Firebase Universität Trier

Document

This document describes how the data will be stored, structured and explains the different features of firebase which will be used in the Crayon application.

Privacy

Humans tend not to share personal information with unknown individuals or companies.
The EU describes personal data as follow:
"Personal data is any information that relates to an **identified or identifiable living individual**. Different pieces of information, which collected together can lead to the identification of a particular person, also constitute personal data." (What is personal data? 2018, paragraph 1). Thus every information on an individual's passport or identity card is personal information. In the digital world, the range of personal information is increased with an email address, Internet Protocol address, and phone number.
Privacy by Design is an approach to build systems that ensures that individuals' privacy is respected at the beginning and throughout the project's construction. Data minimization is a process to minimize personal data collection. The Information Commissioner's Office from the United Kingdom (Principle (c): Data minimisation 2021)describes Data minimization in three key terms:

- adequate - sufficient to properly fulfill your stated purpose;
- relevant - has a rational link to that purpose; and
- limited to what is necessary - only store data that an application needs

Firebase

Firebase is Google's "All-In-One Backend Solution" for mobile and web applications. Firebase provides Software Development Kit (SDK) tools and infrastructure, which allows developers to use efficient APIs to ease out the development process. The key features from Firebase which will be used in the Crayon App are:

- Authentication
- Database
- Cloud Storage
- Scale

Authentication

Authentication is one of the critical security issues of modern applications. Firebase provides a particular service for authentication purposes which is called Firebase Auth. Firebase makes the security updates to maintain the highest security standards for user privacy. In addition, there are two ways to authenticate with Firebase:

- Phone number
- Email
- Username

Phone number authentication, allows an individual to authenticate with his personal phone number. Firebase will send a verification code to the user's phone where he in return has to enter the verification code to access a given application.
On the other hand, email authentication is the most commonly used; it is usually accompanied with a password. Emails are verified by sending an email to the registration email with a specific link. This link needs to be opened by the user and the email is flagged as valid. Both processes allow verification which allows to discard non-human interaction with a given application.
The final authentication is the username with a password which is currently dying out since there can't be a verification process to check whether the user is a computer program or a human.

Database

Google provides two types of databases:

- Cloud Firestore
- Realtime Database

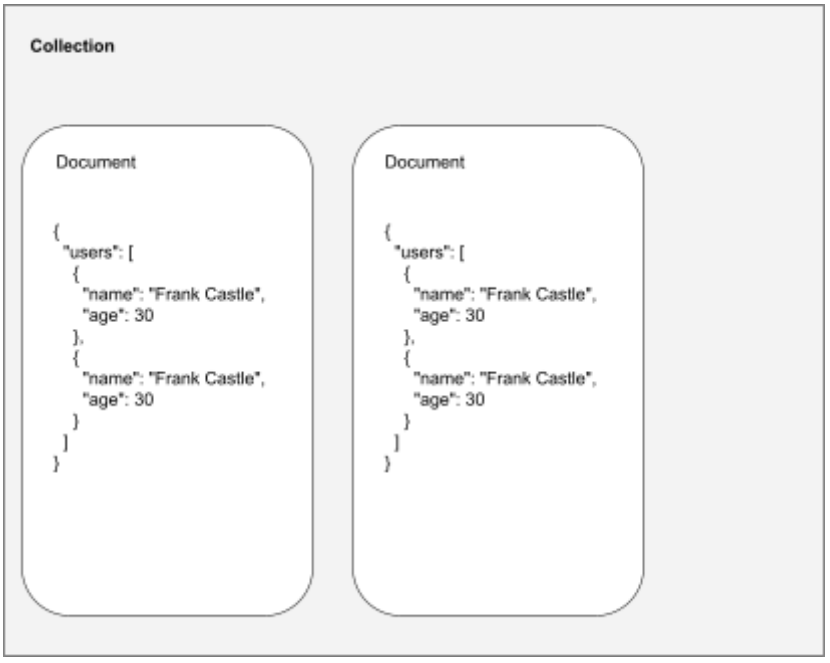
Both Cloud Firestore and Realtime Database are cloud-hosted NoSQL databases. These databases are specifically designed to create schemaless databases. SQL databases have a predefined structure that can hardly be altered depending on the project's needs. NoSQL solves this issue by storing the data in a JavaScript Object Notation format whose acronym is **JSON**.

