Cataloguing for Mechanical Parts Using Firebase: A Study

Ravinder Singh*

Abstract: ***These days Android is the best and the most well-known working frameworks for smartphones. In the transport industry (mainly trucks and buses), the number of parts and spare parts which are available to sale and purchase from the 1st party and 3rd party vendors are huge. If a person wants to buy a part, it becomes very difficult for him to convey what product he wants just based on the name, because most of the products sold with different names as each manufacturer give it a different name. So, to solve this problem, this application was created. The day on which we launch the application on the Google Play Store. From this very day we are constantly making efforts, to create this app more user-friendly and more user useful. Here we present you a brief insight into our Android app, and discuss some advanced features like automatic watermark, integrated search, etc. In addition to the above talk will also provide a brief app statistics stating figures to user rating, average rating, installations, etc.***

Keywords: ***Android Studio, Firebase, NoSQL, real-time database.***

1. INTRODUCTION

Application's objective is to catalog all the available parts and spare parts of trucks and buses which a person wishes to buy. This offers assistance to easily identify the product which they are trying to buy as it displays all the information related to the product like size, an image of the product, number of teeth it may or may not have, the vehicle it's designed for, etc. Customers can contact the seller regarding the price and more details of the product.

2. OBJECTIVES

Application is for the people, who are in the industry of transportation mainly trucks and buses. As all the things require maintenance, so do these vehicles as well.

The purpose of this application is to make it easier for the buyer to find the exact product which he needs.

The spare parts of trucks are very complicated, the seller might have the required part with him but will be unable to make the transaction if the buyer is unable to explain what part he wants to buy/ need.

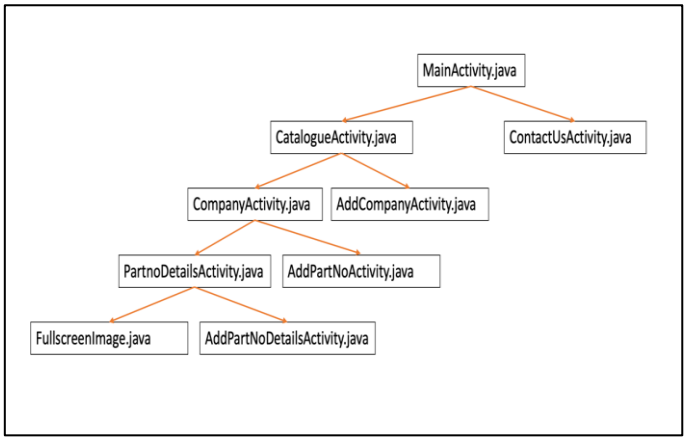
*Kajalravi9@gmail.com

This application comes into play as it contains image of the part as well as the minor details of the part like where it fits, its size, number of teeth it has, etc. seeing and confirming the part, the buyer can easily contact the seller for the product and he can be sure to get the right product the first time around rather than waiting to return and exchange the product, which costs valuable time and money because shipping of most of the parts is difficult as the parts are heavy. This application solves this problem

3. FEATURES AND FUNCTIONALITIES OF THE APPLICATION

- Basic Navigation map
- Firebase as a backend
- Easy addition/ updating of new/old parts using the admin app.
- Integrated live search
- Offline capabilities
- Custom Watermark present on all images

Basic Navigation map of the App



4. FIREBASE AS A BACKEND

Google's Firebase was used as a backend for this project because it offers reliability and constant support. The

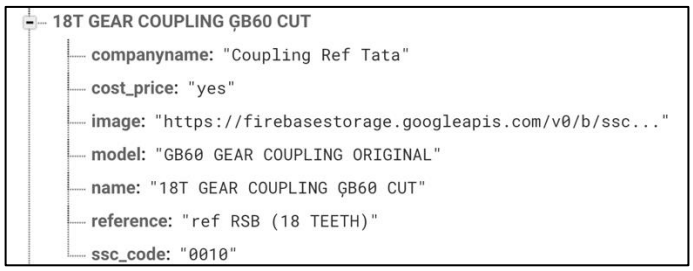
documentation for the Firebase is regularly updated and this technology is widely used so any issues which occurred during the development were easily solved by the support of the community.

To store the data, Firebase Real-time Database was used. Firebase Real-time database offers Real-time connectivity which means instant updating, adding and deleting of data is possible.

