Project CO2Aware

Lorenzo Battilocchi (16341353)

Lorenzo.battilocchi@ucdconnect.ie

Liliana Pasquale

liliana.pasquale@ucd.ie

University College Dublin
Ireland's Global University

Table of Contents

List of Figures

List of Important Abbreviations Used Within

HOV	*High Occupancy Vehicle*
API	*Application Programming Interface*
CF	*Carbon Footprint*
UI	*User Interface*

Abstract (summary report)

My project is called CO_2 Aware, and it consists in developing a mobile solution to raise awareness and reduce people's carbon footprint (CF) in their daily life. This will encourage more sustainable mobility across Dublin City through integrations with the TFI network and the DublinBikes project. The software will allow users to search for itineraries to reach a destination, and suggest "greener" itineraries, for example using public transport, which would lead to a lower carbon footprint. There will be a gamification element to encourage users to race to have the lowest carbon footprint. This will encourage users to take more eco-friendly itineraries during their daily commutes and travels through competitions with friends and relatives on CF achievements. It will feature a backend component to store user data, as well as numerous integrations for various public transport services such as buses and trains within Co. Dublin. Future releases and updates to the app may extend this to the major national routes operated by TFI operator Buseireann, but this is not a core requirement for the first release of the software.

Core Specifications

The objective of this project is to create a mobile app (CO_2 Aware) that provides users suggestions about how to reach specific destinations using different modes of transportation (e.g., car, Uber, bike, walking) including additional information, such as calories that will be consumed, the money spent (if public transport is used), and the CF introduced using each mode of transportation.

Users can also record their commute activity and obtain reward/penalty points depending on the CF introduced. Users will also receive periodical (daily or weekly) feedback indicating their progress in terms of money saving, CF introduced and calories consumed. Notifications, as well as a dedicated “tips” section, will provide tips to improve the user's CF. Finally, users can also add friends and create competitions to identify who introduces the minimum CF during a given period of time.

Many of the APIs I had foreseen to use I realised were deprecated and no longer functional, so it was not entirely possible to adhere to these specifications.

1. Introduction

There is an increased focus on reducing people's everyday CF impact, especially with recent climate changes and the alarmism created around this topic. Alarming evidence is being shown to us by the media each day, urging us to take action. This can be overwhelming for many, as individuals often feel powerless in light of the catastrophic events brought about by climate change. Climate change, as suggested by many papers, does not only come in the form of temperature increase, but also through "human population growth, tree cover loss, fertility rates, fossil fuel subsidies, glacier thickness, and frequency of extreme weather events. All are linked to climate change". This has been observed in the past 40 years at least, and now is the time to act.

Scientists are obliged to warn people about these alarming phenomena, and act quickly in order to rectify this trend. Possible solutions researchers have proposed involve the implementation of drastic energy efficiency and conservation practices, and the replacement of fossil fuels with greener, low-carbon renewables, as well as the protection of wild endangered species and habitats. The increasing human population has also an attributed effect on this process, as the more people there are, the more resources are going to be used to feed, house and cater for them. This will be a major challenge not only for our generation, but also for generations to come, as we try to halt this process in the best way we can.

My project consists of building an Android application through which users will be able to see where they can actively reduce their CF impact through small everyday actions, as well as compete for the lowest CF in a set period of time. The gamification aspect of the app comes in when the app encourages you to make a greener journey to your workplace for example. It will award you badges on your profile for when you're successful, and will put you against your friends to draw out your competitive edge and encourage you to be even greener everyday!

I will be implementing the application in Android Native language Java and I will be making use of the Dublin Bike API, which is capable of providing real-time statistics on availability and location of available Dublin Bikes. By obtaining the user's approximate location through their device's GPS chip, I will be able to suggest nearby Bike Stations with available bikes, or itineraries to Stations that are further away reachable by other means of public transport, such as bus and tram.

The API specifications are available on the Open Data Portal of Dublin City[1], and can be used without requiring any license under certain terms and conditions, which I will investigate as they vary based on the individual services required and the amount of queries made per day. The data is provided as-is and no guarantee of accuracy is made by Dublin City Council or Open Data Dublin, so I will be wary of the possible presence of outlier value and incorrect data in these dynamically generated, real-time datasets. These APIs were found to be deprecated/non-functional, and therefore were not

[1]SmartDublin data portal: https://data.smartdublin.ie/

used in the development process. This was agreed with my supervisor, Dr. Pasquale as these issues were being discovered.

I have nonetheless integrated many features into a useful application that should encourage commuters to take greener forms of public transport in their day-to-day movements across Dublin City.

2. Related Work and Ideas

I researched some similar apps aimed to reduce CF on the Google Play App Store, to have an idea of what has already been done and what I could do that is a little different. I identified 3 main products on the Google Play Store, and one on the Itunes Store for iOS devices only. Please note that I did not have a chance to explore the iOS App Market as I am not in possession of an Apple mobile device. In the rest of this section I summarize contributions and limitations of these apps which I identified after installing and using the apps for a short period of time.

Carbon Footprint (AOSSIE) - Preview[2]

The app is the most similar to what I will achieve in this project. The app aims to automatically detect different ways of transport (via GPS/A-GPS) to reach a destination. , For each suggested way of transport it calculates distance to be travelled and the CO2 emissions and relative Carbon Footprint.

