycled between true and false on button-press, toggling the state of the variable. Using @AppStorage, this value is saved under the UserDefaults value of “isDarkMode”, commonly used to store preferences made within applications, and can be updated per request. The use of this inside the application can be seen in Figures 14 and 15.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure : Lightbulb toggle button (light mode)

Graphical user interface, text

Description automatically generated

Figure : Lightbulb toggle button (dark mode)

### 6.3.5 Installing CocoaPods dependency manager and Google Maps Frameworks

In order to develop the map view UI in later sprints, the environment requires frameworks to run the Google map and heatmap. Using a dependency manager such as Cocoapods provides a method of easy installation of frameworks and makes managing dependencies simple.

To install cocoapods, the gem had to first be requested via the development devices terminal. This was completed using the command “sudo gem install cocoapods”. Once installed on the device, a Podfile (used to describe dependencies needed inside of a project workspace) was created inside of the application folder and the following dependencies were requested: “GoogleMaps” and “google-maps-ios-utils”, as seen in figure 16. Once the Podfile was completed, running the command “$ pod install” installed the requested dependencies, and additional frameworks which were not needed such as “lPods” were removed to avoid errors.

Text

Description automatically generated

Figure : Podfile

## 6.4 Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Outcome | Actual Outcome | Pass/Fail |
| 1.0 | Generating HelpView viewport | Page renders correctly, with logo and title in correct positions | Page renders correctly, with logo and title in correct positions | Pass |
| 1.1 | Generating MapView viewport | Page renders correctly | Page renders correctly | Pass |
| 2.0 | tabView code creates Help & Map buttons | Buttons load correctly, with default tab footer bar colour | Buttons load correctly, with default tab bar colour | Pass |
| 2.1 | tabView swaps between MapView/HelpView | On-click, swap between views | On-click, swap between views | Pass |
| 3.1 | Lightbulb button inverts colour scheme for viewport backgrounds | On-click, viewport background change between light/dark colour scheme | On-click, viewport background change between light/dark colour scheme | Pass |
| 3.2 | Lightbulb button inverts colour scheme for header/footer text | On-click, header/footer contents change between light/dark colour scheme | On-click, header/footer contents change between light/dark colour scheme | Pass |
| 3.3 | Lightbulb button inverts footer/header backgrounds | On-click, header/footer backgrounds change between light/dark colour scheme | On-click, header/footer did not change colour | Fail |
| 4.0 | CocoaPods installation crash-check | Application still runs as intended | Application continues to run as intended | Pass |
| 4.1 | Adding additional frameworks to pods file | After installation, application still runs as intended | Application continues to run as intended | Pass |
| 5.0 | Google Maps SDK implementation | Importing GoogleMaps framework provides no error & references correct framework | Importing GoogleMaps framework provides no error & references correct framework | Pass |
| 5.1 | Google Maps iOS Utils SDK implementation | Importing GoogleMaps iOS Utils framework provides no error & references correct framework | Importing GoogleMaps iOS Utils framework provides no error & references correct framework | Pass |

Table : Sprint 1 tests

## 6.5 Sprint 2

## 6.6 Introduction

Sprint 2 discusses the development of the resources and contacts views, the data entry method, and the migration from a button-based theme controller to a system-settings-based controller.

## 6.7 Development

### 6.7.1 Updating & Fixing accessibility features

The first task in this sprint was addressing issue 3.3 discovered in the testing phase of the first sprint. Upon further research into controlling themes in SwiftUI, the use of ColorScheme was highlighted as an enumeration popular in controlling light and dark themes. This was introduced in place of the lightbulb button and allowed for the identification of the native theme set by the user in their device settings. Figure 16 and Figure 17 both display how using colorScheme can control the entire theme of the UI (including both headers and footers) in a more simplified format in comparison to the previously implemented semi-functional button function. This was included in both the HelpView and MapView files.

Text

Description automatically generated

Figure : HelpView view implementing new theme controller

Text

Description automatically generated

Figure : Header text controlled by new theme controller functionality

### 6.7.2 Implementing auto-generated views for Contacts&Resources View

To add data for the user to access inside of the aforementioned “NavigationLink”s(6.3.3) without the need to implement a new “NavigationLink” manually per data entry. An automated solution was introduced using arrays to hold & reference data to create new accessible links dynamically and provide a consistent UI per element. The following method described was implemented for both Contacts and Resources “NavigationLink”s, but the example is from the ContactsView() respectively.