The most common type of database is relational database or SQL, the database, the Firebase Real-time database does not use SQL type database, rather it uses a NoSQL type database. NoSQL database is very flexible and easy to work with, unlike SQL, one does not need to define all the columns first, it can create as the need arises. Firebase Real-time Database stores data in the .json format. The data is stored as key: value pairs.

In our project the data for a single item is stored with the following attributes:

- Company name
- Cost_price
- Image
- Model
- Name
- Reference
- ssc_code



- Company name: used to fetch all the parts of a particular company name into the companyActivity
- Cost_price: used to determine whether the part is available for the customer or not, this is not visible to the customer. It accepts to values: “yes” and “no”
- Image: this is used to store the URL of the image
- Model: used to store the details about the part
- Reference: used to store more details about the part
- Ssc_code: this is a unique code, which helps to identify each part.

We also use Firebase Storage to store images of the product, the URL of the image is fetched and stored in “image”. Which is later used to load the image onto the device.

5. ADDING/ UPDATING PARTS THROUGH ADMIN APP

As per the request, an admin app was also created which had the ability to add, delete and update the data. It also had the option to temporarily remove the product if it were out of stock as per the user's need. Admin app makes it easier for the user to do all these things without even logging onto the computer and updating them through the firebase database.

All the processing of data like adding a super-category, image compression, adding a default value are all done through the activities of the admin app.

The admin can add/update/delete data through the floating add button which is available throughout the app.

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference().child("PartNo").child(nameinput);
myRef.updateChildren(result).

        addOnCompleteListener((task) → {
                if (task.isSuccessful()) {

                    Toast.makeText( context: addPartnoDetailsActivity.this,  text: "details added successfully", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText( context: addPartnoDetailsActivity.this,  text: "an error occured while uploadig data", Toast.LENGTH_SHORT).show();

                }
                // mProgressDialaog.dismiss();

        });
```

6. INTEGRATED LIVE SEARCH

the application has the feature of the integrated search when the user searches for the part no through the title bar, it fetches all the parts and stores them in a list, and then the substring of each part name is compared and the results are displayed instantly without even pressing the search button. This feature also lets the user search without worrying about the search case.

```
SearchView.OnQueryTextListener queryTextListener = new SearchView.OnQueryTextListener() {
    public boolean onQueryTextChange(String newText) {

        if (newText.equals("")) {
            return false;
        } else {
            newText = toTitleCase(newText);


            Query query = FirebaseDatabase.getInstance()
                    .getReference().child("Companies").orderByChild("name");
            query.keepSynced(true);
            query.addValueEventListener(search(newText));
            // Toast.makeText(getApplicationContext(), "textChanged :" + newText, Toast.LENGTH_LONG).show();

            return true;
        }

    }
```

7. OFFLINE CAPABILITIES

This app has offline capabilities, even when the user is offline, he can still access the parts which were previously fetched by him. This not only helps to save data but also makes the app faster and efficient with each use.

The offline capabilities of the app are implemented through firebase, the following lines of code help to achieve this.

```
super.onCreate();

FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

8. DYNAMIC WATERMARK FOR ALL IMAGES

to solve the issue of the images being used by the competition a custom watermark is added to all images. The watermark contains the SSC code which is unique for each item, this code is added automatically along with the translucent watermark. This Watermark is not added to the actual image which is stored in the firebase storage, but to the app itself, so it saves time as well as processing power to add the watermark. The watermark is just overlaid on top of the image and opacity of the watermark is set to 0.6f which is 60% opacity. The watermark is stored offline, while the image is fetched from the database

```
mWatermark.setAlpha(0.65f);
```

To load the images from the URL which was fetched from the database, an external library called Picasso is used which loads the data onto the position.

```
Picasso.get().load(partNo.image).placeholder(R.drawable.ic_settings_black_24dp).error(R.drawable.ic_settings_black_24dp).into(partNoViewHolder.mItemImage);
```