Upon installing the app, the user is required to sign-in through some social integrations (Google Account/Facebook) or email. It is not possible to currently use it as an unregistered guest. There looks to be a bug, in the sign in screen, since none of the buttons allow you to sign-in with any of the social networks, and only the sign-in via email works correctly.

The app also features a lightweight "journey planner" feature, as shown in Figure 1. However, this does not seem to be working, at least in Ireland. The "Activity" screen is meant to detect the activity automatically, however this only occurs when the screen is active and if another activity (such as an SMS message) comes in, the tracking stops and all counters reset as if you had stopped the current journey and started a new one.

The app also includes a "Friends" section, where users can invite contacts via their email address and compete with them for the smallest CO_2 footprint. Currently, it is only possible to invite and add friends email. I believe more integration with social media would be necessary to add friends to the app, as well as to share your achievements with them on, for example, Facebook or Twitter.

[2] https://play.google.com/store/apps/details?id=com.carbonfootprint

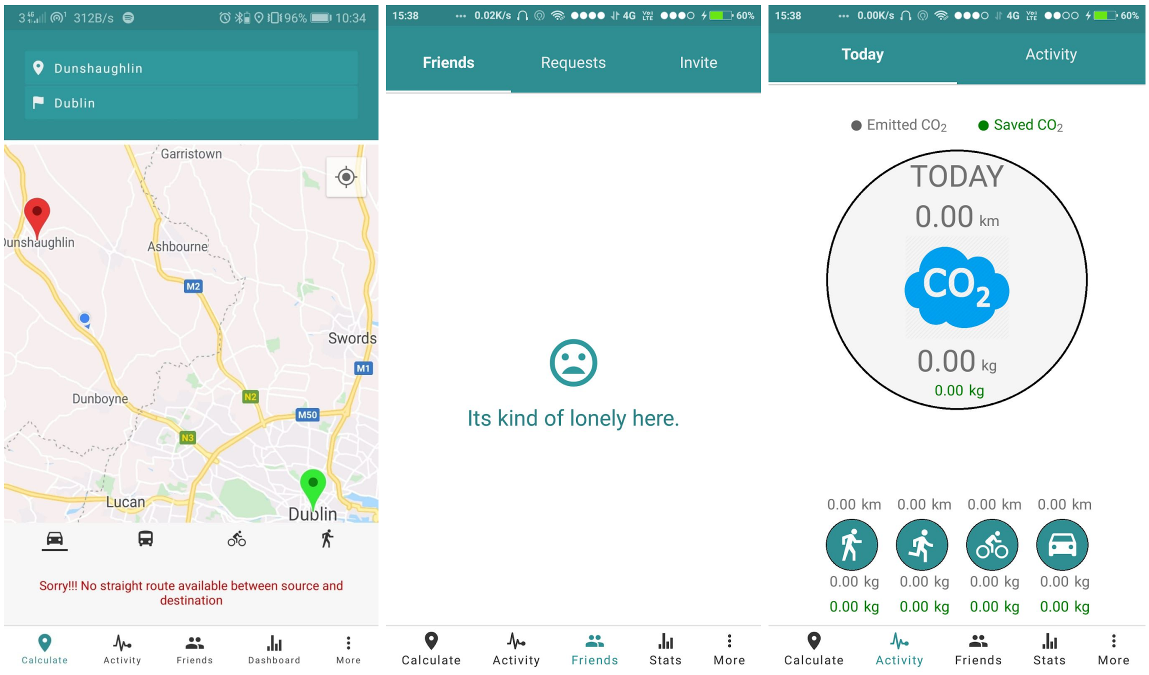


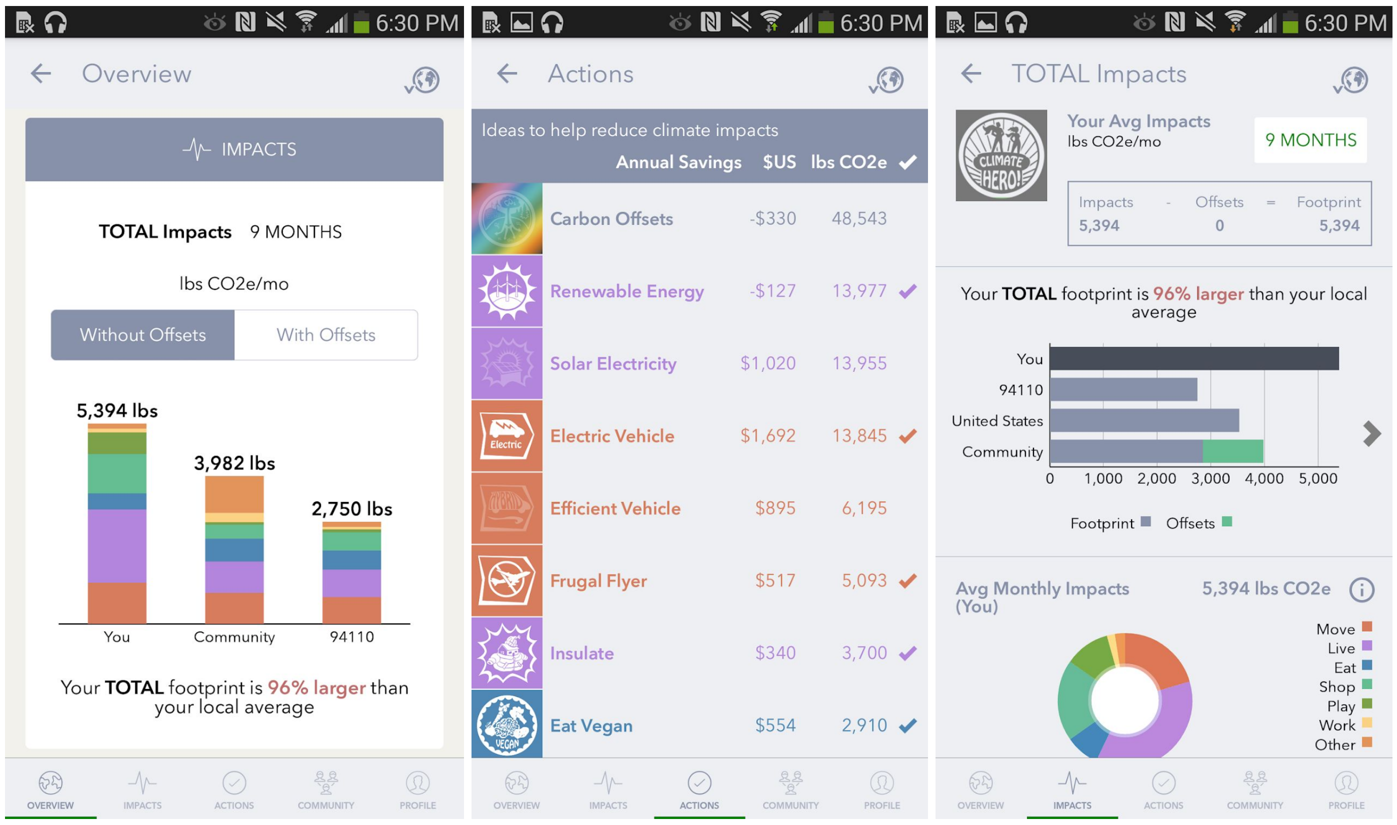
Figure 1: The Carbon Footprint Application. Screenshots taken on my device here in Ireland and from the app's Google Play Store page.