To begin, a struct was defined to contain the contents of the information, with key information included being the name, description, website, email and primary and secondary phone numbers as seen in Figure 19. An array of data was then crafted using available data from the university website (<https://www.uwe.ac.uk/life/health-and-wellbeing>) and assigned to a variable(“contacts”), with all necessary data included in the array as seen in Figure 19.

Text

Description automatically generated

Figure : Contact struct

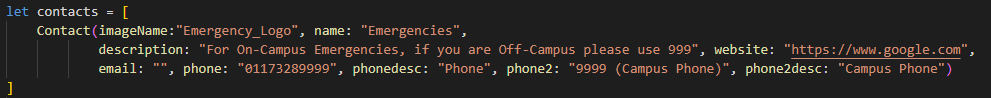


Figure : Contact details inside array

Once the array was outlined, ContactsView() contains two core structs:

ContactRow utilises the array “contacts” to create a new contact row per array element. This utilises the horizontal and vertical stack structures to effectively format the data into presentable rows, making calls to display the data by referencing the array.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : ContactRow View

The ContactsView struct (visible body) iterates through a “for” loop to generate a new Navigation link for each individual array element, creating a UI visible in Figure 21 and directing the user to the “ContactDetailsView”.

Graphical user interface, application

Description automatically generated

Figure : Contacts UI

Text

Description automatically generated

Figure : ContactsView View

ContactDetailsView was created to receive the contents of an element in the array (per for loop iteration) and present the data in a clean and concise manner. Additionally, in the case of having null values for certain niche variables such as “phone2”, inequality operators (Figure 23) were introduced to remove data from the UI that would otherwise return empty.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : ContactsDetailView View

Text

Description automatically generated

Figure : Inequality operators

### 6.7.3 Implementing external links & call prompts

When developing the individual views for each view, “navigationLinks” were found to be often described as the best method to help the user traverse views, however when attempting to forward the user to a website or prompt a phone call, errors such as errors such as “unexpected nil issue” were discovered. This was due to regular “Links” needing to be implemented to exit the navigationView, otherwise the application attempts to locate a view under the specified url or phone number. Once discovered this was quickly corrected and the functionality worked as expected.

Text

Description automatically generated

Figure : Implementation of Links

## 6.8 Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Outcome | Actual Outcome | Pass/Fail |
| 1.0 | Application theme changes on-load | Application theme (light/dark) conforms to system settings | Application theme (light/dark) conforms to system settings | Pass |
| 1.1 | Application theme changes dynamically | Application theme (light/dark) conforms to system settings, and changes without need for re-load | Application theme (light/dark) conforms to system settings, and changes without need for re-load | Pass |
| 2.0 | Contacts generation | Contacts & associated navigation buttons are generated automatically after entering data into “ContactDetails” | Contacts & associated navigation buttons are generated automatically after entering data into “ContactDetails” | Pass |
| 2.1 | Contacts tab navigation | Contacts tab navigates to correct resources on-click | Contacts tab navigates to correct resources on-click | Pass |
| 2.2 | Contacts generation hiding subsections | Subsections such as “website” are hidden if validity check fails (No input) | Subsections such as “website” are hidden if validity check fails (No input) | Pass |
| 3.0 | Resources generation | Resources & associated navigation buttons are generated automatically after entering data into “ResourceDetails” | Resources & associated navigation buttons are generated automatically after entering data into “ResourceDetails” | Pass |
| 3.1 | Resources tab navigation | Resources tab navigates to correct resources on-click | Resources tab navigates to correct resources on-click | Pass |
| 3.2 | Resources generation hiding subsections | Subsections such as “website” are hidden if validity check fails (No input) | Subsections such as “website” are hidden if validity check fails (No input) | Pass |
| 4.0 | Website/Phone Links | Website loaded/phone call pop-up link on-click | Website loaded/phone call pop-up link on-click | Pass |

Table : Sprint 2 tests

## 6.9 Sprint 3

## 6.10 Introduction

Sprint 3 describes the implementation and development of MapView, including the setup of the API, integrating the API into the project, creating a view controller bridge to allow for a swift-generated view to be used by SwiftUI and creating the functionalities to generate the map inside of the application once bridged.

## 6.11 Development

### 6.11.1 Setting up Google Maps API Key

The first task in this sprint was to implement the Google Maps API functionalities, which would allow for calls to generate the map which will be used as a base for the heatmap. To request an API key, a Google Cloud account was created, which provides a 90 day free trial and up to £250 of free credit when using their APIs. Once inside the Cloud portal, a “Maps SDK for iOS” API key was generated and was restricted to only accept requests from an iOS application with the correct bundle identifier (Figure 27).

