

**KABARAK UNIVERSITY**

**SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY**

**DEPARTMENTY OF COMPUTER SCIENCE AND IT**

**INTE 424 IT PROJECT II**

**STUDENT HOUSE AND ROOM MATE FINDING SYSTEM.**

**SUBMITTED BY:**

**BERNAD NJOGU INTE/MG/1908/05/18**

**SUPERVISOR:**

**MR. JOSHUA MUTAI**

29th March, 2022

# DECLARATION

I declare that this is research proposal is my original work and it has not been previously submitted to any university.

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Njogu Bernad Kamande.

## 

# RECOMMENDATION

I recommend this project for examination under my supervision as the university supervisor.

Signature Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mr.Joshua Mutai.

# DEDICATION

This research and project is dedicated to my fellow students who have been suffering on house shortage issue. Hope you will find this helpful.Also i dedicate it to my loving parents whose efforts, love and providence has seen me to this far and the future to come.They are and always will remain the source of my inspiration and handwork and thus is is only fit that this research be dedicated to them for their outstanding parenthood.

# ACKNOWLEDGEMENT

I would like to give thanks to God Almighty for blessing me with life,health and strength to be able to take this task, He is my source of strength whenever I feel like giving up.

I acknowledge the efforts of Mr.Joshua Mutai for offering to be my guide through this task, he has promised to be there ready to advice and help accordingly whenever I face any challenge concerned with this research, for that I am grateful.

Moreover, I am grateful to the Kabarak University Institution for providing me this opportunity to undertake this research together with providing the relevant materials that I may need to this process through.

I acknowledge all the students who cooperated throughout the research.

Lastly I acknowledge my parents for always being there for me and for there promise of continued support throughout this research.

# ABSTRACT

We were stuck with technology when what we really wanted was just stuff that worked. With the previous paradigm shift in technological field, there was an urgent need to embrace and appreciate the power of technology. Housing sector remained vigilant to face the challenges of change by employing a newstrategy that facilitated easy location of rental houses and even room mates for those who may have needed one. Hence there was a need to develop a rental house finding system that could simplify work for the rental owners and tenants so that all their work could be efficient and easy. This however was focused on students in higher learning institutions. A lot of students always struggled to find houses around their schools ussually when joining campus or when leaving hostels to reside outside the school for the very first time. Also, a lot of students wanted a room mate to share their rooms, ussually because they could not afford keeping up with the rent, but still couldn’t find a room mate. Some landlords on the other hand, had houses which were difficult to locate and some ended up empty for a while before any tenant, especially students, were able to locate them. The system we proposed would help students locate houses around their school and also find room mates of their choice. Also it could help landlords advertise their houses where if students were the target, they could easily locate the houses. Also, there are also students who during long holidays, they are unable to keep up with rent payment to their respective landlords. With this system, such students could also advertise their houses so that other students, especially those who are joining, lived in and pay for the house for the time they woudl be on their holiday. This though had to be permitted by the landlord or whoever was incharge of the houses.

The method of data collection we were to use for collecting user requirements was interview and the source of data for the functionalities of system was to be primary data. The development of the system was to follows the waterfall model methodology. This system’s interface was to be written and designed using XML and Java on the android framework. The system had three users: the system administrator, house manager and tenant(student). This system performed the following functionalities: the user could able to register himself, all users of the system could log in, change and reset password, tenants were to book using the system, tenants and system administrator or manager filled in tenant-in and tenant-out forms respectively, the administrator or manager could send a text message and they all viewed or updated information in the database using the system. Therefore this system coulds help students locate houses easily and landlords could advertise their houses which could be done even before the houses were completed gaining them tenants early.

Table of Contents

[DECLARATION i](#_Toc100572598)

[RECOMMENDATION ii](#_Toc100572599)

[DEDICATION iii](#_Toc100572600)

[ACKNOWLEDGEMENT iv](#_Toc100572601)

[ABSTRACT v](#_Toc100572602)

[CHAPTER 1 1](#_Toc100572603)

[1.0 Introduction 1](#_Toc100572604)

[1.1 Background of the study 1](#_Toc100572605)

[1.2 Statement of the problem 2](#_Toc100572606)

[1.3 The purpose of the study 2](#_Toc100572607)

[1.4 Research objectives 2](#_Toc100572608)

[1.5 Research questions 3](#_Toc100572609)

[1.6 Significance of the study 3](#_Toc100572610)

[1.7 Expected Outcome of the Study 3](#_Toc100572611)

[1.8 Justification of the Study 3](#_Toc100572612)

[1.9 Scope of the Study 3](#_Toc100572613)

[1.10 Limitations of The Study 3](#_Toc100572614)

[1.11 Scalability 4](#_Toc100572615)

[CHAPTER 2 5](#_Toc100572616)

[2.0 LITERATURE REVIEW 5](#_Toc100572617)

[2.1 Introduction. 5](#_Toc100572618)

[2.2 Government strategy and incentives in the housing sector 5](#_Toc100572619)

[2.3 The role of the private sector in house management 5](#_Toc100572620)

[2.4 Rental housing around kabarak 6](#_Toc100572621)

[2.5 Existing systems for finding a house 6](#_Toc100572622)

[2.6 The role of object oriented programming (oop) 7](#_Toc100572623)

[2.7 The role of relational database management system (rdbms) 7](#_Toc100572624)

[CHAPTER 3 8](#_Toc100572625)

[METHODOLOGY 8](#_Toc100572626)

[3.0 INTRODUCTION 8](#_Toc100572627)

[3.1 Software development life cycle 8](#_Toc100572628)

[3.1.1 Feasibility study 8](#_Toc100572629)

[3.1.2 Requirement analysis 8](#_Toc100572630)

[3.1.3 Design 8](#_Toc100572631)

[3.1.4 Implementation 8](#_Toc100572632)

[3.1.5 Testing 9](#_Toc100572633)

[3.1.6 Deployment 9](#_Toc100572634)

[3.1.7 Maintenance 9](#_Toc100572635)

[3.2 Waterfall Model 9](#_Toc100572636)

[3.3 Justification of the Selected Methodology 10](#_Toc100572637)

[3.4 Location of the Study 10](#_Toc100572638)

[3.7 data collection techniques 11](#_Toc100572639)

[3.7.1 Interview 11](#_Toc100572640)

[3.7.3 Observation 11](#_Toc100572641)

[CHAPTER 4. 12](#_Toc100572642)

[SYSTEM ANALYSIS AND DESIGN 12](#_Toc100572643)

[4.1 Introduction 12](#_Toc100572644)

[4.2 Existing system 13](#_Toc100572645)

[4.3 Problems of existing system 13](#_Toc100572646)

[4.4 Requirements analysis 13](#_Toc100572647)

[4.4.1 User requirements 13](#_Toc100572648)

[4.4.2 Functional Requirements 14](#_Toc100572649)

[4.4.3 Hardware Requirements 14](#_Toc100572650)

[4.4.4 Software Requirements 14](#_Toc100572651)

[4.5 DESIGN DIAGRAMS 14](#_Toc100572652)

[4.5.1 Use Case Diagram 14](#_Toc100572653)

[4.5.2 Context Diagram 15](#_Toc100572654)

[4.5.3 Data Flow Diagram 17](#_Toc100572655)

[CHAPTER 5. 18](#_Toc100572656)

[SYSTEM IMPLEMENTATION AND TESTING. 18](#_Toc100572657)

[5.1 Design phases. 18](#_Toc100572658)

[5.1.1 Conceptual Design 18](#_Toc100572659)

[5.1.2 Database Design 18](#_Toc100572660)

[5.2 SOFTWARE INTERFACE 20](#_Toc100572661)

[LOGIN ACTIVITY 22](#_Toc100572662)

[MAIN ACTIVITY 23](#_Toc100572663)

[24](#_Toc100572664)

[25](#_Toc100572665)

[26](#_Toc100572666)

[ROOM MATE FRAGMENT 33](#_Toc100572667)

[5.4 Implementation Strategies 34](#_Toc100572668)

[5.5 Testing strategies 34](#_Toc100572669)

[5.6 Test plan 34](#_Toc100572670)

[5.6.1 System Testing 34](#_Toc100572671)

[5.6.2 Acceptance testing 35](#_Toc100572672)

[5.7 Justification of testing and implementation strategy 35](#_Toc100572673)

[CHAPTER SIX 36](#_Toc100572674)

[CONCLUSION AND RECOMMENDATIONS 36](#_Toc100572675)

[6.1 Introduction 36](#_Toc100572676)

[6.2 Recommendations 36](#_Toc100572677)

[6.3 Conclusion 36](#_Toc100572678)

[APPENDICES 37](#_Toc100572679)

[APPENDIX A: REFERENCES 37](#_Toc100572680)

[APPENDIX B: SAMPLES OF CODE 38](#_Toc100572681)

[Samples of code used in both frontend and backend 38](#_Toc100572682)

[Manifest.xml 38](#_Toc100572683)

[activity\_main.xml 39](#_Toc100572684)

[Main Activity 40](#_Toc100572685)

[item\_house 45](#_Toc100572686)

[fragment\_house 47](#_Toc100572687)

[AdapterHouse 48](#_Toc100572688)

[House Fragment 54](#_Toc100572689)

[APPENDIX C: WORK PLAN 61](#_Toc100572690)

[APPENDIX D: RESEARCH BUDGET 62](#_Toc100572691)

Table of Figures

[Figure 1: water fall model 10](#Figure!0|sequence)

[Figure 2: use case diagram 16](#Figure!1|sequence)

[Figure 3: context diagram 17](#Figure!2|sequence)

[Figure 4: data flow diagram(level 1) 18](#Figure!3|sequence)

[Figure 5: firebase realtime database 20](#Figure!4|sequence)

[Figure 6: launch activity 21](#Figure!5|sequence)

[Figure 7: register activity 22](#Figure!6|sequence)

[Figure 8: login activity 23](#Figure!7|sequence)

[Figure 9: next activity 24](#Figure!8|sequence)

[Figure 10: dashboard activity 25](#Figure!9|sequence)

[Figure 11: all listing fragment 26](#Figure!10|sequence)

[Figure 12: wishlist fragment 27](#Figure!11|sequence)

[Figure 13: chats fragment 28](#Figure!12|sequence)

[Figure 14: chats activity 29](#Figure!13|sequence)

[Figure 15: profile fragment 30](#Figure!14|sequence)

[Figure 16: my listings activity 31](#Figure!15|sequence)

[Figure 17: house details activity 32](#Figure!16|sequence)

[Figure 18: add room mate request activity 33](#Figure!18|sequence)

[Figure 19: add house activity 33](#Figure!17|sequence)

[Figure 20: room mate fragment 34](#Figure!19|sequence)

[Figure 21 62](#Figure!20|sequence)

Index of Tables

Table 1: research budget 63

# CHAPTER 1

## 1.0 Introduction

This chapter presents the background of the study,the objectives of the study, the research questions, the scope, the limitations of the study and the justification of the study.

## 1.1 Background of the study

Housing is one of the basic needs of life. It has great economic, social, cultural and personal importance to the quality of life. Right to adequate living especially adequate housing is recognized by the International Human rights law(Nations, 2014). Though a country‟s national prosperity is usually measured in economic terms, increasing wealth is of diminished value unless all can share its benefits and if the growing wealth is not used to redress growing social deficiencies, one of which is housing

(Erguden, 2001). Housing plays a huge role in revitalizing economic growth in any country, with shelter being among key indicators of development. The universal declaration of human rights gives one of the basic human rights as the right to a decent standard of living, central to which is the access to adequate housing (United Nations, The Human Rights-article 25, 1948). Housing as a basic human right demands that urban dwellers should have access to a decent housing, defined as one that

provides a foundation for rather than being a barrier to good physical and mental health, personal development and fulfillment of life objectives (Seedhouse, 1986).Around 546.7 thousand students were enrolled in universities in Kenya during the academic year 2020/21. The number increased from roughly 509.5 thousand in 2019/20. According to the source,the growth by 7.3 percent was related to an expansion in the number of government sponsored students. All over the country, a significant number of students live in rental housing especially private rental housing. For instance in Kabarak university, private rental housing held over 10 percent of the total school population. Most student who may be cant afford the prices offered to live within the school live in private residences around. Therefore, rental houses are in high demand and are tend to be built. This is advantageous to the landlords since they increase their profits through the rents paid by the students. As the number of tenants tend to increase, the rental houses tend to be difficult to find since most of them are occupied and it is evident that most land owners around cant keep up as this sector requires a lot of money to venture in to. Some landlords have properties which are located in areas which may be very hard to find especially for first years who are new to the area. This has resulted to students living very far from the school where they pay extra transport expenses to travel to and fro school. Thus, the focus of this project is to help

students get houses-to-let via technology rather than physically searching for them around the school and landlords to advertise their empty rental houses so that they can get connected with each other. It also helps those who are at risk of being thrown out of their houses to find themselves a room mate and raise up the rent together.

## 1.2 Statement of the problem

The ‘lack-of-houses-problem’ tend to grow time after time since more student increase. Since most students are new to the place , it is hard for them to locate houses around, especially in schools located far from towns where its relatively easier to locate a house. Lack of a computerized and more sophisticated way for the landlords to advertise the information about their houses and students to view and locate this houses from anywhere even before joining the school. Lack of a computerized system where a student can search for a room mate within his/her school. The current means of searching for a house is physically moving around the area to locate houses. Also, to get a room mate is hard as there is no way to know someone needs a room mate apart from asking them personally. With the fact that you have to move around the school searching for a house, new students are not favored as they still don’t know the area very well. Also, to get a room mate, it can be very hard to get one since some people may not be willing to ask. Also the landlords can go for a very long time without tenants simply because the targeted tenants, in this case students, may not be aware of the houses.

## 1.3 The purpose of the study

This study is aimed at conducting a research on the availability of houses around higher learning institutions, how students are able to access them, how owners can help students access them, challenges of how students aren’t able to find room mates and proposes a system that will help minimize all this problems and provide a constant supply of available houses around specific schools within the country and the world.

## 1.4 Research objectives

1. To gather the requirements of RooMateFDR app.
2. To design the RooMateFDR app.
3. To implement the RooMateFDR app..
4. To test and verify the RooMateFDR app.

## 1.5 Research questions

1. How will the system requirements be gathered?
2. How will the RooMateFDR app be designed?
3. How will the RooMateFDR app be implemented?
4. How will the RooMateFDR app be verified?

## 1.6 Significance of the study

This study will analyze the shortage of houses within higher learning education in Africa with a specific focus on Kenya as a developing country and a key area for the shortage in both private and public universities. This study after the analysis will propose a mobile and web based solution for the house shortage problem for students which we believe is also affecting other countries too since there are more people joining higher learning institutions today than ever.

## 1.7 Expected Outcome of the Study

At the end of this study, it is expected that the problem of house shortages around higher learning institutions will have been thorougly analyzed and if proven to be a big problem as speculated, a mobile and a web based solution for the problem will be proposed and its methodology and fixes be explained.

## 1.8 Justification of the Study

Shortage of houses around higher learning institutions is making life very hard for students especially the ones joining since they may not even have prior knowledge of the area making them live very far from school which just adds more expenses to them in most cases where they have to pay for transport to and from school. For this and more reasons, the study is aimed at proposing a solution that could reduce if not completely do with this problem before it worsens since more and more students are joining.

## 1.9 Scope of the Study

The study will explore the currently available systems dealing with this problem, check their impact and decide whether they are efficient or not. The study will also explore the various methods that house owners use to market or advertise their houses. It will also study strategies used by students to acquire room mates especially in cases where the room mate is not someone you have ever met before. At the end we will propose a possible solution to the problem in kenya.

## 1.10 Limitations of The Study

Even though the study is expected to solve the problem of house shortages around higher learning institutions, it faces a few challenges for its completion and implementation. They include the following. First, many house owners may be old and stuck to the ways of the past i.e. the days before technology took over the world. This old people may find it very hard to shift from keeping book records and also advertise their houses in a platform without prior experience with onother of its kind.

The study also involves alot of travelling since the institutions are scattered everywhere. Especially as it is a system to solve this problem in Africa, we require to travel to other countries all over Africa to collect data necessary for the system. The study is also limited to extensive experience in programming to develop the system since it entails alot of features which may require security which have become a necessity today especially if we include money transfer. One also requires people skills which is necessary for our data collection.

