

**İSTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND
INFORMATICS**

**Data Synchronization with CouchDB and
PouchDB for Offline-first Application**

Graduation Project Report

**Lina Fahad A Alrehaili
150160930**

**Department: Computer Engineering
Division: Computer Engineering**

Advisor: Lect. H. Turgut Uyar

May 2019

**İSTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND
INFORMATICS**

**Data Synchronization with CouchDB and
PouchDB for Offline-first Application**

Graduation Project Report

**Lina Fahad A Alrehaili
150160930**

**Department: Computer Engineering
Division: Computer Engineering**

Advisor: Lect. H. Turgut Uyar

May 2019

Statement of Authenticity

I hereby declare that in this study

1. all the content influenced from external references are cited clearly and in detail,
2. and all the remaining sections, especially the theoretical studies and implemented software/hardware that constitute the fundamental essence of this study is originated by my individual authenticity.

Istanbul, 25.05.2019

Lina Fahad A Alrehaili



Data Synchronization with CouchDB and PouchDB for Offline-first Application

(SUMMARY)

The main objective of this graduation project is to achieve data synchronization to keep the local and server data synchronized with CouchDB and PouchDB for an offline-first application. When an application is offline-first, PouchDB will keep the data in a local database until the application is online. When the application is online, the local database within PouchDB will synchronize with CouchDB or with IBM Cloudant ending up with a better user experience both online and offline. It is an amazing advantage because all the data operations as (create, read, update and delete) happens locally on the device, which means the operations work easier and faster. Furthermore, it reduces the limitation of use for the application and can get the latest up-to-date information when the user is offline.

The main target of the combination of PouchDB and CouchDB is to accomplish data synchronization between different devices. For this reason, we need to produce the data in a local database, when the user is online the data is being synced within milliseconds to the server data, and if a user is offline then the data will be synced whenever the network connection is recovered.

When building a synchronized offline-first application, the goal is to make sure the application works online or offline and to make sure the local data sync to the remote data server without any data loss. The application should synchronize reliable data from the local database, and to able to establish data encryption for the chosen target of our database. Hence, there are many ways you can synchronize your application when you are offline. For my final project, I focused on different tools of CouchDB, PouchDB and IBM Cloudant to establish the synchronization mechanism.

The final mobile application I have built is “Fitness Application”, this application will take your workout data that is your workout name, an image snap of your workout, the date of your workout, and the number of workouts done. This data is pushed to the local database of PouchDB and when the network is online is will synchronize it with CouchDB or IBM Cloudant. Furthermore, the application provides a full analysis of your data with charts of your workouts that are done with the same day, and analysis charts of your overall all-time workouts.

The complete work of the final project performed in two semesters in the year of 2018-2019, with a group of tasks to complete the final project. The first task was to research and understand CouchDB, PouchDB and IBM Cloudant service. Comprehensive understanding of the ionic framework of how I can use it to build a mobile application, the steps applied to do the configuration, understanding how the remote Cloudant database achieves the synchronization goal, and at the end to build an offline-first synchronized mobile application. As a result, all my analysis, design and implementation are all included in this report.

Contents

STATEMENT OF AUTHENTICITY	iii
SUMMARY	iv
CONTENTS	v
LIST OF FIGURES.....	vi
LIST OF CODES	vii
LIST OF TABLES	viii
1 Introduction and Project Summary	1
2 Comparative Literature Survey	2
3 Developed Approach and System Model	5
3.1. Mockups	5
3.1.1. Slide Navigation Bar	5
3.1.2. My Workouts Page	6
3.1.3. Analysis Page	7
3.1.4. Snaps Page.....	7
3.1.5. Workout Videos Page	8
3.1.6. About Page	8
3.2. Data Model	9
3.2.1. Document Structure	9
3.3. Structural Model	10
3.3.1. Configuration.....	10
3.3.2. Code and Algorithm.....	12
3.3.2.1. Database Provider.....	12
3.3.2.2. The Add Component	15
3.3.2.3. Others.....	17
3.4. Dynamic Model	18
3.4.1. Client-Server Model.....	18
3.4.2. Data Flow Diagram.....	19
4 Experimentation Environment and Experiment Design.....	20
4.1. Insertion Operation	20
4.2. Deletion Operation.....	21
4.3. System Properties	22
5 Comparative Evaluation and Discussion	23
6 Conclusion and Future Work.....	24
6.1. Summary	24
6.2. Possible Extensions.....	24
7 References.....	25

LIST OF FIGURS

Figure 1: Mockup of the slide navigation bar.....	5
Figure 2: Mockup of My Workout Page	6
Figure 3: Mockup of My Analysis Page	7
Figure 4: Mockup of Snaps Page	7
Figure 5: Mockup of Workout Videos	8
Figure 6: Mockup of About	8
Figure 7: Document Structure.....	9
Figure 8: Database with Cloudant.....	10
Figure 9: Database with CouchDB	11
Figure 10: Client-Server Model.....	18
Figure 11: Data Flow Diagram	19
Figure 12: Elapsed Time of inserting documents	21
Figure 13: Elapsed Time of deleting documents	22

LIST OF CODES

Code 1: Configure Sync for Cloudant.....	10
Code 2: Configure Sync for CouchDB	11
Code 3: Navigation bar Functions	12
Code 4: Encryption Function.....	13
Code 5: addWorkout Function.....	13
Code 6: pushWorkout Function.....	14
Code 7: pushWorkouts Function	14
Code 8: doAnalysis Function.....	15
Code 9: Page Title Conditions	16
Code 10: Decryption Function.....	17
Code 11: Elapsed Time of inserting documents	20

LIST OF TABLES

Table 1: Elapsed Time of inserting documents	20
Table 2: Elapsed Time of deleting documents	21
Table 3: System Properties of CouchDB vs. Cloudant	22
Table 4: Evaluation Criteria of My Workout Page	23
Table 5: Evaluation Criteria of Analysis Page	23
Table 6: Evaluation Criteria of Snaps Page.....	23
Table 7: Evaluation Criteria of Workout Videos Page	23
Table 8: Evaluation Criteria of About Page	23

1 Introduction and Project Summary

I believe that it is essential for developers to understand the synchronize mechanism since a client can go offline and you will need to think about the reconciliation of this data. When adding the synchronizing capability to your application, you can reduce the limitation of access to your application.

For example, In my mobile the weather application cannot work when its offline or when it is in the airplane mode, and the data is not synced as it is preferred, but if you have the ability to synchronize your offline application data, then the weather application can get the latest up to date weather information even when network connection is not available.

Developers may think of the synchronizing mechanism with an offline capability as a time-consuming mechanism, and that its benefit is not worth the investment. However, as stated in previous example it is an important feature and it is not expensive to adapt to your application, and also being offline should not be an error condition.

I have done several kinds of researches to find ways and technologies to establish data synchronization and I came up with two common ways to do it.

The first way is with the help of Kinto technology, which is a generic JSON document store with synchronization and sharing capabilities. Kinto will keep the local database up to date with the Kinto server, and the remote changes perform to the local data, and local changes are uploaded using HTTP headers to control concurrency and overwrite [1].

The second way that I choose for my final project is with the use of CouchDB, PouchDB and IBM Cloudant service, PouchDB will store data locally and when there is network connection it will start synchronizing to CouchDB with IBM Cloudant service to securely host the data.

I choose the second way since I was able to find more researches and information about it online, also because I struggled with configuring and implementing Kinto to my application, to that end I continued with the combination of PouchDB and CouchDB as it was a more widely used way to perform synchronization.

For my project, I have built an offline-first mobile application called Fitness with the use of Ionic framework; this application mainly focuses on adding your workouts data and enables you to review the analysis of your overall workouts.

The application works when you are offline as the data is locally stored until you reconnect again, and the application can work from any device since it has the offline capability and the sync mechanism. Later on, I will include all the details and analysis I have configured to complete this project.

