

South East Technological University

-

Ollscoil Teicneolaíochta an Oirdheiscirt



Fleet Management Project

Research Manual

---

Student Name: Rachel Doogue

Student Number: C00237335

Supervisor: Dr. Joseph Kehoe

Academic Year: 2022/2023

---

<b>Abstract</b>	<b>2</b>
<b>Introduction</b>	<b>3</b>
<b>Web Hosting Services</b>	<b>4</b>
In-House Server	4
Cloud Hosting	5
Conclusion	5
<b>Database Languages</b>	<b>6</b>
SQL	6
NoSQL	6
Conclusion	7
<b>SQL Database Management Systems</b>	<b>8</b>
MySQL	8
MariaDB	9
PostgreSQL	9
Conclusion	10
<b>Cross-Platform Frameworks</b>	<b>11</b>
Xamarin	11
Flutter	12
QT	12
Electron	13
Conclusion	14
<b>Development</b>	<b>15</b>
Flutter Environment Set Up	15
Download Flutter	15
Android Studio Plugins	15
Android Studio SDK	16
Android Studio Licence	16
VS Code Extensions	16
Running Flutter on Emulator	17
Create Project	17
Create Virtual Device	17
Run Emulator	18
Flutter and MySQL	19
Flutter and Firebase	20
Conclusion	20
Connect to Firebase	21
Log on to Firebase	21
Install Firebase CLI	21
Install and Run FlutterFire CLI	22
Initialise Firebase	22
Firebase Libraries	22
Dart Code	23

## Abstract

The aim of this research manual was to document the findings and conclusion that came from developing a cross-platform application as a Final Year Student. It documents the research and thought process that went behind each decision made during both the theoretical and practical development of the project. With the focus being primarily on developing the application using Flutter in conjunction with Firebase for backend database support. Providing steps on how to set up the development environment for Flutter and Firebase. Alongside this there will be details on how the application was created. Specifically it will be covering the all important widgets that were used in the creation of the application.

---

## Introduction

As part of the final year project for software students this research manual has been produced in order to document the new knowledge and or skills that have been learnt throughout project development. The project in question is called 'Daly Tyres Fleet Management'. Its purpose is to help a company called Daly Tyres facilitate repair requests for when vehicles break down.

For this project, research was done first by looking into what I believe were the four main factors of development that I would encounter, web hosting service, database languages, the system for managing that database and finally the best cross-platform frameworks. This part of the research was mostly theory and was used as a starting base for development.

From there the manual will heavily focus on Flutter, Dart and Firebase. As these three technologies became the main tools used for development.

---

## Web Hosting Services

With multiple different users needing access to the fleet management system, all from varying locations. In order to facilitate this the system will need to be hosted online. There are many different options available for this service but for the clients needs it will only be necessary to look at an in-house server and at cloud hosting. Here we will compare and contrast these two services to determine which service will offer the most effective service for the client.



### In-House Server

To begin with, a server is a computer that is outfitted with computer hardware or software or both in order to provide services to other computers on the same network [1]. This is a very broad description and there are many different types of servers. For the purposes of this project a web or HTTP server is necessary. This type of server is purpose built to store any files that are needed to build a website [2]. It can also connect to the internet and carry out HTTP requests [2]. Now with the server type clarified it is time to move onto the pros and cons of using such a piece of equipment in-house.

On the pro side of having the server in-house, the client will have much more control over the server when compared to any other option. No one else will have any access to this server. Control is fully within the clients hands. They will have control of the hardware, meaning that they can tailor it to their needs [3]. They will have control of the software, meaning that they can use their preferred operating environment [3]. They won't be beholden to any contracts, unlike if they were to use another company's services [3]. This control is by far and away the single greatest benefit of an in-house server.

However, with this level of control comes many responsibilities. To start with the client will need to expense the cost of buying the server equipment themselves. From there then the client will need the technical skills and knowledge in the set up, maintenance and security of the system [3]. With support needing to be available around the clock as users of the system will need access at any possible time, day or night [3]. As vehicle maintenance or repairs can hardly ever be considered to be a convenient occurrence.

---

## Cloud Hosting

On the other end of the web hosting spectrum there is cloud hosting. This method, like before, uses a web server to make the client's system accessible by the internet. However, unlike before this server is remote [4]. Meaning the is located in a different location from the main business. These locations are called data centres [4]. Here the client will need to rent space on another company's server [4]. They will not own the server and they will not be able to access it physically themselves.

There are many benefits to this arrangement. The equipment is already bought and operating. This means that not only will the client not need to front the cost of such expensive equipment [5], they also won't need any prior knowledge on how to operate a server. It is already established [5]. All the client has to worry about is creating the files for the website. Paying for only what they use. Additionally, as these data centres are specifically built for this function and specialise in this field, they are much more likely able to provide services at a higher level than a regular non-tech business could. With services such as the backup of data in case of corruption or loss being a given [5].

All of these pros however, come with a lack of control. The hosting service gets to decide how they administer their servers. Largely this issue can be worked around as the client can examine different service providers to find one that suits best. Even then the issue of downtime is inescapable [5]. If there are any technical issues causing the server to be unavailable the client will be at mercy of the service provider. Potentially causing the client and other users great difficulty. There is also the fact that data centres are a major target for hackers as multiple different companies will be using that service [5]. Meaning that even if Daly Tyres isn't the target of such an attack, their data stored there could be swept up in an attack on another company entirely [5].

## Conclusion

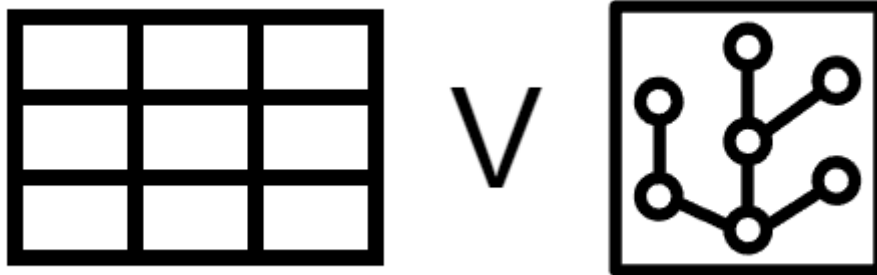
Taking into consideration all the pros and cons of each service it is clear to me that cloud hosting is the much better option. Focusing on the client, Daly Tyres, needs it is unlikely that they have much if any prior experience with operating a private server before. This coupled with the expense of purchasing such equipment to set up an in-house server would make this a costly endeavour. Both in terms of time and resources. Any potential control gained by an in-house service would be lost in a lack of experience in how to utilise it to its fullest for the business. While the concerns that come with cloud hosting will still pose an issue for the client, the high service standards, overall reliability and cost effectiveness make for very attractive features.