Oroeco[3]

Oroeco tries to sensitize communities and individuals to reduce their CF by comparing a user's personal impact to the average impact of the local community, celebrities, friends and family. This encourages mass-driven change in entire communities across the world.

People are encouraged to sign up through a series of giveaways (e.g., tech gadgets) to increase sign-ups to the app (and in turn awareness of emissions) and encourage more eco-friendly behaviour in their everyday life. If the user shares the app with other people and they sign up, their chances of winning the prizes increase. For example, this month they are giving away a NEST smart thermostat for homes. This not only encourages people to sign up and share the app, but also serves the useful purpose of enabling people to save energy by installing the item in their homes.

[3] https://play.google.com/store/apps/details?id=com.oroeco.oroeco&hl=en_IE



Ideas to help reduce climate impacts

Annual Savings	$US	lbs CO2e	✓
Carbon Offsets	-$330	48,543	
Renewable Energy	-$127	13,977	✓
Solar Electricity	$1,020	13,955	
Electric Vehicle	$1,692	13,845	✓
Efficient Vehicle	$895	6,195	
Frugal Flyer	$517	5,093	✓
Insulate	$340	3,700	✓
Eat Vegan	$554	2,910	✓

Figure 2: The Oroeco Application. Source: play.google.com

As the figure above shows, the app features a colourful and fun design. It provides graphs to show your emissions relative to those of neighbouring users in the initial “Overview” screen. These are expanded and presented in more detail in the “Impacts” screen. In the “Actions” screen, the app proposes some eco-friendly actions they can take, along with their cost and the amount of CO_2 that could be saved by taking those actions in your daily routine, the user can take to reduce their Carbon Footprint.

GoCarma Carpooling[4]

GoCarma provides HOV (High-Occupancy Vehicle) Passes to carpooling drivers to reward them of their carpooling initiatives. GoCarma initially sends a HOV Pass (“a small Bluetooth device” that you can “place in the glove box of your car” according to the description on Google Play Store) to the driver in exchange for their registration on the platform. The passengers taking the ride are then required to activate the app when in the car, so the HOV Pass is activated and the compensation calculated and paid to the driver.

A summarising table is given on the next page.

[4] Not personally tested as the app requires users to purchase a High Occupancy Vehicle pass to avail of functionality.

App Name	Functionality	Usability	Complexity
Carbon Footprint (AOSSIE)	✗ Rich in functionalities, some don't work too well as the software is still in preview.	✓ Simple design of the UI. Simple, but effective. ✗ Graphic effects could be improved as they feel like they were designed for older devices.	✗ Friend section is not very clear, would be nice if user could invite/add friends from phone address book/social media, instead of just by email or ID
GoCarma Carpooling	✓ Very rich in functionality, using GPS to retrieve user location and connecting via bluetooth to car HOV pass.	Not tested as not usable without GoCarma HOV pass.	Not tested as not usable without GoCarma HOV pass.
Oroeco	✓ Nice gamification concept, with badges and points as you complete eco-friendly challenges.	✓ Clear UI, plenty of features mainly aimed at the Australian market	✓ Similar concept to the first app. Oroeco, however, is more focused on building a sense of community and gamification of the solution to the C.F. problem

3. Data Considerations

In this project I will not make use of any external data sets, I will only be using publicly available APIs to connect my app to services such as Dublin Bike in order to gather real-time data. There are a few map data providers; some of them are open source, some are licensed. I have decided to use Google Maps Engine, as they offer good integration and unlimited free calls to their api. The other main provider that offers high quality map services globally, is OpenStreetMaps, but this was more difficult to implement due to the lacking documentation on Android Integration. Both of these offer extensive APIs publicly available for developers to use compatible with many different platforms, so OpenStreetMaps would also have been a viable option for my goal.