2 Comparative Literature Survey

Daniel Sauble gives a great example about offline-first applications on his book “Offline First Web Development”, he stated that a VP designer called Susan is responsible for the company’s product design, development, marketing, sales, and others. Susan uses a simple to-do application to store all her lists to organizing everything in her life. Unfortunately, this app behaves poorly when it goes offline. When the internet connection goes down her to-do list fails to load, she doesn’t have confidence that her list is created on her other devices, the app shows scary messages that make her think she lost some data. When Susan has an important thing to write she usually takes fifteen minutes of work, but one day her conference’s WI-FI connection was bad, she continued writing and a few minutes later her to-do list app crashes and Susan loses all her work [2].

CouchDB is an open-source NoSQL database that can store data with JSON (JavaScript Object Notation) documents distributed under the Apache license, if you want to access the documents then it is done via HTTP, and to distribute the data efficiently it is done with CouchDB’s incremental replication [3]. A CouchDB server hosts databases that store documents, and each document has a unique name in the database and for reading and updating the database documents, CouchDB provides a RESTful HTTP API [3].

I choose CouchDB over other different databases like PostgreSQL or MySQL instead, because of the HTTP and sync mechanism. CouchDB communicates with the database directly from the client-side of the app with the use of HTTP, also CouchDB is a cover from the bottom-up to enable easy synchronization between different devices and databases [4].

PouchDB is an open-source JavaScript database inspired directly by CouchDB, and the goal of PouchDB is to emulate the CouchDB API with near-perfect fidelity while running on the client side of the browser or in Node.js [4]. PouchDB is one technology that lets you build apps that store data in the front-end and synchronize its data in the backend. Therefore, PouchDB is a database that can run locally in your browser with the ability to synchronize with the back-end automatically, which means the data will be sent and stored from the front-end to the back-end until it is synchronized with other clients.

IBM Cloudant is an open-source JSON (JavaScript Object Notation) database based on Apache CouchDB. It is different from the traditional databases since all replicas of your data are available for both reads and write, Cloudant and CouchDB share a common replication protocol that lets you synchronize copies of your Cloudant data to remote CouchDB instance in a push of a button [5]. On Cloudant, if you want to connect to a remote database, then you can use PouchDB as a client library to CouchDB by specifying the actual URL of your database to your Cloudant account, and if you have the appropriate permission it would create that database or just reference it as already exist.

Ionic Framework is an open source UI toolkit for building mobile applications using web technologies like HTML, CSS, and JavaScript [6]. Ionic Framework is easy to learn and it focuses on the frontend user experience. It integrates easily with other frameworks and libraries such as Angular. Furthermore, Ionic can build apps for different devices such as Android, iOS, and desktop. It has a clean and simple code structure.

In my project, the greatest challenge was to perform replication and synchronization with PouchDB with CouchDB or Cloudant to the mobile application data. The main advantage of CouchDB and Cloudant is its ability to replicate the local data storage serving data once the connection is available, and being offline requires synchronization of data when a connection is available. Synchronicity and replication is the process of sharing information between multiple redundant storage sources. Documents in Cloudant are stored in a single structure of JSON format; this format simplifies the complex structure of objects, fields, arrays, and scalable types. Then, they are combined into a simple document record. Furthermore, there are several classifications of NoSQL databases, there is a key-value store that is distributed as a hash table, there is bigtable that is a multi-dimensional tabular system, and there is a graph-oriented database. Lastly, there is a document-oriented database such as CouchDB and Cloudant, which are the databases I chose for my final project [7].

From the Book “IBM Cloudant database as a service advanced topics”, I understood an important mechanism of Cloudant, and that is it supports replication for mobile devices. The definition of redundant is the ability to store data redundantly along with the cluster of the entire database and to replicate a subset of documents within one database to another, and there are different database replications types, there is bidirectional that deals with two databases. The continuous type that deals with instance up-to-date replication, and filterable that deals with a subset of the database documents, and many other types. By Cloudant, developers do not need to worry about managing the storage of data on the local device and the server. They do not need to think of how, where, and when to store their data. The API for mobile developers works to remove most of the backend complexity [8]. In addition, Cloudant Sync libraries handle the mapping of JSON documentation to SQLite relational data store on the device. The main reason I worked with Cloudant is that Cloudant sync libraries allow you to push database from your Cloudant cluster on the cloud immediately to your mobile application for local read and write operations, this should work even when your network connection is lost. With Cloudant, you can synchronize your data to many devices by using IBM Cloudant and Apache CouchDB. There are two main types of Cloudant synchronize libraries with different features, a type with API library designed for mobile applications, and a type of an index and query layer that closely matches the requirements of an application over what CouchDB view model does [8].

I worked with PouchDB to satisfy the offline-first sync experience as stated on the research paper “Offline-First Design for Fault Tolerant Applications”, by storing the embedded JSON documents of PouchDB to the local application state while CouchDB on the server side. Many devices can use the application and it will always function based on its current state and synchronize when needed.

Furthermore, I have found helpful solutions to some of the problems I faced during the phases of building my project. Therefore, instead of showing error messages as your connection is not available, try to show the stale data when offline instead. Furthermore, when facing a problem with local data loss for example user overwrite something and as the connection is established the original data can be lost, the solution is to disable editing while offline or to report these actions while offline later on [9].

The hybrid framework I used to build my mobile application is Ionic 3; it is one of the most widely used frameworks. I have learned Ionic by following the steps of the book “Hybrid Mobile Development with Ionic”, the book mainly focuses on Ionic 3 that is

the official version I have used for my application. On this book, I covered how to set up and install Ionic, what are the Ionic components, what are Ionic platform and its services, how to run ionic on a local host, and how to finally build your application and connect it to your mobile device. Furthermore, Ionic frameworks enable you to add your application as a platform easily. For example, if you want to add it by terminal then run “ionic cordova: platform add android” to display this application on your android mobile device [10].

In addition, CouchDB + PouchDB and Cloudant are all NoSQL databases, and I have decided to work with NoSQL databases in the mobile application for several good reasons. On the paper “NoSQL in a Mobile World: Benchmarking Embedded Mobile Databases” it is stated that the popularity of NoSQL databases is increasing over the years for mobile applications, and the reason for this is because NoSQL database has the ability to replicate itself, cloud synchronization capability, and document data model. It stated that NoSQL databases work amazingly for large data centers and that we need to choose the right technology depending on the requirements of our application [11].

Therefore, Perrier and Pervaiz did a benchmark between different NoSQL databases as shown in their paper [11]. The first step was to set up a large data set with random data, and then to create queries and documents depending on each database. After that, they defined a set of matrices to record the benchmark system. Next, they set up each database with an Android device. The result showed that the SQLite database performed great under a high load with complicated queries, and the CouchLite database performed decently on indexed lookups and queries, and DbO4 performed the best at creating and inserting because it was very fast.

To add up, NoSQL offers a lot more than just solving the problems of scaling. NoSQL offer speed even with a small amount of data and between different devices, it offers development time because you do not need to deal with complex SQL queries as joining multiple tables to create your final vision, and almost every NoSQL offer shameless data representation, which means that you don't need to think too much for the data design structure, as adding new data and nesting an existing data. Lastly, plan for scalability that is to plan to never fail, which means your application can be elastic and it should be able to handle spikes of load [12].

A great advantage of NoSQL is that it ensures availability and partition tolerance to our data, and it ensures us data consistency by the replication approach. The main advantage of the NoSQL database is that it uses the scale-out approach. If many users start using the application then another server is added. There is no need to modify the application since the application always sees a single distributed database. Furthermore, usually relational databases are expensive, and you need to purchase the license, whereas NoSQL databases are generally inexpensive, free open source, and priced only to their additional services [13].

CouchDB NoSQL stores their data in the form of documents, it has many advantages as it supports complex data structure and it is easier for debugging, and conceptualizing data. Therefore, it is rather faster than the relational database because it stores an entire object as a JSON document instead of storing and retrieving data from interrelated tables. The conclusion of NoSQL is that it provides a schema-less dynamic flexible data model that is the most preferable for big data and users. NoSQL has the ability to scale dramatically, and it provides improved performance for many users expectations [13].