On the development side of the project cloud hosting is much more beneficial for me. It will enable me to launch the application sooner as there will be no need to set up a server and in the long run I won't have to be burdened with maintaining the server myself. This should allow for a smoother application launch.

---

## Database Languages

It is already a certainty that a database is needed. All that is needed for consideration here is the database language that will be used. Database languages are programming languages that have been specifically created to be used to access, control and manipulate a database [6]. With SQL and NoSQL being two of the more widely known database languages, they will be the two taken into consideration in the following section.



### SQL

Standing for Structured Query Language [7], SQL as the name suggests operates based on a set structure. It can only be applied to databases whose tables conform to its fixed structure. This applies to how tables are connected, tables consisting of rows and columns only or how items are labelled [8]. This strict formatting means that there is a high level of consistency throughout queries. Making it easier to understand between individuals and long term maintainability. Along with this set structure SQL is also considered to be easier to understand than other database languages [8]. Not needing someone that is highly specialised in the language used once again lends itself to its maintainability across individuals.

At the same time having a language that is so reliant on a set structure has its drawbacks. Any changes to the structure of the database can heavily affect any SQL queries already written and make it unusable [8]. This is also what makes it difficult to scale [9]. As traditionally SQL databases have been scaled vertically. Otherwise, it becomes a very difficult task to scale horizontally.

### NoSQL

Taking the opposite approach NoSQL, standing for Not Only SQL [10], does not rely solely on tables consisting of rows and columns. Instead it can store data in many different forms. One of them being document form [10]. This level of flexibility can be a significant assist in creating a highly customised database. Another benefit of NoSQL's lack of a set structure is that it helps it scale horizontally [10]. As data does not need to all be on a single machine.

---

Flexibility does have its drawbacks. NoSQL requires a high degree of effort and time to understand. Accompanied by the fact that the NoSQL community is not as large as the likes of SQL [8] owing to its newer creation status, makes NoSQL somewhat of a less reliable language. As finding others to maintain or expand the database will take more time and is a slightly more specialised language in comparison to SQL.

## Conclusion

Considering that the data that Daly Tyres is going to be handling is of a benign nature the flexibility that NoSQL affords does not seem necessary. Instead the structure of SQL is highly appealing. Especially as once completed and control fully handed over it should be easier for the client to find someone to maintain or even update the system. It is also not evident at this moment that horizontal scaling would be preferably over vertical scaling and vice versa. Given these facts, SQL seems a much more suitable database language in comparison to NoSQL.

This practicality helps save time, as this focus will ensure that very little time is spent on learning tasks that unlimitedly won't be needed. Being able to insert documents into the database may be a nice feature but it would only ever be implemented if there was time to spare and that would be unlikely. Thus the benefits of NoSQL would be unlikely to be realised.

For me, SQL is a much more familiar language. The added complexity and size of the database that will be required should ensure a deeper level of knowledge of SQL will be needed. While having an added benefit of not needing to learn a new system structure with NoSQL.



---

## SQL Database Management Systems

Now that a database language has been decided upon, the next step is to examine a number of different SQL database management systems. These services are used to create, store, access, maintain and update relational databases [11]. While all should offer the same basic function, it is the mechanics that will set these systems apart.

### MySQL



*Figure 1: MySQL logo [22].*

An open-source database, MySQL has been operating for 25 years [12] and has 5 million active installations [12], making it the most popular in the world. It is free and operates under a GNU General Public License [12]. All of that to say that anyone that so wishes can operate, examine, and modify the software freely [13]. With it supporting numerous popular programming languages such as Java, Python, C, C++, and many more [12]. Given its longevity in conjunction with such a large active user base, MySQL can certainly claim to have a wide ranging support system and documentation [12]. Making it unlikely to run into any problem that cannot be solved in a timely manner.

With its single greatest downfall being that it struggles when dealing with databases of substantial size [14], MySQL offers many upsides.

---

## MariaDB



*Figure 2: MariaDB logo [23].*

Similarly to MySQL, MariaDB is free and operates under a GNU General Public License [15]. In fact MariaDB has been developed from MySQL [15]. This means that it shares many of the features. Including support for all the previously mentioned programming languages [16].

Even though MariaDB was created from MySQL this does not mean that the two are still compatible. That may once have been the case but now migrating code between the two now requires work to be done on the code in order to get it to function [17]. This was once a significant advantage for MariaDB.

## PostgreSQL



*Figure 3: PostgreSQL logo [24].*

Similarly PostgreSQL is a free and open-source database management system [18]. However this is where the system starts to differ from the previous two mentioned. Unlike before PostgreSQL does not use a GNU General Public License. Instead it appears to operate under what it calls the PostgreSQL License [19]. Even then it operates under much the same principles as the GNU General Public License. As it allows for anyone to operate, examine, and modify the software freely [20]. The next big distinction is that PostgreSQL is not just a relational database, it is an object-relational database [18]. This means that the databases can handle data of a more complex nature as it allows objects to be stored [21]. An example of such data would be multimedia content [21], i.e. images, video, audio, and so on. Otherwise PostgreSQL supports the same languages as mentioned previously [21].

---

## Conclusion

Given everything above, using MySQL would appear to be a reasonable choice to make. At this moment in time the database and the tables included do not appear to need any special features and are fairly straight forward in their structure. This fact along with its long history makes MySQL a reliable option, with plenty of support. Its popularity is a testament to this fact.

With MySQL there are no obvious marks against it, unlike with MariaDB and PostgreSQL. One of the selling points of MariaDB used to be that it was compatible with MySQL. This shows a willingness to make major changes to the platform. Something that does not appeal when taking consistency and maintainability into account. As for PostgreSQL, not so much a mark against it but, the object-relational databases are not needed. Adding a new licence makes PostgreSQL appear more unnecessarily complex compared to the others.

Taking all of this into consideration from a personal perspective, MySQL appears to be a good tool to learn to use. As its widespread use and the fact that it is used as the basis for other similar applications increases the chance of any experience gained being of use after the project is completed. Making it a valuable skill to learn.

---

## Cross-Platform Frameworks

The client has made it clear that they desire a phone app for drivers to access and did not specify when it came to all other users. There is no control of the phone operating systems that the drivers will be using. This means that both Android and iOS will need to be catered for. For this there are a number of cross-platform frameworks that allow for one application to be built. That can then operate on more than one operating system. That is what will be examined in this section. To determine which one is the best suited for this project.

