CA326
Third Year Project

DCU Hub
Technical Specifications

Joanna Talvo
Chloe Ward

Supervisor: Dr. Malika Bendechache

1. Introduction

1.1 Overview

The product that was developed was a mobile application that is exclusive for DCU students. The aim of the application was to create an environment for DCU students to have a one-stop mobile application for everything DCU related while also making communication easier - especially during a global pandemic. It was developed as a cross-platform mobile application that can be used by both Android and iOS users.

The app includes user registration, user login and user password reset options, as a user must be a verified DCU student in order to avail of the services offered in this application.

Some of the functionalities include a News feed that has various DCU related twitter accounts compacted in a single News Feed. Users also have the ability to make and enter group chats by having a list of available public group chats to view from. The app also includes a timetable which allows users to view the timetable for a specific date and course. Extra functionalities such as a student To-Do list and access to DCU hotlines are also included in the mobile application in order to achieve that one stop shop application for DCU students.

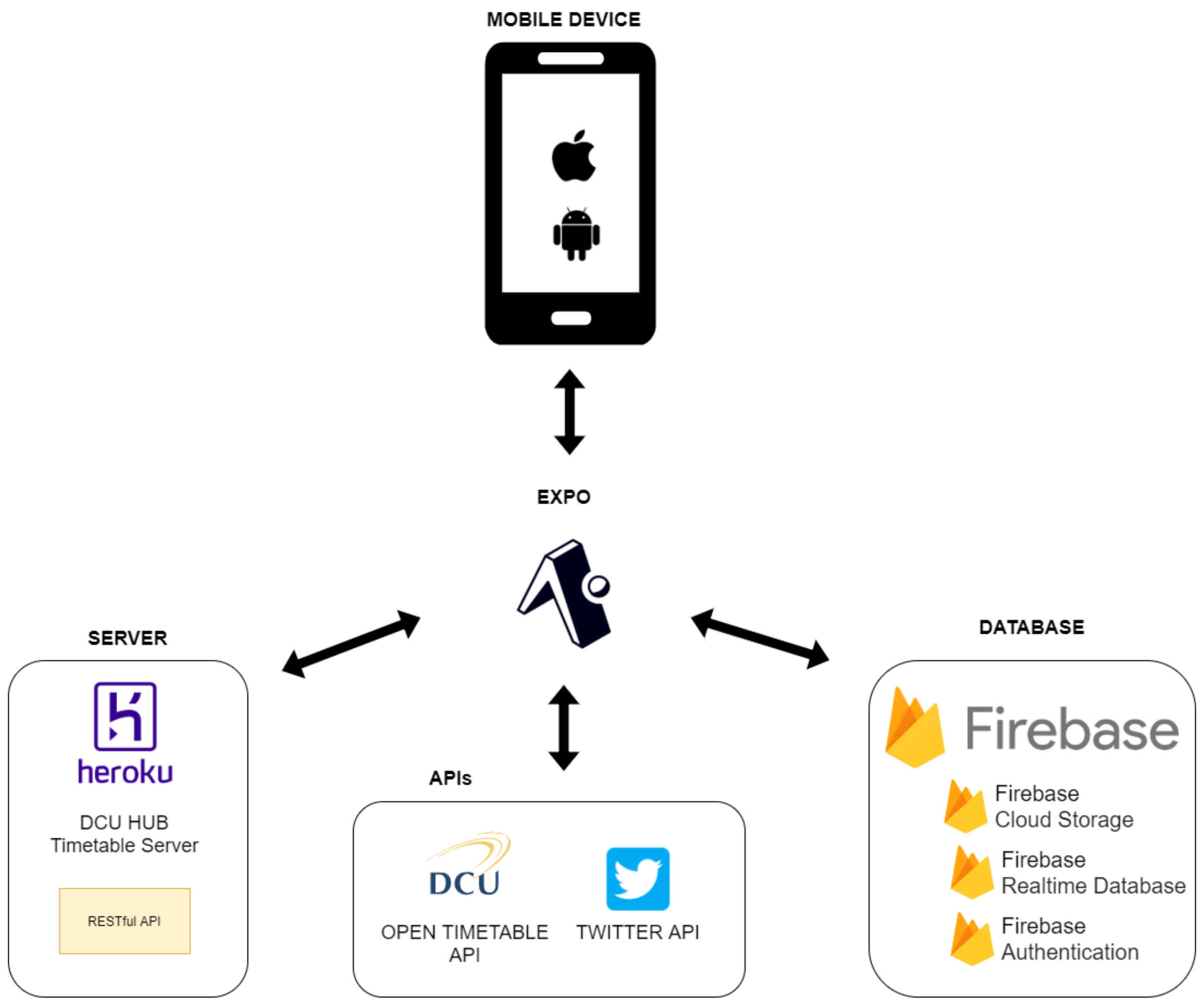
The mobile application was developed using React Native by combining Firebase as the backend. We also make use of the oEmbed Twitter API, RESTful API and JavaScript in the backend in order to achieve the functionality for the application.