Graphical user interface, text, application, email

Description automatically generated

Figure : API Key Setup

Once the key has been generated, it is standard in swift to include this inside of the @main application automatically generated to handle the setup/launch process. The API key is assigned to the variable APIKey and a new class called AppDelegate is created to handle the the functionality of the API as seen in Figures 28 and 29.



Figure : Partially censored API key

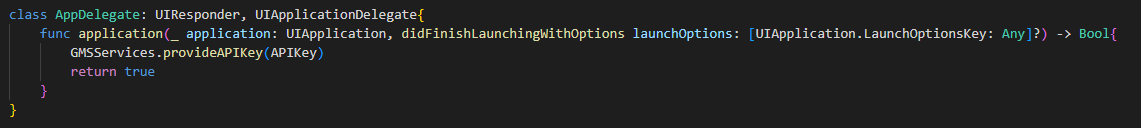


Figure : AppDelegate class

### Creating the Map

To generate the map, first the frameworks: “GoogleMaps” and “GoogleMapsUtils” were imported into the project and a new class “MapViewController” was created to handle the UI and additional components in future sprints. The map is first assigned the class reference “GMSMapView” which acts as the entry point for methods relating to the map. A function is next overwrote (standard Google Maps library has pre-existing loadView contents) under the name of “loadView” and will be used to initially load the custom contents of the map to the UI. A variable “camera” is then created and assigned the value of “GMSCameraPosition.camera(latitude, longitude,zoom)” with the values of the function equating to an overview of Bristol. This variable can then be used as seen in Figure 30 to set the viewpoint for the UI of the map and will focus on this on-load.

Text

Description automatically generated

Figure : loadView function

### 6.11.1 Creating ViewControllerBridge to enable Swift Viewcontroller functionality

As the Maps SDK for iOS is built on top of the UIKit framework and can not provide SwiftUI compatible views, to enable use in SwiftUI requires MapViewController” to be “UIViewControllerRepresentable or UIViewRepresentable” as per the developer documentation (Google, 2022). This allows for SwiftUI to utilise UIKits classes “UIView” and “UIViewController”.

To handle this, a new struct “MapViewControllerBridge” was created to handle the data being received from “MapViewController”. The functions inside of the struct receives the view being generated by “MapViewController” and returns a uiViewController, usable inside of SwiftUI as seen in Figure 31.  
  
Text

Description automatically generated

Figure : MapViewControllerBridge struct

To call this struct as a view, it is initialised inside the body of “MapView” using the line “MapViewControllerBridge()”.

### 6.11.2 Implementing location tracking

To implement location tracking, a new protocol must first be included in the MapViewController class named “CLLocationManagerDelegate” which will allow the receival of events from location manager functions.

A variable “locationManager” is assigned with the property of CLLocationManager which is a class that enables the request and tracking of the user location among others. Once assigned, inside of the previously created “viewDidLoad” function locationManager is delegated to “self” which requests that data retrieved be received by the MapViewController. An if statement (Figure 32) is implemented to verify that the user has location services enabled, and if they don’t to prompt them to enable it for the functionality to be activated.

Text

Description automatically generated

Figure : viewDidLoad function

A new function is then created which will handle the camera focusing on the user position if they allow location services, and if not, the function will be ignored and the map camera will automatically focus on the Bristol Area. The function “locationManager()” retrieves the user position using the previously assigned variable “locationManager” and sets the users location to the last known location using “locations.last” (Figure 33). The coordinates from the last known location are then used by the camera to center around the users position in a zoom effect. Once this task is completed, the application stops updating the users location.

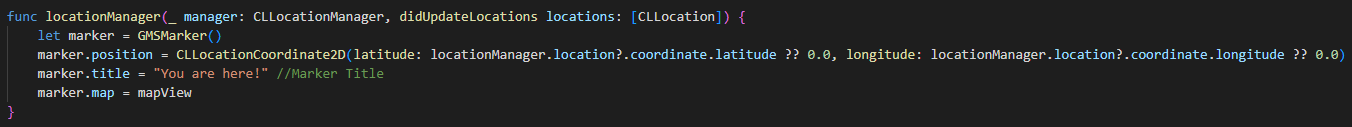


Figure : locationManager function

Additional functions were also implemented to catch errors when handling the location as well as a switch statement to handle denied & restricted authorisations, prompting the user to re-enable location services can also be seen in Figures 34 and 35.