## 1.11 Scalability

Even though the system will be specifically designed to be able to help students to find houses and room mates online and help house owners to advertise their houses, it can be with a great simplicity be scaled to allow payment for houses by tenants within the system and managing the houses for owners instead of using book records. It can also be made in search a way that anybody can use, that is, not students alone, in this country and the whole continent of Africa.

# CHAPTER 2

## 2.0 LITERATURE REVIEW

Literature review is a text written by a researcher to consider an important point of current knowledge including the theory and contribution of the methodology and the critical findings of a particular topic. The primary goal of textual review is to place current research within the literature context and to provide context for the reader (Cooper, 1998).

## 2.1 Introduction.

The rental housing came into play due to the seriousness of housing problem. Many students end up lacking houses on the very first day of joining campus which is due to several factors mainly due to the fact that they are new to the area and may not know where to find houses since the houses which are easily located are already taken or booked. Also the fact that they have not made any friends yet may lead to this problem since it is very hard to find a room mate. This leads to this students finding houses very far from school where they end up paying more expenses on daily transport. The existing techniques of finding houses around Kabarak University is very tiresome since you have to walk all around the place since there are no systems in the school dedicated to this task.

## 2.2 Government strategy and incentives in the housing sector

Some of the dominant strategies for housing and service provision for the Kenya‟s urban poor include slum upgrading and site and service schemes. However, the efficiency of these strategies has been limited by ambivalent government attitude to irregular settlement. These strategies have failed because of a reliance on inappropriate building by-laws and infrastructural standards and modern designs, construction technology and conventional building materials that all make housing unaffordable to the poor, even after subsidies. Thus, government initiatives in assisting house owners in management have proven to be pathetically slow with many of the houses provided being economically and socially irrelevant, this further prompting the rise of informal settlement (Macoloo, 1994).

## 2.3 The role of the private sector in house management

Private sector housing management is defined as any process which is not connected at all with the actions of the state neither directly constructed by state nor financially sponsored by the state where production is not expected to have a social element (Golland, 1996). (Ambrose and Barlow, 1987) have argued that three factors are important in influencing the level of new house building. These are direct capital investment by the state for public housing, state support for production and consumption and

changes in the profitability of house builders in the private sector. The private sector can play an important role in housing provision provided that the state offers sufficient and appropriate incentives to the sector (Mitullar, 2003). The clear motivation that underlies the private sector is profit (or potential profitability) with profit maximizing options being in the context of housing, producing and selling more of the product; reducing the cost of production through lower raw material and wage costs and finally increasing the price of the product or service (Hancock, 1998). Profitability in housing is advocated to be based on three variables; House prices, land prices and building costs, where: Profit=House prices-{Land prices + Building costs} (Golland, 1996).

## 2.4 Rental housing around kabarak

Rental housing around kabarak is mainly confined in two major centers namely Rafiki and Mang’u. Majority of these houses are owned by land owners around the place. This brings up a major problem since some of them have been built in places which are very hard to locate. Around Kabarak the building of the houses have increased due to the growing demand since more and more students are joining every year. But still, since the land is owned by the locals, some of them may not have the neccessary capital required to build houses. This leaves land which sometimes may be very close to the school unbuilt and dedicated to farming just due to lack of funds. For the land owners with requires finances, they may end up being in a location where despite being very close to the school, are not easy to locate which is a very great and stressing problem to students. Also it is a problem to landlords since their houses may go unoccupied for a while which is only due to their location since there is no way around this region that is popular and dedicated to advertising unoccupied houses around. The houses which are very close to the school are also very expensive and thus some students who get in those houses may not be able to afford them in the long run. This brings the need for a room mate where the two or more can share expenses.

## 2.5 Existing systems for finding a house

Currently, the most used system for finding a house is AirBNB. The system was founded in 2008 in San Fransisco, Carifornia, United States of America. This system however is not widely used by house owners targeting students but mainly by those targeting tourists especially from the west or european countries. This makes sense since AirBNB is widely used in western countries and by targeting those who are from the west one can easily get customers. Also the majority of the people who use this platform are not seeking a place to stay for long term. Most of them are just visiting a place for a period of time and move elsewhere or back from their origin later. The property owners here are also seeking for a short term tenant. With this, this platform can not fulfill the long term tenant requirement as this can be very expensive if charges are as per a day of stay. This system is also not widely used locally especially by most students since not most of them travel often. Therefore, the house finding system tends to minimize the problem to both the house owners and students. With the system, the house owners can advertise their properties and the student can book them online. The two can later communicate online through messaging and can make agreements or even give directions if the house is hard to find. Also, those seeking for a room mate can post and potential partners can find them there.

## 2.6 The role of object oriented programming (oop)

The concept of object-oriented modeling is becoming increasingly practical because of its ability to thoroughly represent complex relationships as well as to represent data and data processes in a consistent manner. This concept has been implemented in computer software engineering; ranging from

system analysis, system design, operating system, computer programming and database management system (Cohn, 1996).

## 2.7 The role of relational database management system (rdbms)

(Levin, 1999) Database Management System (DBMS) has replaced the file system data management by having a pool of data that can be shared by multiple application programs and users concurrently. DBMS also provide logical and physical data independence, so that changing of data structure or application program will not affect one another.

# CHAPTER 3

# METHODOLOGY

## 3.0 INTRODUCTION

In this chapter, the system methodology will be explained. It will focus on how to collect data and how they will be analyzed and implemented.

## 3.1 Software development life cycle

Software development methodology is a technique that is used to show how the proposed system will be developed. In this case, the methodology used will be a waterfall model.

### 3.1.1 Feasibility study

This stage will help in gaining understanding about the landlords and students current problems with the existing system via the interviews with the students and landlords. Then the obtained information will determine the importance of the system being proposed in terms of technical, economic and social feasibility.

### 3.1.2 Requirement analysis

This stage involves gathering of information about the customer’s needs via interview and define the problems that the system tends to solve. Both software and hardware requirements will be analyzed. Also from the questionnaires, we will be able to know what a room mate finder requires in the system.

### 3.1.3 Design

This stage will include the overall design of the system, that is, the User Interface and the database design. This stage will help in identify faults from the previous phase (missing information) and its output, which is the design specification, will help in the next stage of implementation

### 3.1.4 Implementation

This is the stage that coding is done as per the design specification(s). The user interface will be implemented with XML for android; Java programming will be used to establish communication between the interface and the database. The database design will be implemented using firebase realtime database. All these will result to the product components built according to a pre-defined coding standard and debugged, tested and integrated to satisfy the architecture requirement of the system.

### 3.1.5 Testing

This is the stage that will test the system to ensure that they are fault free and are made per the user requirements. This will involve both alpha testing that will involve testing at the development site, unit testing of individual code module and system testing of the integrated product. The bugs found will be corrected before moving to the next stage. Preparation, reviewing and publishing of the documentation will be done at this stage.

### 3.1.6 Deployment

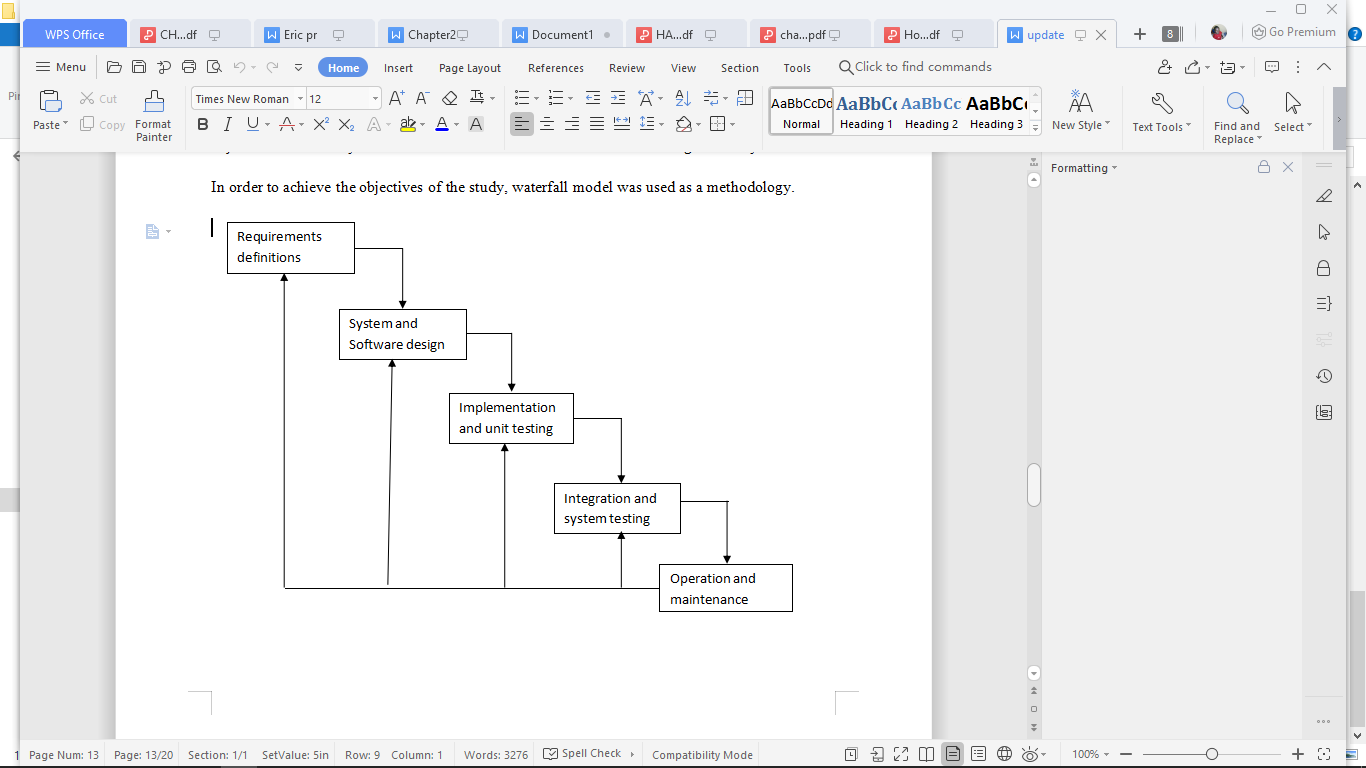
This stage is done once the product has been tested and certified as fit for use. The system is prepared for large scale use.

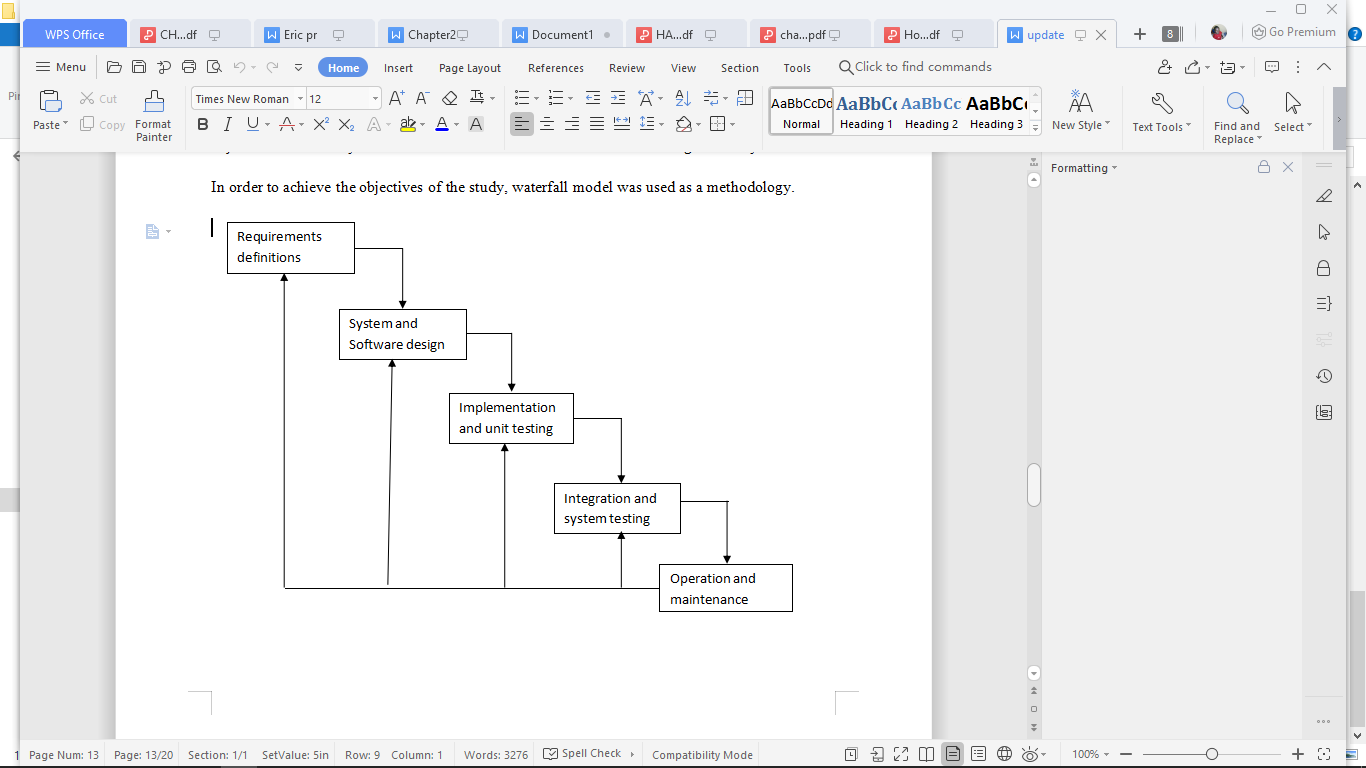
### 3.1.7 Maintenance

This stage occurs after installation. It involves modifications on the system to improve performance per the user requirements or change of technology. Sometimes it is required due to the appearance of bugs that were not initially encountered. These modifications are recorded for documentation and system update.

## 3.2 Waterfall Model

It is comprised of the stages that the developer will use when developing the system. It is a sequential model hence, the name waterfall. One has to finish one stage before going to the next stage therefore one cannot go back to a previous stage once it is done. It comprises of the feasibility study, analysis phase, design phase, coding phase, testing phase, implementation phase and the maintenance phase. It is a simple model and easy to use and understand. The output of one stage is crucial for starting the next stage. Once a stage is approved one is free to move to the next phase.

Figure 1: water fall model

Figure 1: water fall model

## 3.3 Justification of the Selected Methodology

The methodology is simple and easy to use.

The methology is easy to use due to rigidity of model since each phase has specific deliverables and a review model.

Phases are processed and completed one at a time which guarantes completion of the project.

It works well for a small project where requirements are very well understood.

The methodology has well cleary defined goals.

With this methodology it is easy to manage tasks.

With this methodology process and results are well documented making it very easy to follow.

## 3.4 Location of the Study

This research will take place in Kabarak university main campus in Nakuru county kenya.

**3.5** Research Design

The strategies and procedures used to gather and acquire the necessary information needed for the research study are examined in the research design. It specifies the information needed, as well as the techniques for gathering and analyzing data, and how all of this will help to answer the research questions (Creswell, 2013).

Lewis (2015) claims that study design is crucial in reflecting the purpose of inquiry, which in my case is explanatory. The basic goal of explanatory research is to determine the limits of the environment in which the issues, opportunities, or circumstances of interest are likely to exist, as well as the important components or variables that may be located there and be relevant to the research.

3.6 Target Population

This is the total group of people or things that researchers want to apply their findings to. The theoretical population which is also referred to as the target population, has a very wide range of features.

It is the group of people studied by the researcher and whose findings are used to generalize to the rest of the population. In our case we will targeting students from Kabarak University and landlords around the school area together with their properties.

## 3.7 data collection techniques

This shows how data will be collected from the users of the system. The data collection techniques to be used include:

### 3.7.1 Interview

This method will be used to collect information from landlords and students via conversation so as to know how the manual way of marketing a house and finding a house and works from each respectively, its flaws and if and how the digital system might help them.

### 3.7.3 Observation

Observation, as the name implies, is a way of collecting data through observing. Observation data collection method is classified as a participatory study, because the researcher has to immerse himself in the setting where his respondents are, while taking notes and/or recording. Observation will be used to correct data and will be specifically be done during joining of new students come next semester.

surveys, and more, content analysis can be quite useful. This can assist in determining the most critical areas for improvement.