### Xamarin



*Figure 4: Xamarin logo [43].*

As has been seen before, Xamarin is free and open-source [25]. It advertises both tools and libraries that allow for mobile applications to be developed for Android, iOS, and Windows [25] from a single codebase. It boasts of a community of 1.4 million developers [26], with more than 100,000 OSS contributions [25]. This stands for Operational Support System [27] and is used by Xamarin to help maintain their computer network [27].

It lists three different languages C#, F#, and Visual Basic [25]. To get as close to a native application as possible Xamarin can use C#, compile that natively and then .NET is used to implement it to perform cross-platform [25][26]. This in theory should make the mobile application function as close to a native application than it is in reality.

Despite being free and open-source, if any organisation wants to have multiple developers working on Xamarin they will need to purchase a business or enterprise licence for Visual Studio. These licences can get very expensive as an Enterprise Subscription will cost \$250 a month [28]. Fortunately this should not be an issue for this project. It is unlikely now or in the future anything but a simple Individual version will be required. As this project is a solo endeavour.

---

## Flutter



*Figure 5: Flutter logo [44].*

Another free and open-source framework [29]. While Xamarin was a product of Windows [25], Flutter is a product of Google [29]. Once again with Flutter a single codebase can be used to develop an application that can be deployed for Android, iOS, and desktop [29].

Flutter uses the programming language Dart [29]. This is a programming language primarily used for and by Google applications [30] and as such may not be as widely known as other programming languages like C#. This specialisation is definitely a drawback, however the Flutter website has a page on it that has a number of tutorials for developers of different levels with Dart to practise and learn from [31]. Even then Dart is not so new and out there of a language as its syntax is similar enough to Java or C++ to draw recognition [32].

Due to certain features such as the Flutter widgets mobile apps have a tendency to be bulkier than in comparison to native applications. With any application developed on Flutter being a minimum of 4MB [33]. This is less a problem for developers and more so a problem for any users as memory on a phone is more limited than that of desktop.

## QT



*Figure 6: QT logo [45].*

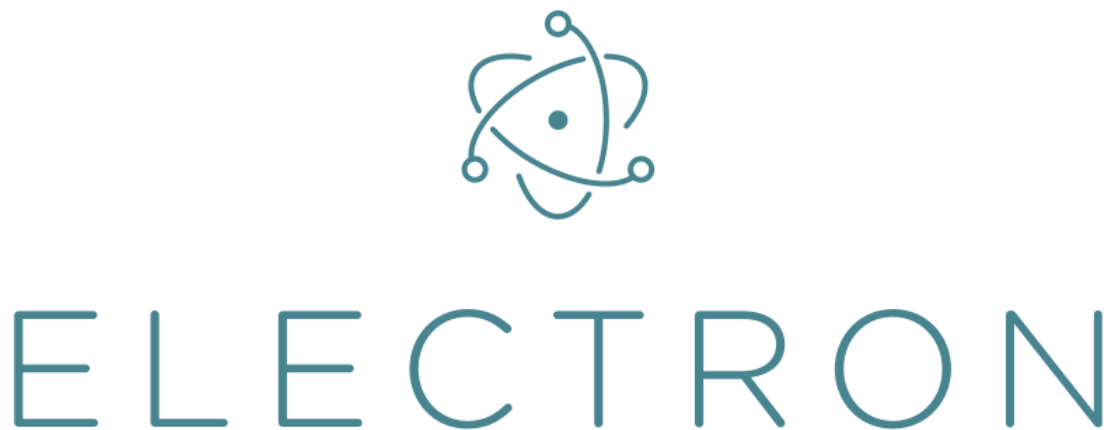
While Xamarin and Flutter are by far and away the more popular cross-platform frameworks [34] there are a number of smaller frameworks that have been around for a far greater number of years. QT being one of those. Released in 1995 [35] QT is free and open-source, operating under the GPL 2.0, GPL 3.0, and LGPL 3.0 licences [35].

---

QT has tools to enable cross-platform application development, advertising, its own IDE, and more [36]. Based on C++ [37] does not have the same learning requirement of a new language like Flutter.

To get access to certain tools such as the quick compiler, a commercial licence is needed [38]. This licence starts at \$180 a month [39] and while all frameworks that have been examined have both a free and a paid version, the performance of QT becomes an issue. As such without the quick compiler starting times for applications can become an issue [38].

## Electron



*Figure 7: Electron logo [46].*

Finally there is Electron, a free and open-source cross-platform framework [40]. Electron is much more geared toward web application development [40]. It can still be used to deploy to Android and iOS since that does not appear to be a primary function; it is more likely than not in among the most popular cross-platform frameworks [34]. Although with there needing to be three separate web apps for Daly Tyres, the fleet operator and the garage operator, in comparison for the one mobile app for drivers, this may not be as big of a hindrance than it first appears.

Similarly to Flutter, the minimum size of the application created is the largest issue at hand. Electron uses Chromium and Node.js to build its desktop applications [41]. Chromium is over 100MB in size [42]. Therefore any web application that is made with Electron will also have to come with this extra bulk. No matter how simple the web application is.

---

## Conclusion

After taking all of the above into account Xamarin has distinguished itself as a suitable cross-platform framework for this project. The fact that a business or an enterprise licence would make the framework an expensive endeavour is not unique to the framework. Even then the odds of one being needed are extremely unlikely.

Its popularity, especially in comparison to QT and Electron, as shown by its large collection of OSS contributors is a good indicator that there will be plenty of resources when implementing the codebase and into the future.

In a similar vein to choosing MySQL, Xamarin is also a widely used platform. Any expertise gained in using the platform now has a greater chance of being applied later. This also applies to the language, C#. With Flutter being the next biggest platform researched it revealed that Flutter uses Dart. This programming language is primarily used for Google applications. It is much more limited in its use. Being ranked 20th [47] on a list of the most common programming languages of 2022. In comparison to C# which is ranked 4th [47], it is clear that C# will have much more utility outside of this project.

---

# Development

Now that the theory portion of the research has been done it is time to move onto the practical. Where the ideas and concepts discussed from before are brought to life. This portion of the manual will cover any research that arose from that development process. As well as discuss any issues that were encountered, and the solutions found.

## Flutter Environment Set Up

To set up Flutter I used Android Studio, VS Code is also an option. Both are needed to install flutter and both were already downloaded on my machine. Here is a step by step guide of that download process.