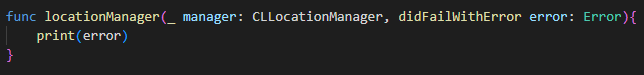


Figure : locationManager Error Handler

Text

Description automatically generated

Figure : LocationManager authorization switch statement

To conclude this functionalities implementation, a new key is added to the “Info.Plist” file (Figure 36) to describe why the application requires permission to access the user’s location data. Without this, the application will automatically fail to launch due to it being a requirement set by Apple when developing applications for their systems.



Figure : Info.Plist Permission Description

### 6.12 Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Outcome | Actual Outcome | Pass/Fail |
| 1.0 | API key validation | API Key working as expected, request tracked. | API Key working as expected, request tracked. | Pass |
| 2.0 | Map Generation | Map is loaded via MapViewController -> MapViewControllerBridge -> MapView successfully | Map is loaded via MapViewController -> MapViewControllerBridge -> MapView successfully | Pass |
| 2.1 | Map focus on Bristol | Map uses camera functionality and given latitude & longitude to focus on Bristol centre | Map uses camera functionality and given latitude & longitude to focus on Bristol centre | Pass |
| 3.0 | Location Tracking | Prompt user to allow access to location tracking while in use. | User is prompted to allow access to location while in use. | Pass |
| 3.1 | Location map zoom | Camera zooms to approximate location on map | Camera zooms to approximate location on map | Pass |

Table : Sprint 3 tests

## 6.13 Sprint 4

## 6.14 Introduction

Sprint 4 introduces the heatmap which is applied over the previously created Google Map, utilizing calls to the UK Police data API to retrieve and JSON values, which are then passed through the appropriate functions to plot said data on the map.

## 6.15 Development

### 6.15.1 Setting up the heatmap overlay

The first task identified in this sprint was to configure the heatmap for use on the current map UI. To do this, the Google Maps iOS Utils framework was utilized as it provides an array of utilities to manipulate the map view previously implemented.

The GoogleMapsUtils library was first imported to allow for its functionalities to be enabled in the project. Once imported, a new function called “setUpHeatMap” was created which configures the heatmap for future use. As seen in Figure 37 the modifiers changed were the radius, opacity and gradient, of which were customised to best represent the data that will be passed through the overlay. With regards to the gradient, two inputs were required , colors and startpoints, which were customized to show a green heatmap marker with an increasingly red centre depending on the amount of data entries for the point as seen in Figure 38.

Text

Description automatically generated

Figure : setUpHeatMap function

A picture containing radar chart

Description automatically generatedA picture containing background pattern

Description automatically generated

Figure : heatmap-generated marker

### 6.15.2 Making Crime API call

To receive local crime data in the Bristol area, a call to the UK Police Data API was decidedly implemented which allows for the definition of a target area, and the retrieval of categorised crimes in the selected area via a readable JSON format.

The target area was first outlined by using the Google Maps web browser to plot the approximate latitudes and longitudes of the area (Bristol) seen below in Figure 39. This was then formatted into a URL request by using the poly parameters outlined in the API documentation (Data.Police.UK, 2022) and specifying the target crime (drugs).

Map

Description automatically generated

Figure : Search radius border (Visual representation)

Once the URL was created, it was then executed in the browser (Police data API does not require a key) so that the results could be examined, and a struct could be developed to hold the data it returns. After examining the JSON results, a struct was created under the name of “Crime” which would hold the details of the request as seen in Figure 40. Using a struct to hold the data allows for easy referencing when attempting to extract the longitude and latitude data from the other “noise” provided by the call.

Text

Description automatically generated

Figure : Crime struct

A new function was then created to return the future results of the API call, using an escaping closure to return the JSON file data which will be formatted into the “Crime” struct.

To ensure the data being received was the most recent available, two variables “thisYear” and “lastMonth” were created, both using the calendar and date functionalities in swift to reference the current year and the month with the most recent data (“API data is delayed by one month, hence the use of -2 seen in Figure 41). This is then inputted into the URL string using \ (escaping character) to input the current year and month retrieved. Once formatted, the URL was set as a variable (“crimeURL”) for later referencing.

Text

Description automatically generated

Figure : Assignment of variables in getCrimes function

**STRUCT REMOVING NOISE – NEVERMIND TALK ABOUT REFERENCING INSTEAD**

### 6.15.3 Constructing the heatmap array data

To construct the heatmap array data, a call inside of the application must be made to the contents of “crimeURL”. In Figure 42, a for loop is implemented to decode and iterate over the JSON data inside of the URL and assign a new variable (“crimes”) the value of each element as a list, constructed using the “Crime” struct. To ensure this code is completed before the results of getCrimes are returned (using “@escaping” closures causes concurrency issues such as returning before values are assigned), a dispatch semaphore was implemented to prevent such issues, which allows the application to manually designate the return of the function. Using “sem.signal()” increments the semaphore value by 1, signalling to return the value of “crimes” using “sem.wait()” and the end of the function.