I have opted to avoid using a backend infrastructure, as they would increase the running cost of the project. Also, user data is constantly being backed up to the user's Google Account since Android 8.1 Oreo update, so this removes the issue of backing up the data myself to external servers and handling all the related privacy and security issues.

This is to reduce the potential for network coverage issues to interfere with the application's functioning. I have not yet defined clearly the frequency at which I will gather the data, as I will have to evaluate the power consumption of an android device's GPS sensor to tune it so as to not overuse the battery, while maintaining a suitable level of accuracy. The user can choose what level of accuracy to provide the app's geolocation feature by selecting the appropriate value on their device through the OS settings menu. Usually, the options available here consist of "High Accuracy", "Battery Saving" or "Disabled" on most devices, although this can vary across different producers and vendors..

Data "cleaning" was not necessary, and I was able to use pretty much all of the data as it came through from the API without much modification to its structure. The accuracy of the data and its validity depends on the accuracy of the sensor(s) in the user's device (eg. GPS) and the API's owners which is outside of my control,.

Initially, I had planned to have a beta user group, however due to the ongoing COVID-19 Crisis and the introduction of the resulting travel restrictions, I have been unable to do so.

4. An Outline of My Approach

I will be using some external open-source libraries to help me with gathering the data and parsing it to display it to the user in a nice, user-friendly manner. To transfer data, I will be using an open-source library for Java HTTP(S) requests known as Retrofit2[5] that allow devices to exchange data over a network. I have parsed the JSON output of the API using Google's Gson library, a JSON serialization-deserialization utility designed for use on the Android Platform. These two libraries integrate well with each other and have been proven to work efficiently in other implementations in the past, which made them the obvious choice.

The diagram below outlines the data exchanges that will take place between the various components on an abstracted view. It shows what data is transferred between client (frontend) and server (backend).

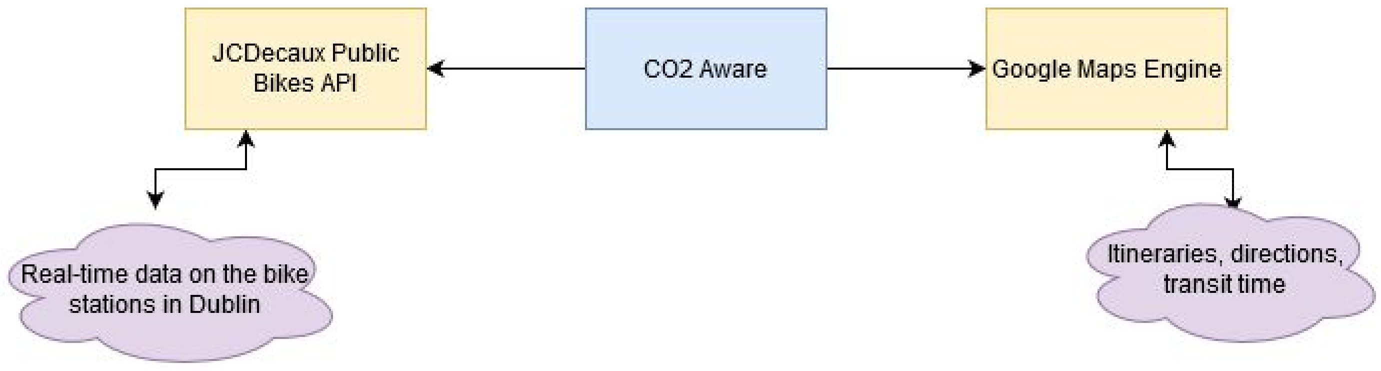
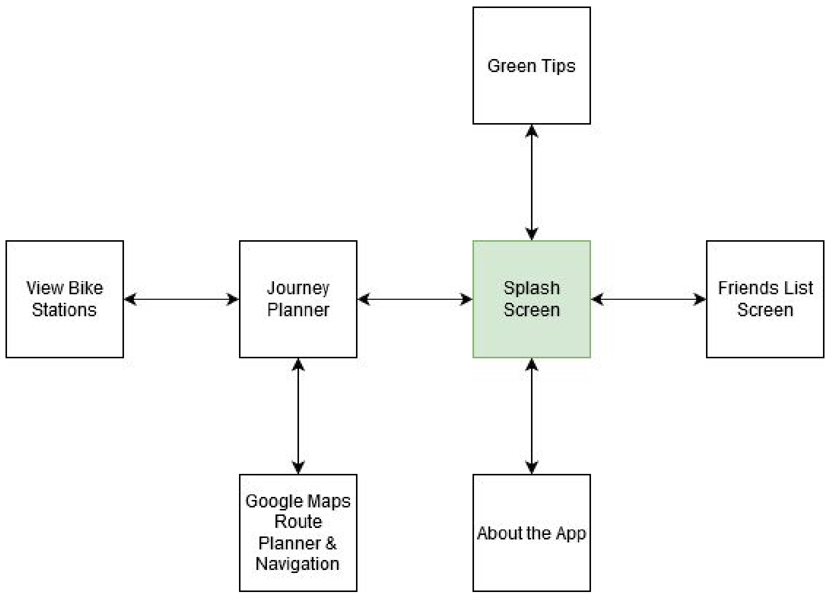

