Database Design

Contents Page

Contents Page 1
Disclaimer 3
Database Models 3
Database Paths 3
Users Database 4
Users Database Structure 4
Users Database Screenshot 4
Users Database Model 4
Users Model Constraints 5
Disclaimer: 5
User Model Constraints 5
Users Database Field Information 6
Following Database 7
Following Database Structure 7
Following Database Screenshot 7
Following Database Model 7
Posts Database 8
1.) PostsUserHasLiked Collection 8
PostsUserHasLiked Collection Structure 8
Database Screenshot 8
2.) userPosts Collection 9
UserPosts Collection Structure 9
Database Screenshot 9
Database Model for Posts Collection 9
PostData Database 10
PostData Database Structure 10
PostData Database Screenshots 10
PostData Screenshot 10
PostData Comments Collection Screenshot 11
PostData Likes Collection Screenshot 11
PostData Database Model 12
PostData Model Constraints 12
PostData Database Field Information 13

Database Design

PostTags Database 14
PostTags Database Structure 14
PostTags Database Screenshot 14
PostTags Database Model 14
PostTags Model Constraints 15
PostTags Database Field Information 15
Bibliography: 16

Database Design

Disclaimer

Database Models

Within this document, there are multiple database models for different collections within this projects firebase database.

However, the approach we decided to use to implement a design was based on the firebase source cited inside the bibliography.

Database Paths

This document also outlines different paths for different collection we have inside our firebase database which looks something like this; `/users/$uid.`

The rules for defining these paths are as follow:

1.) "/collection …." – any text followed immediately by a " / "is the name given to a collection.
2.) "/$documentIDVarialbe" …. – anything followed by a "$" is a document and the text assigned after, the "$" symbol is the variable name for the documentID.

For example, consider a chat application that allows users to store a basic profile and contact list. A typical user profile is located at a path, such as `/users/$uid`. The user `alovelace` might have a database entry that looks something like this:

```
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      "contacts": { "ghopper": true },
    },
    "ghopper": { ... },
    "eclarke": { ... }
  }
}
```

Figure 1 From Firebase Source

Database Design

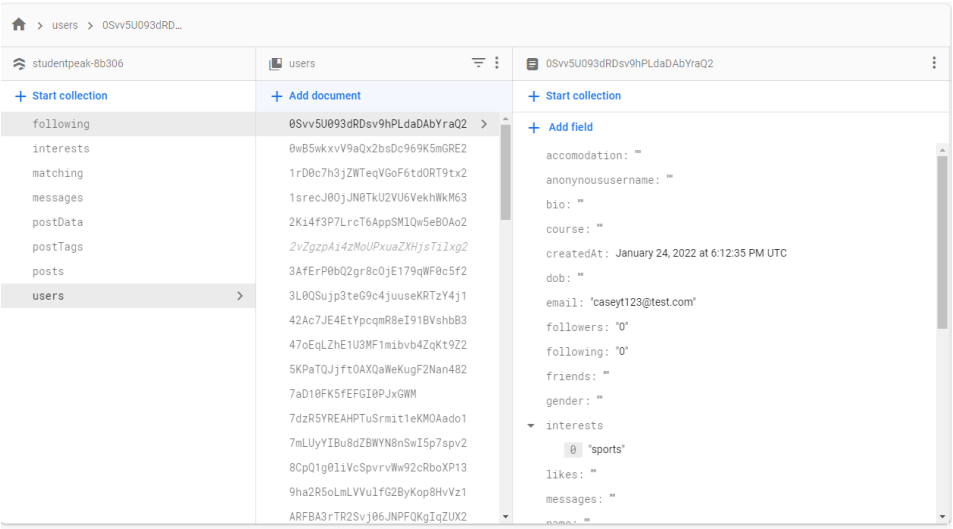
Users Database

Users Database Structure

The details below represent the “users” firebase database which is structured from the following path format; **“/users / $uid”.**

The users database is a collection which is populated with documents which are named after the userID correlating to the user the document is about. This document is then populated with fields regarding information about the user.

