

ReTrade – Second hand product selling/buying app for JUIT

Project report submitted in partial fulfillment of the requirement for the
degree of Bachelor of Technology

in

Information Technology

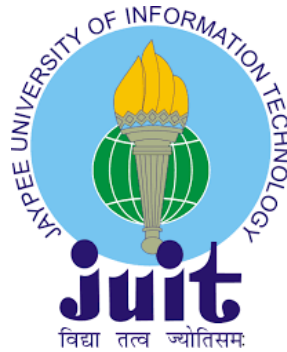
By

Dhruv Dhingra (161457)

Under the supervision of

Dr. Aman Sharma

to



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Waknaghat, Solan-
173234, Himachal Pradesh**

CERTIFICATE

Candidate's Declaration

I hereby declare that the work presented in this report entitled “**ReTrade – Second-hand product selling/buying app for JUIT**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** submitted in the Department of Computer Science Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2019 to May 2019 under the supervision of **Dr Aman Sharma**, Assistant Professor, Computer Science and Engineering and Information Technology Department.

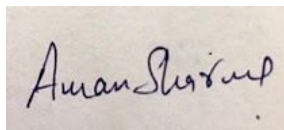
The matter embodied in the report has not been submitted for the award of any other degree or diploma.



(Student Signature)

Dhruv Dhingra, 161457

This is to certify that the above statement made by the candidate is true to the best of my knowledge.



(Supervisor Signature)

Dr Aman Sharma

Assistant Professor

Computer Science Engineering and Information Technology Department

ACKNOWLEDGEMENT

I owe my profound gratitude to my project supervisor Dr Aman Sharma, who took keen interest and guided me all along in my project work titled **ReTrade Second-hand product selling/buying app for JUIT**, till the completion of my project by providing all the necessary information for developing the project. The project development helped me in learning new technologies and I got to know a lot of new things in my domain. I am really thankful to him.

TABLE OF CONTENT	Page No.
Chapter-1 INTRODUCTION	1-5
Chapter-2 LITERATURE SURVEY	6-8
Chapter-3 SYSTEM DEVELOPMENT	9-26
Chapter-4 PERFORMANCE ANALYSIS	27-28
Chapter-5 CONCLUSION	29-32
Chapter-6 REFERENCES	33

List of Figures	Page No.
Figure 3.1 Use Case diagram of ReTrade Application	14
Figure 3.2 Block diagram of ReTrade Application	15
Figure 3.3 Activity diagram of ReTrade Application	16
Figure 3.4 Registration Screen	17
Figure 3.5 Login Screen	17
Figure 3.6 All Items Screen	18
Figure 3.7 My Items Screen	18
Figure 3.8 Item Details Screen	19
Figure 3.9 Notifications Screen	19

Figure 3.10 Drawer Navigator	20
Figure 3.11 Chat Screen	20
Figure 3.12 Add New Item Form	21
Figure 3.13 Edit/Delete My Item Screen	21
Figure 3.14 Firebase Authentication	22
Figure 3.15 Firebase Storage	23
Figure 3.16 users Collection	25
Figure 3.17 items Collection	25
Figure 3.18 contact views Collection	26
Figure 3.19 chats Collection	26

ABSTRACT

One of the major problem students face is finding second-hand textbooks, electronic items and other necessities in college. Fresher students spend a lot of money on purchasing these products which they use only for a brief period of time. The students don't know where to find these products at cheaper costs. To address this issue, we present the ReTrade APP for JUIT students, which is a mobile application; the objective of which is e-sourcing of used products (printed books, stationery, utensils, and electrical/electronic appliances, etc.). This app is designed to facilitate junior students who can buy used products from their seniors thereby saving funds.

This project aims to introduce potential buyers to sellers in a closed community (Jaypee University of Information Technology campus), where buyers can view the sellers' products and can connect with them offline.

React Native is used to coding the front-end, easy-peasy is used for state management of the application and firebase is used to provide various back-end services like authentication, storage, database connectivity and hosting.

Chapter 1

INTRODUCTION

1.1 Introduction

The students usually face several problems in their college life. A problem that they usually encounter is the purchasing of the expensive books, electronic/electric appliances, and other necessities that they require while studying in the college. The students lack a medium to purchase second-hand products.

One of the major obstacles in the sale of second-hand products is the issue of trust between buyers and sellers. Buyers aren't comfortable buying used products from strangers on the internet. This project aims to introduce potential buyers to sellers in a local community that is within our college premises, where buyers can view the sellers' products and can connect with them offline. Our application, called 'ReTrade', aims to reduce the gap between such users.

The goal of our application might often be confused with that of OLX.com or Quikr.com – both of these services provide to the second-hand market. The major difference between ReTrade and these websites is that we plan to cater to a closed community, to maintain the trustworthiness between the seller and the buyer.

The Re-Trade application allows the sellers to advertise their products that they wish to sell off and the potential buyers can look for the products they are willing to acquire at a reduced, reasonable price.

1.2 Problem Statement

We observed these particular problems in our university and decided to resolve it using our application.

1. Fresher students spend mindlessly on books and stationery that are useful only for a semester.
2. Final year students tend to throw most of their belongings which could be useful for the junior students.
3. After the end of a course, a particular book is of no use to a student in a senior class. However, the book can be really useful to a junior student who is going to pursue it in his forthcoming semester.
4. Websites like OLX and Quikr connect anonymous sellers and buyers without even a bit of faith and trust in each other. Our application connects university students among their peers who already know each other thereby ensuring a great amount of trust while trading a product.

We agree that the idea of a second-hand product trading application is not a unique one. There already exist several applications that help you acquire the second-hand products at a high level like trading within a city. This application is developed keeping in view that it should be used only by a closed community where there already exists quite a lot of trust between the trading parties. Since our college is in a remote, hilly area, the neighbouring areas of our college like Shimla and Solan are not easily accessible to the campus students. As a result, there is a great need for junior students of the JUIT community to depend on each other and reuse the products that are no more significant to their seniors.

1.3 Objectives

The objectives of our mobile application, named ReTrade are as follows:

Solve a real-world problem using a software solution.

This mobile application makes it easier for students to find and purchase the second-hand products at a cheaper price online itself.

Study and explore the popular technologies that are widely used for cross-platform (Android and iOS) mobile app development.

The technologies studied and explored for the project include cross-platform application development, React-Native, React, Redux/Easy-Peasy application state management, Firebase, Metro-bundler, Expo and Android Studio.

Make sure the mobile application is scalable.

In the future, it might be possible that this application is used by hundreds of users simultaneously. Therefore, to ensure that the application runs smoothly and efficiently in this scenario, we decided to choose the Google Cloud backend that makes sure to give fast response times no matter any number of users accessing the application at the same time.

Design the database for the application.

Cloud Firestore is used as the database for this project. This database is flexible enough to accommodate any type of data structure and programming can be easily done.

Design the user-interface screens.

The React-Native technology and the related libraries provide the capabilities to design slick, smooth, and responsive interfaces.

Test and debug the application.

Debugging React Native applications is similar to that of debugging JavaScript. Without configuring anything one can simply rely on Chrome developer tools to debug the front-end code of the app.

Deploy the application

This application is deployed on Google's cloud-enabled Firebase which is a mobile application development platform providing numerous services like authentication, storage, analytics, and hosting out-of-the-box.

1.4 Software Development Process

We followed an iterative software development approach. In each iteration, you do a little requirement capture, analysis, design, coding, and testing, facilitating the evolution and refinement of the system. Each iteration produces an executable release that is one step closer to a final solution. These development steps are applied iteratively until you get a final product.

The system is divided into modules. Each iteration produces a partial working implementation instead of a total system at one go. Each successive iteration builds on the work of previous iterations. When partial implementation is successful, you add more functionality, increased performance, etc.

In the first iteration, this is how we worked to achieve a subset of the ultimate goal. We designed a few of the mobile User Interface screens using the Blasmiq wireframing tool and showed them to the end-users of the application. Then we wrote the code for those screens followed by their testing and debugging. Using the appropriate software development tools, we created an Android version of the application and deployed it on the Android phone. The functionality that we implemented during this iteration is User Registration, Login & Logout, Create and Update one or more products by an end-user, and List / View of the products

Along the similar lines of the first iteration, we worked upon the second iteration. We added the Filter functionality to the Product Listing. Then we implemented a complete Chat module in this iteration.

In the final iteration, we added one more functionality which is as follows. When a user decides not to use the application anymore, he can delete himself and consequently, all his products deleted from the system.

Chapter 2

LITERATURE SURVEY

A thorough study of published material related to the project is one of the significant steps in planning and implementing the project. Given below is the survey of several frameworks and libraries that are used for the development of JavaScript mobile applications.

React Native[1]

React Native is a framework for building native apps for iOS or Android using JavaScript. With React Native one doesn't need to know iOS or Android programming unless they want to build a complex app and need to access the native API of those platforms. Application code is written using the JavaScript language only and the same code can be shared across iOS and Android. The User Interface can be represented in an abstract or platform-independent way as a composition of components. The React Native maps these components into their native widgets. React native provides various stateless components like View, Button, Text, etc. that can be composed to build interactive and responsive interfaces. With a rich eco-system built around React Native, it is a very good choice to build cross-platform applications.