1.2 Glossary

1. **API:** Application Programming Interface. It is a gateway that allows two applications to talk to each other. Many different API's exist such as Database API's, Operating System API's. An API allows developers to implement functionality without having to code it themselves.
2. **RESTful API:** Representational State Transfer API is an architectural style for an API which makes use of HTTPS requests to access and use data.
3. **Firebase:** A platform developed by Google to develop mobile and web applications. It gives developers access to a wide range of tools to develop applications quickly and efficiently to a high standard.
4. **Firebase Authentication:** A firebase library that is used to authenticate our applications users by using emails and passwords. When a user signs up it checks to see if the email address is verified and if it is not then an email verification is sent.
5. **Firebase Realtime Database:** No SQL cloud-hosted database that stores and syncs data between our users in Realtime.
6. **oEmbed API:** Programmatically converts a url into an embedded Tweets markup.
7. **Cross-platform:** The practice of developing software or services for various platforms.
8. **Heroku:** is a cloud platform that enables developers to run, deploy and operate applications on the cloud.

2. System Architecture

2.1 System Architecture Diagram



2.2 System Architecture Overview

The major components of our system are the application and Firebase, which acts as the backend. The client connects to the application via Expo which is then connected to the Firebase database. This will then provide a static hosting for the application.

Firebase Authentication library is used to authenticate a user in the database to gain access to the application. This stores the users email and passwords alongside with a unique user identification number. Users are sent a verification email upon signing up. We structure the registration page in such a way that the user's domain name should match the DCU email domain name. This serves the exclusivity of our application for DCU students.

The Firebase Cloud Storage is where the user information and to-do tasks are stored. The data stored in this database is retrieved within the application through the connection of the users to a specific database reference. Data also gets updated whenever a user decides to change anything about their information.

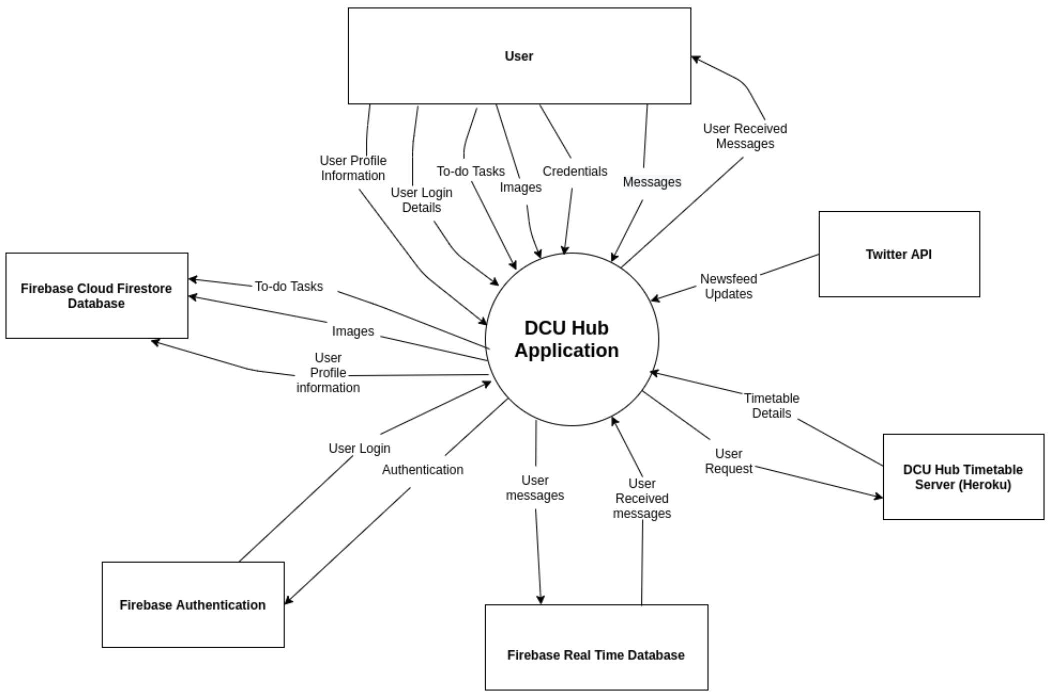
The Firebase RealTime Database is used to store users messages and group chats created by each user. The database is updated in real time by being stored as json strings. Users then receive messages as soon as the data is synchronised to the users which occurs in real time. The synchronization of the data in the real time database is enabled via websockets. This ultimately gives the users a responsive experience in the application.

Our timetable server is hosted on Heroku platform. This provides a secure and fast way of retrieving student’s timetable data as it is running in the cloud 24/7. This method is more efficient than running the server locally each time we wanted to run our application.

The Twitter oEmbed API (Twitter Publish) was implemented to provide news updates about DCU current events. The twitter URL of various DCU accounts is embedded into our application just like a twitter timeline using a React Native library.