### Download Flutter

1. Download the Flutter SDK for Windows from this website:  
<https://docs.flutter.dev/get-started/install/windows>
2. Extract zipped file using the recommended file path. C:\src\flutter\flutter
3. Copy this path
4. Using the Windows search tool enter 'Edit environment variables for your account'
5. Under 'User Variable' click on 'Path' and then click on the edit button.
6. Click on 'New' and paste the copied path from step 3
7. Click 'Ok' until all the screens just opened are gone
8. Using the Windows search tool enter 'cmd'
9. Type 'flutter doctor' and press enter
10. On screen it will show the requirements needed for flutter

### Android Studio Plugins

11. Open Android Studio
12. Click on 'Plugins'
13. Search for 'flutter' and click on install
14. Search for 'dart' and click on install
15. Restart Android Studio
16. Using the Windows search tool enter 'cmd'
17. Type 'flutter doctor' and press enter. 'Android Studio (version 2022.1)' is now complete.



---

## Android Studio SDK

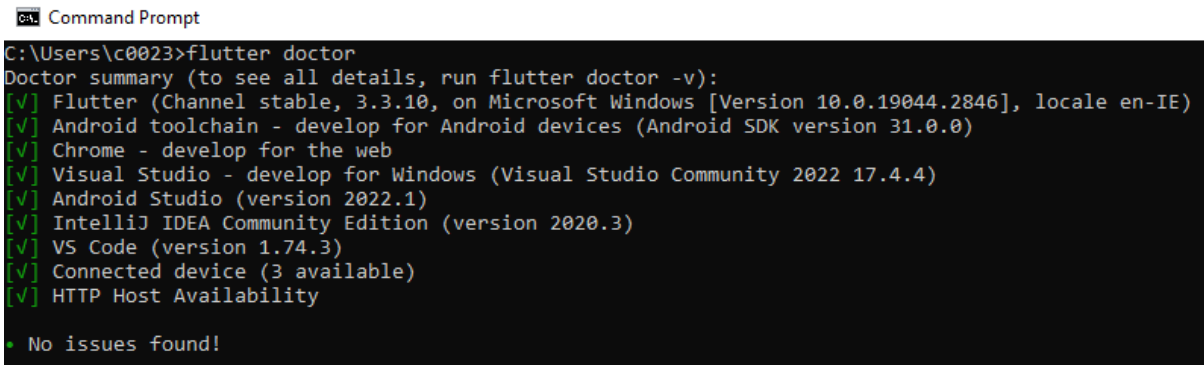
18. In Android Studio open 'SDK Manager'
19. Click on 'SDK Tools'
20. Make sure 'Android SDK Command-line Tools' is selected
21. Click 'Apply'
22. A prompt asking 'Confirm Changed' appears, click 'Ok'
23. Click 'Finish' once download is complete
24. Click 'Apply' and the 'Ok' in the SDK Manager
25. Using the Windows search tool enter 'cmd'
26. Type 'flutter doctor' and press enter. 'Android toolchain' is now partially complete.

## Android Studio Licence

27. Still in the 'cmd' type 'flutter doctor --android-licenses' and press enter
28. Type 'y' and press enter for all prompts
29. Type 'flutter doctor' and press enter. 'Android toolchain' is now complete.

## VS Code Extensions

30. Open VS Code
31. Click on 'Extensions'
32. Search for 'flutter' and click on install
33. Search for 'dart' and click on install
34. Restart VS Code
35. Using the Windows search tool enter 'cmd'
36. Type 'flutter doctor' and press enter. 'VS Code (version 1.74.3)' is now complete.
37. The cmd should now look like the figure below



```
Command Prompt
C:\Users\c0023>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.3.10, on Microsoft Windows [Version 10.0.19044.2846], locale en-IE)
[✓] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop for Windows (Visual Studio Community 2022 17.4.4)
[✓] Android Studio (version 2022.1)
[✓] IntelliJ IDEA Community Edition (version 2020.3)
[✓] VS Code (version 1.74.3)
[✓] Connected device (3 available)
[✓] HTTP Host Availability

• No issues found!
```

*Figure 8: Flutter has successfully been installed*

This means that all requirements for installing Flutter on Windows have been met and that development in either Android Studio or VS Code can now begin. Due to being more familiar with Android Studio this was my IDE of choice.

---

## Running Flutter on Emulator

Having selected Android Studio to develop in, the next step was to create a virtual device. This is a device that runs on an emulator inside Android Studio. Providing a proving ground for the Flutter project without the need to continually connect a physical device to the computer.

### Create Project

1. Open Android Studio
2. Click 'New Flutter Project'
3. Click 'next'
4. Give the project a name. All lowercase, i.e. my\_first\_app
5. A counter project is automatically provided by flutter

### Create Virtual Device

6. Open 'Device Manager'
7. Click 'Create Device'
8. Select 'Phone' and select a model from the list provided
9. Click 'next'
10. Select a system from the list provided
11. Click 'next'
12. Make any changes that are desired, otherwise just click 'Finish'
13. Check the 'Device Manager' tab. A phone is listed.