Text

Description automatically generated

Figure : Retrieving JSON data from URL

### 6.15.4 Generating the heatmap

To generate the heatmap, a new function called “addHeatMap” was created to receive the data retrieved in “getCrimes” and assign heatmap markers to specified coordinates on the heatmap UI. A new list “weightedList” is first created and is initialised with the value of the empty array “[GMUWeightedLatLng]()”. The “getCrimes” function is then called in the form of a for loop to search the array of “crimes”. Another for loop is then called after setting the value of crimes to self, this loop fetches the latitude and longitude of each element in the array and sets their value to a new variable “coordinates” which is then appended to “weightedList”. Once every element in the list is iterated over, the final value of “weightedList” is applied to the heatmaps “weightedData” which holds the longitude and latitude values for the heatmap to utilise in the mapping process.

## 6.16 Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Outcome | Actual Outcome | Pass/Fail |
| 1.0 | Heatmap overlay generation | Heatmap overlay generated successfully | Heatmap overlay generated successfully | Pass |
| 2.0 | Relevant crime API data collected | Data requested from data.police.uk and each element is appended to a list | Data requested from data.police.uk and each element is appended to a list | Pass |
| 2.1 | Retrieve latitude/longitude from list data | Irrelevant data is filtered from list, data is returned as pairs of lat/long | Irrelevant data is filtered from list, data is returned as pairs of lat/long | Pass |
| 2.2 | Updating calendar to show new results | Data requested from data.police.uk and longitude/latitude details are different from the previous heatmap (match correct JSON) | Data requested from data.police.uk and longitude/latitude details are different from the previous heatmap (match correct JSON) | Pass |
| 3.0 | Longitude/Latitude plotted with weights | Longitude/latitudes of data in list plotted on the heatmap overlay successfully | Longitude/latitudes of data in list plotted on the heatmap overlay successfully | Pass |

Table : Sprint 4 tests

## 6.17 Sprint 5

## 6.18 Introduction

Sprint 5 describes the creation and implementation of the “Drug combination tool” and the addition of assets required to ensure the application is visually appealing. It also includes the User Acceptance testing and a review of the requirements described in the requirements chapter (3.0).

## 6.19 Development

### 6.19.1 Creating the Drug combination tool UI

To create this functionality, a new view was first created and a new Zstack containing a title, two pickers (drop-down menus) and a button inside. Ease-of-use was taken into consideration when creating this view as simplicity was of the utmost importance to encourage use. A new array was created, containing commonly used substances to hold the potential values of the pickers and was assigned the variable “drugs”. The elements of “drugs” was then provided to the picker using a for loop which can be seen in Figure 43.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Drug combination tool UI

### 6.19.2 Creating the Drug combination tool functionality

To add functionality to the tools UI, the function “returnCombo()” was created that receives two values “drug1Value” & “drug2Value”. Inside of the function a matrix has been created which contains values which represent the reaction between the combination of two different substances. To retrieve these values, the positions of “drug1Value” & “drug2Value” are located in the initial array “drugs”, finding the first index where their values match, and using the resulting positions, the matrix is indexed to return the value of the combination. An If statement is then implemented to compare the matrix results, and the value which matches returns a string, assigned to the variable “combination”, informing the user of their chosen combinations effects.

Text

Description automatically generated

Figure : returnCombo function

Once the function was created, a button was implemented to the UI along with a text field with the variable “combination” inside as seen in Figure 45. When pressed, the button executes the “returnCombo” function and the blank string is assigned the resulting value.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : Adding returnCombo function call button

### 6.19.4 Adding visual assets to the application

To complete the applications development process, assets were generated for the image calls which referenced non-existing files. First, the logo for the application was made by downloading the font used in the standard UWE logo and acquiring the hex code for the signature UWE red color. This was imported into photoshop and edited to accommodate a “wellbeing” line of text under the usual logo as seen in Figure 46. This was then resized and saved multiple times in accordance with Apples recommendations on different icon sizes for different devices.

Text

Description automatically generated

Figure : Application logo

*(with ruler lines to accurately measure text size)*

Images for contacts and resources were found and then inserted into photoshop (Figure 47), being resized and edited (and in some cases recoloured) to look sharper that what was available, and were then saved as PNG files and named corresponding to the name in the data arrays. All assets were added to the Assets folder inside the application which automatically set them to be used over their blank placeholder as they shared the same file name.