React Hooks[2]

Hooks are a new feature addition in React version 16.8 which allows usage of React features (for example state of a component) without having to write a class. Hooks are useful in sharing stateful logic in a better way. Hooks don't contain any breaking changes and the release is 100 percent backwards-compatible.

Redux[3]

Redux is a state management library for JavaScript applications written using React, Vue like libraries. An application with a complex user interface requires to keep different parts of the UI consistent because the application state is changing in response to every action. The application state is stored inside a central repository that is a single JavaScript object called the store. So, with this architecture, the different pieces of user interface no longer need to maintain their state, instead they get what they need from the central, single store object. Redux makes data flow transparent and predictable. Redux works with any libraries for building UIs; one can use it with React Native, React, Angular and Vue. On the downside, Redux introduces some indirection and complexity in the code. It is usually not used when the application is not so complex.

Easy-Peasy[4]

It is a simple library written on top of Redux that allows the programmer to easily write the state management code making the development process less tedious. In other terms, the programmer does not need to write the Redux boilerplate code.

Cloud Firestore[5]

Cloud Firestore allows us to store the data in the cloud, so the user can sync it across all the devices or share them among multiple users. It also allows structuring of data in ways that make more sense because of its powerful querying and fetching capabilities. It works in near real-time, automatically fetching changes from the database as they take place. It scales automatically as the usage grows. It gives the developer serverless development experience because of no infrastructure issues. The features like setting up the backend, scalability, data replication, recovery from server failure, and other administrative tasks are automatically taken care of by the Google Cloud Firestore.

Firebase Storage[6]

Firebase Storage allows storage and retrieval of user-generated content like images, audio, and video without the need of a self-administered server. The Google Cloud platform takes care of the administration.

Metro bundler[7]

Building and running a React Native project starts up a packager called Metro. The packager bundles all the JavaScript code into a single file and translates newer versions of JavaScript like ES6 into native elements.

React Native - Building Mobile Apps with JavaScript[8]

This book written by Vladimir Novick in 2017, is a handbook for making you understand how to write the mobile applications for the Android and iOS platforms using a single codebase. It makes us learn how Instagram, Twitter-like mobile applications were built using the React-Native framework. It also helps to get acquainted with different APIs provided by the React-Native framework as well as understanding the application release process in detail.

JavaScript mobile frameworks comparison: React Native vs Ionic vs Native Script[9]

This article by Bhagyashree published in 2018 compares three different JavaScript frameworks for mobile application development. Although these frameworks have the same objectives, choosing the best one depends on the user's functional requirements, other requirements discussed below, and the skill set available with the existing developers.

Ionic has the benefit of having a single codebase but it should not be used for graphic-intensive applications. React-Native has better performance and optimization capabilities compared to the other two but has the overhead of mapping generic components to the native elements of Android and iOS. Although the Native Script is quite powerful, the downside is that it makes the application's size large.

Chapter 3

SYSTEM DEVELOPMENT

3.1 Requirements

Following functional requirements have been identified which we are writing in the form of use cases.

1. Sign up User
2. Login User
3. Logout User
4. List All Products
5. List My products
6. Search/Filter Product
7. Add New Product
8. Update Product
9. Delete Product
10. Contact Seller
11. Contact Buyer
12. Chat with Other Users

1. Sign Up User use case

- a. The user enters his email address, name, mobile number, and password.
- b. The system validates the input and saves the profile details in the database.
- c. If the input data is not valid or empty, the system prompts the user with an appropriate message to enter the data again.

2. Login User use case

- a. The user enters his email address and password.
- b. The system validates the input.
- c. If the input data is not valid or empty, the system prompts the user with an appropriate message to enter the data again.
- d. If the email is not found or the password is incorrect, the system prompts the user with a login error message.
- e. After successful verification, the system displays the four-tab buttons at the bottom of the screen. They are: List All Products, List My Products, Notifications, Messages.

3. Logout User use case

- a. The user clicks the logout button.
- b. After successful logout, the user cannot use the system anymore.

4. List All Products use case

- a. When a user signs in, this is the default screen that is displayed to the user.
- b. The system displays the list of all the products that are created by all the users including the products that are owned/created by the signed-in user.
- c. Each row in the list represents an individual product.
- d. A thumbnail of each product's image is displayed on the left-hand side of a row. The product's title and its price are displayed in the middle. The product's category is displayed on the right-hand side of the row.

5. List My Products use case

- a. The user clicks the second tab at the bottom of the screen i.e. List My Products icon.
- b. The system displays the list of only those products that are owned/created by the signed-in user.
- c. Each row in the list represents an individual product.
- d. A thumbnail of each product's image is displayed on the left-hand side of a row. The product's title and its price are displayed in the middle. The product's category is displayed on the right-hand side of the row.

6. Search/Filter products

- a. In both the "List All Products" and "List My Products" screens, the system displays the search bar at the top of the screens.
- b. If a user taps in a Search bar, a list of all the product's categories is displayed. It also includes one more option namely, All.
- c. If the user selects the All option, all the products are displayed. If the user selects a particular category option, let's say, Electronics, the list of all Electronic products is displayed.
- d. The user can also type one or characters in the Search bar. As soon as the user enters the characters, the system filters the list based on the Title of the products. For example, the user types these characters: "Head", the products whose title begins with "Head" like "Headphones" are filtered and displayed. The remaining products are not displayed.

7. Add New Product use case

- a. At the bottom right of the "List My Products" screen, the system displays the 'Add' icon.
- b. The user clicks the Add icon
- c. The system displays a predefined list of categories in the Category field. They are Books, Stationery, Electronics, Accessories, Furnishing, Utensils, and Others.

- d. The user selects a particular category from the list to which a new product to be added belongs.
- e. The user then enters the title and description of the product.
- f. The user enters the price of the product at which he wishes to sell the product.
- g. The user also specifies that the product is available in the Availability Status field.
- h. The user can upload one or more images of the product by clicking the Add Photos button.
- i. The user clicks the Save button.
- j. The system saves the product details including the images in the database.

8. Update Product use case

- a. In the "List My Products" screen, the system displays the list of products that are owned/created by the logged-in user.
- b. The user clicks a particular product whose details he needs to update.
- c. The system displays the details of the selected product and the Update button at the bottom of the screen.
- d. The user clicks the Update button.
- f. The user can now update any of its details. For instance, if the product is no longer available, the user sets its Availability Status to False. Or, if he wants to decrease its price, he can re-enter a new value.
- g. The user may also add new or delete existing images of the product.
- h. The user clicks the Save button.
- i. The system updates the product details in the database.

9. Delete Product use case

- a. In the "List My Products" screen, the system displays the list of products that are owned/created by the logged-in user.
- b. The user clicks a particular product that he needs to delete.

- c. The system displays the product's information and also the Delete button at the bottom of the screen.
- d. The user clicks the Delete button.
- e. The system seeks the confirmation whether the user really wants to delete the product.
- f. When the user confirms, the system deletes the product from the database.

10. Contact Seller use case

- a. In the "List All Products" screen, the system displays the list of all the products that are on sale.
- b. The user taps on the product that he is interested to buy.
- c. The system displays the product's information and the 'Contact Seller' button at the bottom of the screen.
- d. The user clicks the Contact Seller button.
- e. The system displays the Phone mobile app icon.
- f. The user clicks it.
- g. The system automatically transfers the seller's mobile number to the Phone app from where he can make the call.

11. Contact Buyer use case

- a. This is basically an extension to the Contact Seller use case.
- b. When a potential buyer clicks the Contact Seller button (in the Contact Seller use case), the system automatically sends a notification to the seller about the details of the potential buyer.
- c. The seller user can view all such notifications in the third tab i.e. Notifications and gets to know who are the interested buyer users.
- d. He clicks a particular notification.
- e. The system displays the product's details and the "Contact Buyer" button.
- f. He clicks the Contact Buyer button.

- g. The system redirects the user to the Phone app from where he can make the call.
- h. The system displays the Phone mobile app icon.
- i. The user clicks it.
- j. The system automatically transfers the seller's mobile number to the Phone app from where he can make the call.