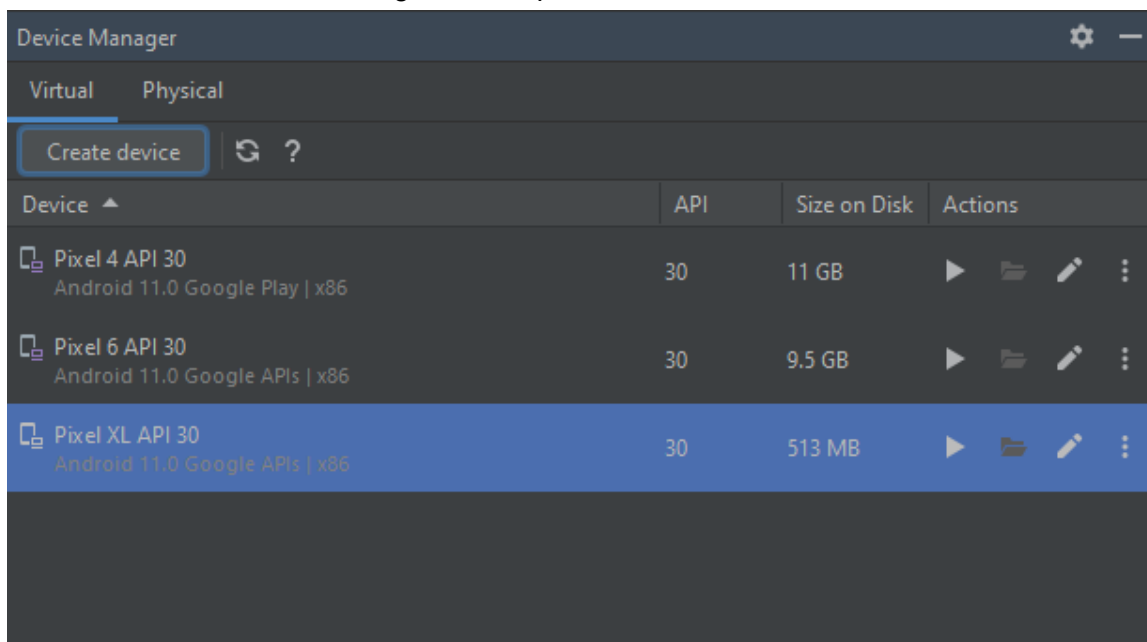
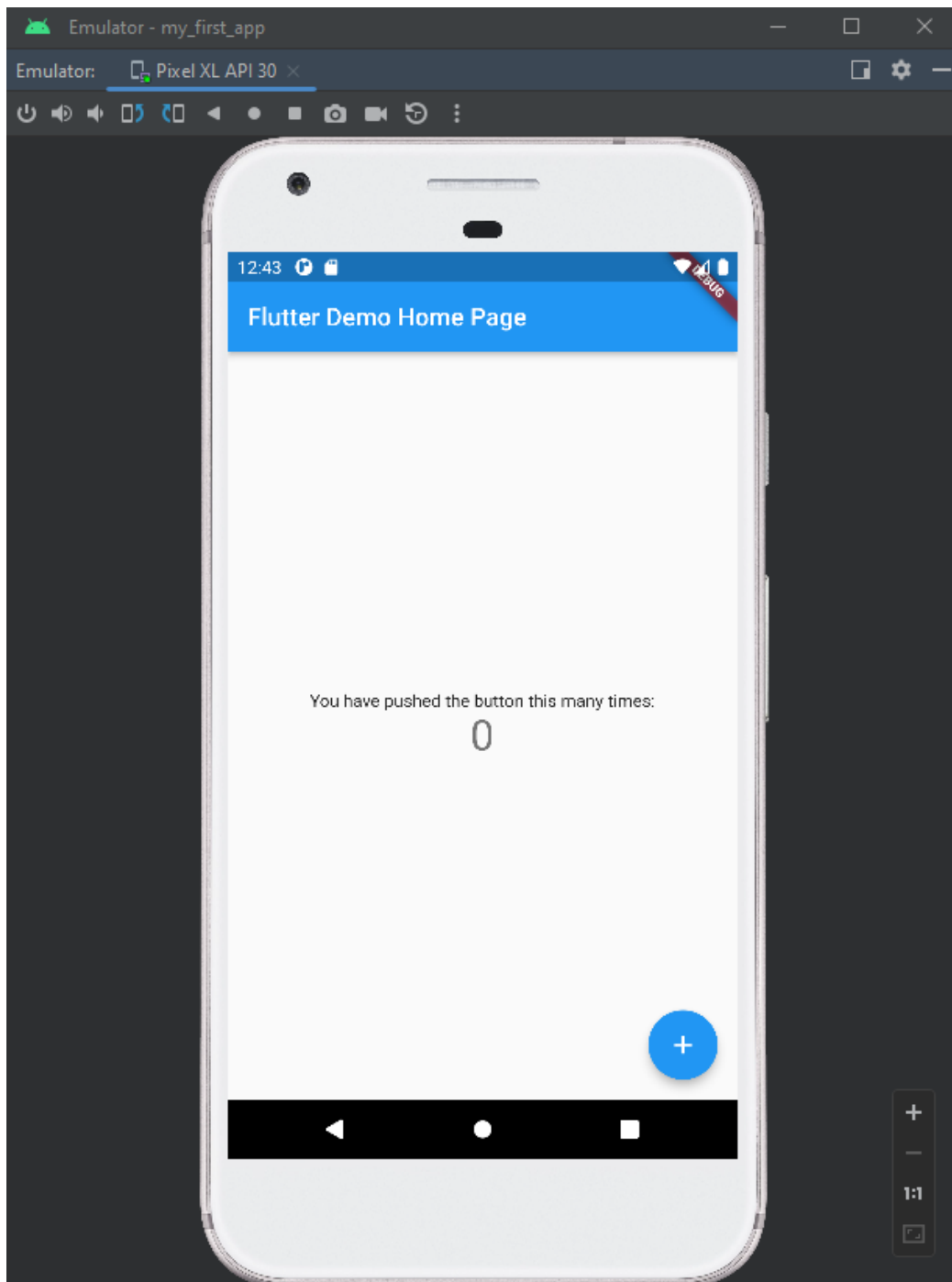


Figure 9: Virtual devices on Android Studio

---

## Run Emulator

14. Click the arrow head symbol next to the device to launch the virtual device
15. Click on '<no device selected>
16. Click 'Refresh'
17. Click on the name of the device just created
18. Click 'Run'



*Figure 10: Default Flutter app running on virtual device inside Android Studio*

Chrome, Edge and Windows are already listed as devices and require no further set up.

---

## Flutter and MySQL

One of the first major problems during the development stage was trying to connect the App to the MySQL database. As Android Studio and Flutter do not have an inbuilt driver to access MySQL. Upon further research a preferred method for those online is to access MySQL databases by using PHPmyAdmin [48]. This however is not desirable as it creates an intermediary between the two applications. Increasing the application's vulnerability to cyber attacks.

After some more looking, a library called mysql1 looked promising. Available on pub.dev this library provides a MySQL driver for the Dart programming language, which is used by Flutter [49]. However, this too was unsuccessful. As even when the Flutter App would run, the MySQL would not connect. Although the reasons behind this failure to connect are not known, it was determined that after so much time spent on the issue with no progress it was time to move onto an alternative database system.

```
void _getCustomer() {
  db.getConnection().then((conn) async {
    String sql = "SELECT email FROM company.customer WHERE number = 12;";
    var results = await conn.query(sql);
    for(var row in results) {
      log('Email: ${row[0]}');
      setState(() {
        email = row[0];
      });
    }
  });
}
```

*Figure 11: Flutter code for SQL*

This was the code used to try and understand the issue with connecting to the MySQL database. When executed it does not reach inside the loop. As the log doesn't print to the console. It also produced this error:

```
[ERROR:flutter/runtime/dart_vm_initializer.cc(41)] Unhandled Exception: SocketException: Connection refused (OS Error: Connection refused, errno = 111), address = localhost, port = 47788
```

---

## Flutter and Firebase

With both Android Studio and Flutter being owned by Google [50][51]; it is unsurprising that other Google products can work with one another without much need for prior setup. As such using the NoSQL database system Firebase was the next sensible step. This proved to be the correct step as Firebase was much easier to connect to and had much more documentation on how to use it in conjunction with Flutter available online.