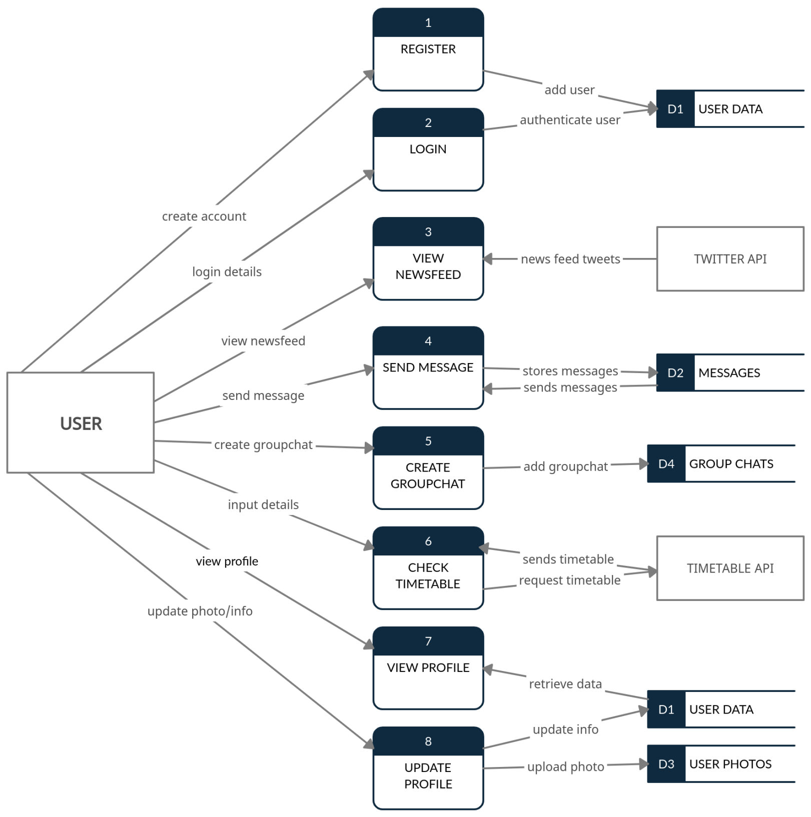
3. High-Level Design

3.1 Context Flow Diagram



This diagram is a high level view of the DCU Hub application, it defines and clarifies the boundaries that exist in the DCU Hub application. It shows the flow of the information as it goes through the application and its external entities that interact with it, such as the firebase libraries, the Twitter API and the DCU OpenTimetable API.

3.2 Data Flow Diagram

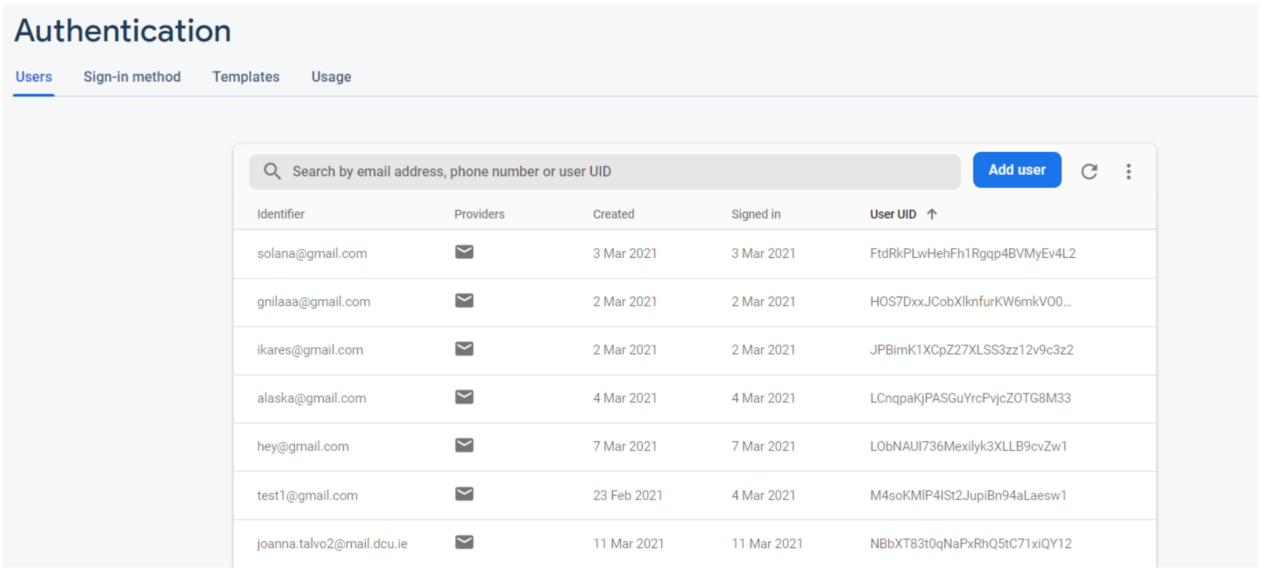


Data flow diagram is used to represent the flow of data of the system. This describes the processes involved in the system to transfer the data from the input to the file storage where it will be stored. For example, the messages are stored in the messages database. External entities such as Twitter API and Timetable API which supply and receive data from the system.

3.3 Firebase Database

Firebase Authentication

The emails used below are all fake emails which were collected during user testing. The DCU email verification was implemented after user testing.