Graphical user interface, application

Description automatically generated

Figure : Photoshop asset layers

The Application with assets now added is visible in Figure 48 below.

Graphical user interface, application, Teams

Description automatically generated

Figure : UI View with assets added

## Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Expected Outcome | Actual Outcome | Pass/Fail |
| 1.0 | Opening the Drugs combination tool view | View loads, with corresponding theme | View loads, with corresponding theme | Pass |
| 2.0 | Enter two of the same values and button pressed | User receives text: “Your selected combination has no interaction” | User receives text: “Your selected combination has no interaction” | Pass |
| 2.1 | Enter two different values and button pressed | User receives text corresponding to the values chosen | User receives text corresponding to the values chosen | Pass |
| X.0 | Assets inside application load correctly | Images appear as expected | Images appear as expected | Pass |
| X.1 | Assets outside the application load correctly (Icon) | Icon appears as expected | Icon appears as expected | Pass |

Table : Sprint 5 tests

## 6.20 Acceptance Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UAT ID | Test Description | Acceptance Test | Prerequisites | Pass/Fail |
| 1.0 | Open application | **Method:**  1. Locate application on home screen.  2. Tap icon to open  **Expected outcome:**  **1.** User is prompted to allow location tracking  2. Help View is loaded |  |  |
| 2.0 | Access the resources view | **Method:**  1. Open the Resources navigation link  **Expected outcome:**  1. User is presented with a list of resources correlating to those in the array data | User is viewing Help View |  |
| 2.1 | Access a University Resource | **Method:**  1. Open any sub-resource navigation link  **Expected outcome:**  1. User is presented with resources and (if appropriate) buttons to redirect them to the associated website | User is viewing Resources View | Pass |
| 3.0 | Access the Contacts view | **Method:**  1. Open the Contacts navigation link  **Expected outcome:**  1. User is presented with a list of contacts correlating to those in the array data | User is viewing Help View | Pass |
| 3.1 | Access a University Contact | **Method:**  1. Open contacts navigation link  2. Open any sub-contacts navigation link  **Expected outcome:**  1. User is presented with contacts and (if appropriate) buttons to redirect them to the associated website or prompt a phone call. | User is viewing Contacts View | Pass |
| 4.0 | Access the Drug combination tool | **Method:**  1. Open Drug combination tool navigation link  **Expected outcome:**  1. User is presented with the Drug combination tool | User is viewing Help View | Pass |
| 4.1 | Combine two of the same drug | **Method:**  1. Select two of the same choice from the pickers (drop down menus)  2. Press combination button  **Expected outcome:**  1. User receives: “Your selected combination has no interaction” on-click | User is viewing Drug combination tool view | Pass |
| 4.2 | Combine two different drugs | **Method:**  1. Select two different choices from the pickers (drop down menus)  2. Press combination button  **Expected outcome:**  1. User receives response specific to selected drugs on-click | User is viewing Drug combination tool view | Pass |
| 5.0 | Return from previous navigation link | **Method:**  1. Click “Back” button in the top-right corner of the view  **Expected outcome:**  1. User returns to previous view | User is already inside a navigation link view | Pass |
| 6.0 | Change theme – Light mode | **Method:**  1. Open system settings  2. Modify light/dark mode toggle button  3. Navigate back to application  **Expected outcome:** 1. Users theme is change to light mode | User is on “Dark mode” | Pass |
| 6.1 | Change theme – Dark mode | **Method:**  1. Open system settings  2. Modify light/dark mode toggle button  3. Navigate back to application  **Expected outcome:** 1. Users theme is change to dark mode | User is on “Light mode” | Pass |
| 7.0 | Access map | **Method:**  1. User clicks on “Map” tab button  **Expected outcome:** 1. Map is loaded within 1s of request | User is viewing Help View | Pass |
| 8.0 | View heatmap | **Method:**  1. User clicks on “Map” tab button  **Expected outcome:** 1. Heatmap is loaded and overlayed over the map within 1s of request | User is viewing Help View | Pass |
| 8.1 | View previous month heatmap | **Method:**  1. User changes system clock to previous month  **Expected outcome:** 1. Heatmap is updated to previous months heatmap (on-load) | User is viewing Map | Pass |
| 9.0 | View location on map | **Method:**  1. User clicks on UI button in bottom right  **Expected outcome:**  1. Map is centred on user location | User is viewing Map | Pass |