However, Firebase does use NoSQL and this is a technology that I had no previous experience in. At the top of the scale Firebase NoSQL stores information in collections. There can be multiple collections in a database. Within these collections are documents. The documents are what hold the individual pieces of data. All of the information contained in the database is stored as a JSON object [52]. The database structure that was initially thought of for the MySQL database is not compatible with this layout.

## Conclusion

Despite the drawbacks in experience and the fact that previous work had to be revised, this was a necessary change and the right one to make. The compatibility of the two technologies has allowed for faster progress. Something was sorely needed.

---

## Connect to Firebase

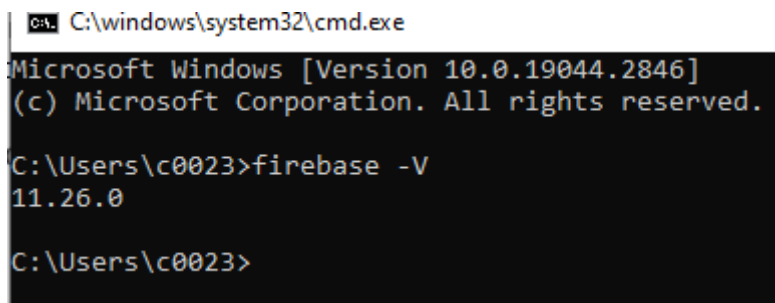
Firebase is an online database and requires no direct insulation. Instead you must create an account and connect to the database over the internet.

### Log on to Firebase

1. Go to this page <https://firebase.google.com/>
2. Click 'Get Started'
3. Enter any name into the Project Name field
4. Turn off Google Analytics
5. Click 'Continue'

### Install Firebase CLI

6. Click on the Flutter Icon at the top of the screen
7. Follow the instructions
8. Install Firebase CLI
9. Once installed enter the CLI
10. Login with your Google account
11. Install Node.js
12. Using the Windows search tool enter 'cmd'
13. Enter 'npm install -g firebase-tools'
14. Copy this path 'C:\Users\you\AppData\Local\Pub\Cache\bin'
15. Using the Windows search tool enter 'Edit environment variables for your account'
16. Under System Variable' click on 'Path' and then click on the edit button.
17. Click on 'New' and paste the copied path from step 14
18. Click 'Ok' until all the screens just opened are gone
19. Using the Windows search tool enter 'cmd'
20. Enter 'firebase -V'. If successfully installed the screen will appear as below.



```
C:\windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.

C:\Users\c0023>firebase -V
11.26.0

C:\Users\c0023>
```

Figure 12: Firebase installed

---

## Install and Run FlutterFire CLI

21. Using the Windows search tool enter 'cmd'
22. Copy and paste the next command given by the instructions into the cmd
23. The second instruction is copied and pasted into a terminal at the root of your flutter project
24. Set up for any devices you wish. For this project it was all.

## Initialise Firebase

25. Copy and paste the next command in the instructions into the main.dart file.
26. Firebase is now ready.

## Firebase Libraries

Copy the following commands into a terminal at the root folder for the project:

- flutter pub add firebase\_database [66]
- flutter pub add firebase\_auth [67]
- flutter pub add firebase\_core [68]
- flutter pub add cloud\_firestore [69]

All dependencies are available at [pub.dev](https://pub.dev)

---

## Dart Code

A modern object-oriented programming language, Dart was developed in 2011 and was developed by Google [53]. The designers of the language are Lars Bak and Kasper Lund [54]. Its creation was influenced by a number of different languages. Some examples being Java, JavaScript, C# [65] and TypeScript [55]. Specifically from looking at how the code is structured, Dart is similar to C. Given all these facts, this gives Dart a strong groundwork for use in web development.

Its use as Flutter's main programming language may reveal what was the purpose of designing such a language. A better performing cross-platform language for development. It markets itself as

Below is a list of some of the common widgets that were used in the creation of this project. As well as a brief explanation of their operation and how they were used:

- `Text()` - this widget was the starting base for much of the project. It displayed the text that is contained within it to the screen [56].
- `TextFormField()` - a combination of two widgets, the `TextField()` and the `FormField()`. With the former contained inside the latter [57]. This widget was used to receive input from the user. With the `TextField()` portion letting the user enter the information [57] and the `FormField()` enabled that information to be handled [58]. This combination of functions made it the perfect widget to use in forms that relied on user inputs.
- `TextStyle()` - executed from within a `Text()` or similar widget, it was used to arrange the composition of the text on display [59]. Such as the font, size and colour of the text [59]. All text on display for the user has been modified using this class.
- `BoxDecoration()` - to make clear where users should put their information in the form, a `BoxDecoration()` class was extremely useful. It allowed for styling of the surrounding form field [60]. Such as the shape and colour of the input boxes.
- `TextButton()` - once a user had filled in a field it was necessary to have an action that would specify when that information could be handled by the backend. To do this a `TextButton()` widget was used. As a button it could have any text written inside it [61]. Making it clear what its purpose was. Most importantly it allowed for an `onPressed()` operation to be carried out. This could be anything from a page navigation to another screen operation, to calling a function to perform an operation on the information contained inside the form [62].
- `Expanded()` - creating a space that will take up any screen space that is available to it [63]. In use this means that if there are two `Expanded()` widgets in use, each will occupy half of the available screen space. This widget was used for setting the layout of each page. To ensure uniformity. It could be orientated either vertically or horizontally [63].
- `Container()` - somewhat similar to `Expanded()`, this widget created a space for its contents [64]. However, unlike `Expanded()` it only occupied the same space as the content specified. Making it a suitable widget to handle each individual element of a page's layout.