# 

# CHAPTER 4.

# SYSTEM ANALYSIS AND DESIGN

## 4.1 Introduction

The system objectives outlined during the feasibility study served as the basis from which the work of system design was initiated. Much of the activities involved at this stage were of technical nature requiring a certain degree of experience in designing systems sound knowledge of computer related

technology and through understanding of computers available in the market and the various facilities provided by the vendors. Nevertheless, a system could not be designed in isolation without the active involvement of the user. The user had a vital role to play at this stage too. Data collected during feasibility study was utilized systematically during the system design. Designing a system is a

creative process which calls for logical as well as lateral thinking Logical approach involves systematic moves towards the end product keeping in mind the capabilities of the personnel and the equipment at each design making step.

## 4.2 Existing system

currently the most property owners do not have a system dedicated for advertising their properties. Students therefore have no place to check for empty houses around the school and most of them have to do pre-visit of the area and get a house before joining. Once students finds a vacant house, they can call the manager of the houses indicating the house they would like rented to them. The property manager can text or call them back giving them all the details about the house they are requesting. The details include;Rent per month, Deposit paid, Terms and conditions to follow acceptance.

## 4.3 Problems of existing system

With the current system locating a house is completely manual and entails a lot of work. One has to walk around the school trying to locate one and it gets worse if the individual has never visited the area. Our system on the other hand provides a platform where the owner can post empty houses and the student can easily book the house. This way even if you pre-visit the area you know exactly where you are going and if you are not satisfied you can get another house almost immediately without having to walk around. The system eliminates most of the limitations of the existing system.

## 4.4 Requirements analysis

Requirement analysis involved defining customer needs and objectives in the context of planned customer use, environments and identified system characteristics to determine requirements for system functions.

### 4.4.1 User requirements

It entailed user involvement and statements of facts and assumptions that define the expectations of the system in terms of mission objectives, environment, constraints and measures of effectiveness and suitability. Basically the users:

1. A system that improves on the efficiency of information storage and retrieval and communication.
2. A system that is easy to learn and use.
3. A system that is fast in processing transactions.
4. A system that is flexible, safe and convenient.

### 4.4.2 Functional Requirements

This is a necessary task, action or activity that was accomplished. The proposed system is able to:

1. Allow property manager to add a houses and their details.
2. Allow the property manager to delete houses and their details.
3. Allow the student to search data in the database.
4. Allow students to advertise for room mate position.
5. Allow communications in messages form between student and manager.
6. Allow student to select multiple houses and make decision later.

### 4.4.3 Hardware Requirements

1. Processor 2.8 Ghz processor speed
2. Memory 8GB RAM
3. Visual Display Unit 800\*600 colors

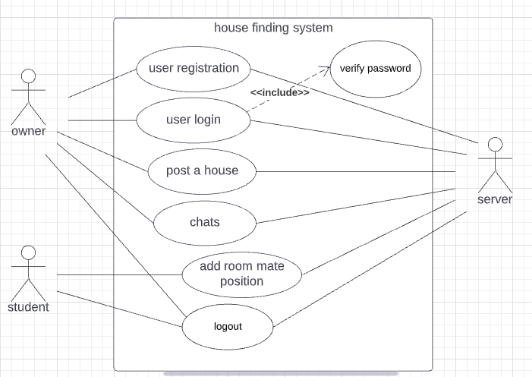
### 4.4.4 Software Requirements

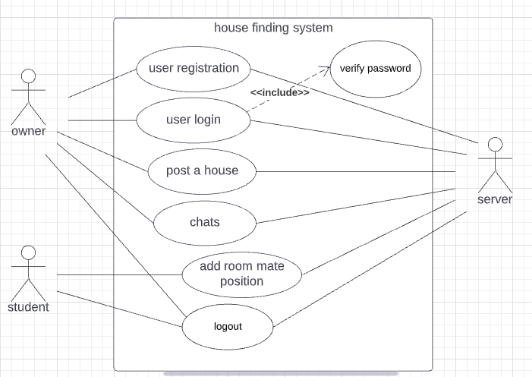
1. Operating System- Ubuntu
2. Microsoft Office Power point- Used during presentation
3. Android studio

## 4.5 DESIGN DIAGRAMS

### 4.5.1 Use Case Diagram

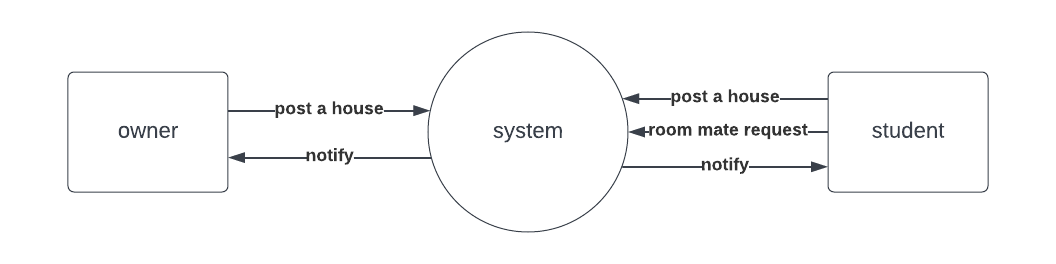
Shows the use case of the system.

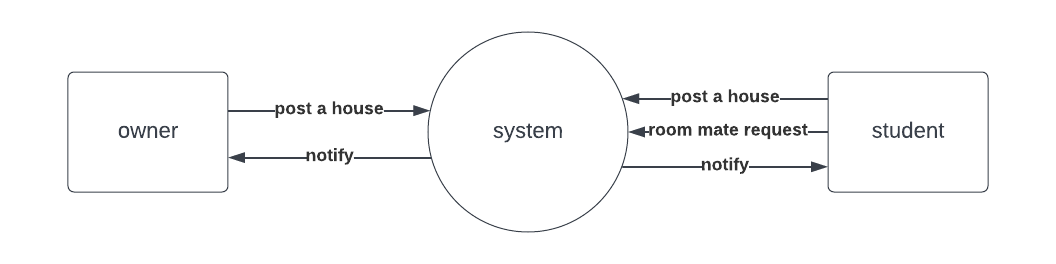
Figure 2: use case diagram

Figure 2: use case diagram

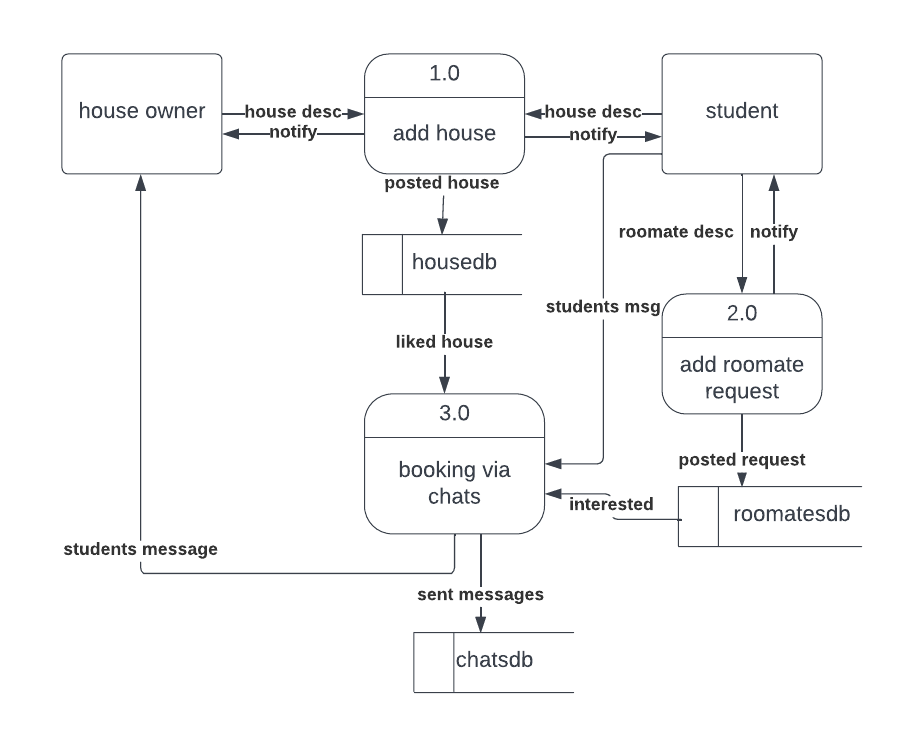
### 4.5.2 Context Diagram

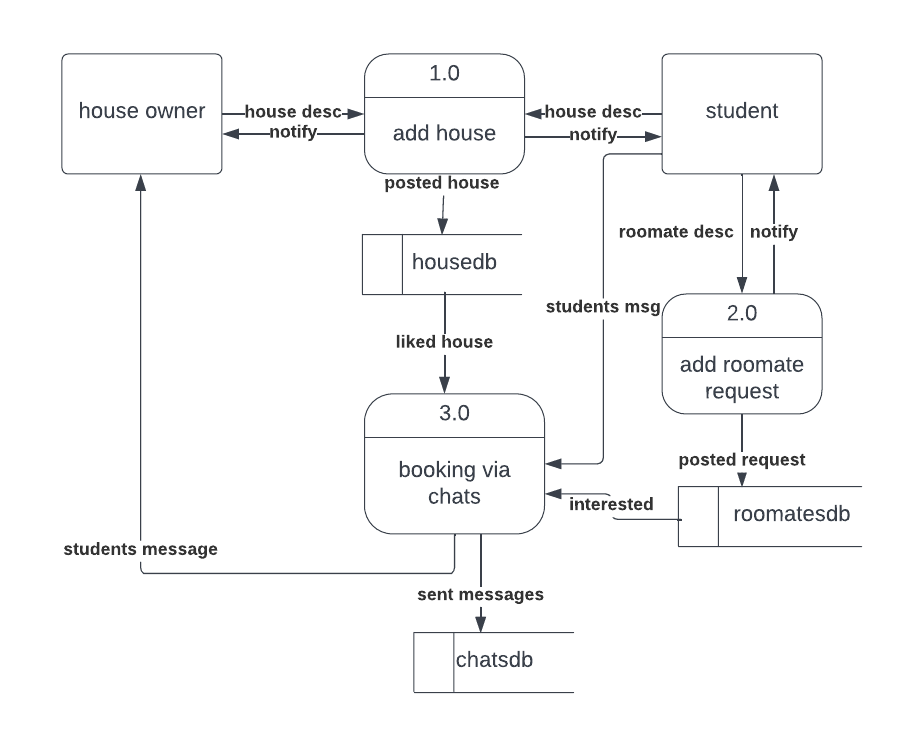
Depicts the users and app interactions. It also shows the admin system interaction.

Figure 3: context diagram

Figure 3: context diagram

### 4.5.3 Data Flow Diagram

Figure 4: data flow diagram(level 1)

Figure 4: data flow diagram(level 1)

# CHAPTER 5.

# SYSTEM IMPLEMENTATION AND TESTING.

## 5.1 Design phases.

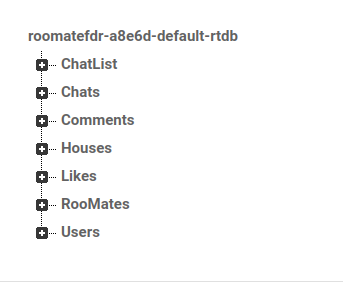
The users requirements document was analyzed for better understanding of what was required of the system. Ways of implementing these requirements were analyzed. Physical modules of the system were designed and identifying of the operating environment in which they were to work on. The system was an android application. The database was updated each time the administrator or the user; add, deletes or deletes data on the system. Its only the administrator who has access to the system to view or make changes when necessary. The system was designed to allow the administrator to view, edit, delete and add data to the database. Each time a customer comes, he/she is registered in the tenant registration table of the database with other relevant details about the tenant. System design involved transforming the software requirements into an architecture that described its top-level structure and identified the software components and developed a detailed design for each software components. For each requirement, a set of one or more design elements was produced.

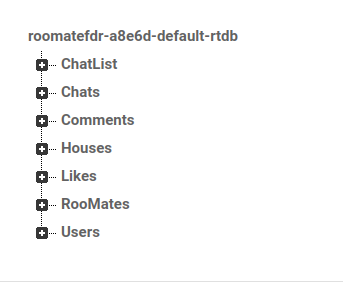
### 5.1.1 Conceptual Design

Conceptual design was the very first phase of design in which drawings or solid models were the dominant tools and products. The conceptual design phase provided a description of the proposed system in terms of a set of integrated ideas and concepts about what it was to do, behave and look like, that was understandable by the users in the manner intended.

### 5.1.2 Database Design

The general theme behind a database is to handle information as an integrated whole. A database is a collection of interrelated data stored with minimum redundancy to serve the users quickly and effectively. After designing input and output, the analyst must concentrate on database design or how data should be organized around user requirement. The general objective is to make information access, easy, quick and flexible for other users. In our case we used a noSQL database called firebase real time database in which no tables are required.