Authentication

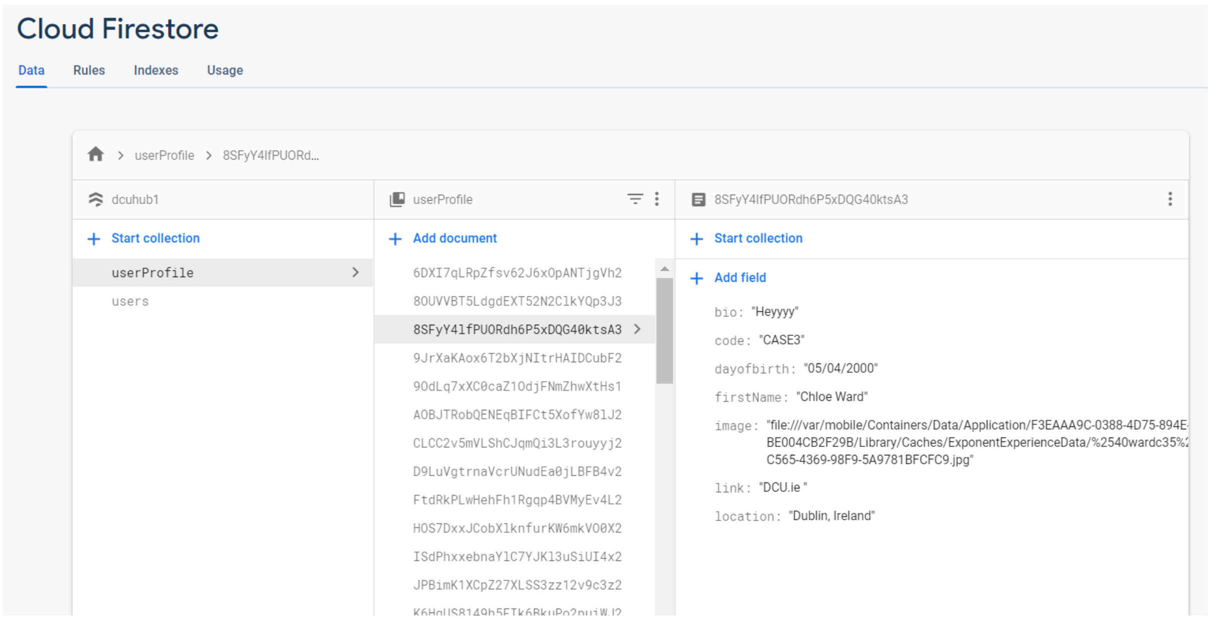
Users | Sign-in method | Templates | Usage

Search by email address, phone number or user UID

Identifier	Providers	Created	Signed in	User UID ↑
solana@gmail.com	✉	3 Mar 2021	3 Mar 2021	FtdRkPLwHehFh1Rgqp4BVMyEv4L2
gnilaaa@gmail.com	✉	2 Mar 2021	2 Mar 2021	HOS7DxxJCobXlknfurKW6mkVO0...
ikares@gmail.com	✉	2 Mar 2021	2 Mar 2021	JPBimK1XCpZ27XLSS3zz12v9c3z2
alaska@gmail.com	✉	4 Mar 2021	4 Mar 2021	LCnqpaKjPASGuYrcPvjcZOTG8M33
hey@gmail.com	✉	7 Mar 2021	7 Mar 2021	LObNAUI736Mexilyk3XLLB9cvZw1
test1@gmail.com	✉	23 Feb 2021	4 Mar 2021	M4soKMlP4lSt2JupiBn94aLaesw1
joanna.talvo2@mail.dcu.ie	✉	11 Mar 2021	11 Mar 2021	NBbXT83t0qNaPxRhQ5tC71xiQY12

Cloud Firestore

Users Profile information and To-Do lists.


Cloud Firestore

Data | Rules | Indexes | Usage

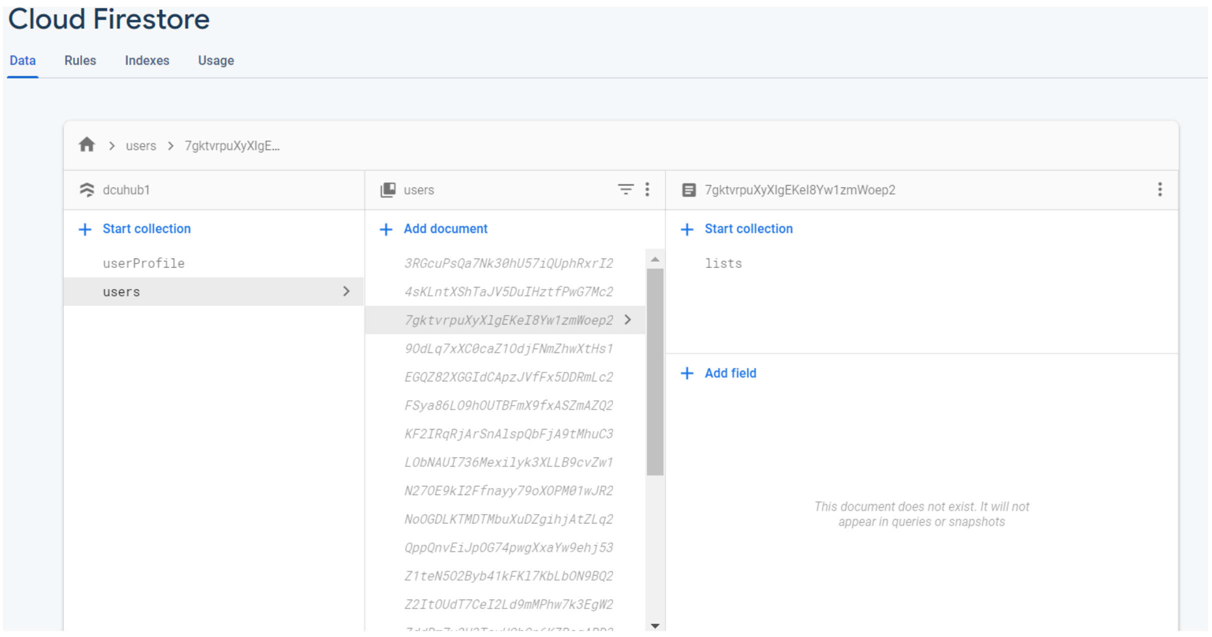
userProfile > 8SFyY4lfPUORd...