Users Database Screenshot



Users Database Model

```
{
    "users": {

        "userID...": {
            "email" : "...",
            "password" : "...",
            "name" : "...",
            "surname" : "...",
            "dob" : "...",
            "gender" : "...",
            "anonynousername" : "...",
            "username" : "...",
            "course" : "...",
            "yearofstudy" : "...",
            "stage" : "...",
            "bio" : "...",
            "topics" : "...",
            "friends" : "...",
            "photos" : "...",
            "messages" : "...",
            "accomodation" : "...",
            "stayaround" : "...",
            "nationality" : "...",
            "placeofstudy" : "...",
            "followers" : '0',
            "following" : '0',
            "videos" : "...",
            "likes" : "...",
            "societies" : "...",
            "profileimage" : "...",
        }
    }
}
```

Database Design

Users Model Constraints

Disclaimer:

This database was designed by group 1 however, the database isn't implemented correctly and doesn't follow expected database protocols like naming conventions and using relevant datatypes for storing fields.

The field names designed in the document cannot be changed because this has to match the already implemented database fields currently existing within the application.

However, the datatypes for these fields we have made corrections in this document outlining how these fields should be stored in the database but, this does not equate to their actual assigned datatype designed by group 1 in the firebase database.

User Model Constraints

- **email** – is wrapped in the format of a string. However, this field being an email has 4 following criteria's which must be met in the same format being; a **username** (string.)**, an @ symbol** (symbol)**, a domain name** (string.)**, a dot** (ascii symbol)**, and the domain** (string.).

- **name –** should have the datatype of a string.
- **surname –** should have the datatype of a string.
- **dob –** should have the datatype of a timestamp.
- **gender –** should have datatype of a string.
- **anonynoususername –** should have datatype of a string.
- **username –** should have datatype of a string.
- **course –** should have datatype of a string.
- **yearofstudy** – should have datatype of a number.
- **stage –** should have datatype of a number.
- **bio –** should have datatype of a string.
- **topics –** should have datatype of an array.
- **friends –** should have datatype of a number.
- **photos –** N/A
- **messages** – should have datatype of a string.
- **accomodation** – should have datatype of a string.
- **stayaround** – should have datatype of a string.
- **nationality** – should have datatype of a string.
- **placeofstudy** – should have datatype of a string.
- **followers –** should have datatype of a number.
- **following –** should have datatype of a number.
- **videos –** N/A
- **likes –** N/A.
- **societies –** should have datatype of an array.
- **profileimage –** should have datatype of a string being an URL.
- **createdAt** – should have datatype of a timestamp.

Database Design

Users Database Field Information

- **email** - this field stores the email address of the user.
- **name** - this field stores the first name of the user.
- **surname** - this field stores the second name of the user.
- **dob** - this field stores the date of birth of the user.
- **gender** - this field stores the gender of the user.
- **anonynoususername** - this field stores username the user has been assigned for chatting in the anonymous sections of the application.
- **username –** this field stores the user's username.
- **course –** this field stores the course the user is studying.
- **yearofstudy** – this field stores the current year the user is studying at university.
- **stage –** this field stores the current stage the user is currently at during university etc; 4.
- **bio –** this field stores the user's bio.
- **topics –** this field stores the an array of topics the user is studying.
- **friends –** N/A
- **photos –** N/A
- **messages** – N/A
- **accomodation** – this field stores the accomodation the user is accommodated in.
- **stayaround** – N/A
- **nationality** – this field stores the nationality of the user.
- **placeofstudy** – N/A.
- **followers -** this field stores the number of follows the user has.
- **following –** this field stores the number of accounts a user is following.
- **videos –** N/A
- **likes –** N/A.
- **societies –** this field stores an array of societies the user is interested in.
- **profileimage –** this field stores the profile image (URL) of the user.
- **createdAt** – this field stores the timestamp of which this account was created at.