Table : User Acceptance Testing

## 6.21 Requirements Review

### 6.21.1 Functional Requirements

|  |  |
| --- | --- |
| Requirement Code | Status |
| FR1 | Pass |
| FR2 | Pass |
| FR3 | Pass |
| FR4 | Pass |
| FR5 | Pass |
| FR6 | Pass |
| FR7 | Pass |
| FR8 | Pass |
| FR9 | Pass |
| FR10 | Fail |
| FR11 | Fail |

Table : Functional Requirements Review

### 6.21.2 Non Functional Requirements

|  |  |
| --- | --- |
| Requirement Code | Status |
| NFR1 | Pass |
| NFR2 | Pass |
| NFR3 | Pass |
| NFR4 | Pass |
| NFR5 | Pass |
| NFR6 | Pass |
| NFR7 | Pass |
| NFR8 | Pass |
| NFR9 | Fail |
| NFR10 | Fail |

Table : Non-functional Requirements Review

# 7 Project Evaluation

## 7.1 Research

The research in this project was a success as many quality references were found that validated the outlined benefits the application developed had on aiding Bristol residents who are some of the largest users of illegal substances in Europe. One issue to highlight was the challenge of finding relevant and formal publications around drug abuse in Bristol, as many were journal/blog showed little value. However the references that were found were high quality.

## 7.2 Requirements

Gathering requirements was a large success as a various range of different functional and non-functional requirements were gathered which led to a simple planning process where requirements could be referenced to ensure all features were included. The requirements were kept concise and summarised the tasks well. The only notable issue was the maintainability of the application (NFR9) as there is room for improvement as later discussed in the Future Improvements (7.7).

## 7.3 Methodology

The chosen methodology provided a method of precise planning and outlined deliverables for each sprint. This aided in tracking progress, documenting the process and encouraged development as progress was visible from start to finish. Unfortunately, due to issues unrelated to the project, keeping pace with the target goals proved difficult, but being able to visualise what was needed by date was useful in case of delays so that less additional time was spent planning what was built first. An issue discovered with the methodology however was the experience felt slightly fragmented, with functionalities feeling less intertwined than preferred.

## 7.4 Design

The design process was very in-depth, with an array of diagrams to illustrate the capabilities intended for the application. There was also the inclusion of wireframes to design the applications UI prior to development, which provided a strong guideline for how the application was designed to look in the implementation phase. One improvement could have been to include composites to represent the UI with user-friendly features such as colour, as this would help envision what was aimed for, prior to development.

## 7.5 Implementation

The implementation process was well-documented and of a high standard, with detailed analysis of functionalities and explanations of concepts within the application. All “Will”, “Should” and “Could” requirements of the MoSCoW method were met excluding one and this was noted throughout the documentation for sake of clarity. However, an issue that was realised throughout the project was the discovery of new functionalities (due to an unexpected lack of knowledge certain capabilities of the Swift language) within Swift and the libraries used led to documentation needing to be altered to adjust for updated code for concurrency with the version being developed upon.

## 7.6 Testing

Testing was conducted smoothly, with the documentation being concise and reflecting issues faced within the project. If and when issues were discovered, they were prioritised in the next sprint, for example when the header and footers were not reacting to the change in theme, a new fix was researched and as a result, a more efficient and cleaner solution was implemented. In retrospect, more testing could have been completed to highlight all minor functionalities being tested, however the choice to keep the testing concise reduced redundancy in the documentation.

## 7.7 Future Improvements

In the future, the system could be improved by providing a form of UI to update the information stored in the Contacts & Resource details files, which would provide the university with an easier means of updating information and contact methods whilst also being more inclusive to those with varying skill ranges. This could be further improved upon by collecting the data on the website via scraping, and automatically implementing relevant details, however this could lead to redundant data being included, degrading the user experience.

Another improvement that could be considered is implementing a system to generate a heatmap around the user rather than a fixed location. While the fixed location system is suitable for the UWE theme applied in the project, if the scope were expanded to a include a wider userbase, using location data the heatmap could quite simply be adapted to retrieve results from a 10-mile radius around the user location. However, a drawback of this could be increased loading times, requiring the map to pre-load before the UI is generated.

A further improvement to consider would be the updating of the drug combination tool to include more niche drugs rather than an umbrella of terms, as more accurate details of combinations could be the difference between someone dismissing the application due to confusing umbrella terms, leading to potentially life-threatening issues, and using the application and staying safe.