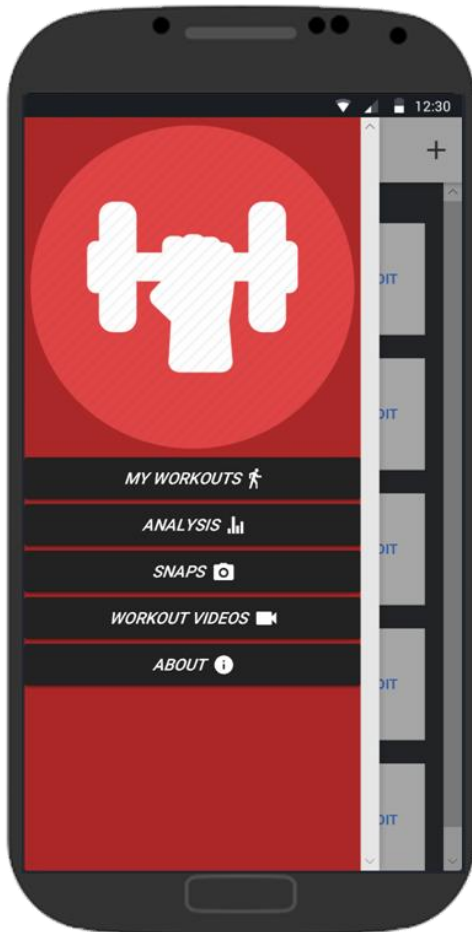
3 Developed Approach and System Model

I have built an offline-first mobile application that is called Fitness, this application will take the user workout information as (name of the workout, date, image, and amount) to store it in a local database when offline, and it will do synchronization with a remote server when a connection is online. Furthermore, the application provides a detail calculation analysis with charts of your today workouts and your overall workouts. Additional information about the application will be included in the below sections.

3.1. Mockups

3.1.1. Slide Navigation Bar

In Fitness Application, there are five pages in the slide navigation bar of the application. I have built this application on an android device with Ionic 3 framework.



The first page on the navigation bar is “My Workouts”, a user can add-delete-edit workouts on this page and can see all of his historical workouts. Furthermore, inside of this page an “Add Workout” page and is shown as a sign “+” symbol, this page is the interface page for filling the necessary data. In addition, users can edit their workouts through the “Edit Workout” page.

The second page is “Analysis”, this page will do the calculation and analysis of your today workouts and of all time, and the result of this calculation will be included in two different charts.

The third page is “Snaps”, in this page the user can take an image snap from “My Workout” page and can review it later on by the “Snaps” page.

The fourth page is “Workout Videos”, on this page I have provided different workouts videos that redirect you to different YouTube links according to the workout you have chosen.

Lastly, the “About” page will include a brief description of what does this application do.

Figure 1: Mockup of the slide navigation bar

3.1.2. My Workouts Page

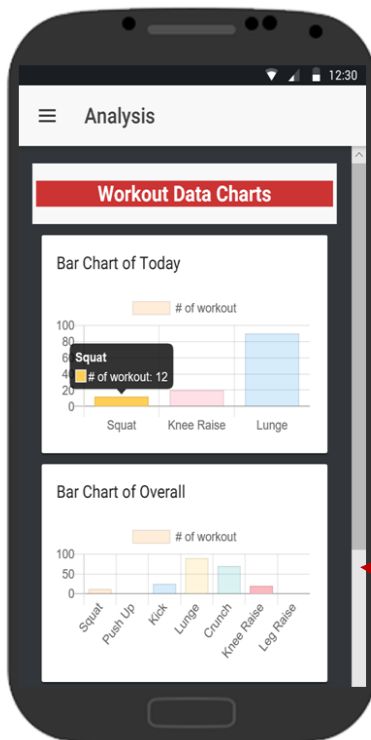
My Workout Page will show all your historical added workouts as seen in the figure. The name of your workout and the amount is shown in a group of boxes, the logo image of each workout is processed to match automatically according to the name of the workout. The “+” sign on the right side is for adding new workouts by the “Add Workout” page.

Each workout can be edited by the Edit button that will direct you to the “Edit Workout” page, and you can also delete a workout by clicking the delete button inside of the edit page.



Figure 2: Mockup of My Workout Page

3.1.3. Analysis Page



Analysis page will include the calculation number of all your workouts that are done today and your overall workouts. There are two types of analysis charts, bar chart, and doughnut chart. The today chart will only include the workouts that have been done today, and the overall chart will sum up all your workouts from the beginning of time until the last.

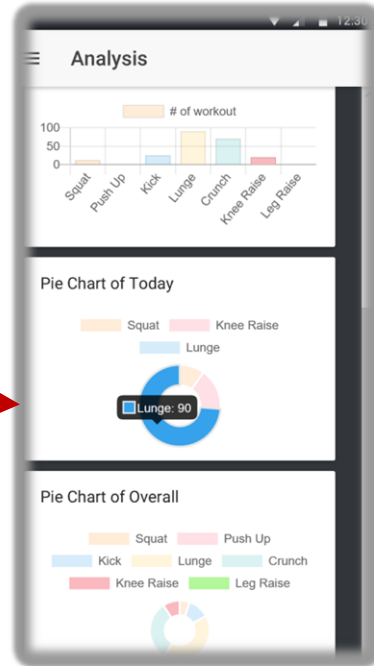
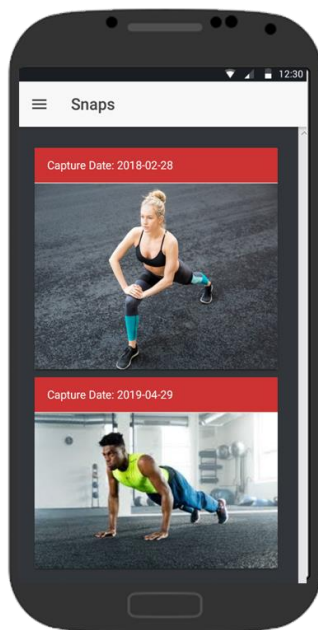


Figure 3: Mockup of My Analysis Page

3.1.4. Snaps Page

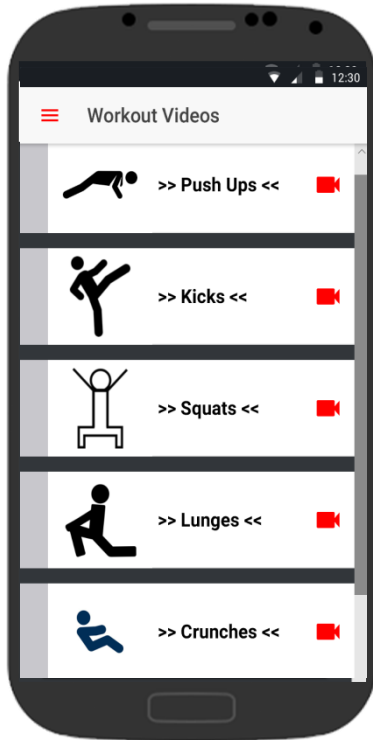


Snaps page will include all of the images you took when adding your workout information, and the date of your workout is included as decryption because date is encrypted to the database.

Each image can be replaced later on by the “Edit Workout” page.

Figure 4: Mockup of Snaps Page

3.1.5. Workout Videos Page

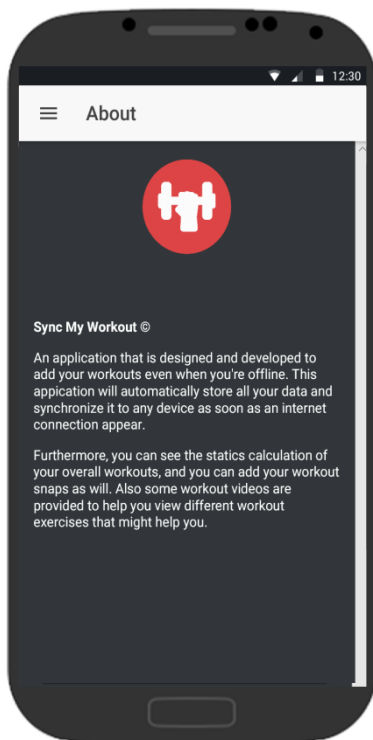


Workout videos page has different workouts lessons, to help users who don't know how to do a specific workout routine.