dcuhub1
- userProfile
- users

userProfile
- 6DXI7qLRpZfsv62J6xOpANTjgVh2
- 8OUVVBT5LdgdEXT52N2ClkYQp3J3
- 8SFyY4lfPUORdh6P5xDQG40ktsA3
- 9JrXaKAox6T2bXjNItrHAIDCubF2
- 9OdLq7xXC0caZ1OdjFNmZhwXtHs1
- AOBJTRobQENEqBIFCt5XofYw8lJ2
- CLCC2v5mVLShCJqmQi3L3rouyyj2
- D9LuVgtrnaVcrUNudEa0jLBFB4v2
- FtdRkPLwHehFh1Rgqp4BVMyEv4L2
- HOS7DxxJCobXlknfurKW6mkVO0X2
- ISdPhxxebnaYlC7YJKl3uSiUI4x2
- JPBimK1XCpZ27XLSS3zz12v9c3z2
- K6HqUS8149h5FTk6BkuPo2nuiWJ2

8SFyY4lfPUORdh6P5xDQG40ktsA3

bio: "Heyyyy"
code: "CASE3"
dayofbirth: "05/04/2000"
firstName: "Chloe Ward"
image: "file:///var/mobile/Containers/Data/Application/F3EAAA9C-0388-4D75-894E-BE004CB2F29B/Library/Caches/ExponentExperienceData/%2540wardc35%2
C565-4369-98F9-5A9781BFCFC9.jpg"
link: "DCU.ie "
location: "Dublin, Ireland"


Cloud Firestore

userProfile
users

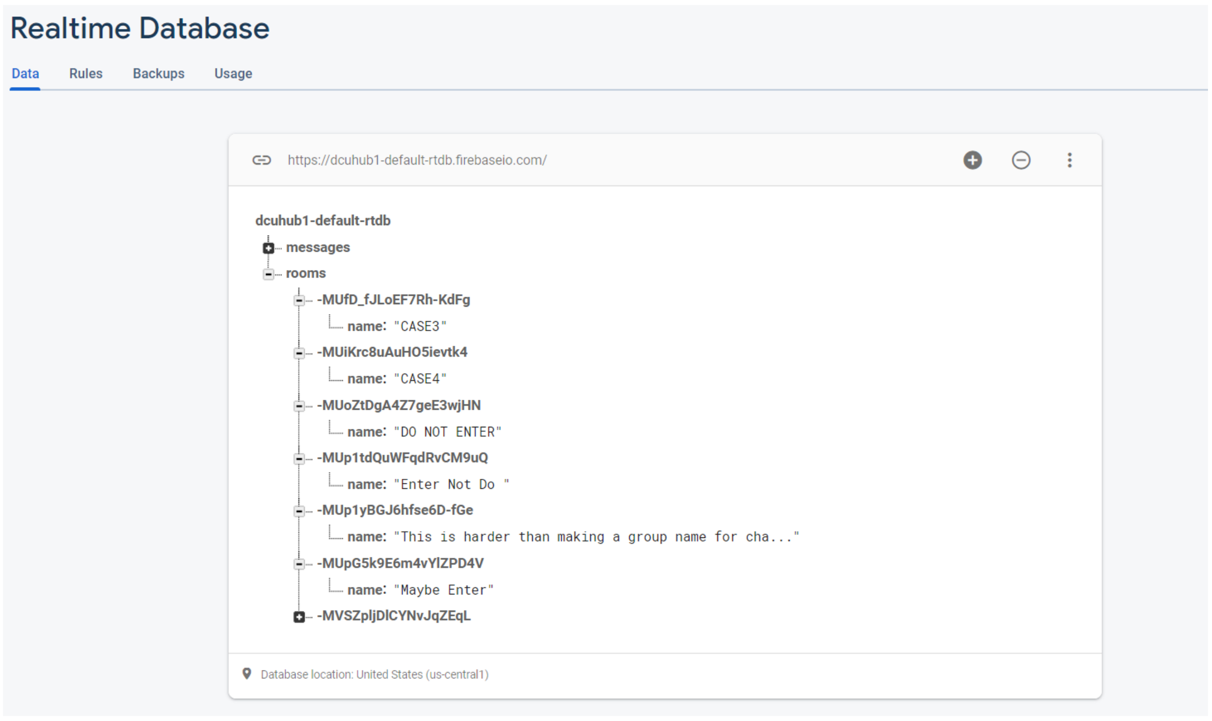
3RGcuPsQa7Nk30hU57iQUphRxrI2
4sKLntXShTaJV5DuIHztfPwG7Mc2
7gktvrpuXyXlgEKeI8Yw1zmWoep2
9OdLq7xXC0caZ1OdjFNmZhwXtHs1
EGQZ82XGGIdCApzJVfFx5DDRmLc2
FSya86LO9hOUTBFmX9fxASZmAZQ2
KF2IRqRjArSnAlspQbFjA9tMhuC3
LObNAUI736Mexilyk3XLLB9cvZw1
N27OE9kI2Ffnayy79oXOPM01wJR2
NoOGDLKTMDTMbuXuDZgihjAtZLq2
QppQnvEiJpOG74pwgXxaYw9ehj53
Z1teN5O2Byb41kFKl7KbLbON9BQ2
Z2ItOUdT7CeI2Ld9mMPhw7k3EgW2