---

## References

1. Study.com. (n.d.). *What is a Server? - Definition & Explanation - Video & Lesson Transcript*. [online] Available at:  
<https://study.com/academy/lesson/what-is-a-server-definition-lesson-quiz.html>.
2. MDN Web Docs. (2019). *What is a web server?* [online] Available at:  
[https://developer.mozilla.org/en-US/docs/Learn/Common\\_questions/What\\_is\\_a\\_web\\_server](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server).
3. Migrator (2012). *Pros and cons of in-house hosting*. [online] Available at:  
<https://www.nibusinessinfo.co.uk/content/pros-and-cons-house-hosting>.
4. Labfolder. (2017). *Cloud vs local Server - Where should you store your data?* [online] Available at: <https://www.labfolder.com/cloud-vs-local-server/>.
5. honestproscons.com. (2020). *10 Advantages and Disadvantages of Cloud Hosting*. [online] Available at:  
<https://honestproscons.com/advantages-and-disadvantages-of-cloud-hosting/>.
6. Indeed Career Guide. (n.d.). *Types of Database Languages and Their Uses (Plus Examples)*. [online] Available at:  
<https://www.indeed.com/career-advice/career-development/database-languages#:~:text=What%20are%20database%20languages%3F>.
7. www.w3schools.com. (n.d.). *SQL Introduction*. [online] Available at:  
[https://www.w3schools.com/sql/sql\\_intro.asp#:~:text=SQL%20stands%20for%20Structured%20Query](https://www.w3schools.com/sql/sql_intro.asp#:~:text=SQL%20stands%20for%20Structured%20Query).
8. admin (2022). *SQL vs. NoSQL: Pros and Cons*. [online] W3schools. Available at:  
<https://www.w3schools.blog/sql-vs-nosql-pros-and-cons>.
9. www3.technologyevaluation.com. (n.d.). *SQL vs. NoSQL Databases: What's the Difference? | TEC*. [online] Available at:  
<https://www3.technologyevaluation.com/research/article/sql-vs-nosql-databases-what-s-the-difference.html>.
10. www.couchbase.com. (n.d.). *NoSQL Databases – What They Are and Why You Need One*. [online] Available at:  
<https://www.couchbase.com/resources/why-nosql#:~:text=of%20modern%20businesses.->
11. IBM (2010). *What is a database management system?* [online] www.ibm.com. Available at:  
<https://www.ibm.com/docs/en/zos-basic-skills?topic=zos-what-is-database-management-system>.
12. Oracle.com. (2021). *What is MySQL?* [online] Available at:  
<https://www.oracle.com/mysql/what-is-mysql/>.
13. OpenSource (2019). *What is open source?* [online] Opensource.com. Available at:  
<https://opensource.com/resources/what-open-source>.
14. W3Schools (n.d.). *MySQL advantages and disadvantages - W3schools*. [online] W3Schools. Available at:  
<https://www.w3schools.blog/mysql-advantages-disadvantages>.

15. MariaDB.org. (n.d.). *About MariaDB Server*. [online] Available at: <https://mariadb.org/about/#policies>.
16. MariaDB. (n.d.). *Open Source Database (RDBMS) for the Enterprise*. [online] Available at: <https://mariadb.com/docs/ent/connect/programming-languages/>.
17. opensource.com. (n.d.). *Comparing 3 open source databases: PostgreSQL, MariaDB, and SQLite | Opensource.com*. [online] Available at: <https://opensource.com/article/19/1/open-source-databases#:~:text=installed%20with%20MariaDB.->.
18. PostgreSQL (2019). *PostgreSQL: About*. [online] Postgresql.org. Available at: <https://www.postgresql.org/about/>.
19. www.postgresql.org. (n.d.). *PostgreSQL: License*. [online] Available at: <https://www.postgresql.org/about/licence/>.
20. opensource.org. (n.d.). *The PostgreSQL Licence (PostgreSQL) | Open Source Initiative*. [online] Available at: <https://opensource.org/licenses/postgresql>.
21. IONOS Digitalguide. (n.d.). *PostgreSQL: a closer look at the object-relational database management system*. [online] Available at: <https://www.ionos.com/digitalguide/server/know-how/postgresql/>.
22. Logo Mysql PNG Images, Free Download - Free Transparent PNG Logos. (n.d.). *Logo Mysql PNG Images, Free Download - Free Transparent PNG Logos*. [online] Available at: <https://www.freepnglogos.com/pics/logo-mysql>.
23. TM/©MariaDB (2009). *The logo of MariaDB – Database management system, relational, open source, community developed fork of MySQL*. [online] Wikimedia Commons. Available at: [https://commons.wikimedia.org/wiki/File:MariaDB\\_colour\\_logo.svg](https://commons.wikimedia.org/wiki/File:MariaDB_colour_logo.svg).
24. IconScout. (n.d.). *19 Postgresql Icons - Free in SVG, PNG, ICO*. [online] Available at: <https://iconscout.com/icons/postgresql>.
25. Microsoft. (n.d.). *Xamarin | Open-source mobile app platform for .NET*. [online] Available at: <https://dotnet.microsoft.com/en-us/apps/xamarin>.
26. AltexSoft. (2019). *The Good and The Bad of Xamarin Mobile Development*. [online] Available at: <https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/>.
27. SearchNetworking. (n.d.). *What is operational support system (OSS)? - Definition from WhatIs.com*. [online] Available at: <https://www.techtarget.com/searchnetworking/definition/operational-support-system-OSS#:~:text=An%20operational%20support%20system%20>.
28. visualstudio.microsoft.com. (n.d.). *Pricing and Purchasing Options | Visual Studio*. [online] Available at: <https://visualstudio.microsoft.com/vs/pricing/?tab=enterprise>.
29. flutter.dev. (n.d.). *Flutter - Build apps for any screen*. [online] Available at: [https://flutter.dev/?gclid=Cj0KCQiA99ybBhD9ARIsALvZavVNnioQW\\_Mu-fduXOhJnt0oslzJdOtA1POqNlcNe7wFCoY23HjDO-caAt21EALw\\_wcB&gclidsrc=aw.ds](https://flutter.dev/?gclid=Cj0KCQiA99ybBhD9ARIsALvZavVNnioQW_Mu-fduXOhJnt0oslzJdOtA1POqNlcNe7wFCoY23HjDO-caAt21EALw_wcB&gclidsrc=aw.ds).
30. dart.dev. (n.d.). *Who uses Dart*. [online] Available at: <https://dart.dev/community/who-uses-dart#:~:text=Google%20engineers%20use%20Dart%20to>.
31. flutter.dev. (n.d.). *Learn*. [online] Available at: <https://flutter.dev/learn>.
32. AltexSoft. (n.d.). *The Good and the Bad of Flutter App Development*. [online] Available at:

- <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/>.
33. AltexSoft. (n.d.). *The Good and the Bad of Flutter App Development*. [online] Available at: <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-flutter-app-development/>.
  34. Statista. (n.d.). *Cross-platform mobile frameworks used by global developers 2020*. [online] Available at: <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>.
  35. Wikipedia Contributors (2019). *Qt (software)*. [online] Wikipedia. Available at: [https://en.wikipedia.org/wiki/Qt\\_\(software\)](https://en.wikipedia.org/wiki/Qt_(software)).
  36. The Qt Company (2019). *Qt | Cross-platform software development for embedded & desktop*. [online] [Www.qt.io](http://www.qt.io). Available at: <https://www.qt.io/>.
  37. SCAND. (n.d.). *Why Should You Use Qt for Mobile App Development?* [online] Available at: <https://scand.com/company/blog/mobile-app-development-in-qt/>.
  38. [www.veprof.com](http://www.veprof.com). (n.d.). *Advantages and Disadvantages of Using QT for Mobile Applications*. [online] Available at: <https://www.veprof.com/blog/technology/qt-mobile-applications>.
  39. [www.qt.io](http://www.qt.io). (n.d.). *Pricing and Packaging | Software Stack | Tech Stack | Qt*. [online] Available at: <https://www.qt.io/pricing>.
  40. [www.electronjs.org](http://www.electronjs.org). (n.d.). *Electron | Build cross-platform desktop apps with JavaScript, HTML, and CSS*. [online] Available at: <https://www.electronjs.org/>.
  41. [electronjs.org](http://electronjs.org). (n.d.). *Introduction | Electron*. [online] Available at: <https://www.electronjs.org/docs/latest/>.
  42. [groups.google.com](https://groups.google.com). (n.d.). *Size of Chromium codebase*. [online] Available at: [https://groups.google.com/a/chromium.org/g/chromium-discuss/c/AiMkyb3\\_WPg](https://groups.google.com/a/chromium.org/g/chromium-discuss/c/AiMkyb3_WPg).
  43. Xamarin (2010). *English: Logo of Xamarin*. [online] Wikimedia Commons. Available at: <https://commons.wikimedia.org/wiki/File:Xamarin-logo.svg>.
  44. flutter.dev. (n.d.). *Brand*. [online] Available at: <https://flutter.dev/brand>.
  45. Project, Q. (2016). *English: Qt framework's logo*. [online] Wikimedia Commons. Available at: [https://commons.wikimedia.org/wiki/File:Qt\\_logo\\_2016.svg](https://commons.wikimedia.org/wiki/File:Qt_logo_2016.svg).
  46. Gritskevich, A. (n.d.). *Meeting Electron.js – ISS Art Blog | AI | Machine Learning | Computer Vision*. [online] Available at: <https://blog.issart.com/meeting-electron-js/>.
  47. [www.orientsoftware.com](http://www.orientsoftware.com). (n.d.). *Most Popular Programming Languages in 2022 & Beyond*. [online] Available at: <https://www.orientsoftware.com/blog/most-popular-programming-languages/>.
  48. Grondman, T. (2022). *How To Connect Flutter to PHP*. [online] Medium. Available at: <https://betterprogramming.pub/how-to-connect-flutter-to-php-8be032df0f55>.
  49. *Dart packages*. (n.d.). *mysql1 | Dart Package*. [online] Available at: <https://pub.dev/packages/mysql1>.
  50. Wikipedia. (2023). *Android Studio*. [online] Available at: [https://en.wikipedia.org/wiki/Android\\_Studio#:~:text=Android%20Studio%20is%20the%20official](https://en.wikipedia.org/wiki/Android_Studio#:~:text=Android%20Studio%20is%20the%20official).

51. *www.google.com. (n.d.). flutter ownership - Google Search. [online] Available at: [https://www.google.com/search?q=flutter+ownership&rlz=1C1RXQR\\_enIE927IE927&oq=flutter+ownership&aqs=chrome..69i57j0i22i30i4j0i390i650i4.6080j0j9&sourceid=chrome&ie=UTF-8](https://www.google.com/search?q=flutter+ownership&rlz=1C1RXQR_enIE927IE927&oq=flutter+ownership&aqs=chrome..69i57j0i22i30i4j0i390i650i4.6080j0j9&sourceid=chrome&ie=UTF-8).*
52. *Firebase. (2019). Structure Your Database | Firebase Realtime Database | Firebase. [online] Available at: <https://firebase.google.com/docs/database/web/structure-data>.*
53. *[https://en.wikipedia.org/wiki/Flutter\\_\(software\)#:~:text=Flutter%20is%20an%20open%2Dsource,web%20from%20a%20single%20codebase](https://en.wikipedia.org/wiki/Flutter_(software)#:~:text=Flutter%20is%20an%20open%2Dsource,web%20from%20a%20single%20codebase).*
54. *Wikipedia. (2023). Dart (programming language). [online] Available at: [https://en.wikipedia.org/wiki/Dart\\_\(programming\\_language\)#cite\\_note-5](https://en.wikipedia.org/wiki/Dart_(programming_language)#cite_note-5).*
55. *Anon, (n.d.). The Dart Team Welcomes TypeScript. [online] Available at: <https://news.dartlang.org/2012/10/the-dart-team-welcomes-typescript.html>.*
56. *api.flutter.dev. (n.d.). Text class - widgets library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/widgets/Text-class.html>.*
57. *api.flutter.dev. (n.d.). TextFormField class - material library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/material/TextFormField-class.html>.*
58. *api.flutter.dev. (n.d.). FormField class - widgets library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/widgets/FormField-class.html>.*
59. *api.flutter.dev. (n.d.). TextStyle class - painting library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/painting/TextStyle-class.html>.*
60. *api.flutter.dev. (n.d.). BoxDecoration class - painting library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/painting/BoxDecoration-class.html>.*
61. *api.flutter.dev. (n.d.). TextButton class - material library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/material/TextButton-class.html>*
62. *api.flutter.dev. (n.d.). onPressed property - TappableChipAttributes class - material library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/material/TappableChipAttributes/onPressed.html>*
63. *api.flutter.dev. (n.d.). Expanded class - widgets library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/widgets/Expanded-class.html>.*
64. *api.flutter.dev. (n.d.). Container class - widgets library - Dart API. [online] Available at: <https://api.flutter.dev/flutter/widgets/Container-class.html>.*
65. *www.javatpoint.com. (n.d.). Flutter: What is Dart Programming - Javatpoint. [online] Available at: <https://www.javatpoint.com/flutter-dart-programming>.*
66. *Dart packages. (n.d.). firebase\_database | Flutter Package. [online] Available at: [https://pub.dev/packages/firebase\\_database/install](https://pub.dev/packages/firebase_database/install).*

67. Dart packages. (n.d.). firebase\_auth | Flutter Package. [online] Available at:  
[https://pub.dev/packages/firebase\\_auth/install](https://pub.dev/packages/firebase_auth/install).
68. Dart packages. (n.d.). firebase\_core | Flutter Package. [online] Available at:  
[https://pub.dev/packages/firebase\\_core/install](https://pub.dev/packages/firebase_core/install).
69. Dart packages. (n.d.). cloud\_firestore | Flutter Package. [online] Available at:  
[https://pub.dev/packages/cloud\\_firestore/install](https://pub.dev/packages/cloud_firestore/install).