After the user clicks the workout name, it will be redirected to a specific Youtube video according to what has been clicked on.

Figure 5: Mockup of Workout Videos Page

3.1.6. About Page



The application description is mentioned in the about page, it is written that:

“an application that is designed and developed to add your workouts even when you're offline. This application will automatically store all your data and synchronize it to any device as soon as an internet connection appear. Furthermore, you can see the statics calculation of your overall workouts, and you can add your workout snaps as well. Also, some workout videos are provided to help you view different workout exercises that might help you.”

Figure 6: Mockup of About Page

3.2. Data Model

3.2.1. Document Structure

In this project, PouchDB and CouchDB both are a NoSQL database, and dealing with NoSQL database is different than dealing with the traditional SQL database; because NoSQL stores data in a JSON format with key/value pairs instead of tables with relations style. The main advantage of the NoSQL database is that it can be an optimal solution for massive data with a huge number of operations.

The data object in NoSQL is called a document, and the project has a number of data objects (documents) listed in IBM Cloudant. The PouchDB database will be connected to a remote link with username and password of the IBM Cloudant. From the server you can (get, add, edit and delete) the located documents establishing the necessary synchronization mechanism.

In consequence, the document structure of the project has five keys and values, the first key “id” has a unique value by date time, and the second “key” has the same value as id in order to be unique. The next key is “value” which is a pointer to the location of the data.

The last field is “doc” that has six nested keys and values, the first two keys have the same value as I explained before, the next key is “name” which is for the name of the workout. “amount” key has the value of the workout amount number, and the “date” key has been encrypted by the transform-pouch package as a sensitive document field [14], to print this data you must decrypt it by the same package.

The last key is “image”, it stored your captured image but in below structure, the user didn’t add an image which is acceptable.

```
id "2019-04-29T22:40:44.785Z"

{
  "id": "2019-04-29T22:40:44.785Z",
  "key": "2019-04-29T22:40:44.785Z",
  "value": {
    "rev": "1-7ebd8781cca3cd29d6183c974cf96af5"
  },
  "doc": {
    "_id": "2019-04-29T22:40:44.785Z",
    "_rev": "1-7ebd8781cca3cd29d6183c974cf96af5",
    "name": "Crunch",
    "amount": "88",
    "date": "TRDHNN4aCdPYSwt71It0cg==",
    "image": ""
  }
}
```

Figure 7: Document Structure

3.3. Structural Model

3.3.1. Configuration

To configure the synchronization setup of the application, I have installed Apache CouchDB, PouchDB and signed in to my IBM Cloud account. I have installed Cloudant that is a free instance from the catalog search, and then I have defined a new service credential for the instance.

Next, we need to do the database and remote setup, you can choose CouchDB or Cloudant as your remote service. Thus, at the start I will show how to configure with Cloudant, from the Cloudant dashboard we can add a new database, and then click generate an API key to add its value in the username field of the code for permission purpose, and when the API key is created a password is given along. The remote link of Cloudant is a copy from the JSON page next to the options button. Finally, we call sync function with the remote to start synchronizing the local PouchDB database with the Cloudant as shown in the figure below.

```

this.db      = new PouchDB('workout');
this.username = 'illedsterydredsouseselow';
this.password = 'db128bb23d671ad6db080f85e57fd5d0057daf31';
this.remote  = 'https://928971b9-95f0-4267-bb5a-714e7044f264-bluemix.cloudant.com/workout';

let options = {
  live: true,
  retry: true,
  continuous: true,
  auth: {
    username: this.username,
    password: this.password
  }
};

this.db.sync(this.remote, options);

```

Code 1: Configure Sync for Cloudant

_id	amount	date	image	name
2019-04-29T21:59:...		ytLW4nV9MQslj62kL...	data:image/jpeg;bas...	
2019-04-29T22:00:...		An/NW6hVRY7cc5Im...	data:image/jpeg;bas...	
2019-04-29T22:40:...	88	TRDHHN4aCdPYSWt...		Crunch

Figure 8: Database in Cloudant

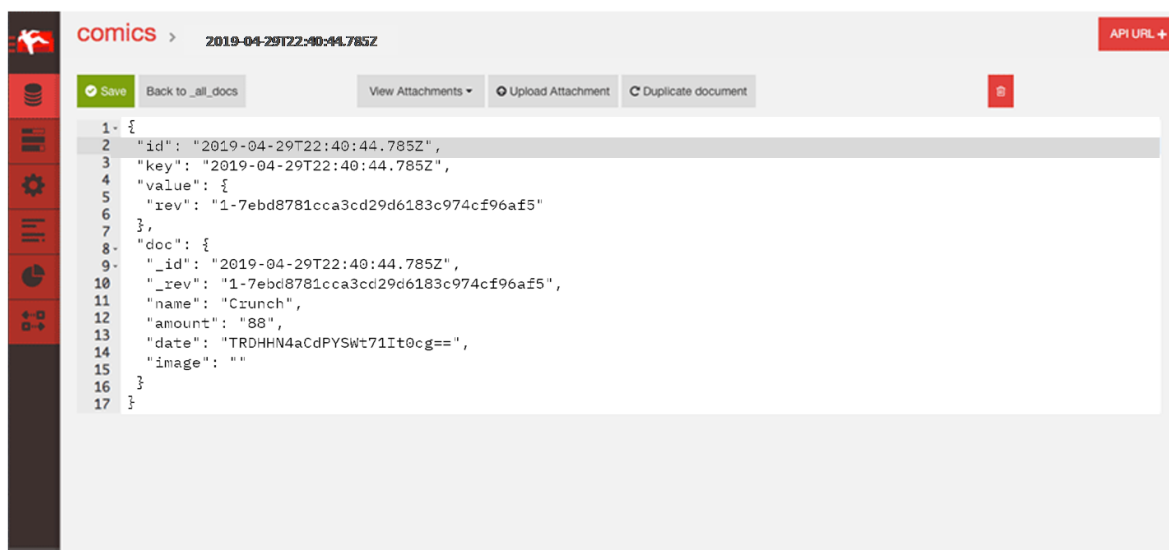
For this part, I will show how to configure sync with the CouchDB remote. First, we need to add a database with the name “workout” in CouchDB. With the link of the database, we need to add it to the remote field. After calling the sync function, it will start synchronizing the local PouchDB database with CouchDB as shown in the figure below.

```
this.db      = new PouchDB('workout');
this.remote  = "http://192.168.1.70:5984/workout";

let options = {
  live: true,
  retry: true,
  continuous: true,
  auth: {
    username: this.username,
    password: this.password
  }
};

this.db.sync(this.remote, options);
```

Code 2: Configure Sync for CouchDB



The screenshot shows a CouchDB interface for a database named "comics". The document ID is "2019-04-29T22:40:44.785Z". The document content is a JSON object with the following structure:

```
1- {
2-   "id": "2019-04-29T22:40:44.785Z",
3-   "key": "2019-04-29T22:40:44.785Z",
4-   "value": {
5-     "rev": "1-7ebd8781cca3cd29d6183c974cf96af5"
6-   },
7-   "doc": {
8-     "_id": "2019-04-29T22:40:44.785Z",
9-     "_rev": "1-7ebd8781cca3cd29d6183c974cf96af5",
10-    "name": "Crunch",
11-    "amount": "88",
12-    "date": "TRDHHN4aCdPYSwt71It0cg==",
13-    "image": ""
14-  }
15- }
16- }
17- }
```

Figure 9: Database in CouchDB

3.3.2. Code and Algorithm

In this part, I will describe the code implementation and how each algorithm work. I have worked with Ionic framework version 3 and installed a list of package libraries. The main libraries that I have used such as 'transform-pouch' that is used to encrypt and decrypt the required sensitive document fields [14], and 'ionic-native/camera' for handling images taking by your mobile, and also 'chart.js' for showing the calculated data in analysis charts.