JCDecaux Public Bikes API
CO2 Aware
Google Maps Engine
Real-time data on the bike stations in Dublin
Itineraries, directions, transit time

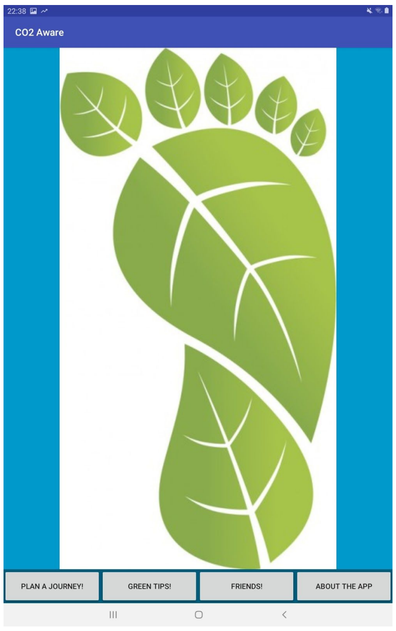
Figure 3: Diagramatic representation of the application's data flow. Yellow rectangles represent the remote services on which the app relies to function correctly. Purple bubbles represent the information obtained from each service.

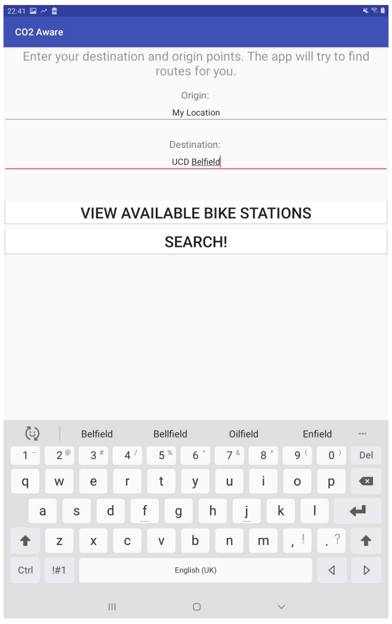
[5] Retrofit2: A type-safe HTTP client for *Android* and Java. © Square Open Source

Screen workflow



The diagram aside shows the interactions for when the user launches the application on their Android device. The application will initially show the splash screen (Figure 4.1 below) which will contain the buttons to reach the other parts of the app.



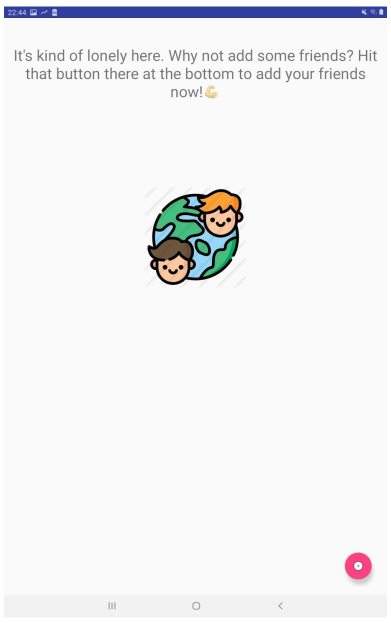


CO2 Aware

Enter your destination and origin points. The app will try to find routes for you.

Origin:
My Location

Destination:
UCD Belfield

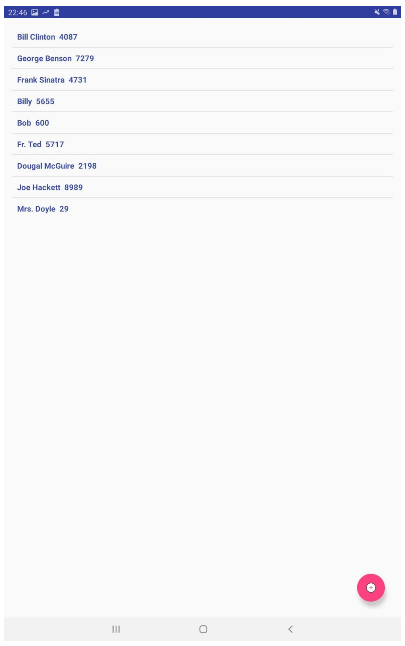


It's kind of lonely here. Why not add some friends? Hit that button there at the bottom to add your friends now!