Database Design

Following Database

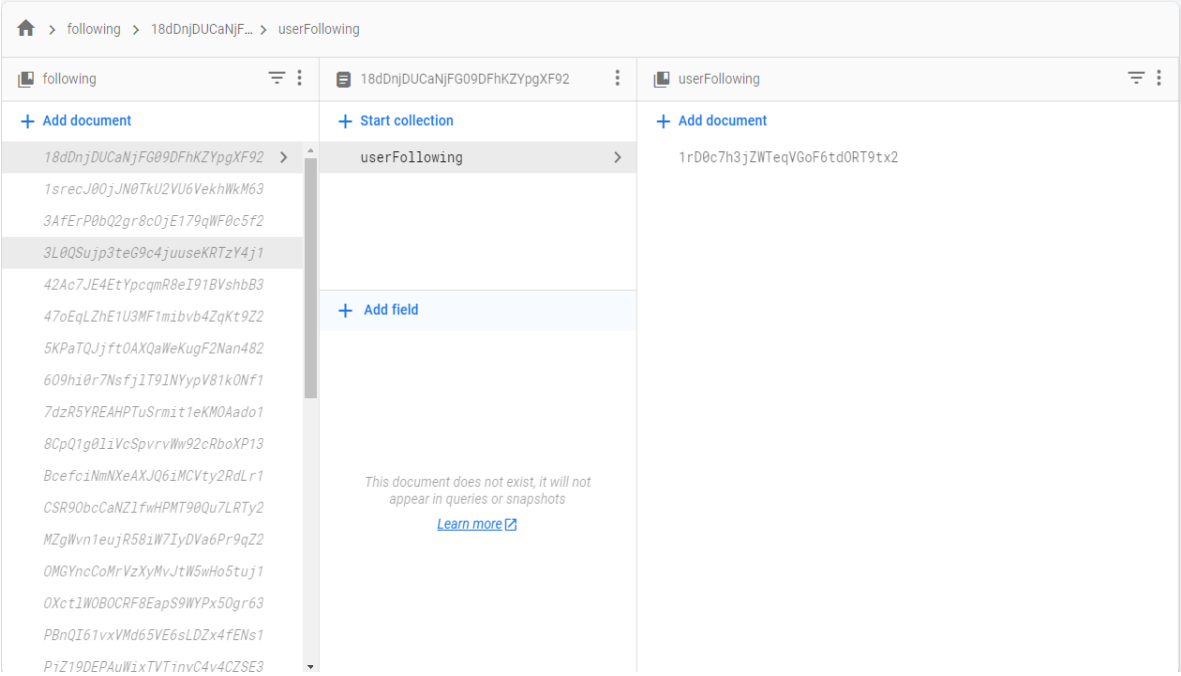
Following Database Structure

The details below represent the “following” firebase database which is structured from the following path format; **“/following / $userID / userFollowing”.**

The following database is a collection which is populated with documents which are assigned a userID value which correlates to the user in the database which this collection regarding who this is user is following is about.

This document has a collection inside of it called “**userFollowing”** which is populated with a bunch of documents which are assigned the value of the userID that correlates to the user this user is following.

Following Database Screenshot



Following Database Model

```
{
    "following": {

        "userID": {
            "userFollowing": {
                "userID" : {}
            }
        }
    }
}
```

Database Design

Posts Database

The “posts” firebase database is split into 2 collections:

1.) PostsUserHasLiked Collection

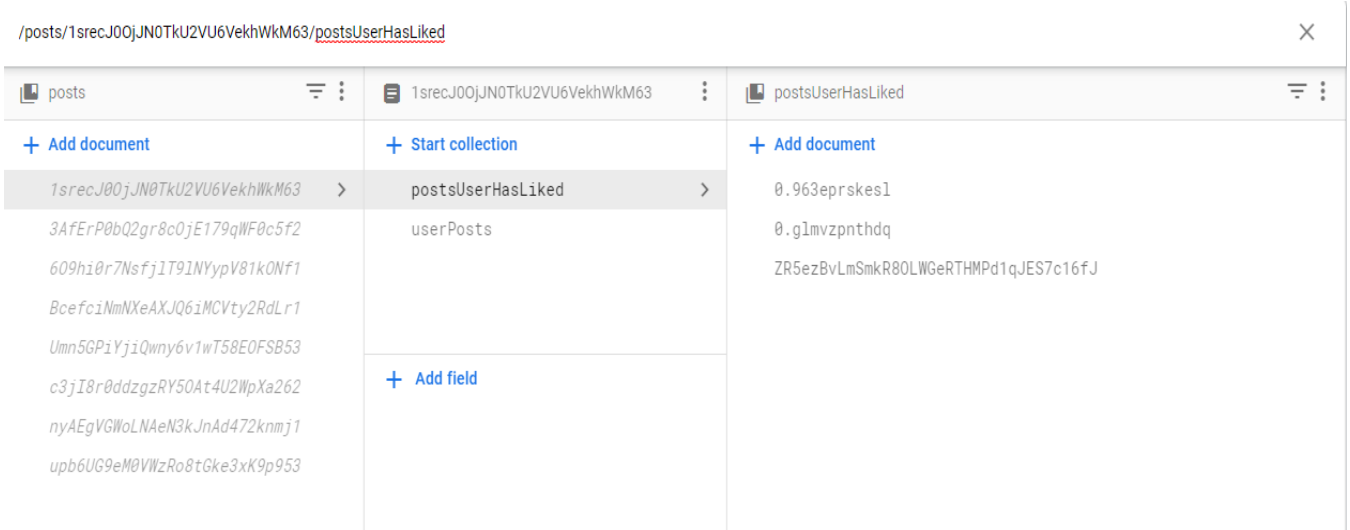
PostsUserHasLiked Collection Structure

The details below represent the “postsUserHasLiked” collection inside of the firebase database called “posts” which has following path format: **“/posts / $uid / postsUserHasLiked”.**

From the posts database which is a collection that is populated with documents which are named after the userID, correlating to the user the document is about.

Then inside this document is a collection called “**postsUserHasLiked”** which is populated with a bunch of documents which are assigned the value of the postID that correlates to the post the user liked.