Figure 5: firebase realtime database

Figure 5: firebase realtime database

## 5.2 SOFTWARE INTERFACE

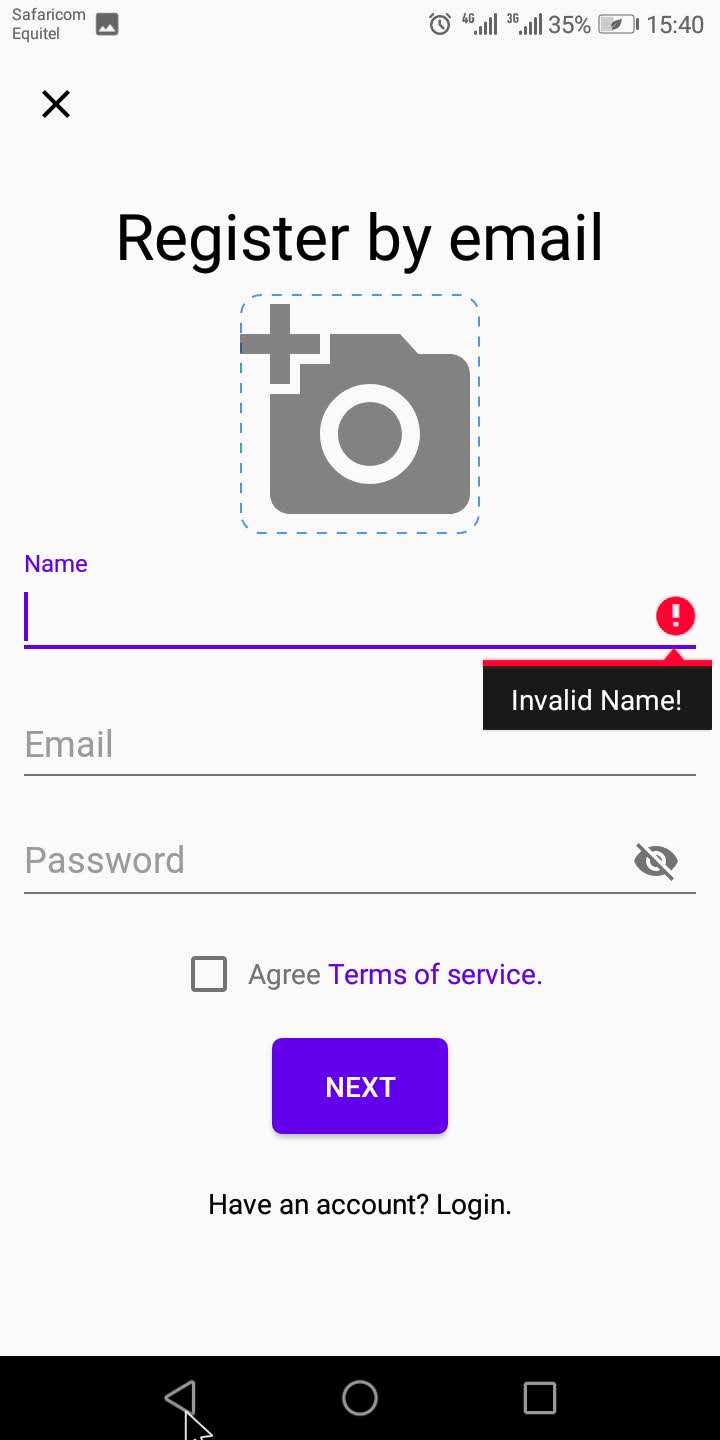
LAUNCH ACTIVITY

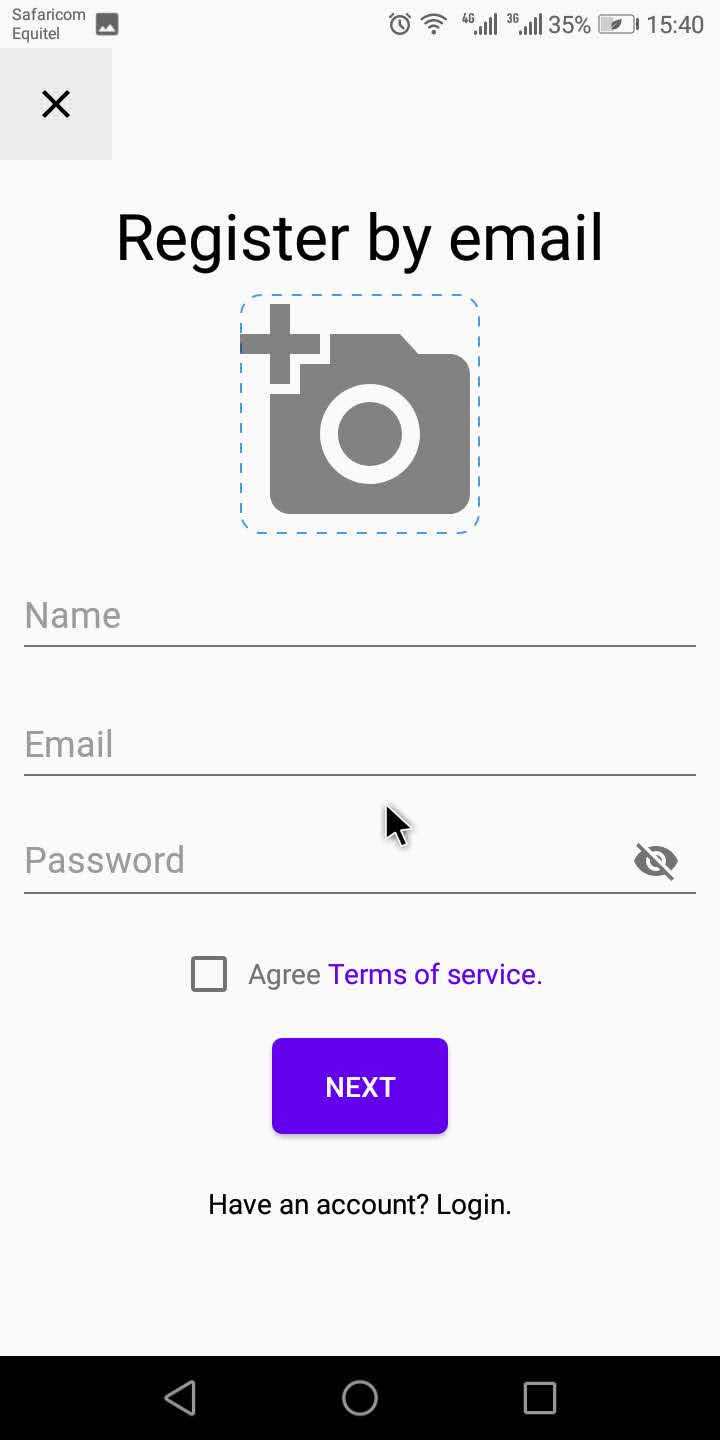
Figure 6: launch activity

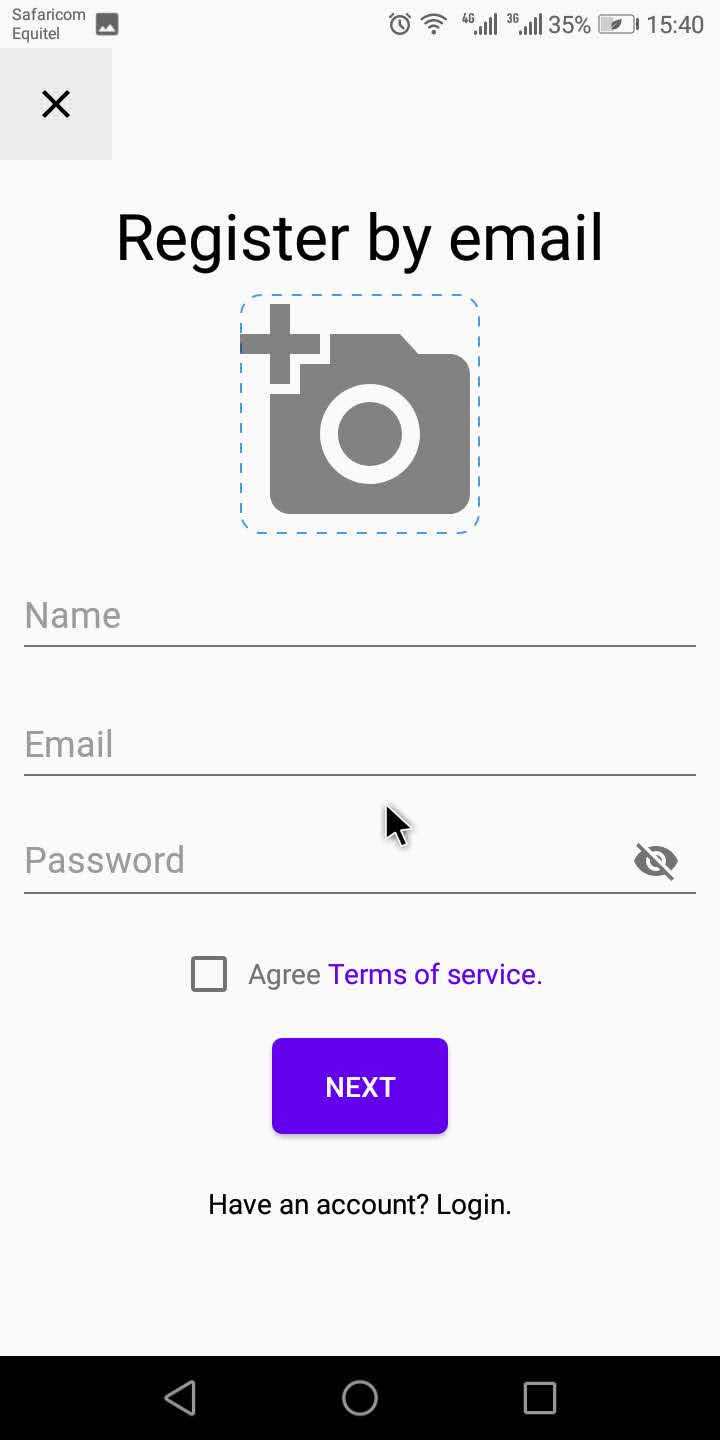
Figure 6: launch activity

The launch activity is the very first activity that you see when you open the app. This is where you choose whether to register(if he doesn’t have an account) or login if he has an account.

REGISTER ACTIVITY

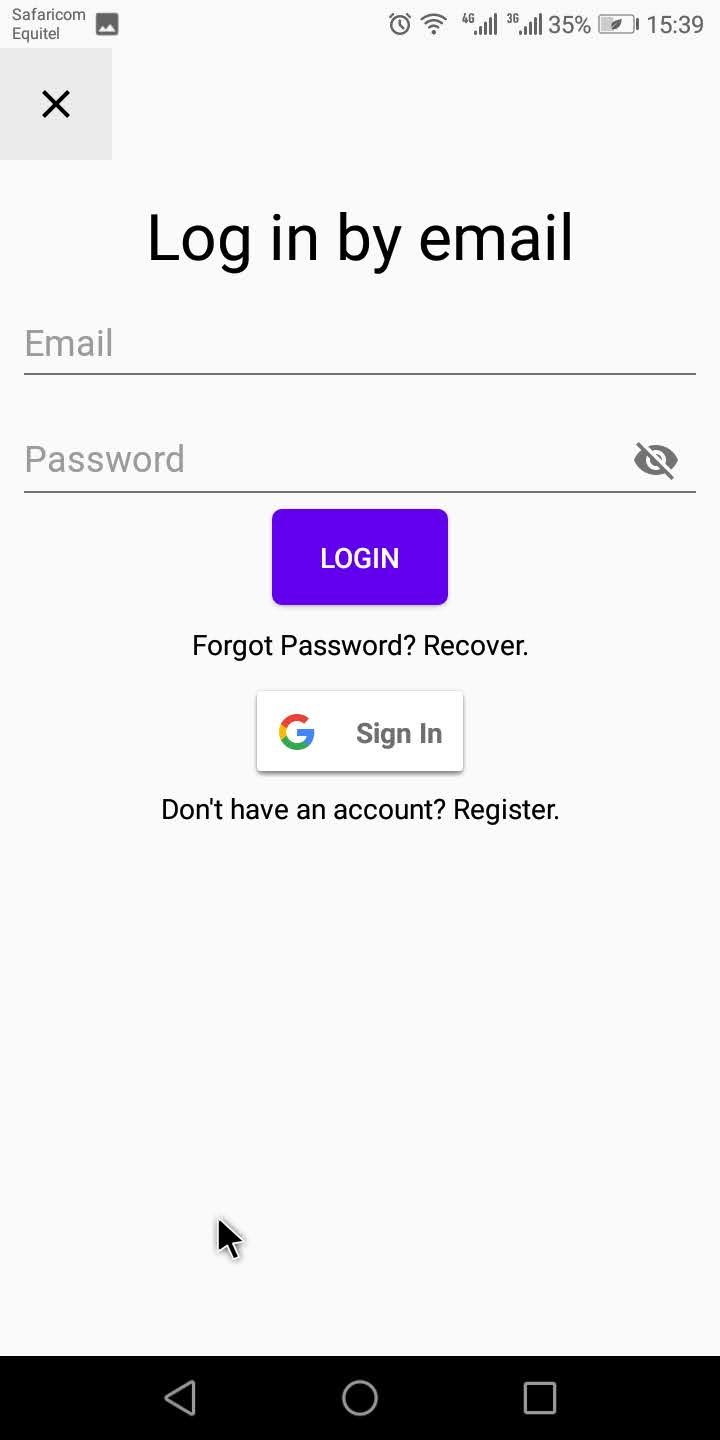


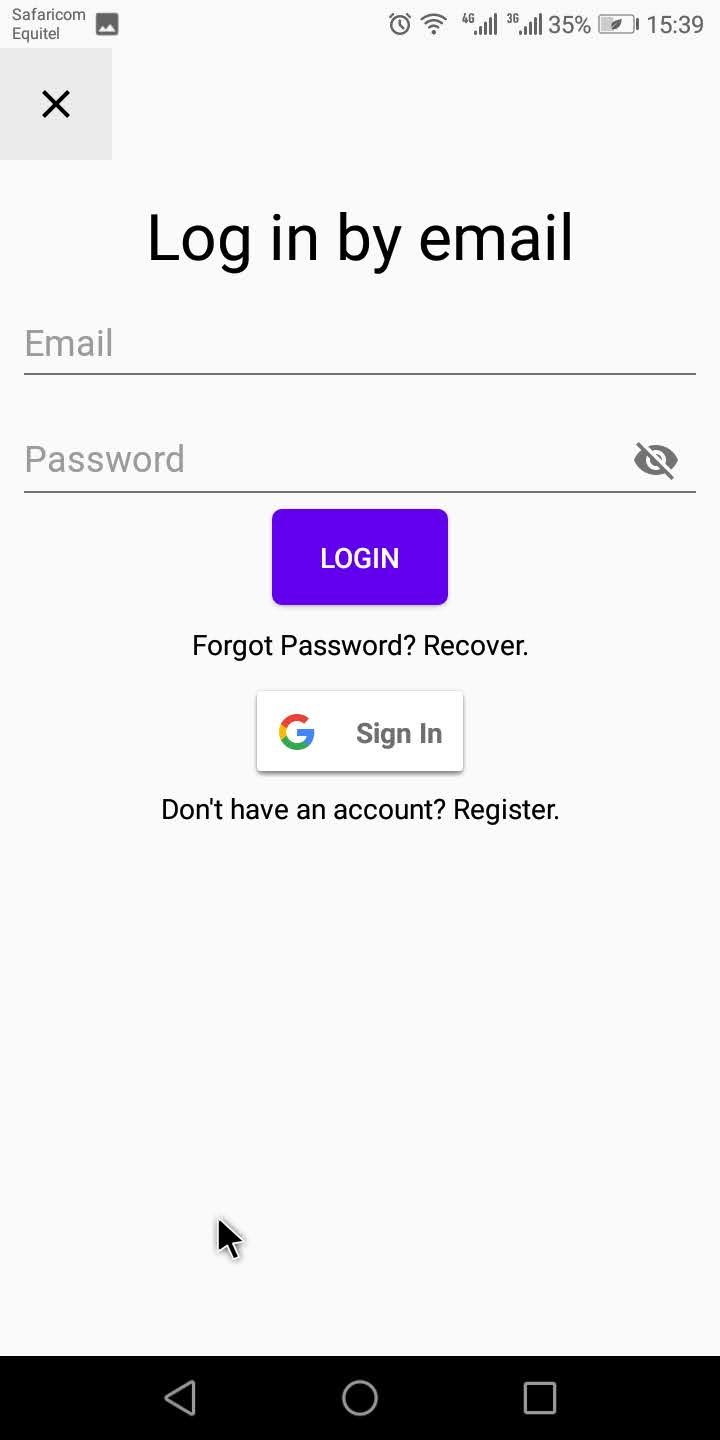
Figure 7: register activity

Figure 7: register activity

The two photos above show the register activity. The image on the right shows validation in action if you try to use invalid details.