9. SCREENSHOTS

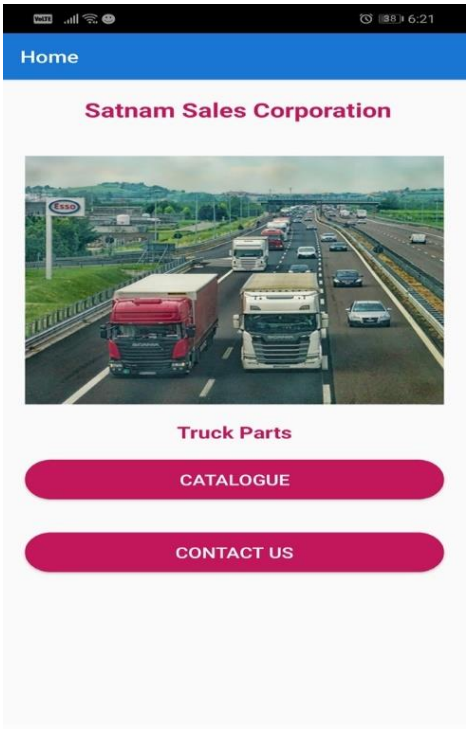


Fig. 1. (main Activity.java)

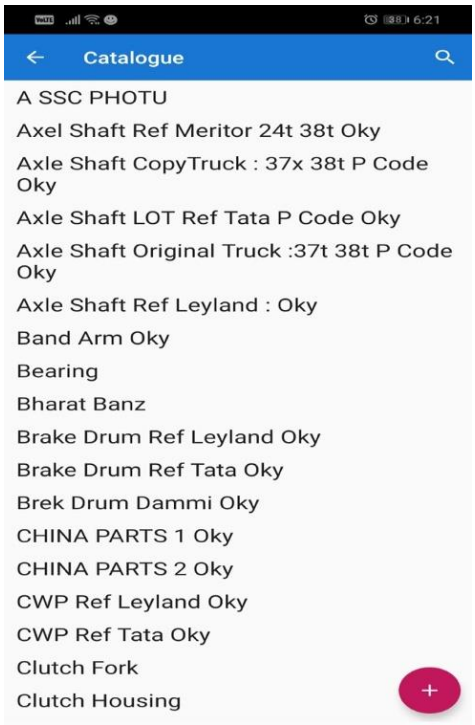


Fig. 2. (CatalogueActivity.java

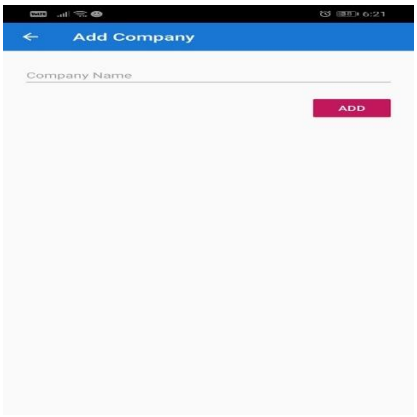


Fig. 3. (AddCompanyActivity.java)

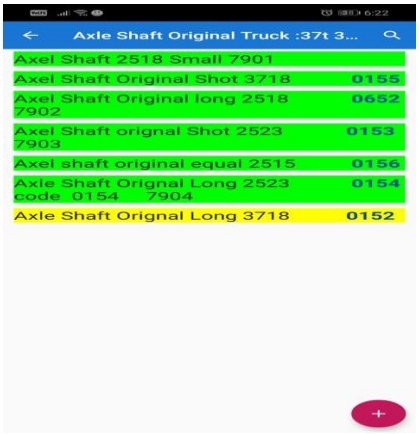


Fig. 4. (Comp

Fig. 5. (Add Partno Activity.java)

Fig. 6. (PartnoDetails.java)

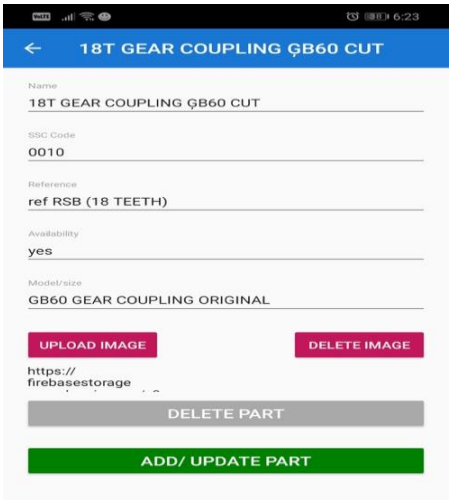


Fig. 7. (Add Partno Details.java)

Fig. 8. (ImageActivity.java)

REFERENCES

[1] http://firebase.google.com
[2] www.youtube.com
[3] www.stackoverflow.com
[4] geeksforgeek website