Realtime Database stores the JsonObjects in one big **JSON** object, which the developers can control in real time. The illustration **[4],** is an example of a big JSON file. Naturally, developers would split up chat messages into multiple files to create a hierarchical data structure and benefit from a better overview of the data. In this case, Realtime databases will not accept a better structure.

Multiple Chats in one big JSON file **[4]**

In case a user of an application pulls the data from the Realtime database API each time the data changes the given application will be notified of the changes of the data.

On the other hand, Firestore allows the same and more functionalities than the Realtime database solution. The data isn't stored in one huge JSON object but in a much more structured approach. The data is subdivided in collections and documents. Documents can have a maximum size of 1 MB if the size is exceeded the JSON object can not be stored. The collections contain documents and the documents contain data and can point to other subcollections.



The additional data structure allows firestore to have important advantages to perform queries. Firestore queries are shallow, which allows the retrieval of a single document without retrieving any other linked subcollections. Moreover, Realtime

database does not allow querying over multiple fields which means the data needs to be denormalized, meaning that a json object requires an additional field where the query needs to be performed on. Firestore does not need such a denormalizing process of data and thus allows querying over multiple fields. However, querying over multiple fields in firestore can not be done by default. Firestore by default indexes each field to increase querying speeds. Composite indexing is required to perform multiple fields querying which must be set up by the developer.

Cloud Storage

A firebase customer isn't required to purchase hardware such as harddrives or ssd to store their data online. The cloud storage has close to no limits of storage and accepts every binary file.

Scale

In a real world scenario, a complex application makes Post requests on a server to retrieve data. However, if the application gains more and more users the backend servers tend to get slower to process the requests which results in slower response times. One countering measure is to add more servers to minimize the workload on each server. This particular problem was normally solved by the developers by writing scaling code which is more complicated. Firebase removes the scaling problem, by providing an infrastructure which scales automatically, thus allowing the developer to focus on their primary goal, the success of the application.

Pricing

Firebase has different pricing types namely Spark and Blaze.
The spark plan is free however if the application consumption increases a firebase user is required to use the blaze plan. The Blaze plan also named “Pay as you go” can have a high price if the developer doesn’t structure the data correctly.

	Free quota per day	Price beyond the free quota (per unit)	Price unit
Document reads	50,000	$0.06	per 100,000 documents
Document writes	20,000	$0.18	per 100,000 documents
Document Deletes	20,000	$0.02	per 100,000 documents

Firebase pricing [1]

From the table [1], the different CRUD operations such as reading, writing and deleting a document have different pricing ranges. Reading is free up to 50.000 documents per day and writing or deleting is free for 20.000 documents per day. If the application has a higher consumption the administrator/application owner has to pay for additional tranches of read or/and write operations. Thus, adding additional document reads (50.000) will cost 0.06 $ and afterwards 100.000 for each extra tranche. The writing to documents is 3 times higher than document reads. Finally, the cheapest operation is deleting files which only costs 0.02$.

	Free quota per day	Price beyond the free quota (per unit)	Price unit
Storaged Data	1GB storage	$0.18	GB/Month

The application not only requires documents but also cloud storage to store images and the respective lecture slides. Storing 1Gb of data is free, adding additional storage will each time cost $0.18 per GB.

Firebase Universität Trier

Firebase Pricing Simulation

The university of trier has approximately 15.000 students according to timeshighereddiction. On average a student makes 30 credits per semester where 5 credits are assigned to one course. As a result a student has six lectures per semester, each one of these lectures might also have one exercise session per week. In addition, one semester has 15 weeks of courses according to Southern Utah University.

If each one of the 15.000 students would use the application in this specific scenario each student would have 12 lectures((30/5)*2) per week. Moreover, a semester has 15 week resulting in 180 lectures. This would result in at least 180 document reads if the student only opens the application only for lecture purposes. 180 documents * 15.000 students will result in 2.7million document reads per semester. In addition, each lecture contains pdfs which needs to be downloaded where each pdf has 1 MB on average. The downloads for pdfs would be 12 MB per student and in total 15 GB.

This scenario in document reads would cost $1.62 and for storage $2.7. The overall cost isn't high. However as soon as we use the realtime changes on document these prices can skyrock.

The next scenario

Which data is required?