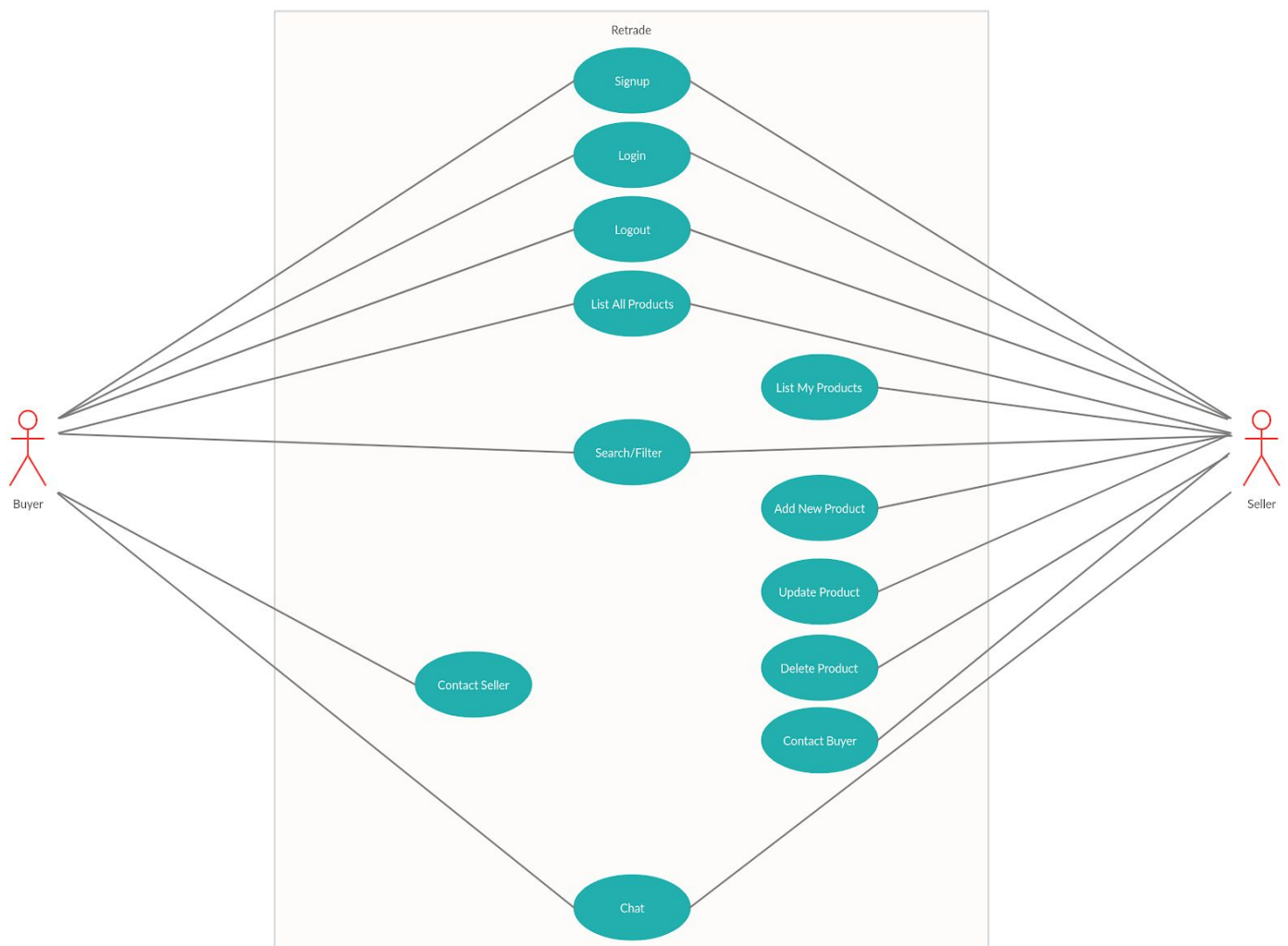


Figure 3.1 Use Case diagram of ReTrade Application

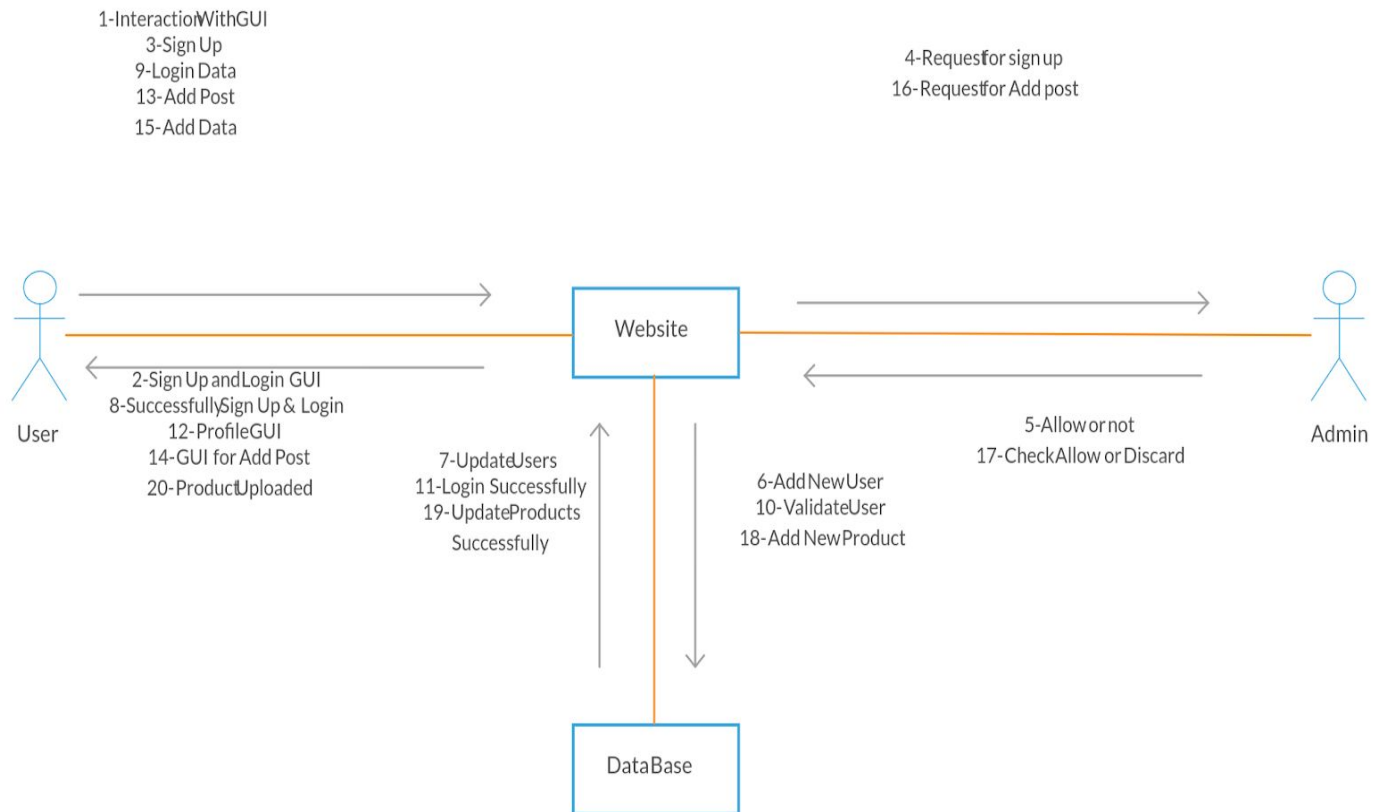


Figure 3.2 Block diagram of ReTrade Application

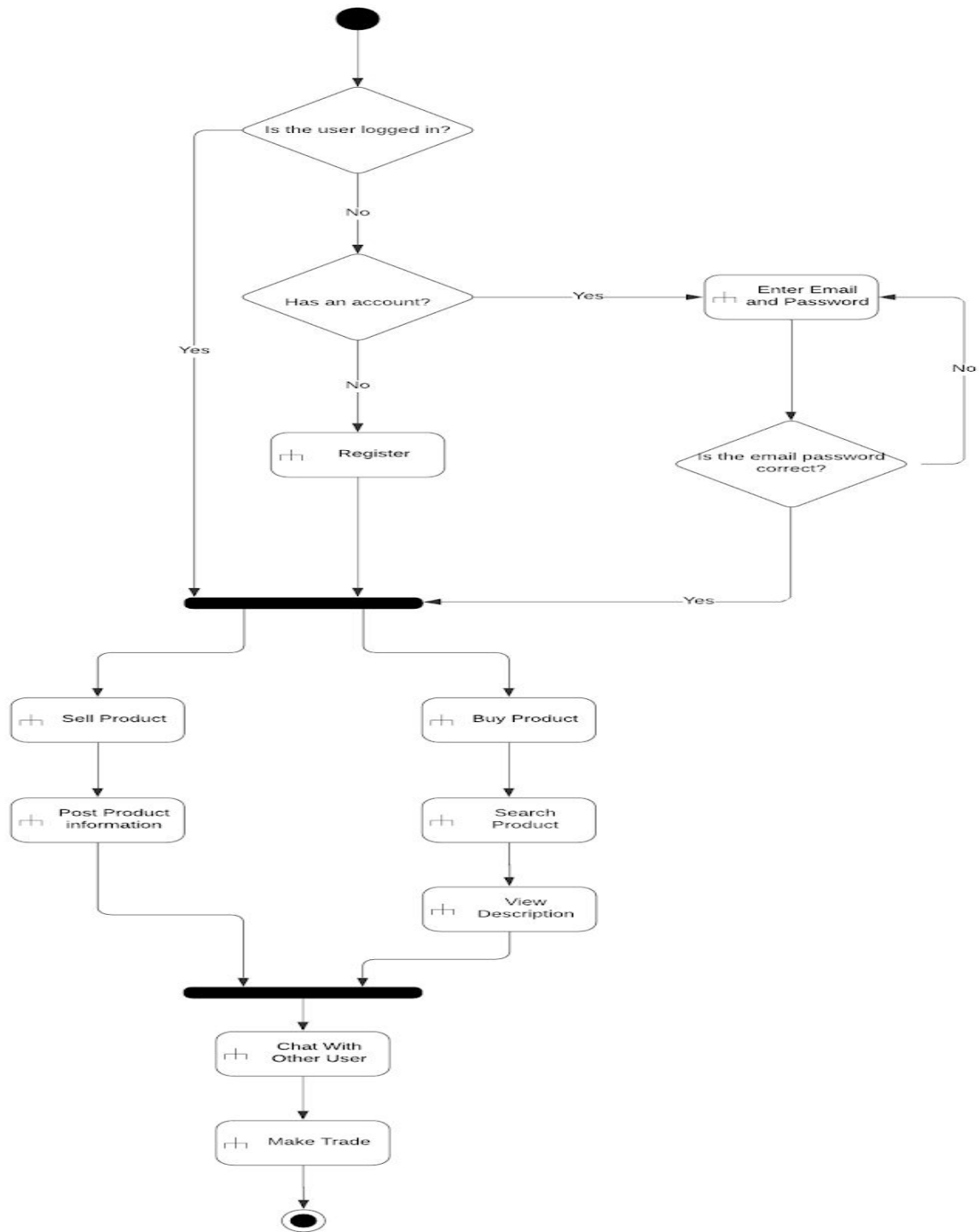
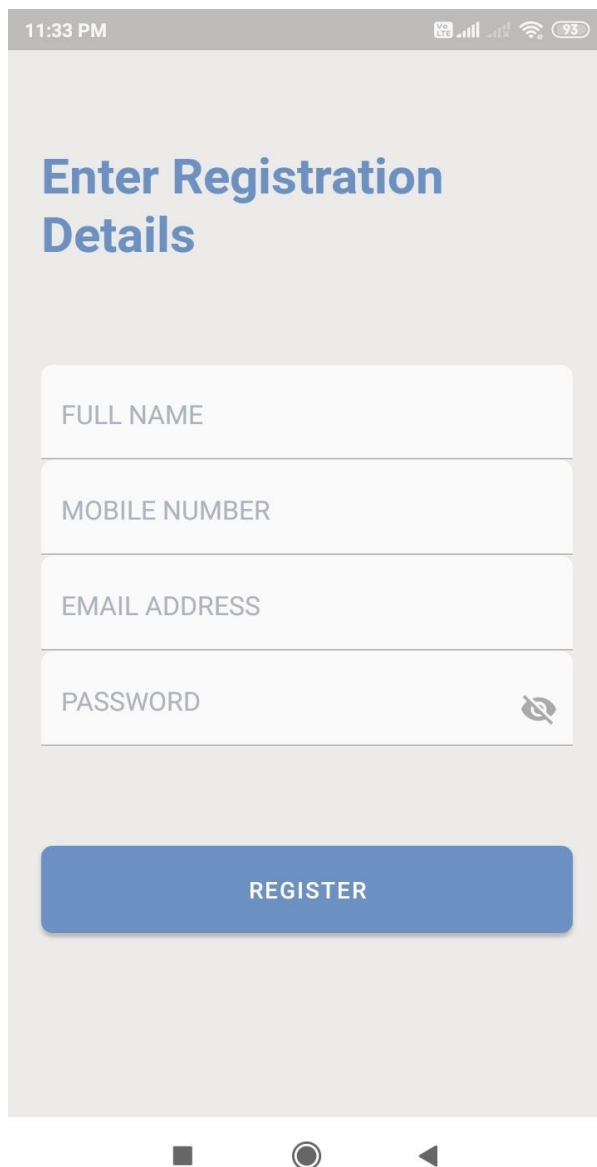


Figure 3.3 Activity diagram of ReTrade Application

3.2 User Interfaces



11:33 PM

Enter Registration Details

FULL NAME

MOBILE NUMBER

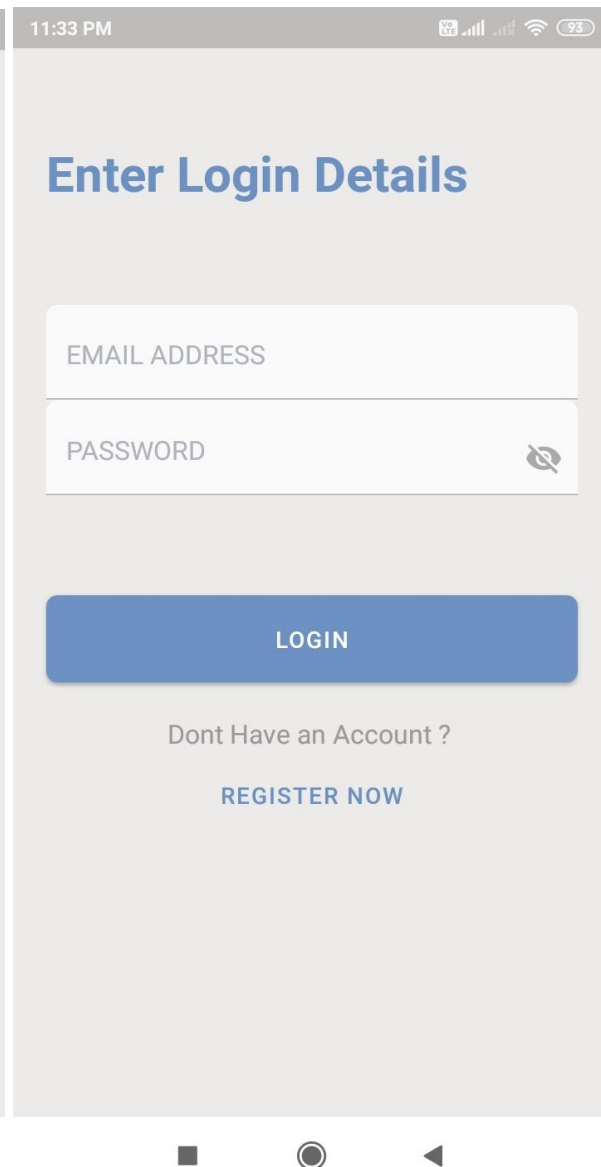
EMAIL ADDRESS

PASSWORD

REGISTER

The registration screen features a light gray background. At the top, the status bar shows the time as 11:33 PM, along with LTE, signal strength, Wi-Fi, and a 93% battery icon. The title 'Enter Registration Details' is in a bold blue font. Below the title are four white input fields with gray borders and placeholder text: 'FULL NAME', 'MOBILE NUMBER', 'EMAIL ADDRESS', and 'PASSWORD'. The password field includes a small eye icon on the right. A large blue 'REGISTER' button is positioned at the bottom of the form area. The Android navigation bar is visible at the very bottom.

Figure 3.4 Registration Screen



11:33 PM

Enter Login Details

EMAIL ADDRESS

PASSWORD

LOGIN

Dont Have an Account ?

REGISTER NOW

The login screen has a light gray background. The status bar at the top shows 11:33 PM, LTE, signal strength, Wi-Fi, and a 93% battery icon. The title 'Enter Login Details' is in a bold blue font. There are two white input fields with gray borders and placeholder text: 'EMAIL ADDRESS' and 'PASSWORD'. The password field has an eye icon on the right. A large blue 'LOGIN' button is centered below the input fields. Below the button, the text 'Dont Have an Account ?' is displayed in a small gray font, followed by a blue 'REGISTER NOW' link. The Android navigation bar is at the bottom.