lists

This document does not exist. It will not appear in queries or snapshots

Real Time Database

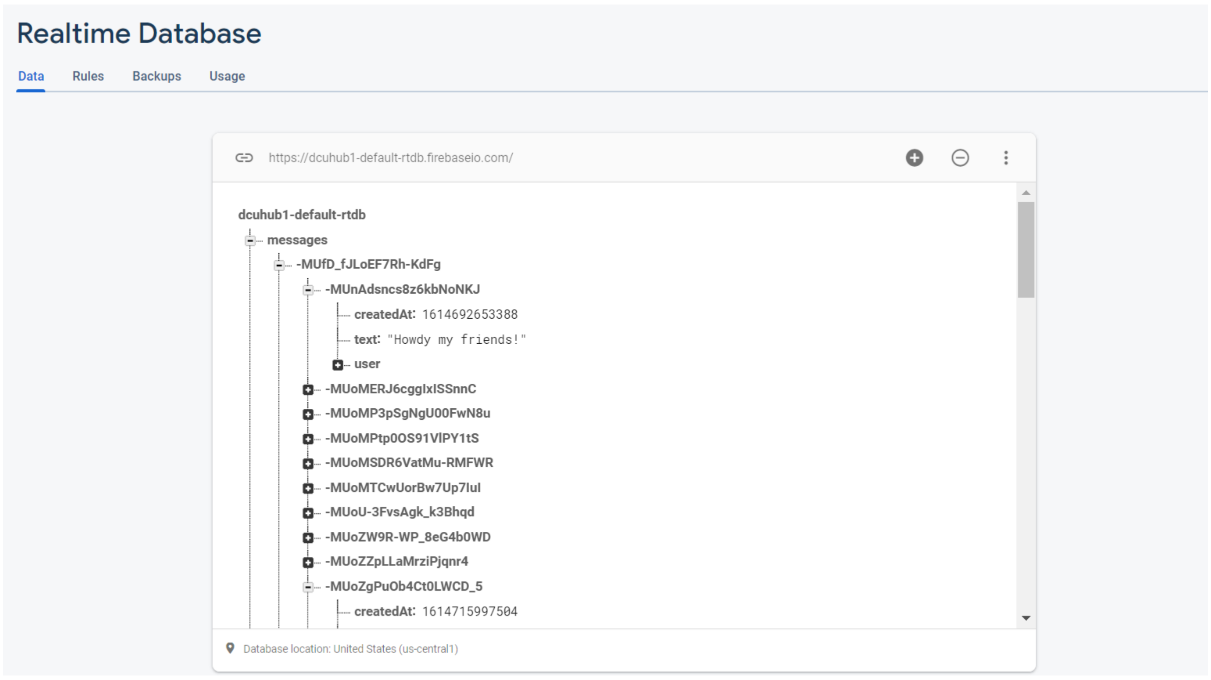
Messages and Group chat rooms as JSON objects in the Firebase Realtime Database.


Realtime Database

https://dcuhub1-default-rtdb.firebaseio.com/

```
dcuhub1-default-rtdb
  messages
  rooms
    -MUfD_fJLoEF7Rh-KdFg
      name: "CASE3"
    -MUiKrc8uAuHO5ievtk4
      name: "CASE4"
    -MUoZtDgA4Z7geE3wjHN
      name: "DO NOT ENTER"
    -MUp1tdQuWFqdRvCM9uQ
      name: "Enter Not Do "
    -MUp1yBGJ6hfse6D-fGe
      name: "This is harder than making a group name for cha..."
    -MUpG5k9E6m4vYlZPD4V
      name: "Maybe Enter"
    -MVSZpljDlCYNvJqZEqL
```

Database location: United States (us-central1)



```
dcuhub1-default-rtdb
  messages
    -MUfD_fJLoEF7Rh-KdFg
      -MUnAdsncs8z6kbNoNKJ
        createdAt: 1614692653388
        text: "Howdy my friends!"
        user
      -MUoMERJ6cggIxISSnnC
      -MUoMP3pSgNgU00FwN8u
      -MUoMPtp0OS91VlPY1tS
      -MUoMSDR6VatMu-RMFWR
      -MUoMTCwUorBw7Up7lul
      -MUoU-3FvsAgk_k3Bhqd
      -MUoZW9R-WP_8eG4b0WD
      -MUoZZpLLaMrziPjqnr4
      -MUoZgPuOb4Ct0LWCD_5
        createdAt: 1614715997504
```

Database location: United States (us-central1)

4. Problems and Resolutions

4.1 Issues

1. **User Interface Development**
    - A good user interface gives the user a pleasurable user experience. UI is one of the most important parts of our application that we put a lot of effort into. We had initial design prior to building the application so we had an idea how to design our application in code. React Native Framework is something new to us and a lot of trial and error was done in order to achieve the design we wanted. Some of the components on our initial design were implemented and some were not. We had to take into consideration the things that we can and cannot do.
    - We also decided to divide up the contents of one specific screen (Profile) as it looked too crowded which was not appealing from a user perspective.

2. **Cross Platform Deployment**
    - Deploying an application that works for both platforms (iOS and Android) was indeed a challenging one. We are both iOS users so we had to download an Android Emulator in order to get the view of our application on an Android device. The installation process was complicated enough and only one of us was able to get it working. We almost decided to stick to just getting the

application working on iOS but we still pushed through and managed to get the application working on both.

- Although React Native and Expo provide an avenue for this to work, some packages/libraries don’t function the same for both. Sometimes, the package that we needed only worked for Android so we had to look for other libraries that we can use to work on both. It took a lot of time researching until we found the one that applies on both.
- Some React Native libraries that we wanted to use weren’t compatible with Expo. We were debating whether to deviate from using Expo in the middle of the development but decided to just stick to it and work with what libraries are available for us to use instead.