Management tool

The management tool will only be used by a specific group of users, namely teachers or lecture givers. A primary login and registration process is, therefore, a requirement. There are several ways to implement an authentication process. First of all, phone numbers are more and more in use in modern-day applications. It is easier for the user since he does not need to remember a password. *A password study was conducted by HYPR which states that 78% of people had to reset a password they forgot in the past 90 days [3]*. Phone authentication would allow a majority of users not to remember a password and reset their password. However, data minimization states that an application should only need what is required and thus, the Crayon application will use email authentication.
The teacher's registration information is the role that will be a professor and will be set automatically. In addition, the required fields are the name, email, and password. The email and password will be used as login information.

```
{
  "UID": "Unique_ID",
  "role": "Teacher",
  "name": "Nicolas Tesla",
  "email": "nicola_tesla@edu.com",
  "password": "ElonUsesMyTeck13"
}
```

Registration Information

The Dashboard contains lectures that one specific teacher gives. The JSON object has a unique id of the lecture, allowing the students to identify a specific lecture and enroll in the course. In addition, the lecture can contain an image that describes the course and the last PowerPoint title. Moreover, a lecture has two types:

- Lecture (Theoretical session)
- Exercise (Practice session)

The lecture and exercise have a room and date times on which dates the respective lectures are happening.

```
{
  "id":"Unique_ID_LECTURE",
  "PID":"Unique_ID_PROFID"
  "title": "Operating Systems",
  "last_power_point_title": "Introduction",
  "image": "image/path/operatingsystems",
  "dates": [
    {
      "room": "B12",
      "day": "monday",
      "starting_time": "12:00",
      "ending_time": "14:00",
      "type": "lecture"
    },
    {
      "room": "A2",
      "day": "friday",
      "starting_time": "15:00",
      "ending_time": "16:30",
      "type": "exercise"
    }
  ]
}
```

Lecture Information

A lecture requires slides for presentation purposes which are stored on the firebase filesystem. Additional information needs to be added to the lecture data. One of them is the title of a given presentation and the respective file path to the lecture slides. This needs to be stored in an array since a lecture can have multiple slides.

```
{
  "pdfs": [
    {
      "ids": "Unique_ID",
      "title": "Introduction",
      "path": "pdfs/courses/operatingsystems",
      "date_of_creation": "11.10.2021-13:43:45"
    }
```

```
  ]
}
```

Lecture Slides Information

The lecture giver can also start a quiz during a lecture. The quiz requires a unique identifier to let the students participate. A quiz session has a title and can contain multiple questions which can have multiple responses. A response is completed by a boolean value that describes if a response is correct or a wrong answer.

```
{
  "id": "Unique_ID",
  "title": "Introduction Quiz",
  "questions": [
    {
      "question_ID": "Unique_ID",
      "title": "What is an Operating System?",
      "responses": [
        {
          "response": "Response 1",
          "is_Response": true
        },
        {
          "response": "Response 2",
          "is_Response": false
        }
      ]
    }
  ]
}
```

Quiz Information

Students can respond to quizzes started by the teacher. Responses from students contain the responses to the questions and the time taken to respond to each question. To ensure the anonymity of a student, the student can choose his username. .

```
{
  "responses": [
    {
      "userName": "Flying_Car",
      "UID": "Unique_ID",
```

```
      "responses": [
        {
          "question_ID": "QID",
          "time_taken": "10s",
          "was_timed_out": "false",
          "response": "(a): Response 1"
        }
      ]
    }
  ]
}
```

Student Response Information

Student application

The student application uses most of the data created by the teacher. On the registration level, the user can also provide no name. The role will be set automatically for students. The Name is not a required field for the student's privacy. In addition, the email is required for verification purposes. This email will not be provided to other users nor the professor,

```
{
  "UID":"Unique_ID",
  "role": "Student",
  "name": "Evergreen Rocky",
  "email": "evergreenRock@gmail.com",
  "password": "123456"
}
```

The userId is used to guarantee the uniqueness of the names. Moreover, the username can be any name chosen by the student and is valid for the entire quiz. The response is an array of objects containing the timeTaken, which will be used for scoring a student, and the questionId describes to which question the student answered the question. The response string contains the answer clicked by the user.

```
{
  "UID": "Unique_ID",
  "userName": "Flying Car",
```