Database Screenshot



Database Design

2.) userPosts Collection

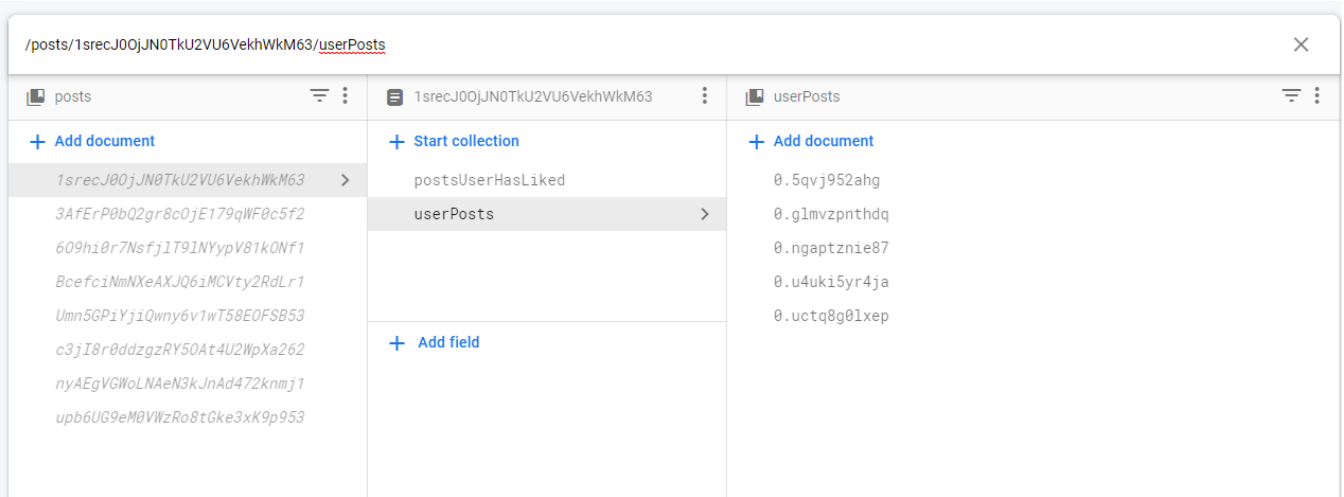
UserPosts Collection Structure

The details below represent the “userPosts” collection inside of the firebase database called “posts” which has following path format: **“/posts / $uid / userPosts”.**

From the posts database which is a collection that is populated with documents which are named after the userID, correlating to the user the document is about.

Then inside this document is a collection called “**userPosts”** which is populated with a bunch of documents which are assigned the value of the postID that correlates to the post the user created.

Database Screenshot



Database Model for Posts Collection

```
{
    "posts": {

        "userID": {

            "postsUserHasLiked": {
                "postID" : {}
            },

            "userPosts": {
                "postID" : {}
            }
        }
    }
}
```

Database Design

PostData Database

PostData Database Structure

The details below represent the “postData” firebase database which is structured from the following path format; **“/postData / $postID”.**

The postData database is a collection which is populated with documents which are named after the postID correlating to the post the document is about. This document is then populated with fields regarding information about the post.

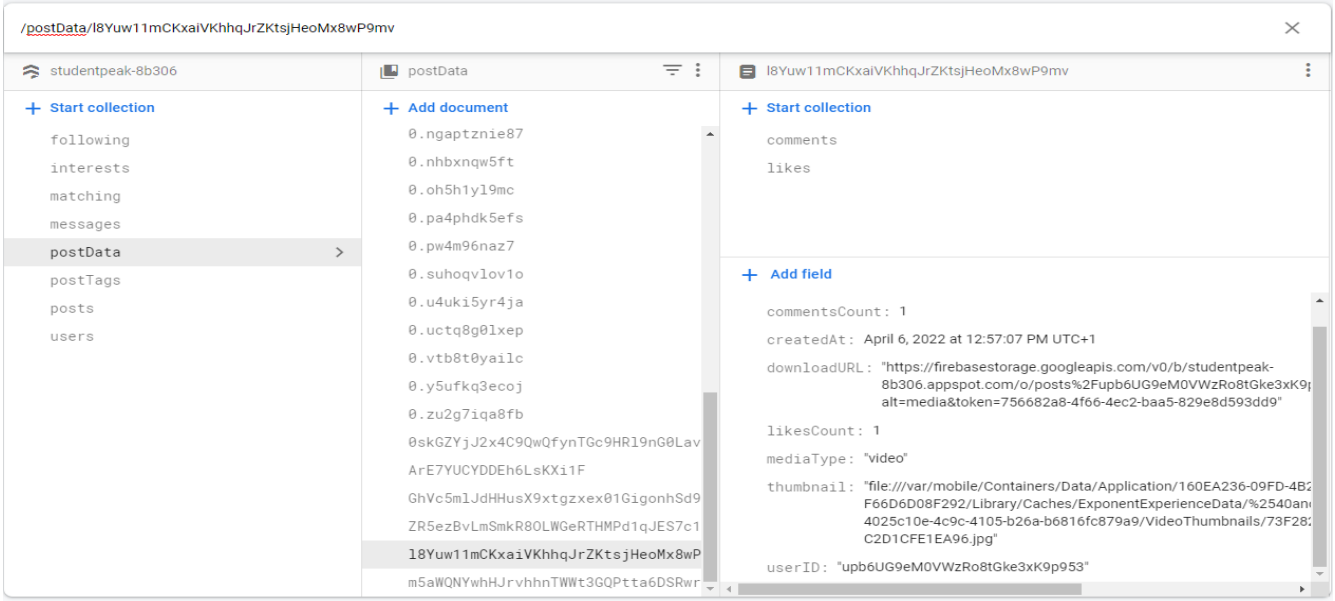
However, there maybe potentially 2 collections inside this document being the;

1.) **Likes** – this collection is then populated with documents which are named after the userID of the user who this document correlates to who liked this post.