3. **OpenTimetable API**
    - We were able to find a repository on github that allows us to fetch classes for a specific weekday from the DCU OpenTimetable API. However, this was written in JavaScript and we had no prior knowledge with this language. We could have used Python for creating our server but we managed to study the basics of JavaScript, Node.js and Express in order to create our own server. We implemented this github repo into our server and we were able to connect this with our front-end.
    - Another issue was that our timetable functionality works only for Android devices for some reason. For iOS devices however, our server receives the request and sends the data back but our application can’t process the data. We changed our ports and even used all the other networking libraries available on React Native but none of those worked. After some research, we figured that it might be an iOS security issue since we were hosting our server locally. We decided to host our server using Heroku instead. After the deployment on Heroku, our application works fine on both platforms.

4. **Firebase**
    - We implemented Firebase authentication into our application for when a user signs up, login or resets password. This feature was particularly easy to implement as we didn’t have to spend a considerable amount of time implementing these features as we spent the first week of our development process studying Firebase and how it works. The real problem came when we had to write user data to Firestore. We struggled to be able to write a user’s text input to firestore for when a user updated their profile. We expressed our concerns to our supervisor that we can’t figure out how because none of the articles we read online covered this feature in immense detail.
    - After spending another few days trying to figure out how to write to Firestore, we eventually implemented code that worked. We taught ourselves how to use React Native hooks and functions and this approach ultimately worked in order to allow a user to write directly to Firestore. React Native hooks and functions were quite complicated to get the hang of first, but once we understood we started implementing functions and hooks into our mobile application more. This allowed us to successfully implement the user edit profile and user profile pages, as the users previously inputted data in the edit

profile page would be rendered from firestore by using a function which was wrapped in a useeffect in order to achieve the effect that we see on many social media edit profile pages.

- Another issue we had with firestore was that when we were implementing features that would call and write to firestore, we would exceed our daily database limits and all types of firebase errors would appear on our application. We resolved this by googling online, and would have to make a new firebase app in order to continue working on our project as we exceeded the firebase free plan rules.

5. **Lack of knowledge, experience and exception handling**
    - We were completely new to mobile app development and had no prior knowledge on other languages aside from Python. During this global pandemic, it was even harder for us to communicate and do the project more efficiently. Due to these problems, we did not know the correct procedures to follow for app development. We tried to find sources that would help us and modify it to meet our needs. A considerable amount of time was wasted on this instead of studying the basics and working on other functionalities of the app.
    - We also kept on adding functionalities without considering the errors/exceptions that may arise later on. It was harder to fix the errors/exceptions in the end. We realized this midway of the development and planned on thinking ahead of the possible errors/exceptions that will arise and taking it into account as we go along.

6. **API Access**
    - We applied for access to the Twitter Developers API. We had to put in a lengthy application in order to get our own API keys to develop a Twitter Fetch program. We were still new to developing programs in javascript, so we wasted some time on simple javascript syntax as we mostly code in Python. We developed two programs that fetched Twitter tweets from various DCU platforms. The first program was not fetching the tweets live as it would save the tweets in a json file and then the program would stop. The second program would fetch the tweets live providing the program was running at the same time, but this program failed to get us previous tweets as well as live tweets.
    - We worked on the Twitter API for almost two weeks, but it just was not working. Then we discovered the oEmbed API inorder to get tweets from specific twitter accounts. We implemented this feature in our NewsFeed to get our desired output. Looking back, we wasted a considerable amount of time trying to get the Twitter API programs working to our desired output, when that time could have been used to further develop our chat feature.
    - We also decided not to implement the Instagram API into our application as we realised that both of the content posted on Twitter and Instagram from DCU related accounts were the same. This made adding this feature redundant as it would have led to the news feed containing repeated content, which we were not actually aware of when writing the functional specification.

7. **Chats**
    - Implementing the chat was a feature we really wanted to add to our application, as during the global pandemic and college being moved online we wanted to allow students to be able to communicate with each other easier like never before. We faced many problems when trying to implement a chat application in react native. Many packages and tutorials were not compatible with Expo which is an open source platform for making cross platform applications for IOS and Android. We spent a lot of time trying to figure this out, especially being new to developing in javascript and having such a short timeframe to complete this feature alongside our other features.
    - To the end of our development process, we made some progress in our chat functionality. It was quite overwhelming due to the time constraint of the project, but we finally got a chat functionality working. It allows users to create public chats and also view chats. These chats are public and this wasn't our desired outcome for the chat functionality but due to the project deadline nearing we had to settle on something that works. The chat still achieves our aim of allowing DCU students to communicate with each other especially those in each other's courses. We struggled with this feature as it was also connected to the firebase real time database which we were also new to, we were not aware of how hard a chat application would be to implement but we are proud that we got the functionality of group chats working as it was one of our main functionalities.