Figure 3.5 Login Screen

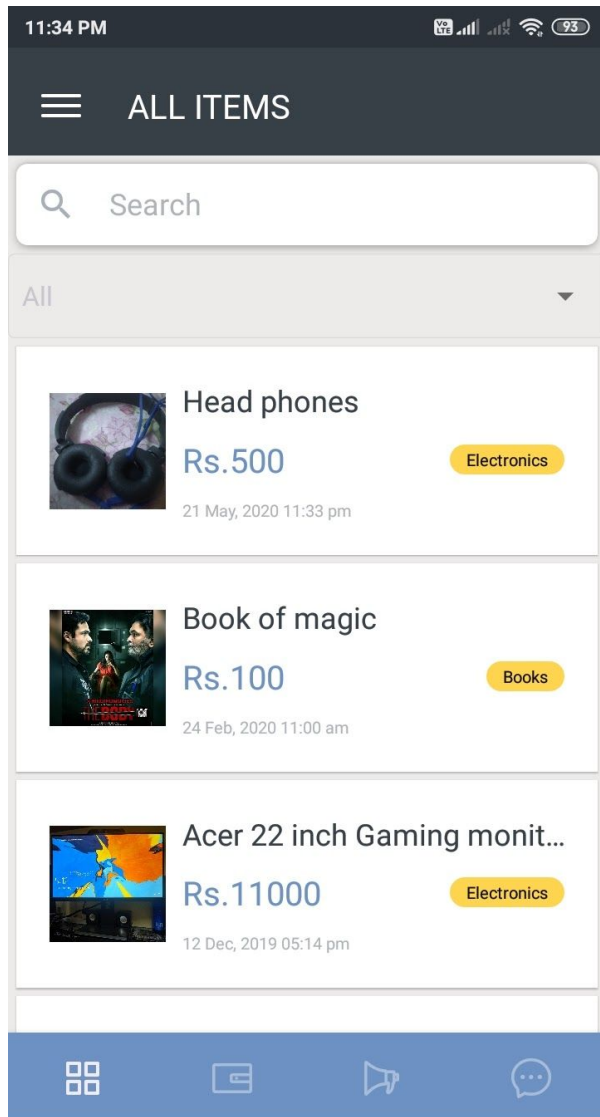


Figure 3.6 All Items Screen

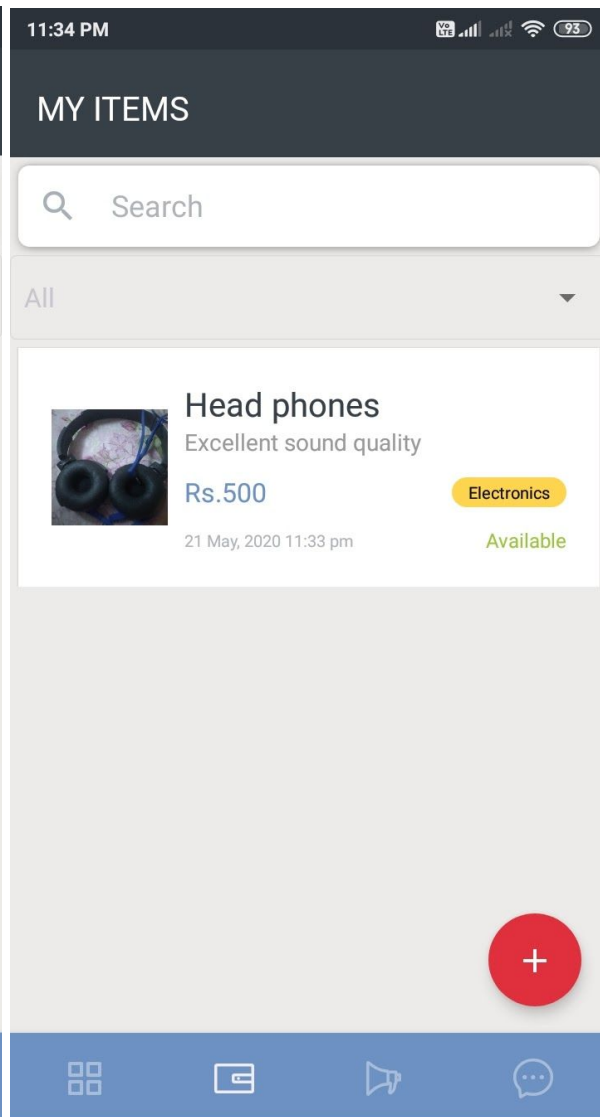


Figure 3.7 My Items Screen

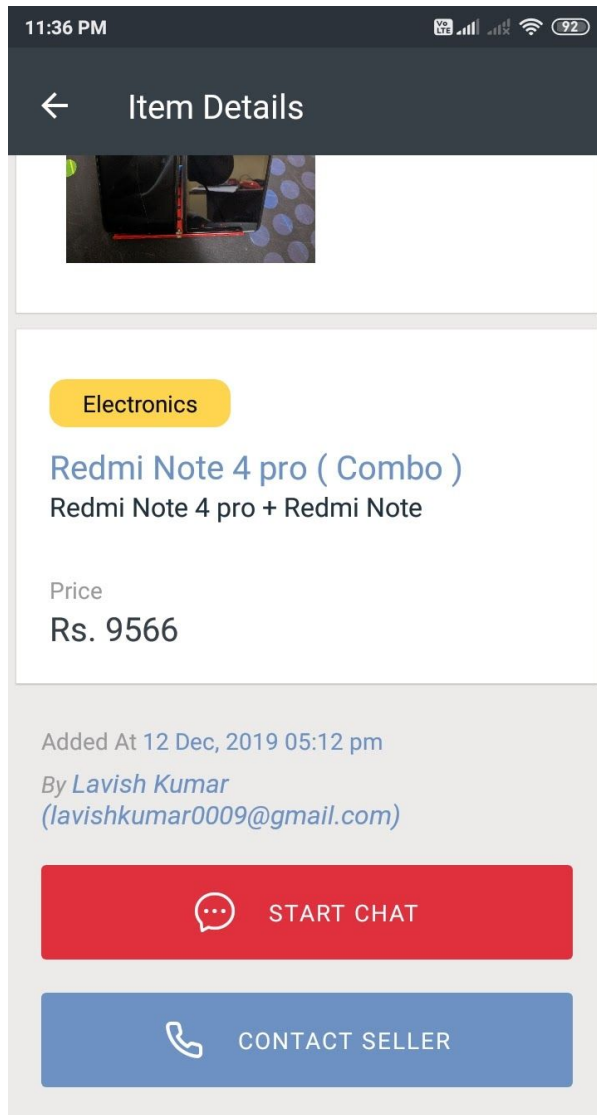


Figure 3.8 Item Details Screen

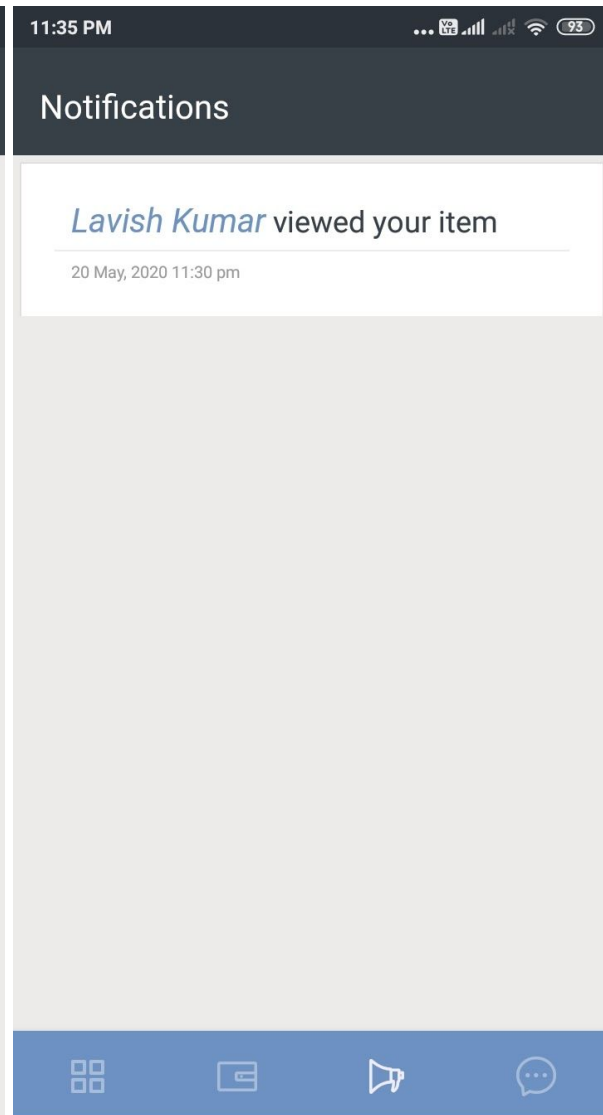


Figure 3.9 Notifications Screen

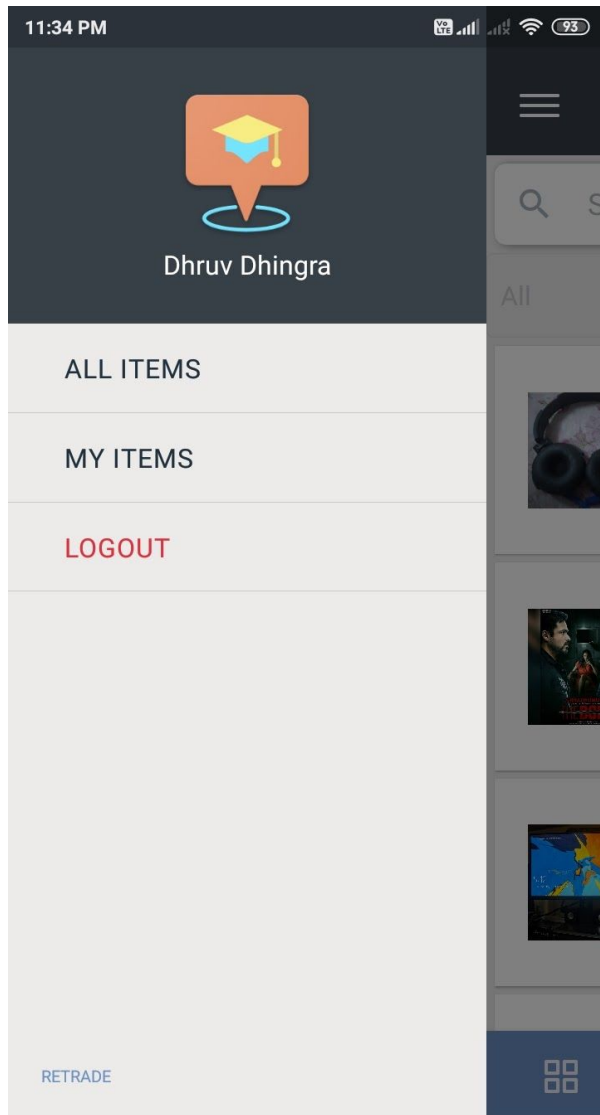


Figure 3.10 Drawer Navigator

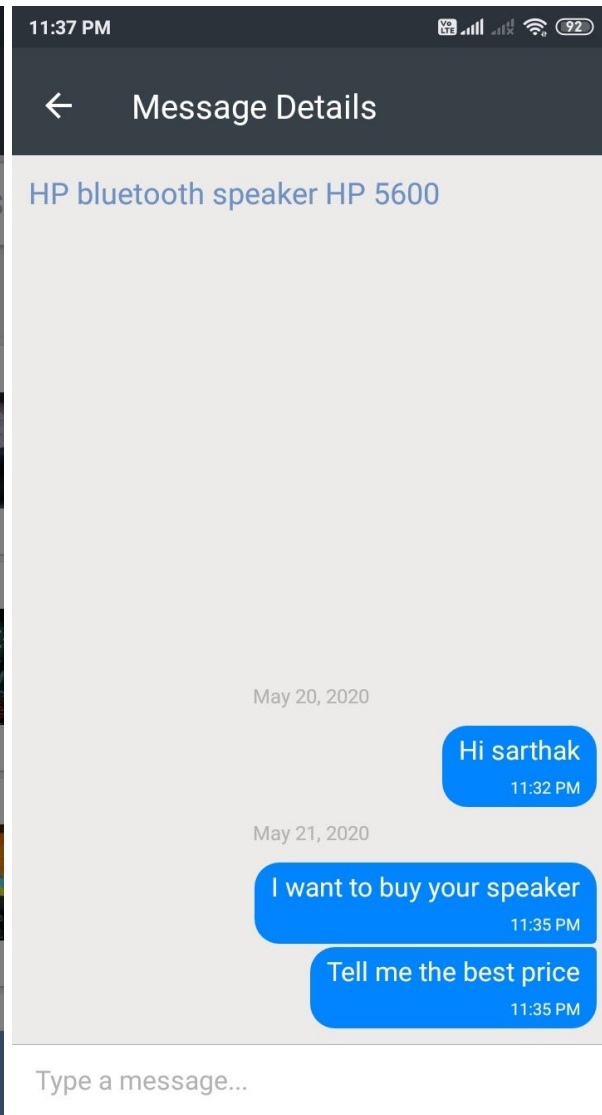


Figure 3.11 Chat Screen

2:35 PM

← Create New Item

Add New Item

Select a category to which the item b.. ▼

Item Title
Enter a suitable title for the item

Item Description
Describe it in not more than 50 characters

Item Price
Price, specify 0 if you want to give it for free


Is Item Available? ☒

Add Files SELECT PHOTO

Figure 3.12 Add New Item Form

2:40 PM

← Item Details



Electronics

Head phones
Excellent sound quality

Price **Rs. 500** Availability Status **Available**

Added At 21 May, 2020 11:33 pm
By You (dhruv@gmail.com)

EDIT

DELETE

Figure 3.13 Edit/Delete My Item Screen

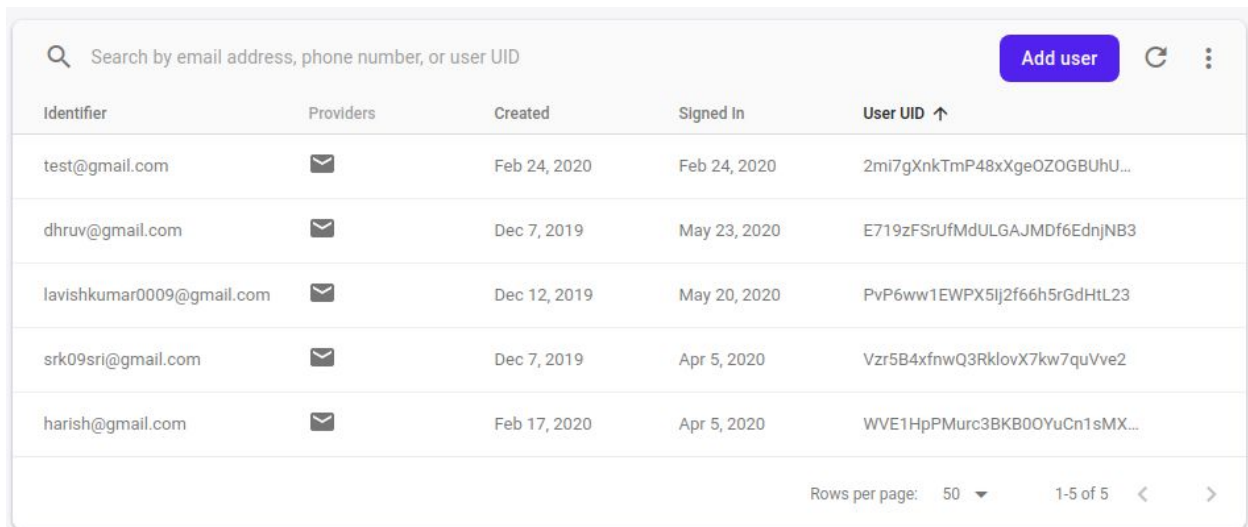
3.3 Backend Services

Firestore Authentication

Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices.

Firestore Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Firestore Authentication integrates tightly with other Firestore services, and it leverages industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.



The screenshot shows the Firebase Authentication console interface. At the top, there is a search bar with the placeholder text "Search by email address, phone number, or user UID". To the right of the search bar are two buttons: "Add user" (in a blue box) and a refresh button (circular arrow icon). Below the search bar is a table with five columns: "Identifier", "Providers", "Created", "Signed In", and "User UID" (with an upward arrow icon). The table contains five rows of user data. At the bottom right of the table, there is a pagination control showing "Rows per page: 50" and "1-5 of 5" with navigation arrows.











Identifier	Providers	Created	Signed In	User UID ↑
test@gmail.com	✉	Feb 24, 2020	Feb 24, 2020	2mi7gXnkTmP48xXgeOZOGBUhU...
dhruv@gmail.com	✉	Dec 7, 2019	May 23, 2020	E719zFSrUfMdULGAJMDf6EdnjNB3
lavishkumar0009@gmail.com	✉	Dec 12, 2019	May 20, 2020	PvP6ww1EWPX5Jj2f66h5rGdHtL23
srk09sri@gmail.com	✉	Dec 7, 2019	Apr 5, 2020	Vzr5B4xfnwQ3RklovX7kw7quVve2
harish@gmail.com	✉	Feb 17, 2020	Apr 5, 2020	WVE1HpPMurc3BKB00YuCn1sMX...


Rows per page: 50 1-5 of 5 < >


Figure 3.14 Firestore Authentication