For the slide navigation bar, I defined the following functions to redirect you to the selected page on "app.component.ts file", the below functions will be called on the "app.html" file to be a template for all five pages.

```

goToMyWorkouts(params) {                                // My Workouts Page
  if (!params) params = {};
  this.navCtrl.setRoot(MyWorkoutsPage);
}
goToAnalysis(params) {                                  // Analysis Page
  if (!params) params = {};
  this.navCtrl.setRoot(AnalysisPage);
}
goToSnaps(params) {                                    // Snaps Page
  if (!params) params = {};
  this.navCtrl.setRoot(SnapsPage);
}
goToWorkoutVideos(params) {                             // Workout Videos Page
  if (!params) params = {};
  this.navCtrl.setRoot(WorkoutVideosPage);
}
goToAbout(params) {                                     // About Page
  if (!params) params = {};
  this.navCtrl.setRoot(AboutPage);
}

```

Code 3: Navigation bar Functions

As I mentioned before there are five pages (My Workouts, Analysis, Snaps, Workout Videos and About). Furthermore, there is one provider called the database. A provider is a service that enables other components to use its functions, and it is done by importing the provider from the page that wants to use it.

In order to be able to use the provider on the application, we need to add it to "src/app/app.module.ts" with all the other pages.

3.3.2.1. Database Provider

In the following part, I will explain how the database server works. At the start, we need to import the necessary packages and mainly the PouchDB package for setting up the sync configuration as explained in the previous part "4.3.1. Configuration" pg.10, and to import "transform-pouch" package to establish encryption for sensitive document fields [14].

The `encrypt` function will be applied to each “date” field of the document. Since I considered the date to be the sensitive part of our data because we do not have any password and pin number to consider, and given that the encryption can apply to any field of the data but what I focused on is how we can implement it. Therefore, the following code shows the encryption function that will transform the data before it is stored on the database.

```
encrypt(text) {
  var crypto = require('crypto');
  var cipher = crypto.createCipher('aes-256-cbc', 'password');
  var crypted = cipher.update(text, 'utf8', 'base64');
  return crypted + cipher.final('base64');
}
```

Code 4: Encryption Function

The “`addWorkout`” function will take the necessary document inputs from the add page, these inputs will be formed into a JSON format by workout object, and you can notice that the date field used the `encrypt` method in this part. Later on by the “`put`” method of PouchDB it will store the workout object into the local database only if the operation is successful, else if failure it will show the proper error to console.

```
addWorkout(name, amount, date, image)
{
  var idByTime = new Date().toISOString(),
      workout = {
        _id      : idByTime,
        name     : name,
        amount  : amount,
        date    : this.encrypt(date),
        image   : image
      };
  return new Promise(resolve =>
  {
    this.db.put(workout).catch((err) =>
      ... some code ...
    )
  })
}
```

Code 5: addWorkout Function

The “`updateWorkout`” function has the same algorithm as the previous function, so there is no need to explain it again.

Next, we have the “`push Workouts`” function that is used to take one input parameter, which is the unique key of the selected document, then by “`get`” method of PouchDB it will retrieve all the documents in the database by a given key, and after it matches with the required document, it will return the required document in an array. This function of the database is used later on by other functions in the add page, that is when the user clicks edit workout or delete workout.

```

pushWorkout(id)
{
  return new Promise(resolve =>
  {
    this.db.get(id, {attachments: true})
      .then((doc)=>
      {
        var item = [];
        item.push(
        {
          id      : id,
          rev     : doc._rev,
          amount  : doc.amount,
          name    : doc.name,
          date    : doc.date,
          image   : doc.image
        });
        resolve(item);
      })
    });
  });
}

```

Code 6: pushWorkout Function

Later on, we have the “pushWorkouts” function that is used to retrieve and get all of the saved workouts from the database. The needed PouchDB method is the “allDocs” method. I have included options for this method as “include_docs” to include the full content of the documents, also the “descending” method to return documents in descending order. The “pushWorkouts” is used on “My Workout” page to list all of the entered workouts. Furthermore, I have included an extra item called logo that is responsible for choosing the proper logo according to the workout name; once everything is pushed to the items array, it will be returned.

```

pushWorkouts()
{
  ... some code ...
  for(k in row) {
    var item = row[k].doc;
    let pic :string

    // If name is push up then image is push up etc
    if("Push Up" === item.name){ pic = "assets/img/pushup.png" }
    ... etc ...
    items.push(
    {
      id      : item._id,
      rev     : item._rev,
      amount  : item.amount,
      name    : item.name,
      date    : item.date,
      logo    : pic,
      image   : item.image
    });
  }
  resolve(items);
});
}

```

Code 7: pushWorkouts Function

After that, we have the “doAnalysis” function that used on the Analysis page; this function will do the calculation by counting the total workouts done for each workout type. The needed PouchDB method is the “allDocs” method as mentioned in the previous function. At the end, it will return this result along with the entire document data as an array.

```
doAnalysis()
{
    ... some code ...
    var total_pushups    : number = 0;
    var total_kicks      : number = 0;
    ... etc ...
    for(k in row)
    {
        var item = row[k].doc;
        // .....count total overall push ups..... //
        if("Push Up" === item.name){
            total_pushups = total_pushups + (+item.amount); }

        if("Kick" === item.name){
            total_kicks = total_kicks + (+item.amount); }
        ... etc ...
    }
    ... some code ...
    items.push(
    {
        id          : item._id,
        rev         : item._rev,
        name        : item.name,
        ... etc ...
        total_pushups : total_pushups,
        total_kicks  : total_kicks,
        ... etc ...
    });
    }
    resolve(items);
}
```

Code 8: doAnalysis Function

The last function we have on the Database provider is “deleteWorkout”, this function will take two parameters the “id” and “rev” which is for referring to which document we want to delete, the PouchDB method we used in this part is “remove”. After calling the remove method with the required document, it will remove the required record if it is successful, else if fail it will show the proper error to console.

3.3.2.2. The Add Component

In the following part, the add component will be responsible for all of the entry operations, and it will call the necessary functions from the database provider. Therefore, we will import the necessary packages and mainly the “ionic-native/camera” for handling images taking by your mobile, and we need to import the database provider to be able to use its functions. Initially, we will define a group of public variables that are used in the class.

- Form: is for blocking the group control if one of the group members is invalid by checking from the “FormBuilder” class for validation.
- workoutAmount: is the ngModel value from the HTML control.
- workoutName: is an ngModel value from the HTML control.
- workoutDate: is an ngModel value from the HTML control.
- keyId: is the key value of a document in the database.
- revisionId: is the rev value of a document in the database.
- Selected: is for determining if the page is add or edit (to disable delete button).
- Title: is for the title of the page withers “add workout” or “edit workout”.
- doEmpty: to empty out the ngModel values at the end.

The following code show if the page is add or edit workout by checking the key and rev values, if the values are given the page will be edit and “this.selected” is true which means the delete button is not hidden. On the other hand, if key and rev values are not given the page will be add, and “this.selected” is false which means the delete button is hidden.

```

if(navP.get("key") && navP.get("rev"))
{
    this.keyId           = navP.get("key");
    this.revisionId     = navP.get("rev");
    this.Selected       = true;
    this.selectWorkout(this.keyId);
    this.Title          = 'Edit Workout';
}
else
{
    this.keyId           = '';
    this.revisionId     = '';
    this.Selected       = false;
    this.Title          = 'Add Workout';
}
}

```

Code 9: Page Title Conditions

The first function is “selectWorkout(id)” that is responsible for retrieving the database documents for the edit page by the given unique id value, because we can edit one document at a time, and we will call “pushWorkout” function of the database provider to help do what we need with the database.

Later on, we have “getPicture()”, “readImage(event)” and “getImage()” for handling the process of taking an image, if successful, it will read the image accordingly else it will show alert message “Unable to take photo”.