2.) **Comments** – this collection is then populated with documents which are name randomly as the doc is generated and correlate to a comment which was made by a user. However, each document is populated with some fields regarding the information of the comment made.

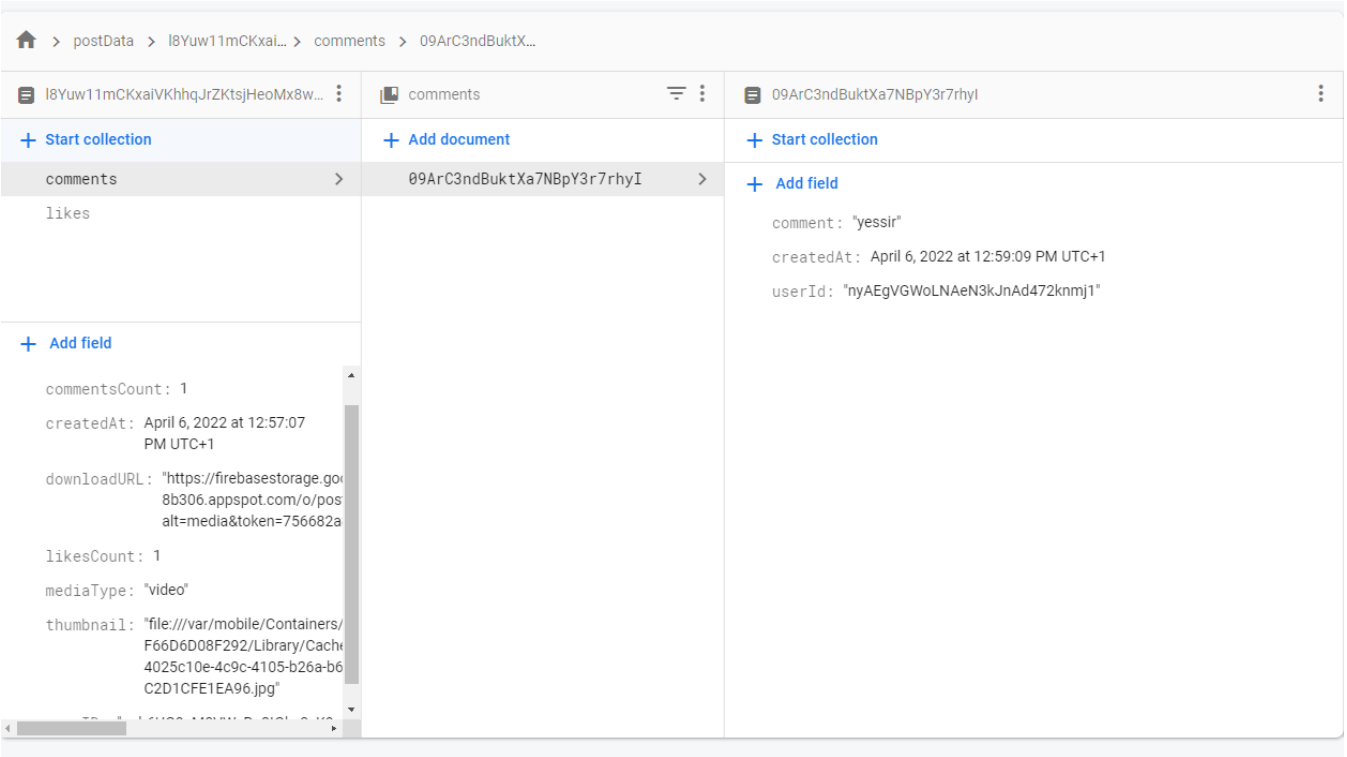
PostData Database Screenshots

PostData Screenshot

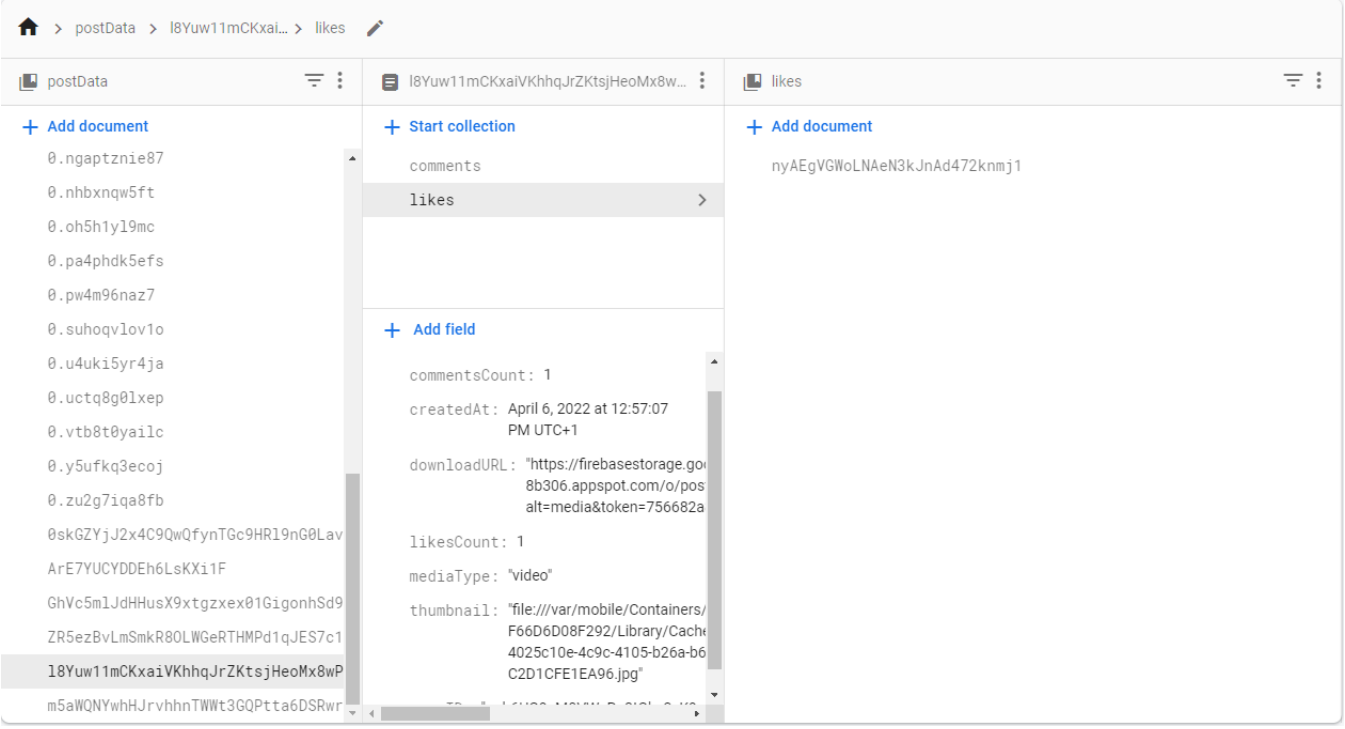


Database Design

PostData Comments Collection Screenshot



PostData Likes Collection Screenshot



Database Design

PostData Database Model

```
{
    "postData": {

        "postID": {

            "likes": {
                "userID" : {}
            },
            "comments": {
                "commentID" : {
                    "comment" : "...",
                    "createdAt" : "...",
                    "userId" : "...",
                }
            },

            "caption" : "...",
            "commentsCount" : "...",
            "createdAt" : "...",
            "downloadURL" : "...",
            "likesCount" : "...",
            "mediaType" : "...",
            "userID" : "..."
        }
    }
}
```

PostData Model Constraints

- **caption –** has the datatype of a string.
- **commentsCount –** has the datatype of a number.
- **createdAt -** has the datatype of a timestamp.
- **downloadURL -** has the datatype of a string.
- **likesCount -** has the datatype of a number.
- **mediaType -** has the datatype of a string.
- **userID - –** has the datatype of a string.