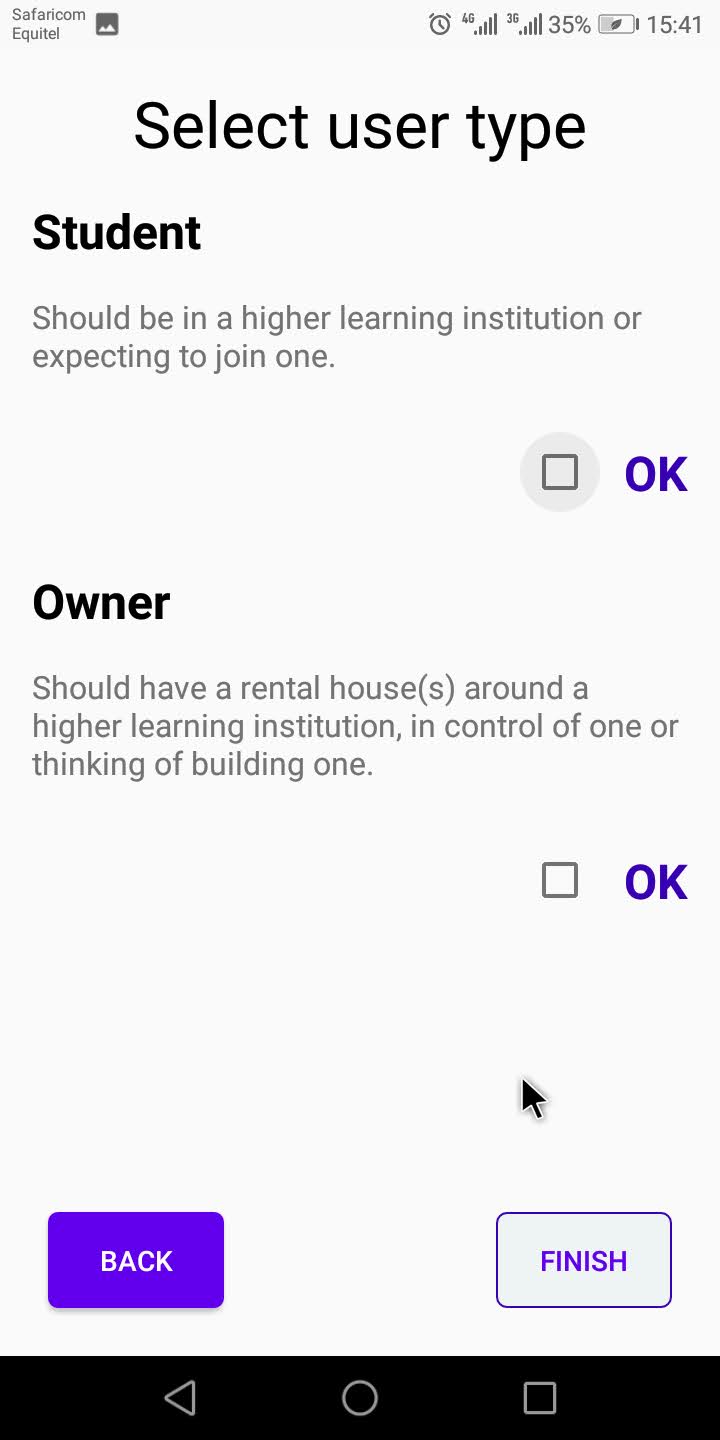
## LOGIN ACTIVITY

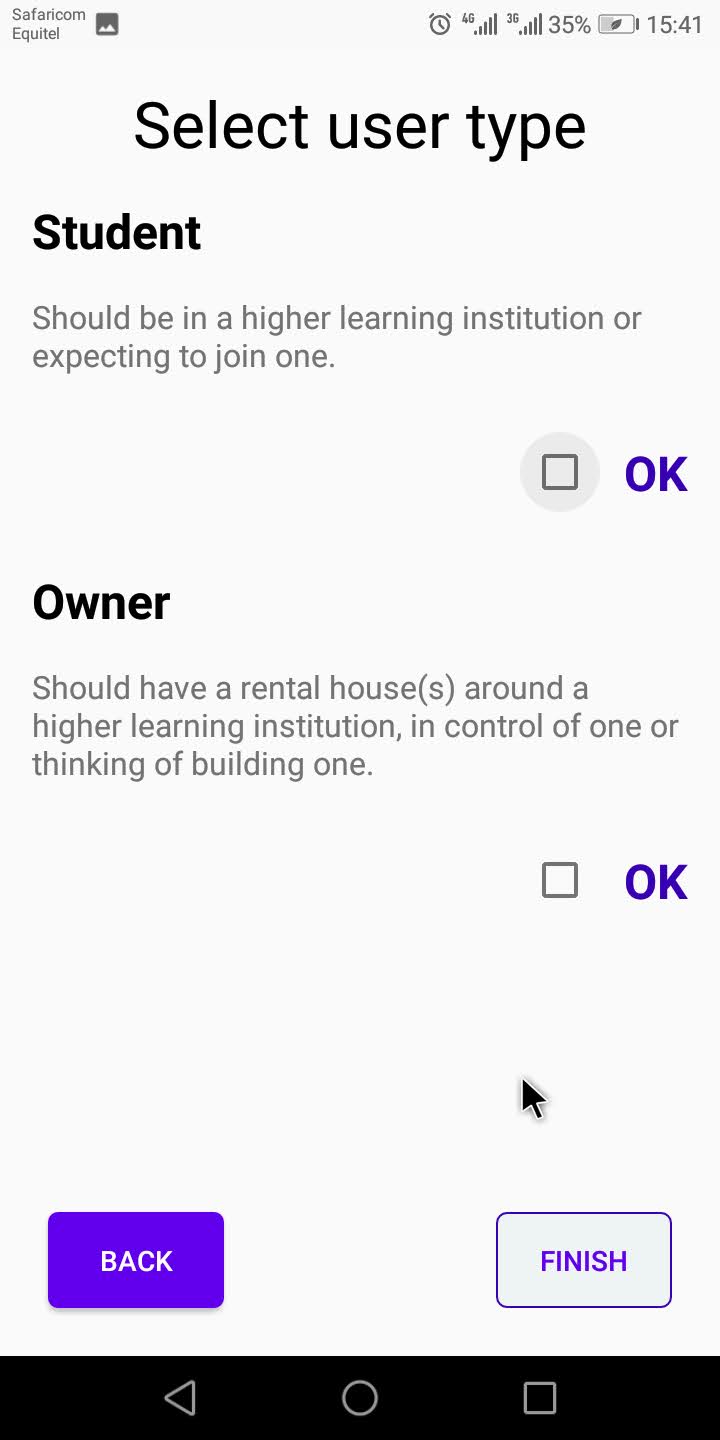
Figure 8: login activity

Figure 8: login activity

The above images show the log in page. In this case you need an email and your password to log in to your account. Invalid data keeps you out as demonstrated by the image on the right.

NEXT ACTIVITY

Figure 9: next activity

Figure 9: next activity

In this activity the user selects his/her user type. This indicates whether the user is a student or a house owner.

## MAIN ACTIVITY

## 

## 

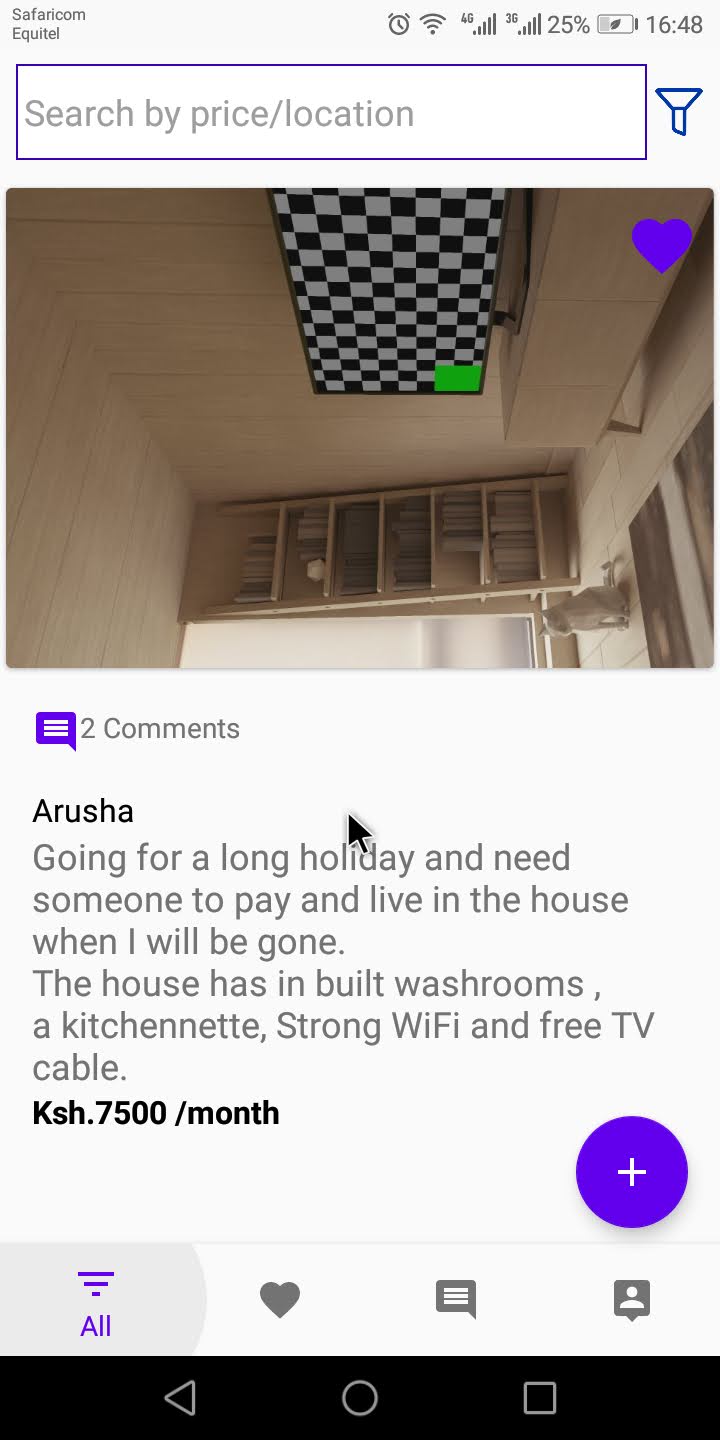
Figure 10: dashboard activity

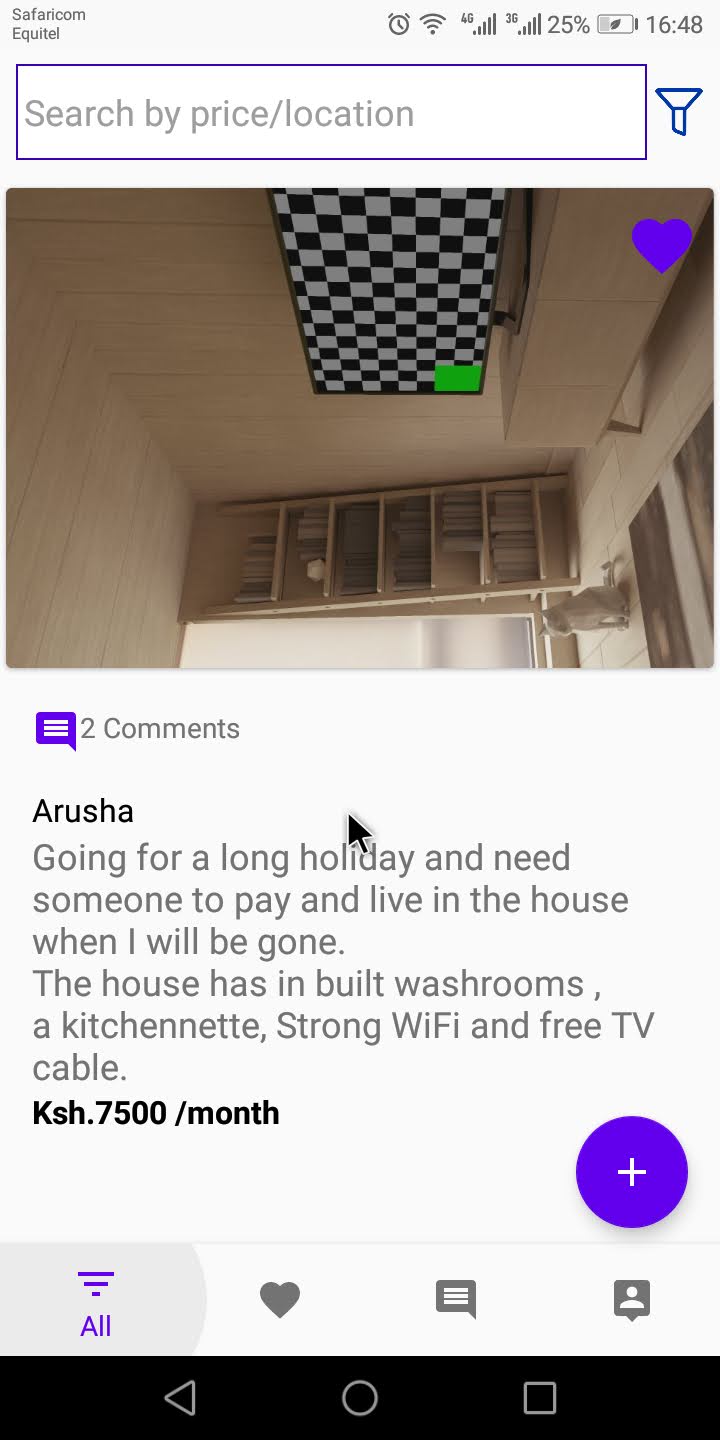
Figure 10: dashboard activity

This is where you land after registration or login. Its the dashboard containing all the functionalities of the application.

ALL LISTINGS FRAGMENT

## 

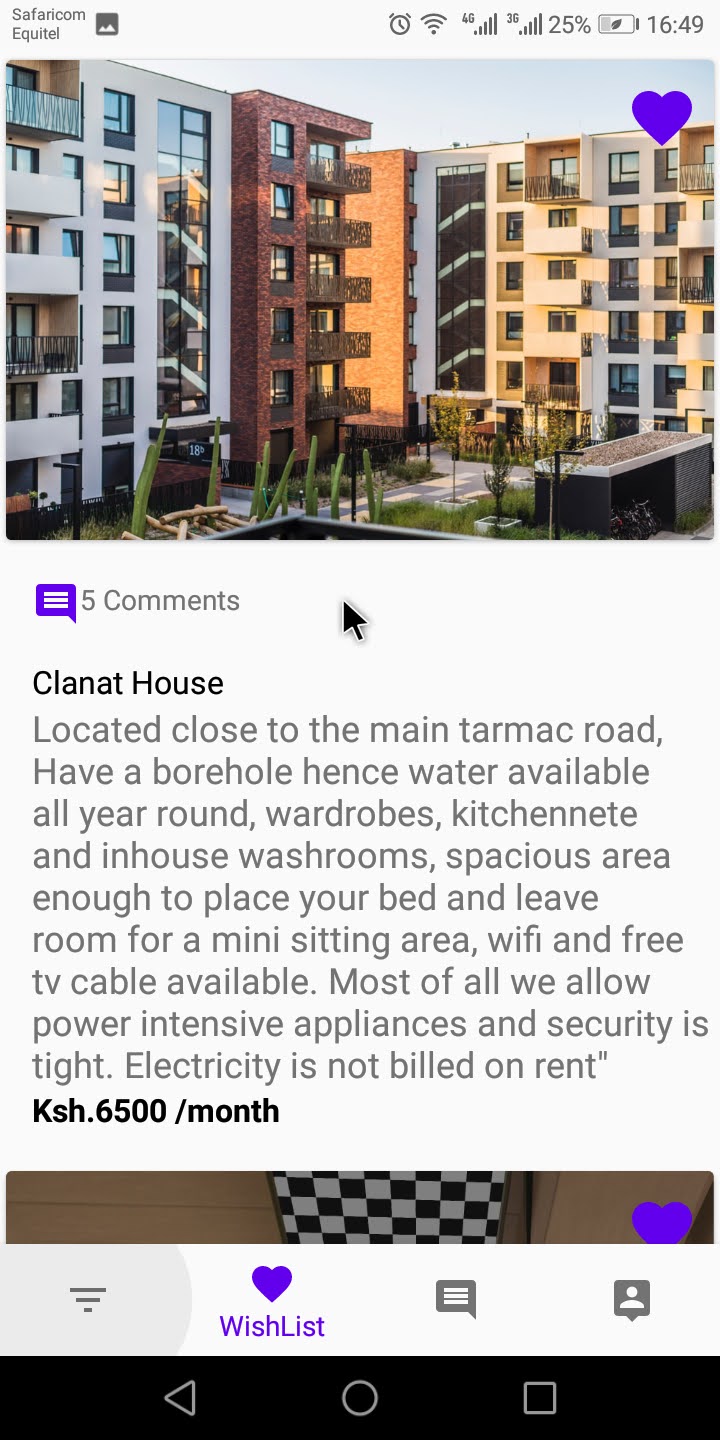
Figure 11: all listing fragment

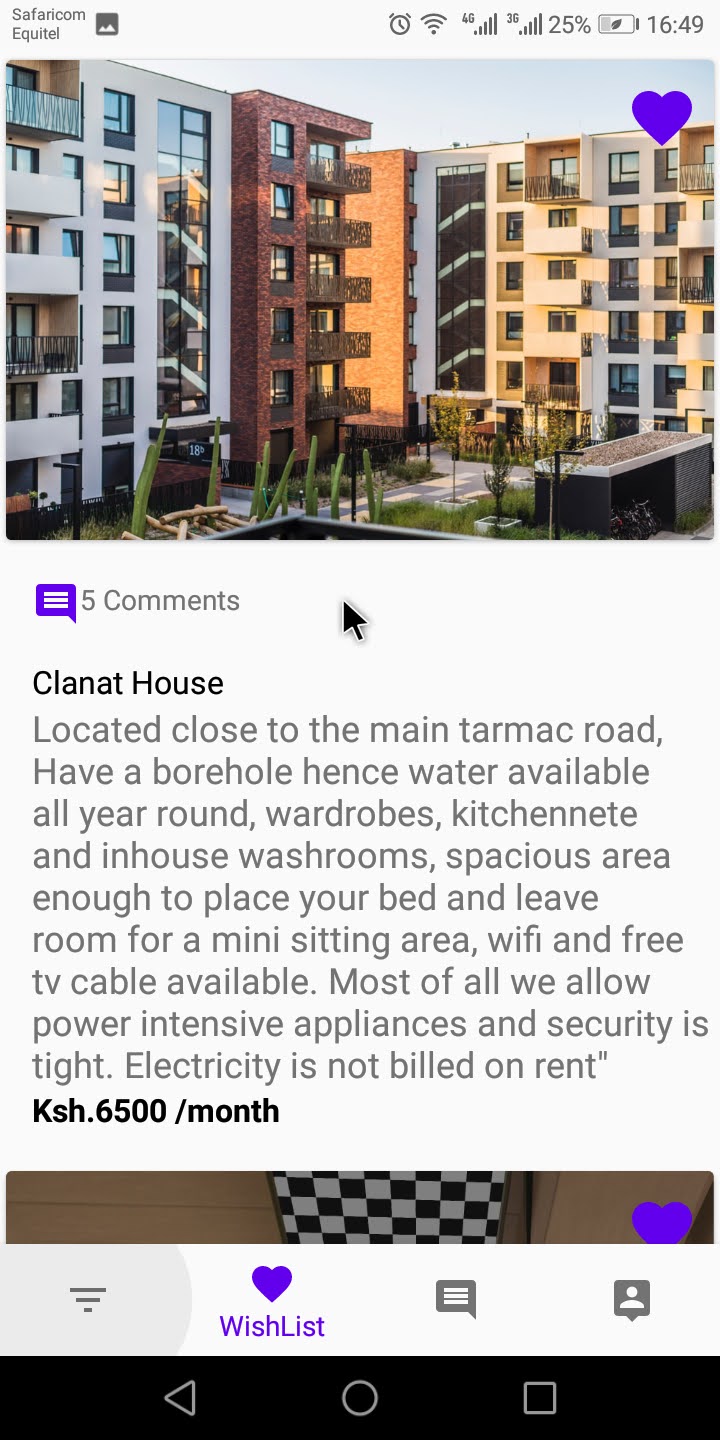
Figure 11: all listing fragment

On this fragment users can see all houses listed. They can like the houses that interest them and place them in a cart from which they can do the elimination.