5. Installation Guide

5.1 How to install locally

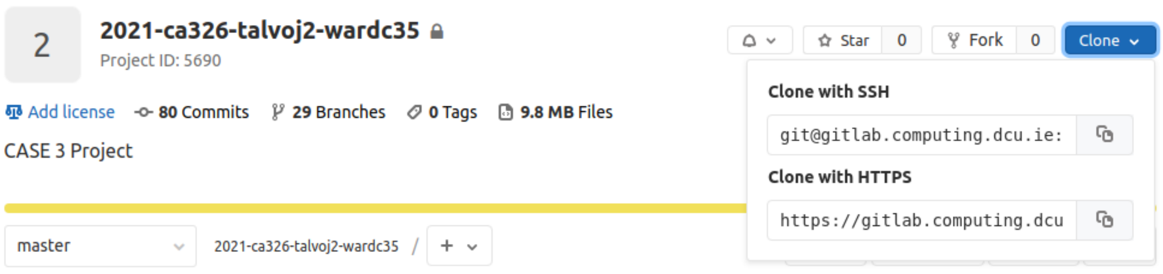
Ensure that the following are installed on your machine

- Git
- Node.js
- Node Package Manager (npm)
- Expo-cli

Ensure that "Expo Go" is installed on your device. This can be done by downloading the app on PlayStore (for Android users) and App Store (for iOS users) in order view the application from a mobile point of view. You can also use an emulator if you wish.

1. **Download the project's repository**

- Download the repo by cloning the project’s repository.
- Press the clone button to show the URL for cloning using SSH or HTTPS.
- To clone the repository using HTTPS URL, copy the HTTPS link and run the following command on your command line prompt: git clone <https url repo>


2

2021-ca326-talvoj2-wardc35

Project ID: 5690

Add license · 80 Commits · 29 Branches · 0 Tags · 9.8 MB Files

CASE 3 Project

Clone with SSH

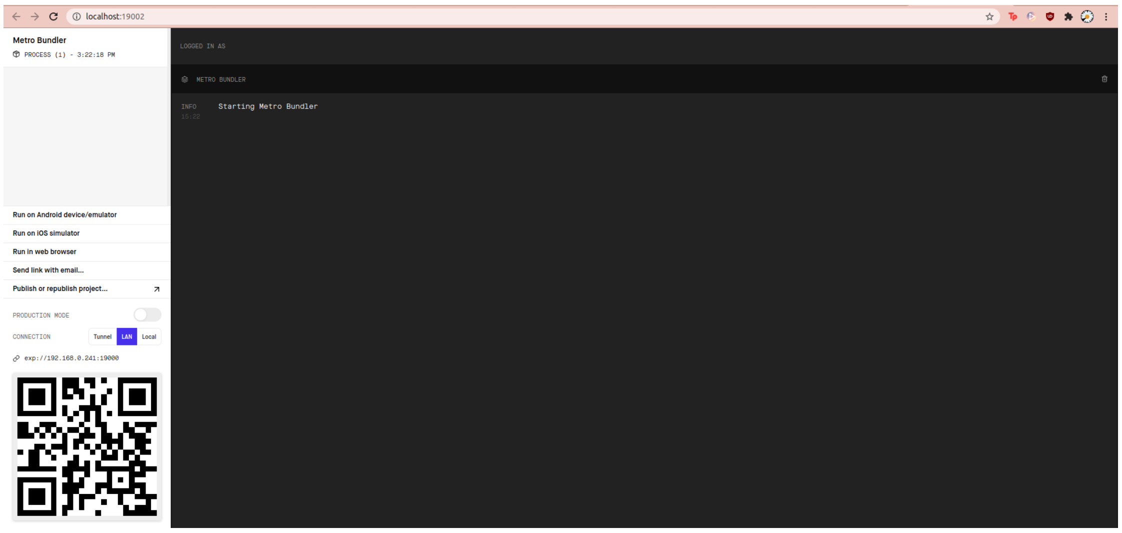
`git@gitlab.computing.dcu.ie:`

Clone with HTTPS

`https://gitlab.computing.dcu`

2. **Navigate to the project code directory**
   - All the code is located in the code directory. On your command line, type “cd code” and it should take you to the code directory.
   - Then, type “cd dcuhub” where all the code is located.

3. **Deploying the app**
   - On your command line, run “npm install”.
   - Then, run “expo start”
   - Scan the QR code that is displayed on your command line or on the expo web browser using your mobile device.
   - In case a problem arises, change the connection on the Expo server, which is deployed on your browser, from “LAN” to “tunnel” and scan the QR again.



exp://192.168.0.241:19000

```
INFO     Starting Metro Bundler
15:22
```

6. Appendices

6.1 Resources

- **Firebase**
  - https://heartbeat.fritz.ai/how-to-build-an-email-authentication-app-with-firebase-firestore-and-react-native-a18a8ba78574
  - https://medium.com/@ericmorgan1/change-user-email-password-in-firebase-and-react-native-d0abc8d21618
  - https://rnfirebase.io/reference/auth/user
  - https://blog.logrocket.com/react-hooks-with-firebase-firestore/
  - https://medium.com/@emilybartlettz/full-stack-react-firebase-tutorial-build-and-launch-real-world-app-1c51618b12a3
- **Twitter API**
  - https://developer.twitter.com/en/docs/twitter-api/v1/tweets/post-and-engage/api-reference/get-statuses-oembed
- **React Native**
  - https://gilshaan.medium.com/react-native-hooks-how-to-use-usestate-and-useeffect-3a10fd3e760c
  - https://www.freecodecamp.org/news/react-native-firebase-tutorial/
- **Chat**
  - https://heartbeat.fritz.ai/chat-app-with-react-native-part-4-create-chat-ui-screens-with-react-native-gifted-chat-7ef428a60d30
- **Expo**
  - https://docs.expo.io/
- **Node.js and Express (RESTful API)**
  - https://www.robinwieruch.de/node-express-server-rest-api
  - https://www.freecodecamp.org/news/how-to-deploy-your-site-using-express-and-heroku/