Firestore Storage

Cloud Storage for Firestore is a powerful, simple, and cost-effective object storage service built for Google scale. The Firestore SDKs for Cloud Storage add Google security to file uploads and downloads for your Firestore apps, regardless of network quality. You can use their SDKs to store images, audio, video, or other user-generated content. On the server, you can use Google Cloud Storage, to access the same files.

retrade ▾ Storage					Go to docs 🔔 d	
<input type="checkbox"/>	Name	Size	Type	Last modified		
<input type="checkbox"/>	 1575729249013.jpeg	1 MB	image/jpeg	Dec 7, 2019		
<input type="checkbox"/>	 1576138883230.jpeg	1.26 MB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576138924656.jpeg	1.27 MB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576139937951.jpeg	1.12 MB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576141465841.jpeg	951.75 KB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576147204787.jpeg	445.71 KB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576150628588.jpeg	976.62 KB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576150759265.jpeg	921.75 KB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576150902651.jpeg	1.39 MB	image/jpeg	Dec 12, 2019		
<input type="checkbox"/>	 1576151019910.jpeg	1.13 MB	image/jpeg	Dec 12, 2019		

 1576150628588.jp... ✕



Name

[1576150628588.jpeg](#) 🔗

Size

1,000,060 bytes

Type

image/jpeg

Created

Dec 12, 2019, 5:07:27 PM

Updated

Dec 12, 2019, 5:07:27 PM

File location

▾

Other metadata

▾

Figure 3.15 Firestore Storage

Cloud Firestore

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform.

Like Firebase Realtime Database, it keeps your data in sync across client apps through real time listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity.

Cloud Firestore also offers seamless integration with other Firebase and Google Cloud Platform products, including Cloud Functions.

Document databases store data in documents similar to JSON (JavaScript Object Notation) objects. Each document contains pairs of fields and values. The values can typically be a variety of types including things like strings, numbers, booleans, arrays, or objects, and their structures typically align with objects developers are working with in code. Because of their variety of field value types and powerful query languages, document databases are great for a wide variety of use cases and can be used as a general purpose database. They can horizontally scale-out to accomodate large data volumes.