WISHLIST FRAGMENT

## 

Figure 12: wishlist fragment

Figure 12: wishlist fragment

In this fragment the houses a user like are added. This is where you select your favourite and may be remove the rest.

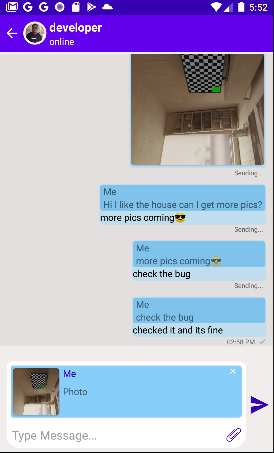
CHATS FRAGMENT

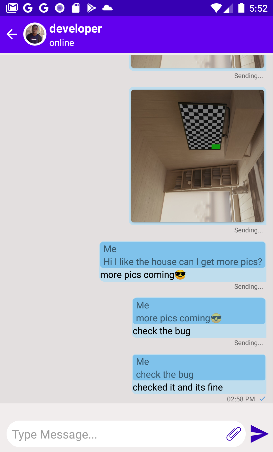
Figure 13: chats fragment

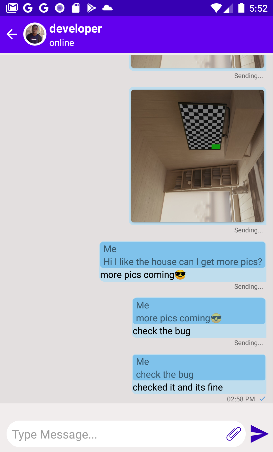
Figure 13: chats fragment

In this fragment all the contacted users are listed.

CHATS ACTIVITY

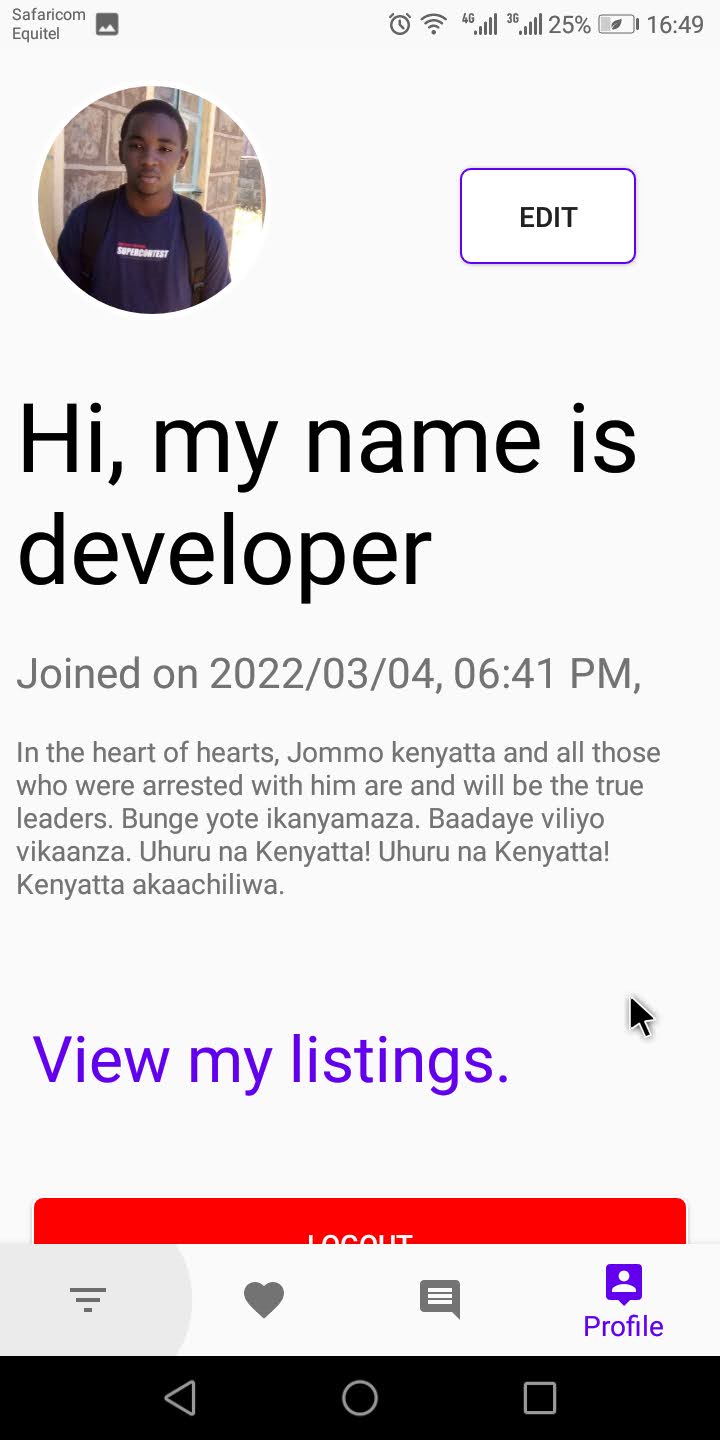


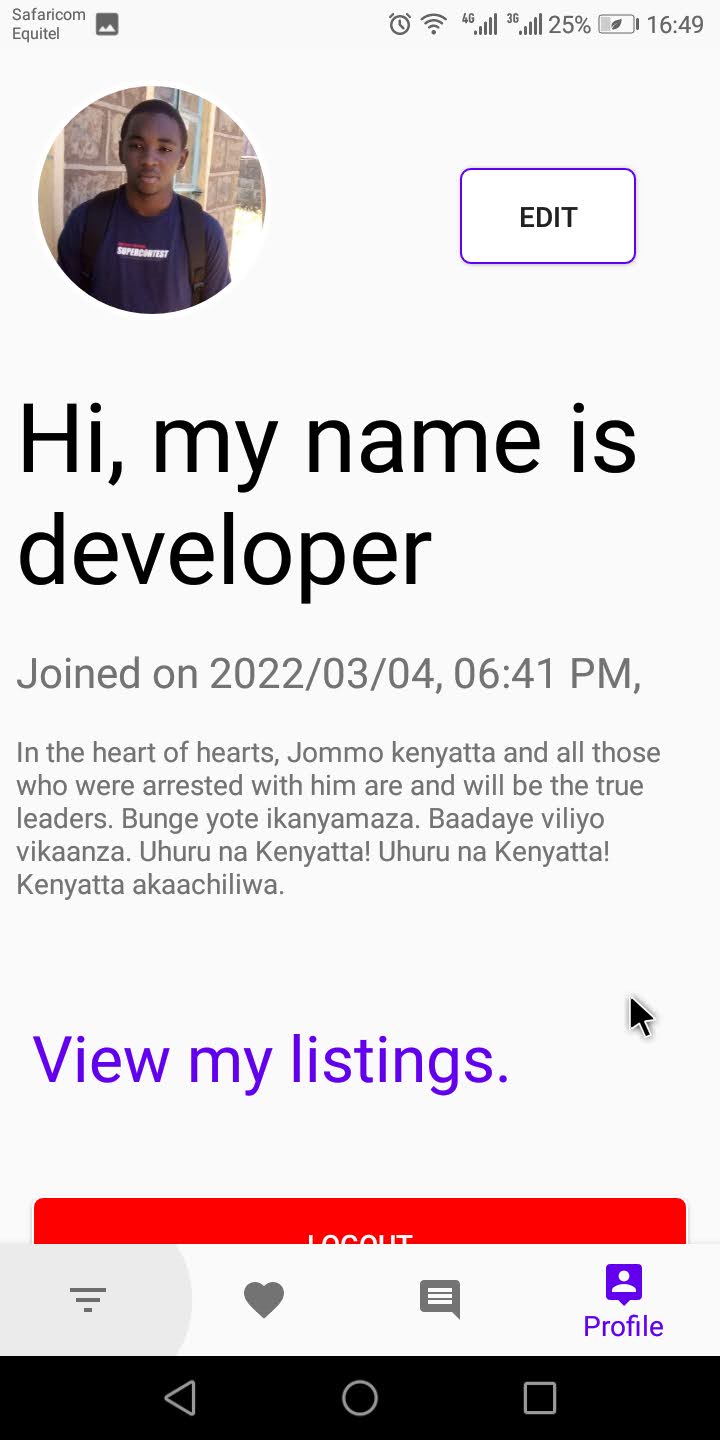
Figure 14: chats activity

Figure 14: chats activity

In this activities the actual messages sent between users are displayed.

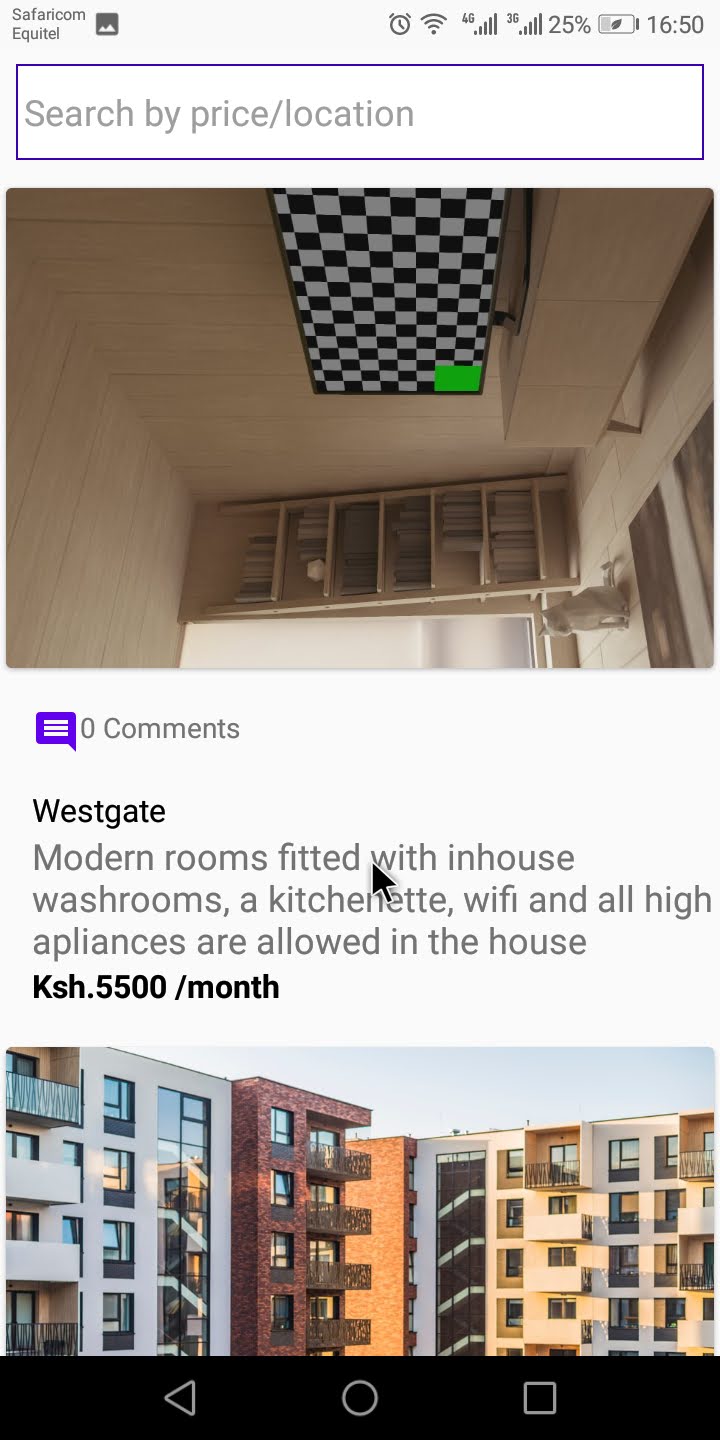
PROFILE FRAGMENT

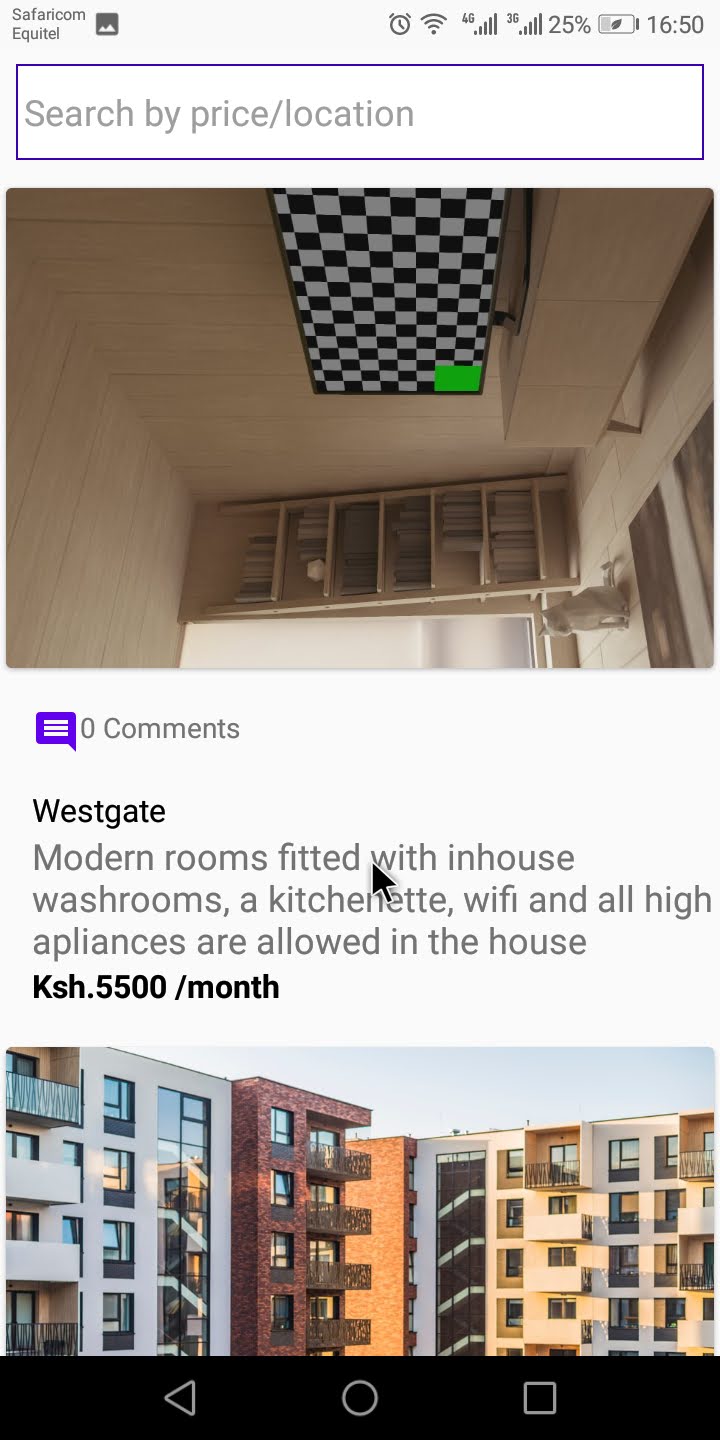
Figure 15: profile fragment

Figure 15: profile fragment

In this fragment a user gets to see his profile details. Also in this fragment a user can edit profile view his listings and also logout from the system.