Next, the “submit()” function for submitting the added or the edited workout to the database, if the operation is successful it will show the proper message for it and add the

change to the database by the “addWorkout” or “updateWorkout” function from the database provider class.

Afterward, we have the “delete()” function for deleting entries from the database. First, it will retrieve the database with the given unique id with “pushWorkout” method, and then by “deleteWorkout” method of the database provider it will delete the specified entry and show the proper message of deletion.

Lastly, “showMessage(message)” function is for showing the message of the parameter to the user by the ToastController component.

3.3.2.3. Others

In this section, I will give the main work for each page, since the pages do not have many complicated operations. Each of the typescript files has the “ionViewWillEnter” method, which is an ionic method that runs when the page is active to ensure the data retrieved from the database is the latest.

. Therefore, starting with “my-workouts.ts” page it will check if there is an existing data. if data exist, it will keep it in module “workouts” as public to use it by “my-workouts.html”. By HTML, it will retrieve data by the for-loop, and list the workouts on the page. There are two functions “addWorkout()” and “viewWorkout(param)”, the first function will push the page “Add” to do the add workout operation, the second function will push the page “Edit workout” page to edit an existing workout.

Next, is with “analysis.ts” page that will take the calculations on charts, I assigned two modules chart and chart2 for bar chart and doughnut chart, I have also assigned four @ViewChild’s which is an angular decorator that is used as a reference to the HTML page. The decorators are “todayCanves” of today’s workout with doughnut chart, and “overallCanves” of overall workouts with doughnut chart. “barTodayCanvas” and “barOverallCanvas” similar as previous, first for today’s workout with a bar chart, second for overall workouts with a bar chart.

Furthermore, I have used the “decrypt” function of “transform-pouch” package to decrypt the “date” data, because it is stored as an encrypted data in the database.

```
decrypt(text) {
  var crypto = require('crypto');
  var decipher = crypto.createDecipher('aes-256-cbc', 'password');
  var dec = decipher.update(text, 'base64', 'utf8');
  return dec + decipher.final('utf8');
}
```

Code 10: Decryption Function

Therefore, by using the “doAnalysis” method of database provider, we will retrieve the database and I assigned two lists called “listData” and “listAmount”, the first is for storing the workouts name, and the second is for storing the amount of the workout. These two lists are only of today’s workouts by the if-condition:

“if (this.decrypt(data[i].date) == today)”, that is if decrypted date is equal to today’s date. In the end, the necessary charts structure of today and overall workouts is completed and used in the HTML page.

3.4. Dynamic Model

3.4.1. Client-Server Model

Client-server is a system that generates the functions of the client and server to establish the sharing of information between them. It allows a group of users to access the same database at the same time, and the information will be stored in that database. The main advantages of the client-server model is that it splits the processing of applications across multiple machines, and it allows easier sharing of resources from client to server, and it reduces data replication by storing data on each server instead of client [15].

In the client part, when a user enters his workout data in an offline mode, the data is going to be sent to PouchDB local database, and PouchDB will start synchronizing with the remote of the server side Cloudant (CouchDB) when online again as shown on figure below. To understand the configurations setup of the server-client then return back to part “4.3.1. Configuration” pg.10.

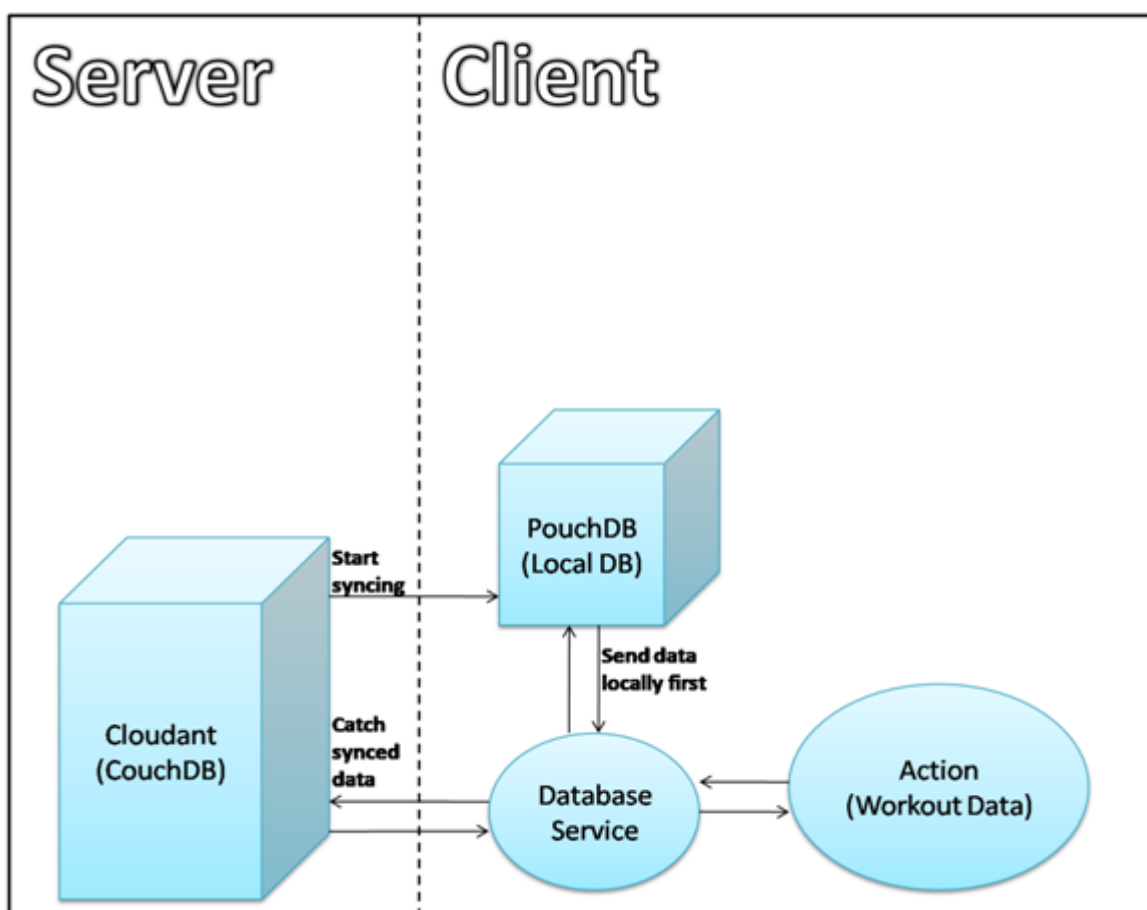


Figure 10: Client-Server Model

3.4.2. Data Flow Diagram

In this part, I will explain the data flow between the device (interface) and the database (PouchDB and Cloudant). The actions are as below according to the given diagram:

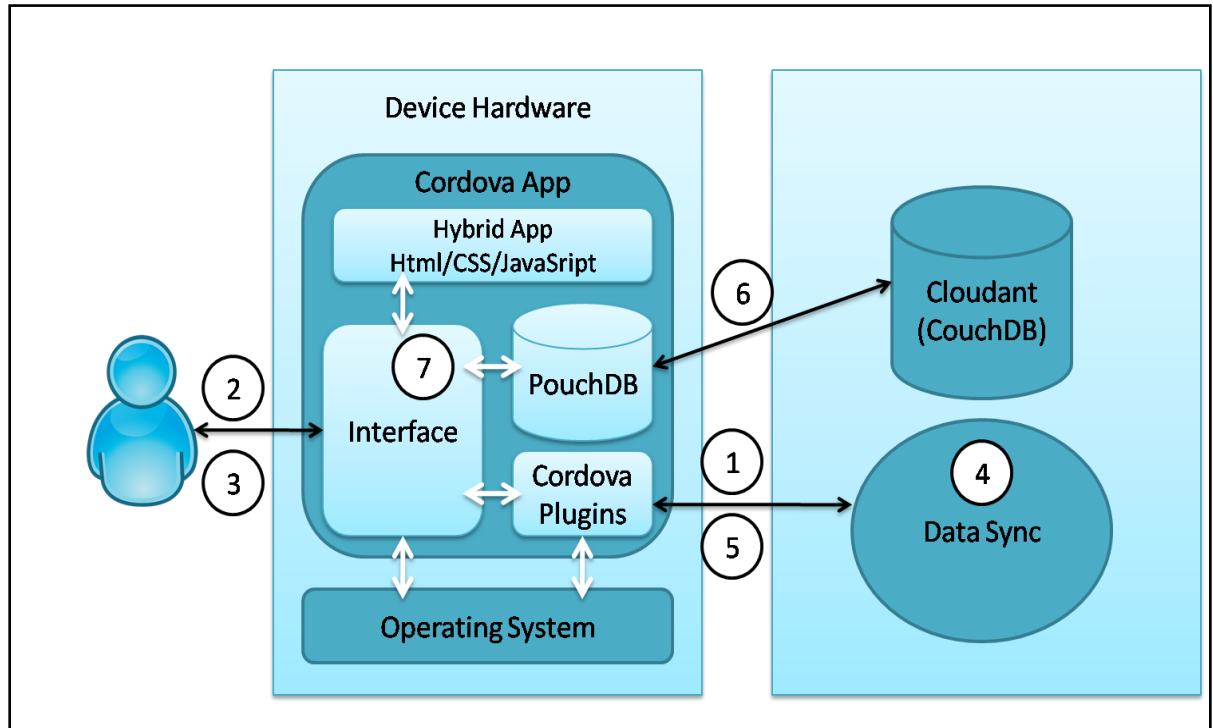


Figure 11: Data Flow Diagram

- 1- A user launches the mobile application. Cloudant service is called to fetch the latest data.
- 2- The workout page appears first as the main page, and all saved workouts fetch from the database to the workout page.
- 3- User wants to add a new workout and clicks submit with the new workout.
- 4- Checking the validity of the data entry.
- 5- If success, data is responding to the server request from Cloudant.
- 6- Local PouchDB and Cloudant DB start synchronization.
- 7- After synchronizing, mobile application updated with the latest data by the handleChange method of angular NgZone.

4 Experimentation Environment and Experiment Design

4.1. Insertion Operation

In this part, I will show the performance comparison between the CouchDB NoSQL database and Cloudant NoSQL database for the application. As I mentioned before, PouchDB can start synchronization with CouchDB and Cloudant. Therefore, I applied some performance tests to benchmark CouchDB and Cloudant sync performance and to see which database performs better in terms of speed for synchronization. The test results are as given below.

```
var start = new Date().getTime();

    for (let i = 0; i < 1000; i++) {

        this.db.addWorkout(name, amount, date, image)
            .then((data) =>
            {
                this.doEmpty();
            });
    }
var end = new Date().getTime()
alert("Elapsed Time is= " + (end - start) );
```

Code 11: Elapsed Time of inserting documents

I have run the above code to insert documents with the number of (10, 100, 1000, 10.000, 100.000, 1000.000) documents, with CouchDB remote and Cloudant to compare their performances. Given that, documents added with the “put” method. The results are as following:

Number of documents to be inserted	Elapsed Time of inserting with CouchDB	Elapsed Time of inserting with Cloudant
10	0.045 seconds	0.058 seconds
100	0.196 seconds	0.264 seconds
1000	1.042 seconds	0.887 seconds
10.000	3.565 seconds	4.670 seconds
100.000	30.121 seconds	26.572 seconds
1000.000	109.102 seconds	119.256 seconds

Table 1: Elapsed Time of inserting documents

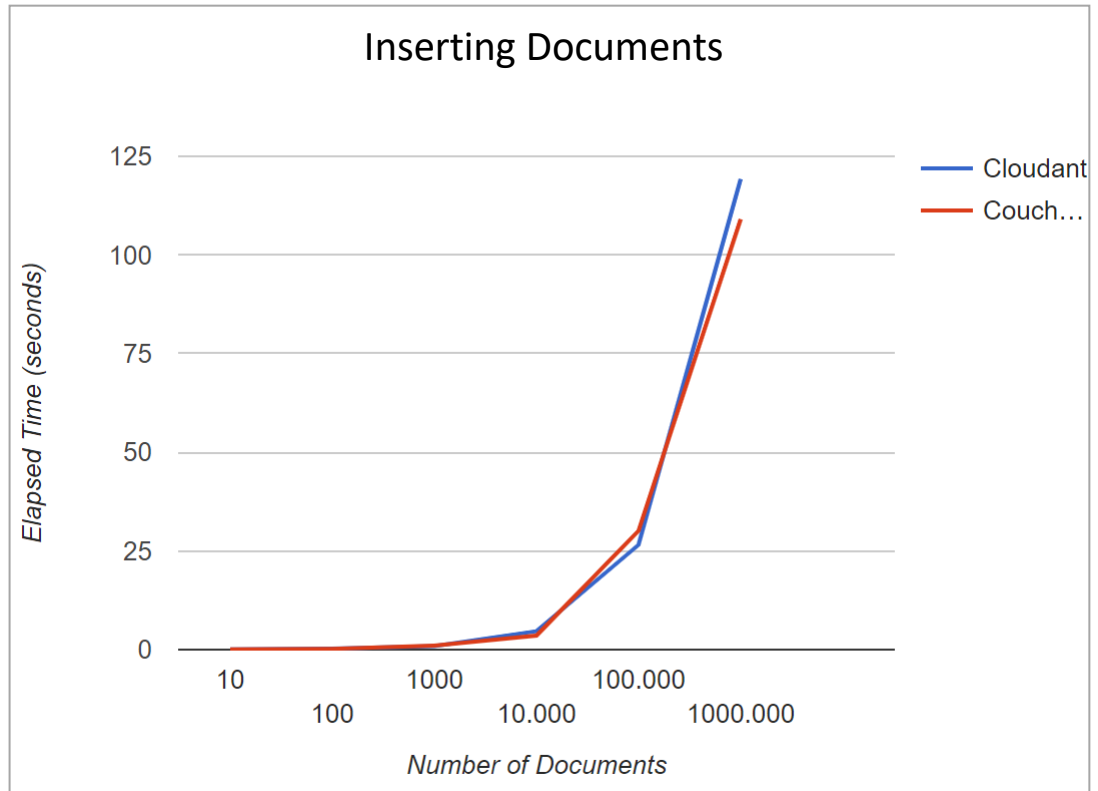


Figure 12: Elapsed Time of inserting documents

As I expected, the performance of both CouchDB and Cloudant databases are similar when inserting a number of documents from the database.

4.2. Deletion Operation

In this part, I tested again but with delete operation for deleting documents that were previously inserted with a number of (10, 100, 1000, 10,000, 100,000, 1000,000) documents, with CouchDB remote and Cloudant to compare their performances. Given that, documents deleted with the “`remove`” method. The results are as following:

Number of documents to be deleted	Elapsed Time of deleting with CouchDB	Elapsed Time of deleting with Cloudant
10	0.012 seconds	0.039 seconds
100	0.284 seconds	0.146 seconds
1000	0.993 seconds	1.350 seconds
10,000	3.010 seconds	3.968 seconds
100,000	27.834 seconds	24.731 seconds
1000,000	118.937 seconds	110,201 seconds