ReTrade application has 4 collections namely: -

1. users
2. items
3. contactViews
4. chats

Database Design

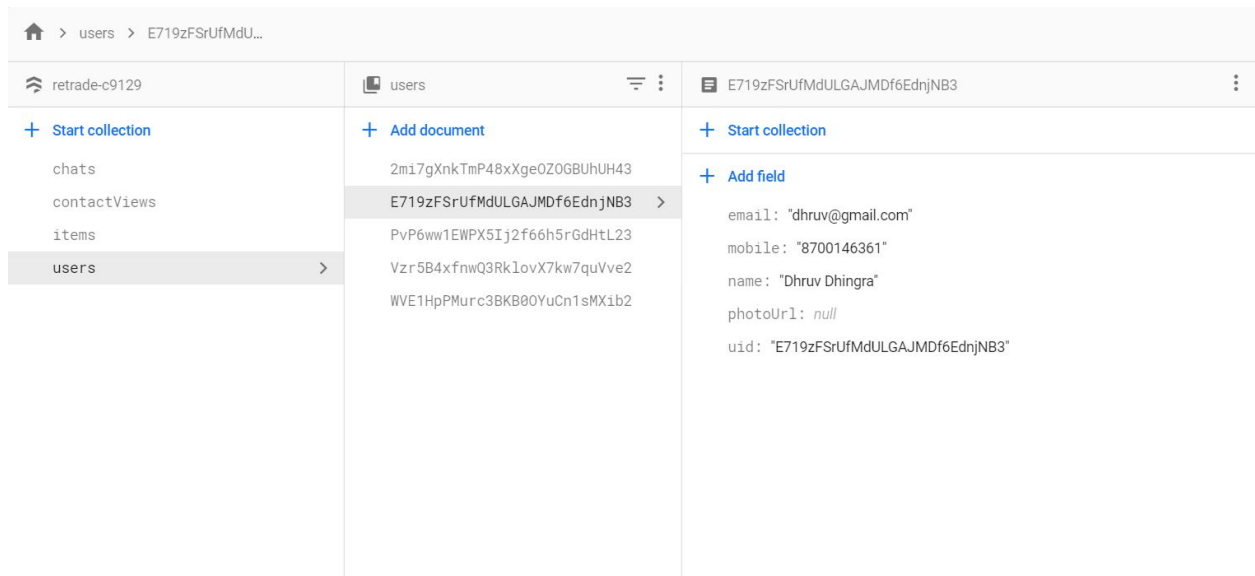


Figure 3.16 users Collection

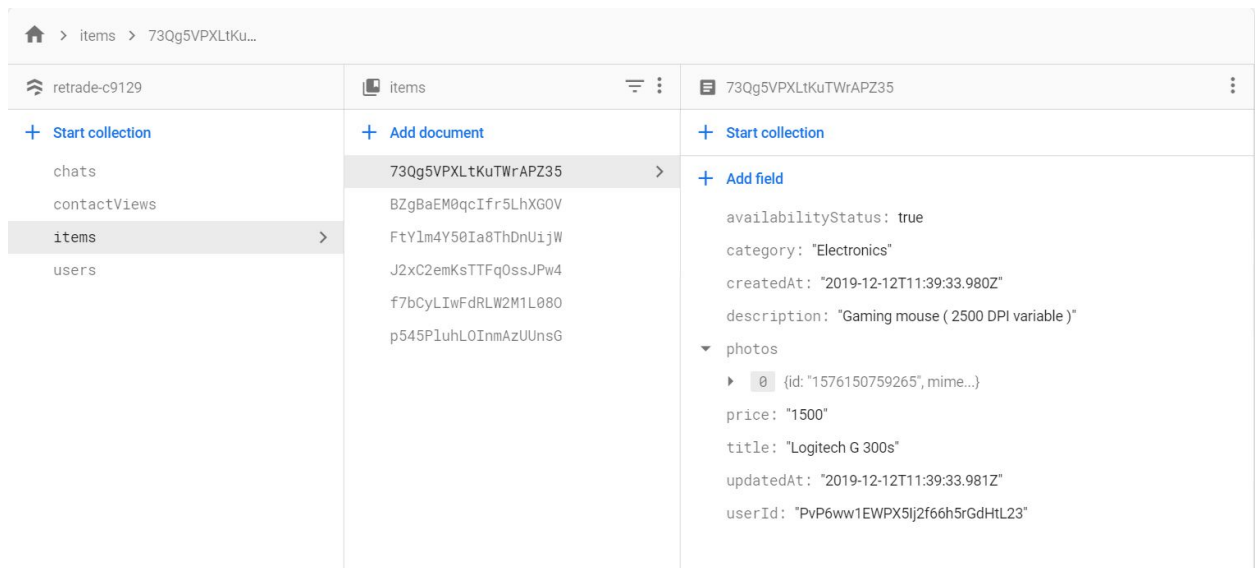


Figure 3.17 items Collection

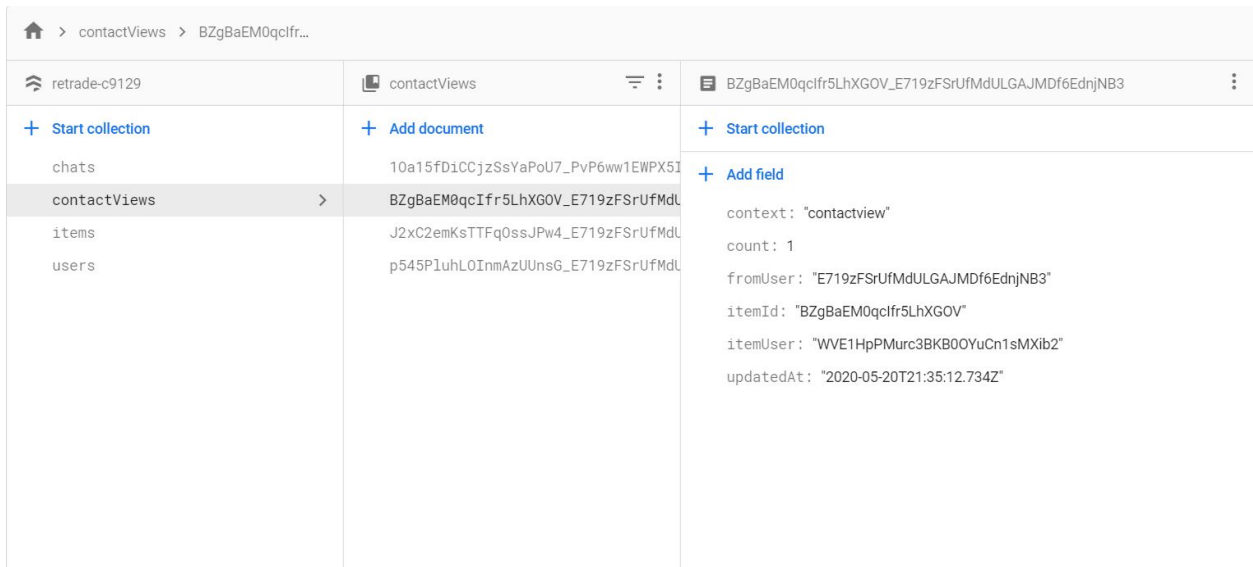


Figure 3.18 contactViews Collection

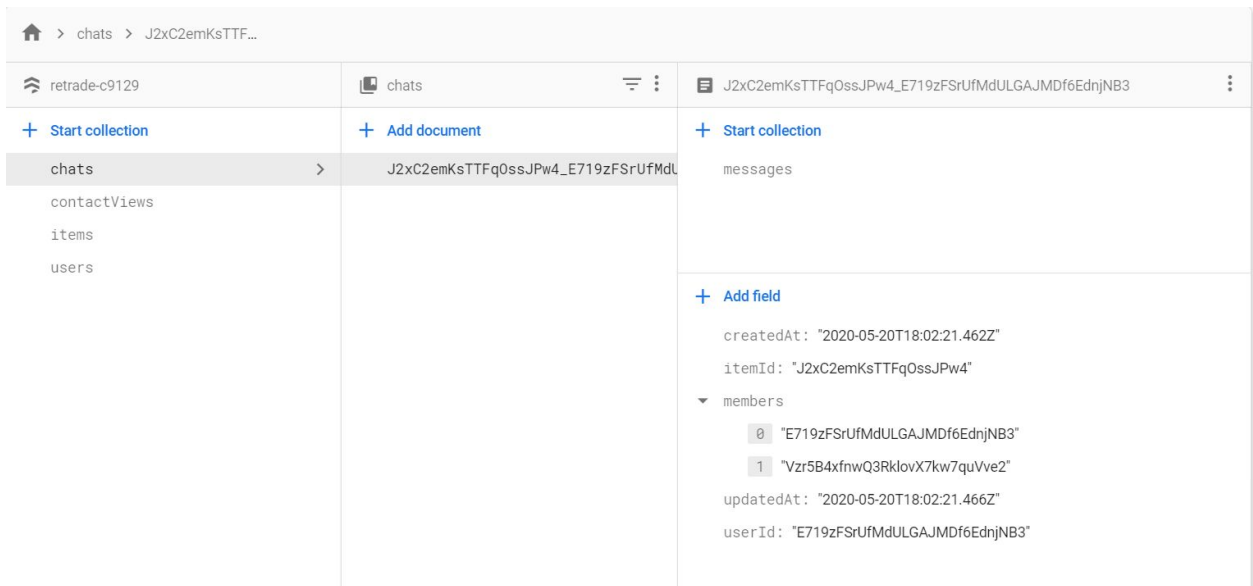


Figure 3.19 chats Collection

Chapter 4

PERFORMANCE ANALYSIS

We tried to deliver a smooth user-interface experience by manually doing these performance optimisations.

Removing console.log statements

When running a React-Native application, console.log statements impedes the execution of JavaScript threads. We used a babel plugin that automatically removes all the console.* statements in the deployed versions of the project.

Using FlatList component instead of ListView component

The initial rendering of the ListView component is too slow. Also the scrolling performance is bad for larger lists.

We used the FlatList component instead of the ListView component. FlatList component simplifies the API and has notable performance enhancements like: -

1. constant usage of memory for any number of rows.
2. optimized rendering speed.

Using useMemo hook

We used the useMemo hook to avoid time-consuming calculations on every render.

Innate performance optimization of React-Native

Just like Reactjs's Virtual DOM, React-Native creates a tree hierarchy to define the initial layout and creates a diff of that tree on each layout change to optimize the renderings. Except React-Native manages the UI updates through a couple of architecture layers that in the end translate how views should be rendered while trying to optimize the changes to a minimum in order to deliver the fastest rendering possible.