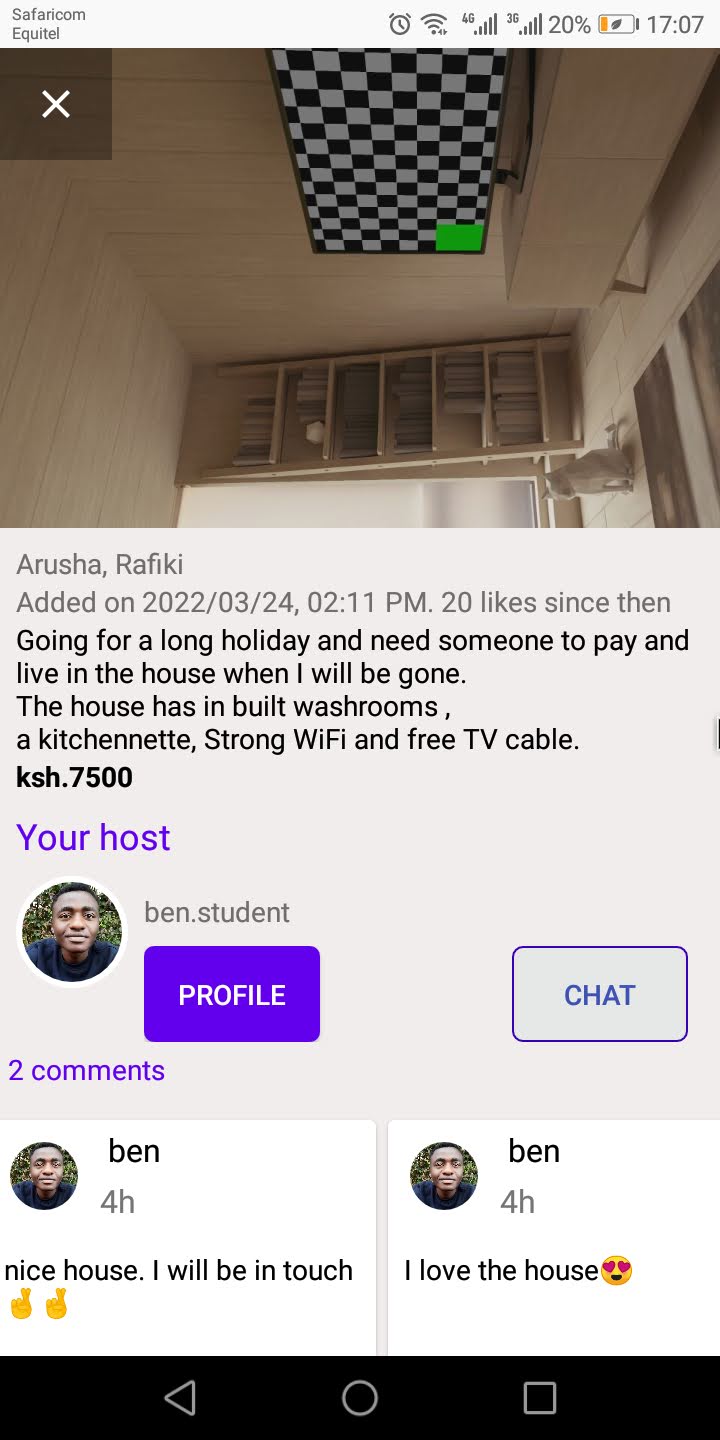
MY LISTINGS ACTIVITY

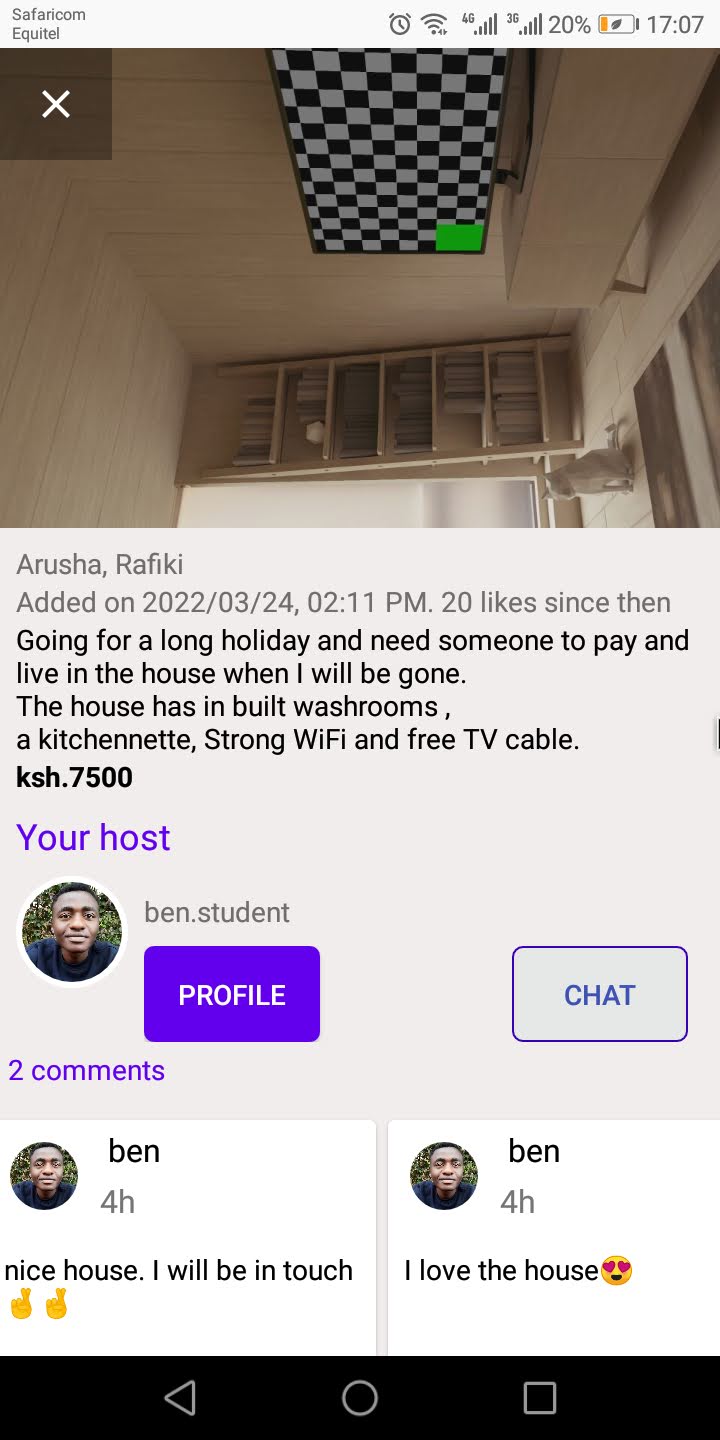
Figure 16: my listings activity

Figure 16: my listings activity

In this activity the user who is currently logged in gets to see all his/her listings.