# 8 Conclusion

In conclusion, the project met the objectives defined in the requirements section, providing an application for UWE residents to use when needing wellbeing support and assistance. The application allows users to view a heatmap of Bristol, identifying key areas of issue with regards to drug crime via API calls and weighted coordinate plotting, and provides on-the-go tools and resources to individuals who may be struggling at/with university life.

# Table of Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Meaning** |
| API | Application Programming Interface |
| FR | Functional Requirements |
| GMS | Google Maps Service |
| GPS | Global Positioning System |
| iOS | iPhone Operating System |
| NFR | Non-Functional Requirements |
| SDK | Software Development Kit |
| UAT | User Acceptance Testing |
| UML | Unified Modelling Language |
| UI | User Interface |
| URL | Uniform Resource Location |
| UWE | University of the West of England |

# Bibliography

Ally, M. and Prieto-Bláquez, J. (2014) What is the future of mobile learning in education? Mobile Learning Applications in Higher Education. *International Journal of Educational Technology in Higher Education* [online] 11, pp.142–151 [Accessed 29 May 2022].

Black, P. and Wiliam, D. (1998) Assessment and Classroom Learning.*Assessment in Education: Principles, Policy & Practice*. 5(1), pp.24-25. [Accessed 29 May 2022].

Bradshaw, J. and Richardson, D. (2009)An Index of Child Well-Being in Europe. *Child Indicators Research* 2, pp.319–351 [Accessed 27 May 2022].

Cameron, A. (2019) *Bristol has no strategy for city's drug problem as drug-related deaths reach record numbers*. Available from: https://www.bristolpost.co.uk/news/bristol-news/bristol-no-strategy-citys-drug-3224979 [Accessed 29 May 2022].

Clayton, R, Cattarello, A, and Johnstone, B. (1996) The Effectiveness of Drug Abuse Resistance Education (Project DARE): 5-Year Follow-Up Results. *Preventative medicine.* 25, pp.309 [Accessed 29 May 2022].

Crane, J. (2018) Bristol’s ketamine dealers: trapped in luxury. *Addicted Bristol: Life And Death.* Available from: https://thebristolcable.org/2018/12/bristols-ketamine-dealers-trapped-in-luxury/ [Accessed 27 May 2022].

European Monitoring Centre for Drugs and Drug addiction (2018) *Wastewater analysis and drugs — a European multi-city study*. Available from: https://www.emcdda.europa.eu/publications/html/pods/waste-water-analysis\_en [Accessed 27 May 2022].

Google Developers (2022) *Add a map to your iOS app with SwiftUI (Swift)*. Available from: https://developers.google.com/codelabs/maps-platform/maps-platform-ios-swiftui [Accessed 3 June 2022].

Li, C., and Lalani, F. (2020) *The COVID-19 pandemic has changed education forever. This is how.* Available from: https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/[Accessed 30 May 2022].

OECD (2013) *Education at a Glance 2013: OECD Indicators*. Available at: https://www.oecd.org/education/eag2013.htm [Accessed 28 May 2022].

Office of National Statistics (2016) *Deaths related to drug poisoning in England and Wales*. Available at: https://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/bulletins/deathsrelatedtodrugpoisoninginenglandandwales/2016registrations [Accessed 27 May 2022].

Police UK (2022) *Home | data.police.uk.* Available from: https://data.police.uk/ [Accessed 6 June 2022].

Public Health England (2018) *Alcohol and drug prevention, treatment and recovery: why invest?* [online] *Available from: https://www.gov.uk/government/publications/alcohol-and-drug-prevention-treatment-and-recovery-why-invest/alcohol-and-drug-prevention-treatment-and-recovery-why-invest* [Accessed 27 May 2022].

Santor, D. (2009) *Taking mental health to school.* Available from: https://canadacommons.ca/artifacts/1200519/taking-mental-health-to-school/1753638/[Accessed 28 May 2022].

Torrance, J, Khouja, J, and Crick, E. (2018) *Bristol in Brief: Drugs in the South West*. Available from: https://www.bristol.ac.uk/policybristol/policy-briefings/bristol-in-brief-1-drugs-in-the-south-west/ [Accessed 27 May 2022].

Waters, K. (2009) *Prioritisation using MoSCoW.* Available from: https://comp.anu.edu.au/courses/comp3120/local\_docs/readings/Prioritization\_using\_MoSCoW\_AllAboutAgile.pdf [Accessed 28 May 2022].

Wood, J (2022) *These 3 charts show the global growth in online learning.* Available from: https://www.weforum.org/agenda/2022/01/online-learning-courses-reskill-skills-gap/ [Accessed 30 May 2022].