Chapter 5

CONCLUSION

5.1 Future Scope

If we add/implement a few more features in this application, it could have a very bright scope in future and can be used by thousands of users worldwide.

The most important feature that is still not implemented is an online payment when a buyer is supposed to pay to a seller. Effectively speaking, a transaction of an item is considered to complete only when the buyer gets the money. It'd be great if we implement this feature in the next version of the application.

There are students who are rich and kind enough who are willing to donate their books and other belongings for free before they leave the campus. It is possible that there could be no student who needs the things at that time when a senior student wants to donate them. So the college authorities can make an arrangement where all such things can be kept. The system administrator can maintain the details of those things on the system that are available for free. So whenever a new/junior student needs something which he thinks is quite expensive for him to buy may browse the list of those things and get it from there.

There could be a scenario where a buyer thinks that a seller is demanding more price than it is actually worth. So we can implement a feature where the two parties can negotiate the pricing in real-time. If you look at this feature from a sophisticated point of view, things can be re-traded just like the price of shares in a stock exchange are bought and sold and their price being negotiated in real-time.

Right now, the application has a simple User Interface. Based on the users' feedback, we can continually refine the User Interface (screens) of the application and make it a very elegant and more user-friendly.

Few of the university campuses are built on a huge area and students have to walk a lot from one place to another, like for example, a hostel could be far off from his/her place of study. If we talk about our campus, it is located in a remote hilly area which is 4 kms away from a town called Wagnaghat. It becomes really difficult when one has to go shopping to Wagnaghat all the way walking. Or, a student needs to hire a taxi from the college to Wagnaghat, when he needs to visit his home during the college holidays.

Since this application is intended to be used by a closed community, for example, a college or a university campus, we can upload the application on Google Play Store and let other closed communities like other colleges may use the same application. This application would become a top-notch if we introduce a feature called vehicle pooling. Few students have their own cars or bikes, they can inform the system when they are willing to share a ride with others.

5.2 Conclusion

We have put in a lot of effort in developing this application. As discussed in the "Future Scope" section of the report, we still need to put more effort in order to make the application really usable. We learnt a lot while implementing this real-world project. I'd like to share my most important experiences that we had during this project. They are:

a. The proper analysis of the functional requirements of an application to be developed is a must. The thinking of a developer should be in perfectly alignment with what the end users really need to have. Sometimes the functional requirements of the application are perceived from a different perspective what the end users really want. And the project ultimately fails when the final system

is shown to the end users. A constant interaction with the end users from time to time is essential. And the analyst must pay full attention to the users' viewpoint.

b. It is rightly said that a picture is worth a thousand words. We consistently put efforts to draw the User Interface screens during the analysis itself and tried our best to get the users' feedback in order to ensure that our thinking remained in sync. During the process, we made constant refinements to the User Interface.

c. Choosing the right architecture including the platform and technologies was another challenge. We needed to make sure that the data became available and visible to the user in an instant. In technical terms, the application should be scalable and it should have the ability to give the same smooth experience to all users no matter thousands of them using the application at the same time. If we study the experiences of thousands of developers, we come to the conclusion that the main bottleneck in scalability is due to use of the relational databases where there are many database tables and an application needs to make use of the JOIN in order to retrieve data from database tables. The more the number of JOINS, the more slow the application becomes. So we decided to use an alternative NoSQL database technology. It is becoming very popular and the data access is really fast when we use NoSQL databases in the backend.

It is a nightmare when an application goes in production and one has to administer it. If we use our own server to deploy an application, it becomes our responsibility to do the different administration tasks 24*7. The tasks include, for instance, handling server failures, application scalability, database backups, database replication. Therefore we decided to use the Cloud functionality that has really become so popular in the last several years and gives us a "serverless" experience meaning that we don't need to bother about the administrative tasks including the scalability. Everything is taken care of by the company that is offering us the ability to deploy our application on the Cloud and they take care of our server activities.

As everybody knows, the most popular mobile platforms are Android and iOS. If we want to release an application for these two different platforms, we need to write the code and maintain it further in two different native languages. For example, Swift is used to write the iOS app code while Java is used to write the Android code. But it quickly becomes a nightmare when we need to maintain the two different codebases in the long run. Also, we need expertise in two different languages. But the IT industry has a solution for it. We can create an application by writing the 'hybrid' code for both iOS and Android using the same language, i.e. JavaScript and there exist libraries that wrap our code in two different iOS and Android formats.

We decided to use React-Native which is one of the most popular libraries that lets us write the hybrid JavaScript code. React-Native is written on another very important library named React which makes sure that the application becomes really fast to refresh the screens as the data changes are by an end user. This React library is the most widely used in the IT industry these days to write the mobile and web applications.

d. To conclude, I'd like to summarize my experience. It takes time to learn the latest technologies of our own and use them to write the application code. One has to have enough patience to learn and then write, debug, and test the code. This project helped me grow not only learning new technologies but other aspects and enhance our skills like analysis, constant communication with end users, and taking bold decisions of our own.

REFERENCES

1. <https://reactnative.dev/> accessed on 9th October 2019.
2. <https://reactjs.org/docs/hooks-intro.html> accessed on 12th November 2019.
3. <https://redux.js.org/> accessed on 30th November 2019.
4. <https://easy-peasy.now.sh/> accessed on 3rd December 2019.
5. <https://firebase.google.com/docs/firestore> accessed on 8th December 2019.
6. <https://firebase.google.com/docs/storage> accessed on 10th December 2019.
7. <https://facebook.github.io/metro/> accessed on 10th December 2019.
8. Vladimir Novick, published on Aug 7 2000, React Native - Building Mobile Apps with JavaScript.
9. JavaScript in mobile applications: React native vs Ionic Vs Native Script Vs Native Development
<https://hub.packtpub.com/javascript-mobile-frameworks-comparison-react-native-vs-ionic-vs-native-script/> accessed on 10 th September 2019.

ORIGINALITY REPORT

16%

SIMILARITY INDEX

8%

INTERNET SOURCES

2%

PUBLICATIONS

14%

STUDENT PAPERS

PRIMARY SOURCES

1	spgon.com Internet Source	2%
2	www.mongodb.com Internet Source	2%
3	Submitted to President University Student Paper	2%
4	firebase.google.com Internet Source	1%
5	Submitted to Multimedia University Student Paper	1%
6	Submitted to Kingston University Student Paper	1%
7	Submitted to Universiti Teknologi MARA Student Paper	1%
8	hub.packtpub.com Internet Source	1%
9	Submitted to University of Essex Student Paper	1%

10	Submitted to University of Ulster Student Paper	1 %
11	espeo.eu Internet Source	<1 %
12	Submitted to Taibah University Student Paper	<1 %
13	Submitted to Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur Student Paper	<1 %
14	John Ciliberti. "ASP.NET MVC 4 Recipes", Springer Science and Business Media LLC, 2013 Publication	<1 %
15	Submitted to University of Surrey Student Paper	<1 %
16	jetrockets.pro Internet Source	<1 %
17	Submitted to Daffodil International University Student Paper	<1 %
18	Submitted to De Montfort University Student Paper	<1 %
19	Submitted to University of East London Student Paper	<1 %
20	"Management Report", University/Business and	

Administrative studies/Management Studies,
2008-11-13

Publication

<1 %

21

Submitted to Nelson Mandela Metropolitan
University

Student Paper

<1 %

22

Submitted to University of London External
System

Student Paper

<1 %

23

Submitted to University of Greenwich

Student Paper

<1 %

24

Submitted to University of Westminster

Student Paper

<1 %

25

Pranav Shriram, Sunil Mhamane. "Android App
to Connect Farmers to Retailers and Food
Processing Industry", 2018 3rd International
Conference on Inventive Computation
Technologies (ICICT), 2018

Publication

<1 %

26

Submitted to University of Mauritius

Student Paper

<1 %

27

Submitted to Asia Pacific University College of
Technology and Innovation (UCTI)

Student Paper

<1 %

28

Submitted to Curtin University of Technology

Student Paper

<1 %

Exclude quotes On

Exclude matches Off

Exclude bibliography On

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date:15 July 2020.....

Type of Document (Tick): ☐ PhD Thesis ☐ M.Tech Dissertation/ Report ☒ B.Tech Project Report ☐ Paper

Name: _____ Dhruv Dhingra _____ Department: _____ CSE&IT _____

Enrolment No _____ 161457 _____

Contact No. _____ +91 8700146361 _____ E-mail.

_____ dhruv.dhingra11@gmail.com _____

Name of the Supervisor: _____ Dr. Aman

Sharma _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____ RETRADE - SECOND HAND
PRODUCT SELLING/BUYING APP FOR JUIT _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 40
- Total No. of Preliminary pages = 6
- Total No. of pages accommodate bibliography/references = 1

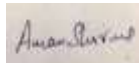


(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)



Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Abstract & Chapters Details	
Report Generated on	<ul style="list-style-type: none">All Preliminary PagesBibliography/ Images/Quotes14 Words String		Word Counts	
			Character Counts	
		Submission ID	Page counts	
			File Size	

Checked by
Name & Signature

Librarian

.....
Please send your complete Thesis/Report in (PDF) & DOC (Word File) through your Supervisor/Guide at plagcheck.juit@gmail.com