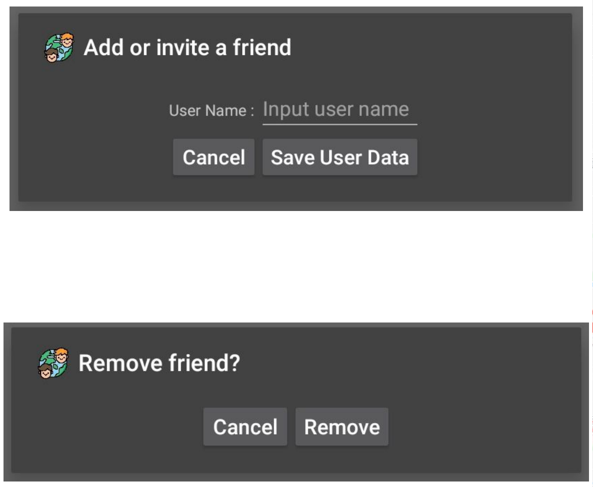
1

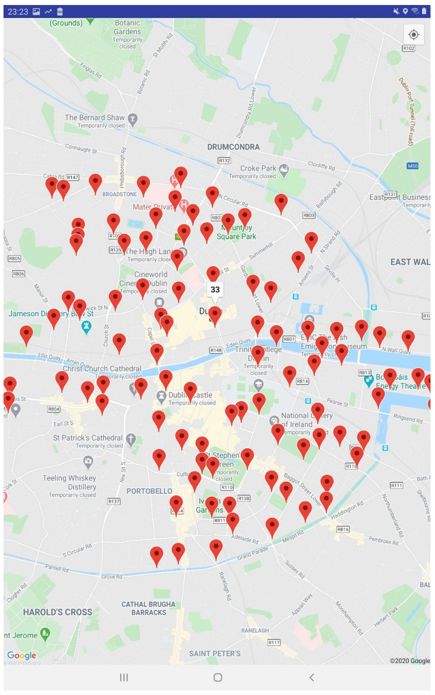
2

3



Bill Clinton 4087
George Benson 7279
Frank Sinatra 4731
Billy 5655
Bob 600
Fr. Ted 5717
Dougal McGuire 2198
Joe Hackett 8989
Mrs. Doyle 29





4 5 6

Figure 5: Mock-ups for my UI screens

Across the bottom of the screen will be a navigation bar to point to other screens offered to the user. This element will be present throughout the various UI screens, so users won't have to interact with the Android system buttons to navigate it.

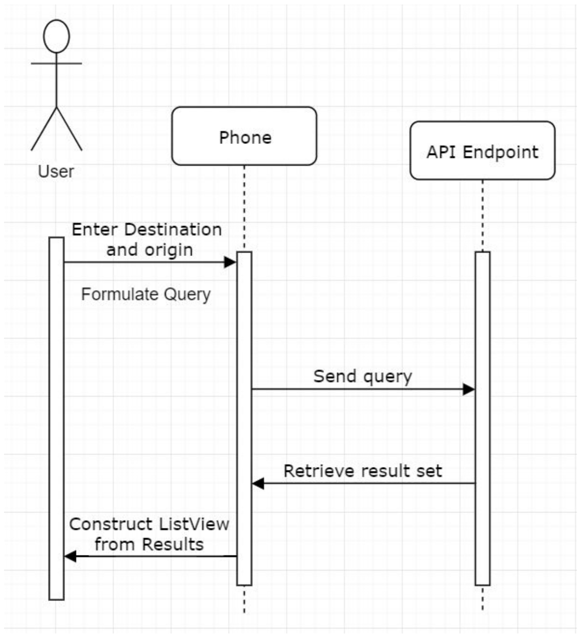


Figure 6: API Call structure

Figure 2 (page 13) shows the "transport planner" section, where the user will have the option of choosing starting point and destination. Due to the restriction imposed by Google in their Maps API Terms and Conditions, I am forced to redirect the user to the Google Maps Application for route planning, or alternatively show a webview of Google Maps in the device's browser. Fig. 6 shows the general structure for my API requests.

Figure 5.3 shows the Friends screen when no friends have been added by the user. Once they begin adding them, the list view is populated and appears similar to Figure 5.4.

Figure 5.5 shows the pop-ups the user is presented when they add and remove a friend from the list. Friends are added and removed from an ArrayList of String elements, which are committed to an instance of SharedPreferenceManager to allow them to persist across reboots of the app. In this way, the friends list is not erased when the app is quit or when the device is rebooted. Should the list be longer than the screen size, it is automatically made scrollable thanks to a ScrollView element implemented in the Activity's layout XML file.

Figure 5.6 shows my representation of the bike stations around Dublin City Center. The markers shown on the map only appear when there are bikes available at the bike station and are dynamically retrieved from the JCDecaux API Endpoint for the city of Dublin. Tapping on one of the markers retrieves the marker's number and shows it in the small text box that appears.

A section titled "Green Tips" contains an article extracted from www.conserve-energy-future.com on 40 ways to go green. This is static content stored inside the app, so that it is always available for people to view even in case of lack of connectivity to Internet. Similarly to the Friends screen, this section of the app also implemented a ScrollView element, to facilitate reading the text.

The final screen is the "About the App" Activity. This contains a brief account of the motivation behind such an app, as well as references to some of the technologies I have relied upon to build the application.

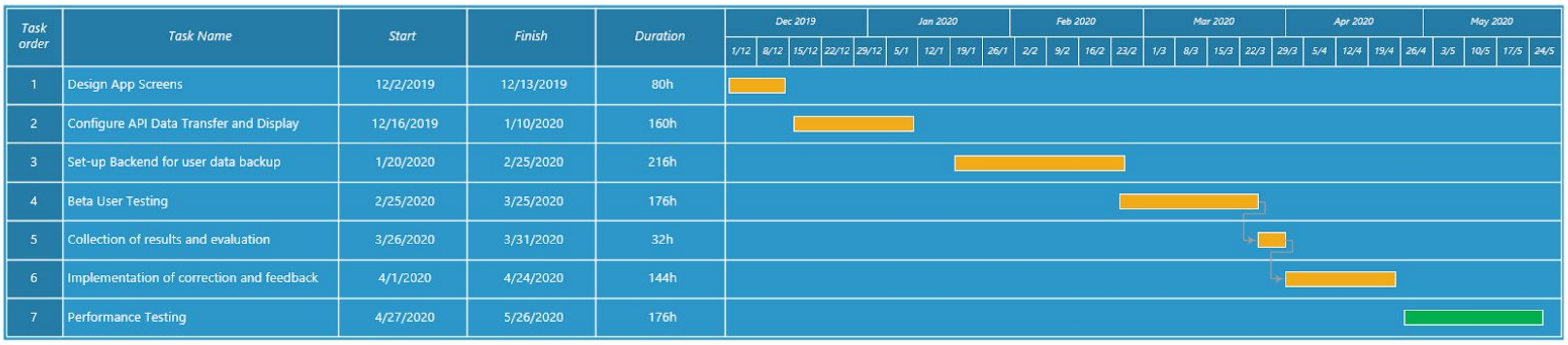
I felt it was not necessary to provide "guiding overlays" for users of the app, as it is fairly self-explanatory what each section does and how to use it.