Database Design

PostData Database Field Information

- **caption -** this field stores the caption of the post correlating to this document has.

- **commentsCount -** this field stores the numerical count value for the number of comments the post correlating to this document has.

- **createdAt -** this field stores a timestamp value for the date of creation for the post correlating to this document.

- **downloadURL -** this field stores the download URL for the media file this post correlates to.

- **likesCount - -** this field stores the numerical count value for the number of likes the post correlating to this document has.

- **mediaType -** this field stores the mediaType (picture/video) for the media file this post correlates to.

- **userID** - this field stores the userID of the user who created this post which this document correlates to.

Database Design

PostTags Database

PostTags Database Structure

The details below represent the “postTags” firebase database which is structured from the following path format; **“/postTags /Picture/ posts /”.**

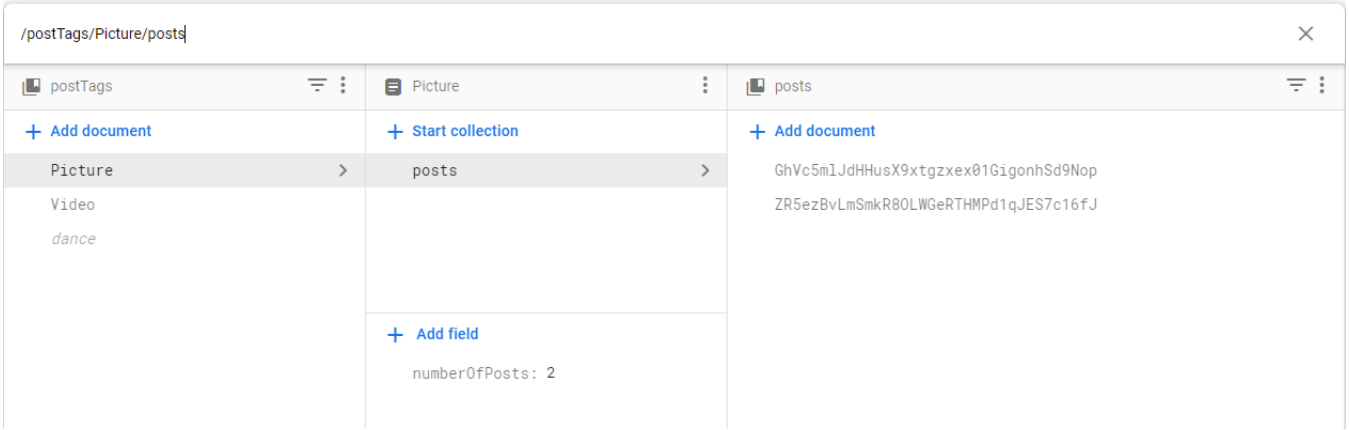
The postTags database is a collection which is populated with documents which are named after the postTagID correlating to the postTag the document is about.

This document is then split into 2 parts:

1.) The document itself which is then populated with 1 field.
2.) Another collection which is called “post”

Inside the “posts” it is populated with documents which are named after the postID correlating to the post the document is about.

PostTags Database Screenshot



PostTags Database Model

```
{
    "postTags": {

        "postTagID": {

            "numberOfPosts" : "...",

            "posts": {
                "postID" : {}
            }
        }
    }
}
```

Database Design

PostTags Model Constraints

- numberOfPosts – has the datatype of a string.

PostTags Database Field Information

- numberOfPosts – this field stores the numerical value of the number of posts the postTag correlating to this document has.

Bibliography:

Firebase. (2019). *Structure Your Database | Firebase Realtime Database | Firebase*. [online] Available at: https://firebase.google.com/docs/database/web/structure-data.