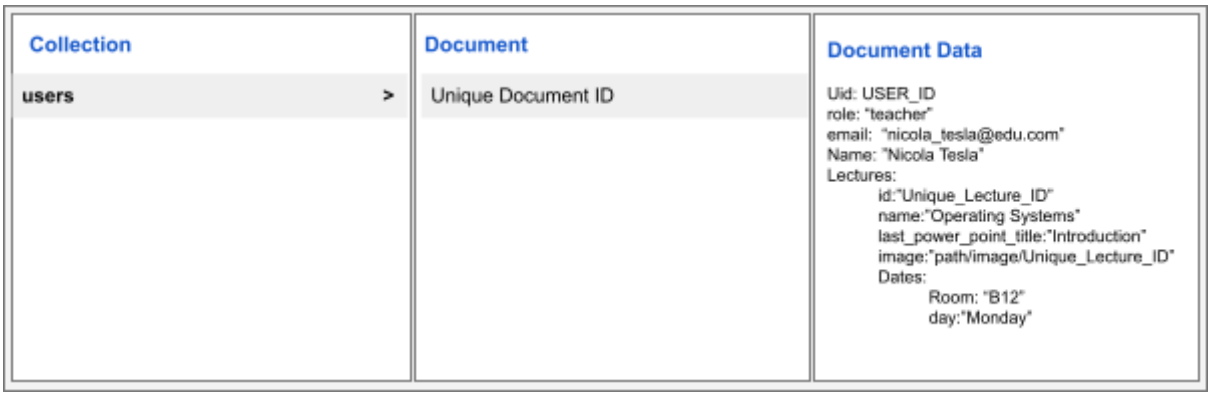
```
  "responses": [
    {
      "response": "Response 1",
      "timeTaken": "10s",
      "was_timed_out": "false",
      "QID": "Unique_Question_ID"
    }
  ]
}
```

Data design

The password and email will be stored in firebase authentication. The unique user id will be generated by firestore.

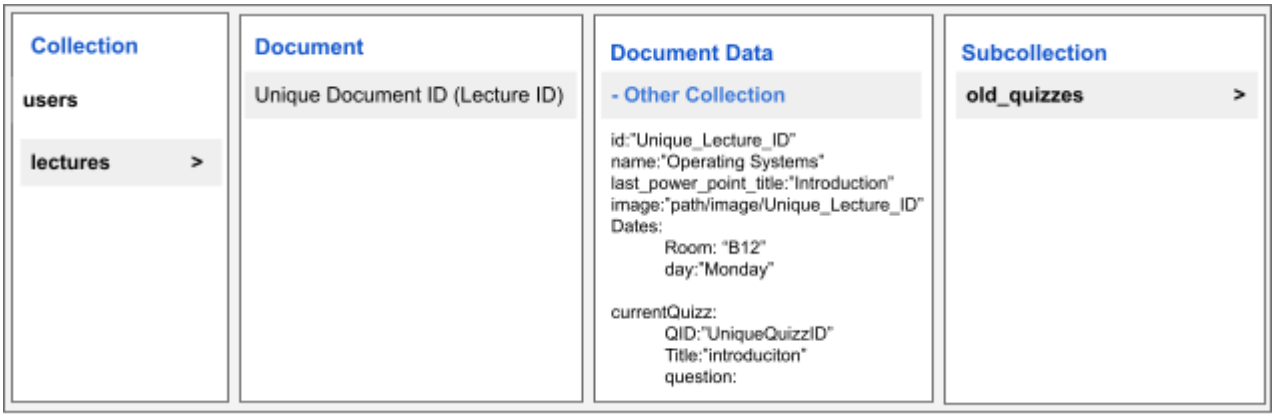
```
{
  "User UID": "Unique_ID",
  "email": "nicola_tesla@edu.com"
}
```

A teacher has a profile that contains all data with the lecture data. The user_ID can, in this specific case, be used as a unique document id since firestore guarantees the uniqueness of user ids.
Student's profiles will be stored in the same user's collection. However, students' information will not contain any personal information except the email address. Cloud firestore allows users to set rules which users can access or modify a given document. The access to personal data will be restricted if any other student wants to watch another student's personal information.



Teacher data Template

There are two methods of storing the quiz data. One of them is to store the quiz data directly into the lecture document itself. The illustration **[5],** shows the storage of a quiz in the same lecture document. This has one drawback: unnecessary or unchanged data will be loaded.



Data design [5]

On the other hand, storing the data into a separate subcollection will not lead to downloading unchanged data which will then in return reduce cost of operation. Both solutions contain a subcollection of old_quizzes which contains the already executed quizzes.