User Testing of my App

This was not carried out due to the ongoing COVID-19 Pandemic Crisis and advice from Authorities to avoid all unnecessary movement using public and private transport means.

5. Project Work Plan

Gantt Chart – Project Breakdown and Deadlines



Task order	Task Name	Start	Finish	Duration
1	Design App Screens	12/2/2019	12/13/2019	80h
2	Configure API Data Transfer and Display	12/16/2019	1/10/2020	160h
3	Set-up Backend for user data backup	1/20/2020	2/25/2020	216h
4	Beta User Testing	2/25/2020	3/25/2020	176h
5	Collection of results and evaluation	3/26/2020	3/31/2020	32h
6	Implementation of correction and feedback	4/1/2020	4/24/2020	144h
7	Performance Testing	4/27/2020	5/26/2020	176h

Figure 7: Original Gantt chart showing expected project workflow. Completion date (Estimated): May 2020

The above Gantt chart shows my estimated progression in the project subtasks. I had originally identified seven major "steps" to bring the project to completion. I will start off by designing the application's screens in Android Studio, before progressing to populate them with data from the aforementioned API. This will allow me to see limitations in space of the Android user interface and adapt accordingly. The next stage will be setting up a connection to the API and be able to pull down data from them, interpret it and display it in a user friendly manner. I have allocated about a month for this as this kind of task can sometimes pose numerous challenges in cases where the API is not properly documented, which can happen especially with "stale" deprecated APIs.

A small set of unit tests have been designed initially to ensure that my buttons were working correctly in the SplashScreen, as I had some issues at first with them not showing on the UI correctly.

The fact that Android 8.1 and above automatically back up user data in case of uninstall was very useful to me, and helped me save time and money on a potentially expensive backend infrastructure.

6. Application Design and Implementation

The design of the application deviated slightly from the original proposed layout outlined in Section 4. This was mainly due to the APIs becoming unavailable as the project was progressing. Therefore, some changes were implemented in order to make the available content fit in more naturally into the screens.

One of the first things I had to change was the home screen. I had envisaged to track how people moved around, but upon a discussion with friends and family, some concerns emerged over the use of these data and their storage. This was why I decided to opt for a much simpler welcome screen without any graphs calculated by external sources.

The “Friend List” screen was kept almost identical to the original design. I used a random number generator script here to give users random scores due to the ongoing pandemic which made it once again difficult to test the app. If a user enters an email address (or even a name with the “@” symbol in it), the application automatically switches to “invite” mode. This shows the email address with “(requested)” beside it, and no score as the user is supposedly a new user to the app. The app then generates a Snackbar to inform the user that an email has been sent to the specified email address. This is simulated, as the project specification did not include the creation of an “invitation” mechanism with links to join the app sent by email.

I also changed the proposed “Settings” activity into an “About the app” activity. This screen now contains the credits and attributions section, as well as a small introduction and my motivation for building such an application.

Another important change in my application’s layout was found while I was implementing the navigation section of the app, where the user inputs their origin and destination points and selects their desired means of transport. I came across a section of Google Maps’ terms and conditions of use which stated that making replicas of copies of Google Maps, partly or in whole, was unauthorized[6].I verified and examples of this include implementation of navigation aids that rely on Google Maps APIs. Their suggested approach to obtaining a similar result was using Intents to launch (if installed on the device) the Google Maps application, or alternatively the respective web view. Since intents are also capable of transmitting data between apps, it was also possible to provide Google Maps with all the required parameters to trace the itinerary such as destination, origin and mode of transportation. I am gathering all of these information in the ChooseDestinationActivity through a series of EditText text boxes and buttons, so I am simply extracting the parameters from those in text form, constructing the Maps intent and finally executing it. The Android subsystem is in charge of handling that intent at this point, and in case the Maps application is not installed, the intent launches in the web view.

A screen with the public bike stations available has been implemented and shows available bike stations (ie ones with available bikes or parking), through a call to the JCDecaux API.

During the course of the implementation of the project, I became aware of the fact that many of the public transport API I had envisaged to use had been deprecated and/or shut down. This created the need to rethink the core layout/concept of the idea, and I had to spend some time looking for alternatives. I finally settled for some widely accessible Google Maps APIs such as the “Directions API”. All these APIs are publicly available and free to use thanks to UCD’s agreement with Google for their development tools. It was therefore a natural choice to go with Google Maps and associated APIs for my app’s intended use.