Table 2: Elapsed Time of deleting documents

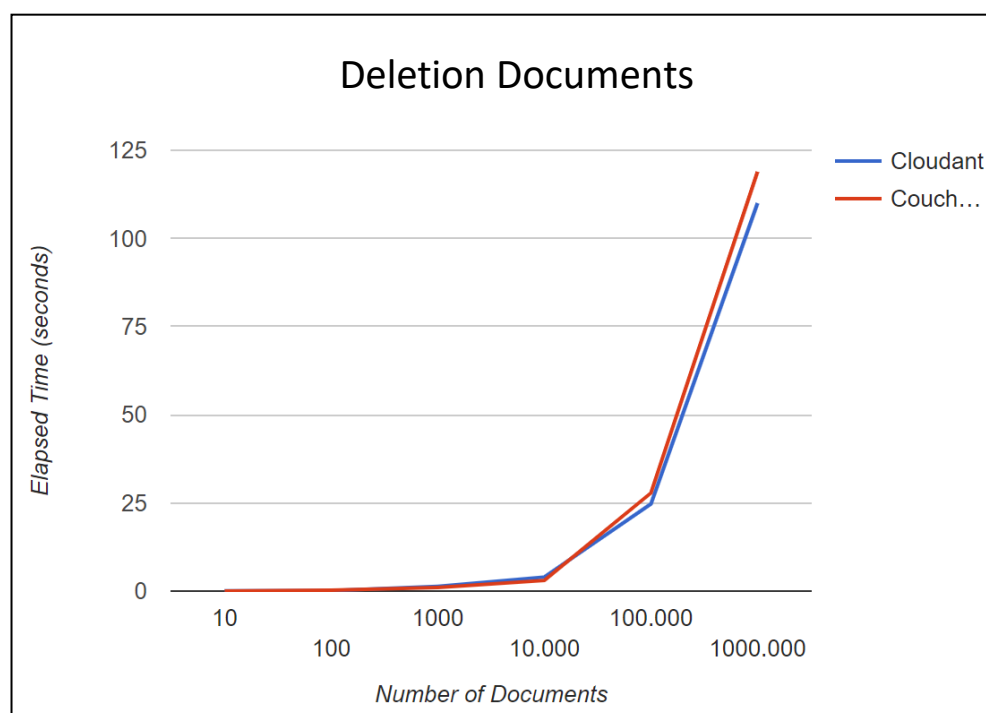


Figure 13: Elapsed Time of deleting documents

Once again, the performance of both CouchDB and Cloudant databases are similar when deleting a number of documents from the database.

4.3. System Properties

Since the scalability and performance of both CouchDB and Cloudant are the same. I searched about the differences between them in terms of system properties, one of the main differences is that CouchDB is an open-source that needs to be installed (i.e. Windows, Linux), and Cloudant is a service with a cost that does not need to be installed (i.e. Hosted). The following schedule is a comparison of their differences:

	CouchDB	Cloudant
Horizontally Scalable	No	Yes [16]
Full-text Search	No	Yes [16]
Geospatial Querying	No	Yes [16]
License	Open Source	Monthly subscription or perpetual [17]
Hosting Operating System	No	Yes [17]

Table 3: System Properties of CouchDB vs. Cloudant

In conclusion, if you were willing to pay for the additional features existed in Cloudant service, then Cloudant would be a better choice for you to start the. In opposition, if you do not need to use the additional features for your project, then CouchDB would be a better choice for you.

5 Comparative Evaluation and Discussion

The main goal for my final project was to establish the synchronization of an offline-first application across multiple different devices. After a series of researches and consulting my advisor in a two-semester period, I was able to build an offline-first mobile application called Fitness that adds and organizes your workouts at any time, to any device when it is offline. The following tables show the evaluation criteria for each page of the application:

Criteria	Did it meet?
1. Add workouts successfully.	✓
2. Edit workouts successfully.	✓
3. Delete workouts successfully.	✓
4. Display all saved workouts on page.	✓
5. Use camera to add image successfully.	✓
6. Encrypt data to the database (Date).	✓
7. Select the appropriate logo image for the added workout.	✓

Table 4: Evaluation Criteria of My Workout Page

Criteria	Did it meet?
1. Display analysis bar chart of today's workouts.	✓
2. Display analysis bar chart of overall workouts.	✓
3. Display analysis par chart of today's workouts.	✓
4. Display analysis par chart of overall workouts.	✓

Table 5: Evaluation Criteria of Analysis Page

Criteria	Did it meet?
1. Display all images taken of workouts.	✓
2. Display the date for an image by decrypting the date from the database.	✓

Table 6: Evaluation Criteria of Snaps Page

Criteria	Did it meet?
1. Direct the selected workout to the right YouTube link.	✓

Table 7: Evaluation Criteria of Workout Videos Page

Criteria	Did it meet?
1. Provide a brief description of the application.	✓

Table 8: Evaluation Criteria of About Page

6 Conclusion and Future Work

6.1. Summary

PouchDB is an amazing tool for storing and synchronizing your data, it is powerful and it can store data directly to your local database. Furthermore, it can replicate this database to a remote server. Working with PouchDB at first was hard because I never had any experience with a NoSQL database. Therefore, if you came from a SQL background like me, then you need to understand all the concepts and approaches of NoSQL before choosing to work with it.

In addition, the performance of CouchDB and Cloudant are similar when adding and removing documents, the real differences exist in their system properties as I mentioned in the schedule of system properties. Therefore, you can choose Cloudant by cost if you need its additional features such as being able to Full-text Search, or you can choose the free open-source CouchDB which is widely used and it fits your requirements.

In conclusion, I would recommend PouchDB + (CouchDB or Cloudant) to anyone thinking to work with an offline-first application, and if you want to understand everything you need to learn in detail then you can read the free documentation provided by PouchDB, CouchDB, and Cloudant, These documentation helped me a lot throughout the process of building the final graduation project code [18].

6.2. Possible Extensions

In the future, I might extend the analysis page by calculating the number of calories burned, and by calculating the amount of muscle gained with the added workout. In addition, I might extend a new page called “Gym near me” to find the location of all gyms near your area, and to be able to rate the chosen gym.

Furthermore, I might check out some new synchronization tools such as Kinto, to have a wider knowledge of the different tools used to establish synchronization for an offline-first application.

7 References

- [1] Kinto-storage 2.1.0 Documentation,
<http://docs.kinto-storage.org/en/2.1.0/api/1.x/synchronisation.html>
- [2] Daniel Sauble, *Offline First Web Development*, Published by Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.
- [3] CouchDB Documentation,
<https://docs.couchdb.org/en/stable/index.html>
- [4] PouchDB Documentation,
<https://pouchdb.com/guides/>
- [5] IBM Cloudant: FAQ,
<https://www.ibm.com/cloud/cloudant/faq>
- [6] Ionic Framework Documentation,
<https://ionicframework.com/docs/intro>
- [7] Miler, Mario; Medak, Damir; Odobasic, Drazen, *Two-Tier Architecture for Web Mapping with NoSQL Database CouchDB*, Salzburg, Austria, 5-8.07.2011,
mmiler@geof.hr.
- [8] Bienko, Greenstein, E Holt, Philips, *IBM CLOUDANT Database as a Service Advanced Topics*, 1st addition, restricted by GSA ADP with IBM Corp. April 2015, U.S.
- [9] Linklate, Marais, Herbert, Irwin, *Offline-First Design for Fault Tolerant Applications*, Centre for Research in Information and Cyber Security, Nelson Mandela University,
2s204005264@mandela.ac.za
- [10] Gaurav Saini, *Hybrid Mobile Development with Ionic*. Packt Publishing Ltd. April 2017, 35 Livery Place. UK.
- [11] Pervaiz and Perrier, *NoSQL in a Mobile World: Benchmarking Embedded Mobile Databases*, fahadp@cs, tperrier@cs.
- [12] Gaurav Vaish, *Getting Started with NoSQL*, March 2013, Birmingham B3 2PM, UK.
- [13] Zaki, *NoSQL DATABASES: NEW MILLENNIUM DATABASE FOR BIG DATA, BIG USERS, CLOUD COMPUTING AND ITS SECURITY CHALLENGES*, BMS College of Engineering, Bangalore, India
- [14] Transform Pouch Package,
<https://www.npmjs.com/package/transform-pouch>
- [15] Haroon Shakirat Oluwatosin, *Client-Server Model*, School of Computing Universiti Utara Malaysia Kedah, Malaysia, IOSR Journal of Computer Engineering, Feb. 2014.
- [16] IBM Cloudant: What is Cloudant?,
<https://www.ibm.com/cloud/cloudant>
- [17] Wikipedia IBM Cloudant,
<https://en.wikipedia.org/wiki/Cloudant>
- [18] Graduation project codes (Fitness mobile application),
<https://github.com/lina-alrehaili/Graduation-Project>