[6] 3.2.3 Restrictions Against Misusing the Services.- Section (d): No Re-Creating Google Products or Features. https://cloud.google.com/maps-platform/terms

Google Maps has a very interesting Android SDK, which with just a few lines of code allowed me to generate an Activity with a Map and the markers for all of the bike stations around Dublin City. I had originally envisaged to maintain the geolocation coordinates local to try to maintain the data usage on my app to a minimum. However, I soon realised it wasn't making a huge impact (the difference was a mere 13kb more per request), and that I was anyway forced to download the entire JSON response from the server each time. I therefore opted to use the dynamic data pulled down from the server each time, rather than storing anything locally. This also makes the app more future proof, in case the number of bike stations were to change, the app is able to adapt and still show the expected results. This also helps to ensure that the app does not misbehave or crash in case of corrupt data received from the API endpoint, which can happen in case the phones use unreliable internet connections.

I used Retrofit2[7] to carry out these API calls, and parsed the results with Gson plugin by Google[8] for Android. The main advantage of these two libraries is their ease of integration and simplicity of use, which made them my number one choice in this case. I had never used Retrofit2, but I had used the previous version of that library in another personal project, so I was quite familiar with the integration process and how to make a call.

Due to the current pandemic situation, and the consequent lockdown, I was not able to carry out the user beta testing plan contained in the previous version of this report. Instead, I wrote some unit tests and created a script to run them at every commit through TravisCI (https://www.travis-ci.com), an online code-checking and auto-deployment SaaS service. I had already used this service in the past, so I was familiar with the process of setting it up and working with it, so I went with this provider over others. They also offered a nice deal with my GitHub premium account, which was obviously a plus. A travisCI status "badge" is available on the GitHub Repository README.MD file. This contains an updated status of compile/testing errors on the repository, which update at every commit on the master branch.

My initial plan to use OpenStreetMaps as a map engine turned out to be more difficult than expected, so I turned to Google Maps instead. Android is owned and maintained by Google, and so is Maps. The two services can be connected and used together fairly easily, as most of the code required for the integration is boilerplated from the SDK on generation of the classes.

The fact that I had to test the application's functionality myself, instead of having a Beta User group doing it, took away a few weeks, during which I had originally planned to carry out the backend construction work. I also asked some of my friends shortly after the pandemic began, looking for people's privacy concerns regarding storing their data on remote servers. What I found was that people aren't so keen to change their devices, and are nowadays very worried about where their

[7] RetroFit2: "A type-safe HTTP client for Android and Java" - https://square.github.io/retrofit/
[8] Google Gson: "A Java serialization/deserialization library to convert Java Objects into JSON and back" https://github.com/google/gson

data resides. They would rather not use an app if they did not trust it fully. This is why I opted to have more content on the app, and not to focus on the backend architecture work. This would have also entailed a lot of work on my side, and since I had assignments and other coursework to carry out as well, I decided to keep everything on the user’s device. Most people, however, do not even back up their contacts when changing devices, so I did not see this as a major concern.

8. Further Developments

The layout choice I adopted when changing the originally proposed “Setting” activity into an “About” screen is more in line with most apps, and is a good implementation in case the app was to go cross-platform. In fact, iOS apps do not include a settings menu in-app, but transfer the user over to the OS Settings menu, where entries containing the settings for each application are available. I see the idea of porting to Apple iOS and targeting other device operating systems as an interesting move in the future, in case this application was to gain success. In fact, almost half of the devices currently on the market are non-Android, so the likelihood that someone’s friend might own an iOS-powered device and want to use this app is high. It is therefore worthwhile to try to keep the app’s layout as consistent as possible between the two vendors’ operating systems.

I will be continuing development of this application in a separate GitHub Repository than the one used for submission, in order to perfection and continue to work on the skills I gained during the course of the project.

9. Summary and Conclusions

I do not believe my project raises any major ethical concerns and hence I applied for the *low risk* ethical research exception to UCD Research Ethics Committee as I had oriinally planned to run user testing on the application. This has obviously not been possible, due to the COVID-19 pandemic and the lockdown it caused worldwide. Therefore, I was not able to run these beta tests on my app, although I implemented code-checks to automatically run on every commit to validate my code against standards. I also had inputs from numerous experts in the sector among my colleagues in Oneview Healthcare Plc, who highlighted some important issues in the usability of the application in the early phases of the development lifecycle.

Acknowledgements

The work presented here is entirely my own, apart from the screenshots taken from the different apps in the “Related Work and Ideas” section, which I have referenced below every description with relevant footnotes.

I wish to thank my Supervisor, Dr. Liliana Pasquale, as well as my friend, Megha, for helping me write this document and for their precious suggestions on various aspects of the project, which I would probably never have thought about on my own.

A word of thanks also to Martin Digulio from VHI Healthcare, for suggesting many times useful optimisations to improve user experience in the app.

Furthermore, I wish to acknowledge the contribution of my esteemed colleagues at Oneview Healthcare, for their continued support and encouragement, as well as Marina and Bruno, my parents, who always supported me in this journey.

References

- Newsome, T. and Ripple, W. (2019). *11,000 scientists warn: climate change isn't just about temperature*. [online] The Conversation. Available at: http://theconversation.com/11-000-scientists-warn-climate-change-isnt-just-about-temperature-126261.
- William J Ripple, Christopher Wolf, Thomas M Newsome, Phoebe Barnard, William R Moomaw (2019). *World Scientists' Warning of a Climate Emergency*, *BioScience*, , biz088, https://doi.org/10.1093/biosci/biz088
- All screenshots from existing products were taken from the Google Play Store pages.
- App Mockups designed with Balsamiq Mockups 3.5.17
- Sequence Diagrams deigned with Draw.io Free version
- Gantt Chart design: Microsoft Office Visio 365