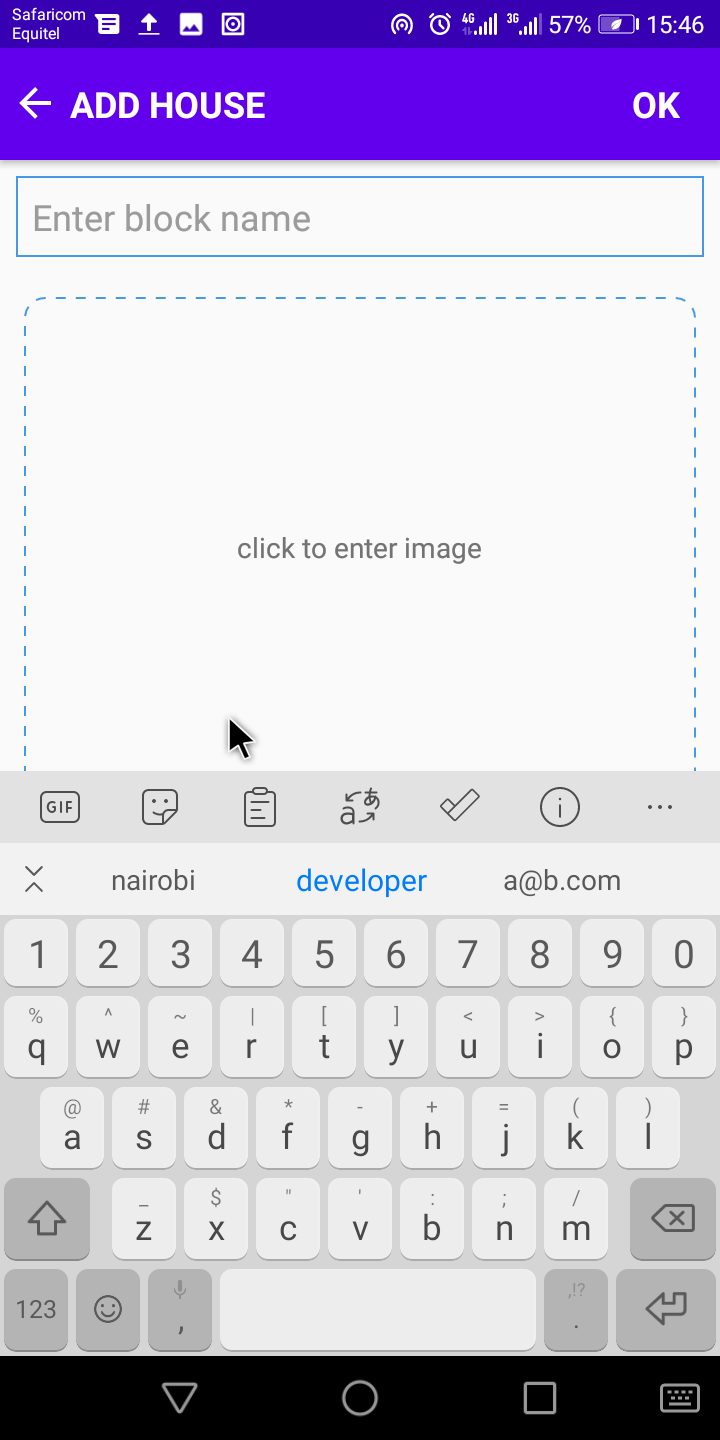
HOUSE DETAILS ACTIVITY

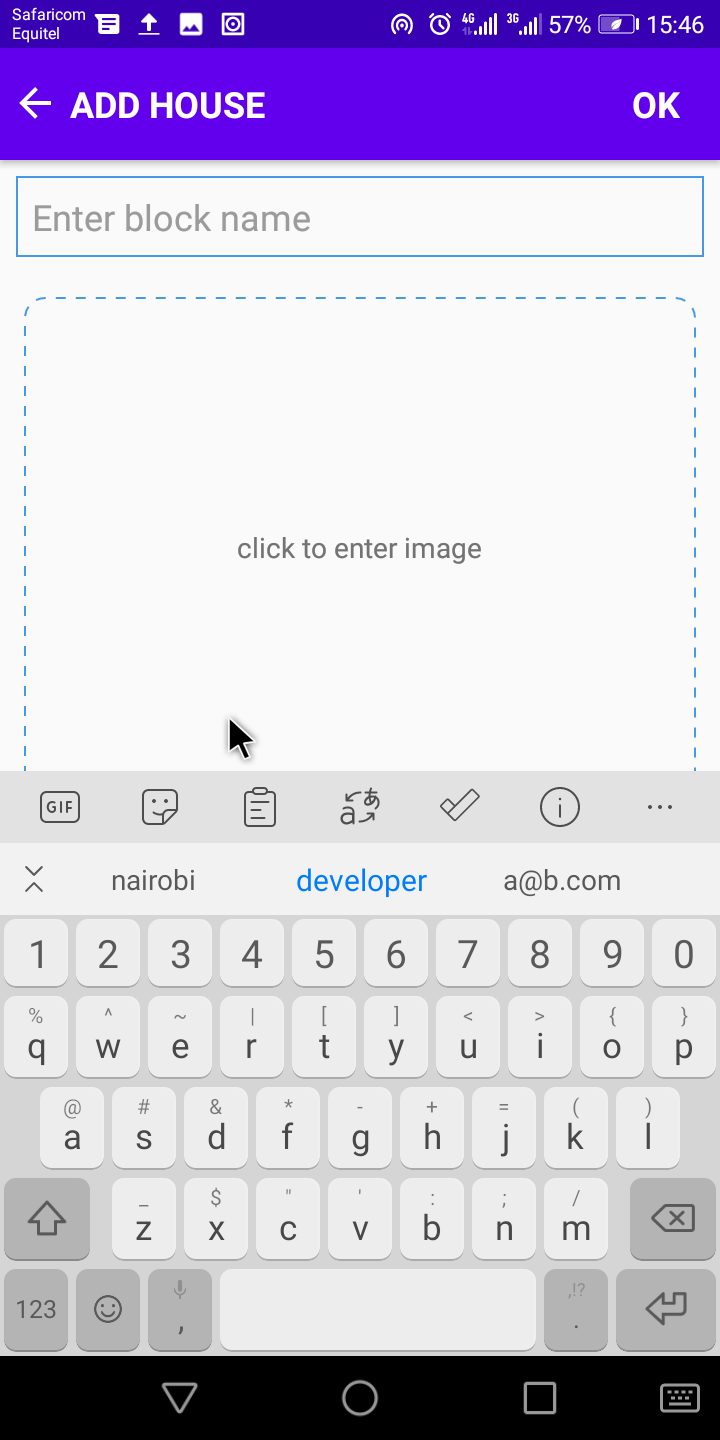
Figure 17: house details activity

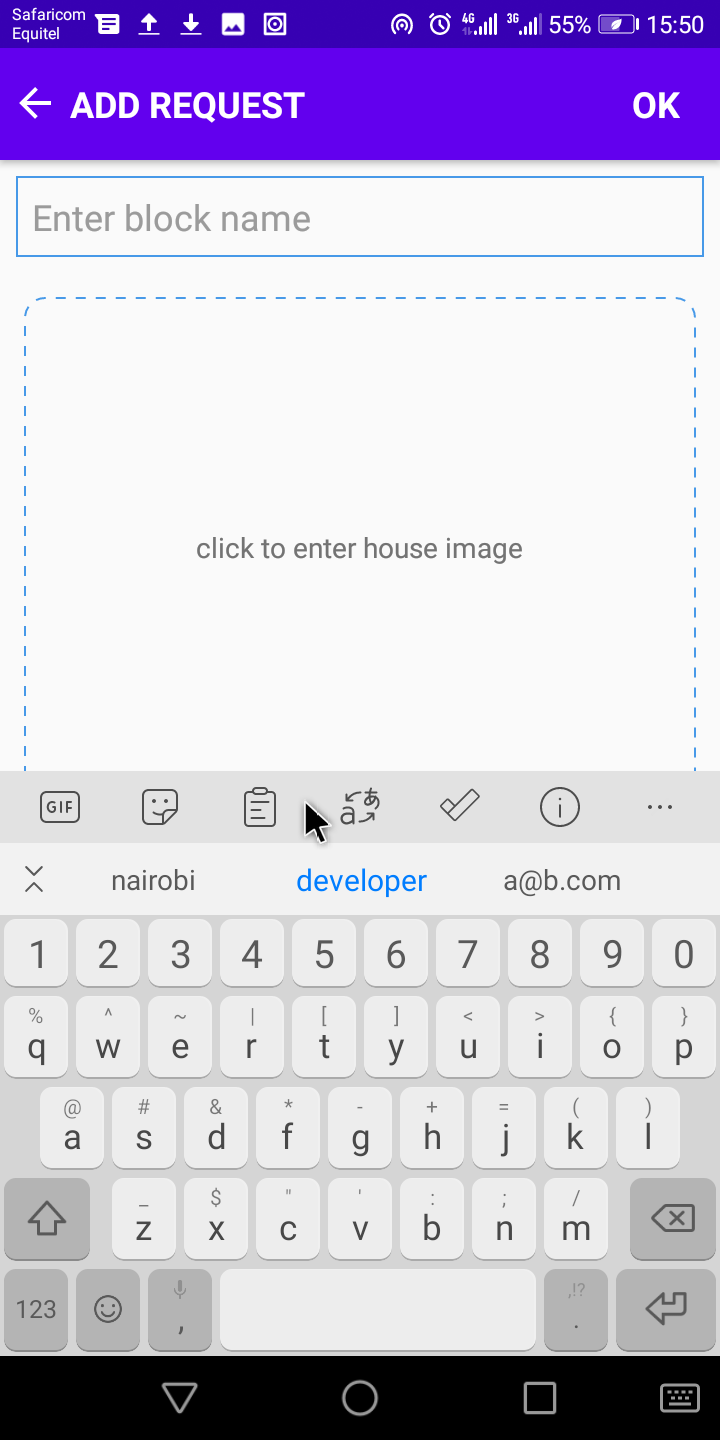
Figure 17: house details activity

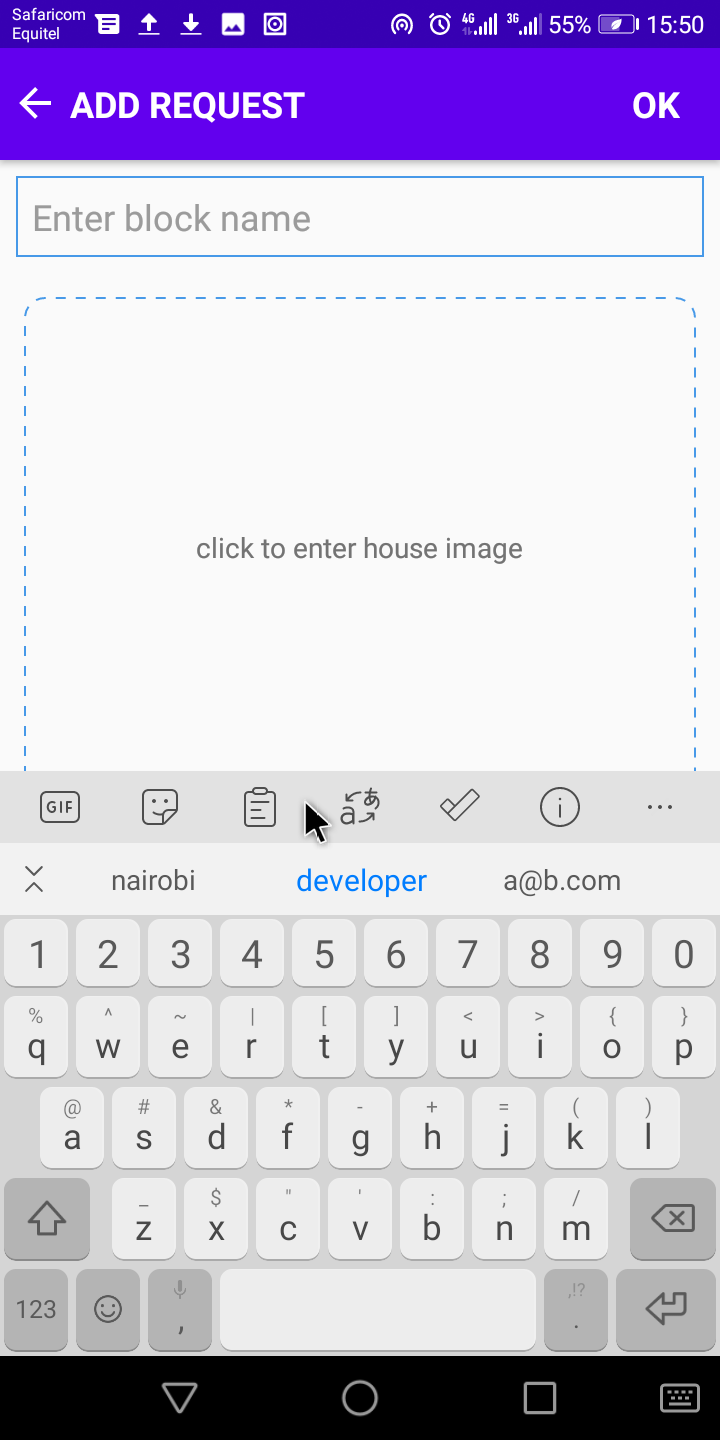
On clicking the house item on the houses fragment, it brings you to the activity above. In this activity, you get to see the owners details and the house comments from other users. You can also view the owners profile or chat with the user. You can also leave a comment.

ADD HOUSE AND FRIEND REQUEST ACTIVITIES

Figure 19: add house activity

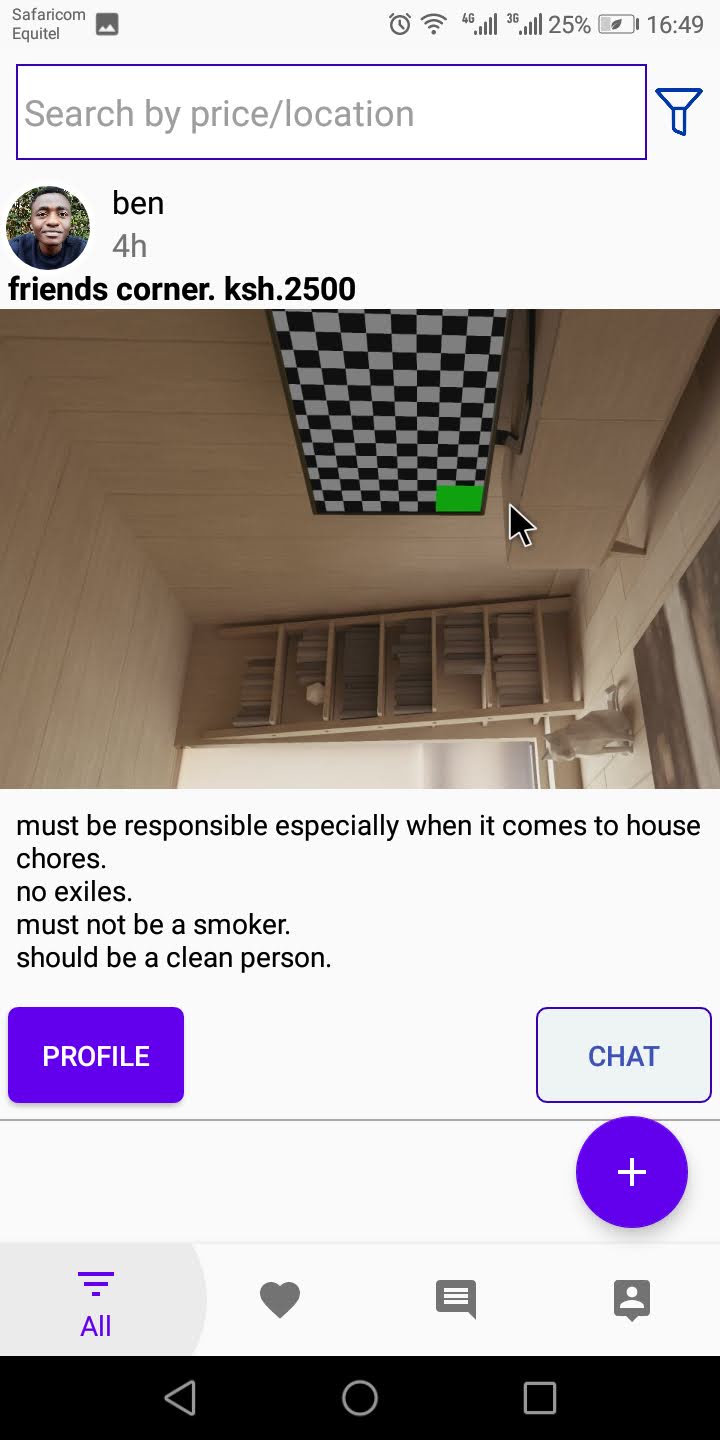
Figure 19: add house activity

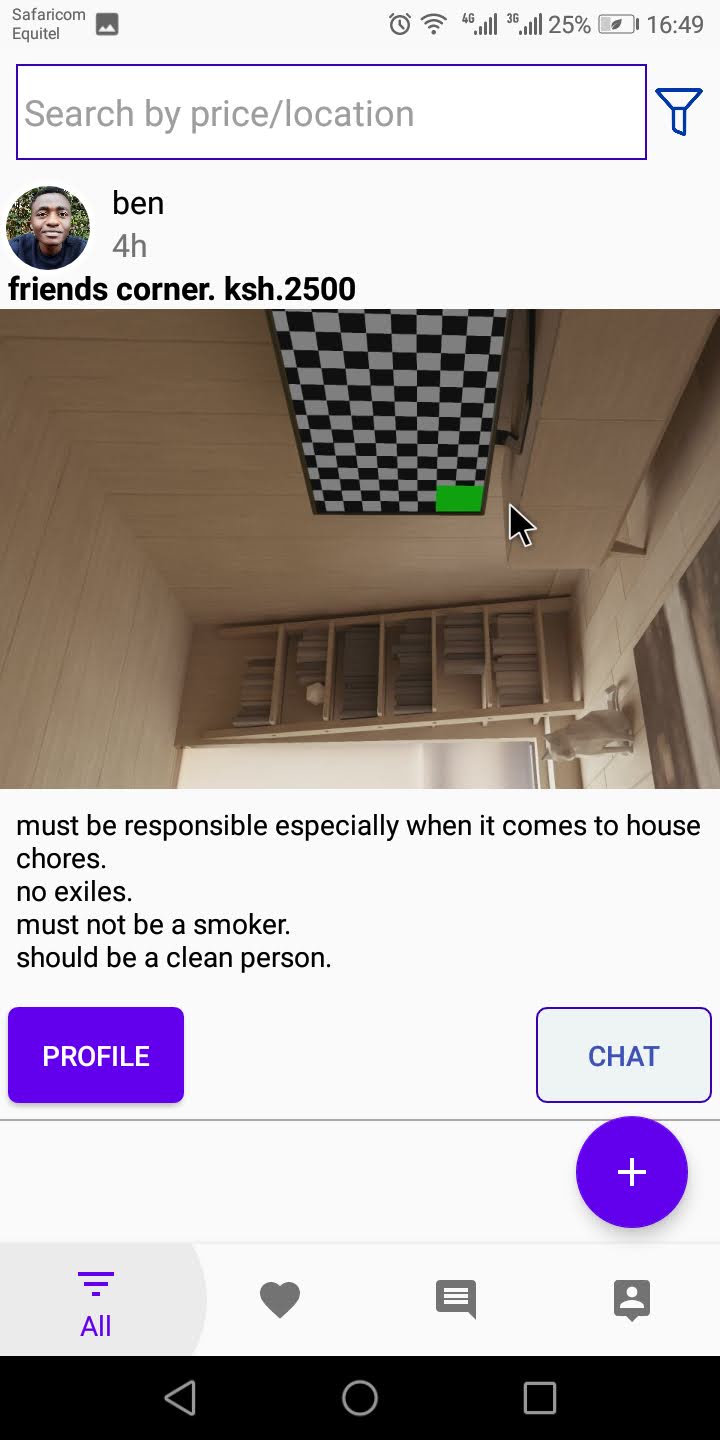
Figure 18: add room mate request activity

Figure 18: add room mate request activity

The activities above are for adding a house and a room mate request respectively. This are in turn displayed for users to select from.

## ROOM MATE FRAGMENT

Figure 20: room mate fragment

Figure 20: room mate fragment

The fragment above displays the request from the db. In this case you can either view profile of the owner or chat the owner .

## 5.4 Implementation Strategies

The process of installing software and hardware and getting the system up and running is known as system implementation. Agile methods were used to develop this software. The agile methodology divided the project into discrete modules, with testing and feedback phases in between. The methodology divided the system software implementation into a number of self-contained segments, each of which delivered a measurable business result. It also made sure that the users became accustomed to the system's operation, flexibility, and utility, as well as provided individual training to prepare users for the new House finding system.

## 5.5 Testing strategies

This represents the testing methods that were employed in this project. They outline how the product risks were addressed at the test level, as well as the types of testing conducted and the entry and exit criteria used.

## 5.6 Test plan

A test plan provides a comprehensive document that includes the test strategy, testing objectives, testing resources, test schedule, test estimation, and test deliverables. This proved efficient in guiding the development of a comprehensive and completely integrated system guaranteeing that all user needs were met and all system capabilities executed smoothly and without glitches.

These are the two types of software testing that were performed:

* Whitebox testing

a test case design that uses the control structure of the procedural design to drive test case.

* Blackbox testing.

focuses on the functional requirements of the software. This is black box testing enables the engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors e.g. interface errors, perfomance in data structures etc.

### **5.6.1** **System Testing**

In this phase,the testing was on the completed software product before it’s release to the general public. The testing was based on a variety of factors, including the system's usability, recovery, and functioning, among others.

### **5.6.2 Acceptance testing**

This was done when the system was put to the test by the users themselves. It was used to determine whether the system met the needs of the clients and whether they would accept it.

## 5.7 Justification of testing and implementation strategy

When designing an object oriented system, the optimal testing technique is module integration and testing. Because the system's modules or objects are integrated and tested, this is the case. This technique assured that the modules are interoperable and work together to produce a fully functional system, as in our case. Because updates to the system are made immediately and even late in the development phase, the Rapid Application Development (RAD) methodology is the optimal implementation strategy for this system. This implementation technique is also based on code reusability, which has been implemented in the development of this system.

# CHAPTER SIX

# CONCLUSION AND RECOMMENDATIONS

## **6.1** **Introduction**

This chapter concludes the description of how the House finding system works. The accomplishments and lessons learned, as well as the findings and recommendations, are highlighted.

## 6.2 Recommendations

Our project is meant to satisfy the needs of rental house owners. Several user friendly interfaces have also been adopted. This package shall prove to be a powerful in satisfying all the requirements of the users. It is with utmost faith that I present this software to you hoping that it will solve your problems and encourage you to continue appreciating technology because it is meant to change and ease all our work that seems to be very difficult. I don‟t mean that my project is the best or that I have used the best

technology available it just a simple and a humble venture that is easy to understand. However, I would encourage anyone who has the ability to advance it using advanced technologies so as to increase its capabilities.

## 6.3 Conclusion

The usage of computer based house finding system means that students can be able to locate and book a house at the comfort of their smart devices and with ease, this will tremendously reduce the work usually done to find houses around Kabarak university especially by first years on the joining day. I will help all the stakeholders in their specific ways all positively. In a distinctive and bright way, the system introduces simplicity and ease.

# APPENDICES

## APPENDIX A: REFERENCES

Erguden, S. (2001), Low cost housing policies and constraints in developing countries, International conference on spatial development for sustainable development, Nairobi.

Golland, A. (1996), Housing supply, profit and housing production: The case of the United Kingdom, Netherlands and Germany, Journal of Housing and the Built Environment, vol.11, no1.

Hancock, T. (1998), Caveat partner: Reflection of Partnership with the private sector, Health promotion international, vol. 13, no 3

Levin, K. (1999), Database Management Systems: How to use Relational Databases, vol. 2, no 4.

Macoloo, G. (1994), The changing nature of financing low income urban housing development in Kenya, Housing Studies, vol. 9, Issue 2, pages 189- 281.

Mitullah, W. (2003), Urban Slums Report: The case of Nairobi Kenya, Understanding Slums: Case Studies for the Global Report on Human Settlements.

Seedhouse, D. (1986), Foundation for Health Achievement, Health Policy, vol. 7, issue, 3.

Cooper, M. (1998), Ideas to develop a literature review, vol. 3, page, 39.

United Nations, (1948), The Bill of Human Rights.

DiMarzio, J. (2017), Begining Android Programming with Android Studio, vol. 4.

Laurence, M. (2017), The Definative Guide to Firebase: Build Android Apps on Google’s Mobile Platform.

Herbert, S. (2019), Java a Beginners Guide, vol. 8.

## APPENDIX B: SAMPLES OF CODE

### Samples of code used in both frontend and backend

#### Manifest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.keytech.roomatefdr">  
 <!-- Internet permission -->  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.CAMERA" />  
 <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
 <activity android:name=".MyListingsActivity"  
 android:theme="@style/AppThemeNo"></activity>  
 <activity  
 android:name=".SendFilesActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".ChatsActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".TheirProfileActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".CommentsActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".HouseDetailActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity android:name=".FullImageActivity" />  
 <activity  
 android:name=".AddRooMateActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".AddHouseActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".EditActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".NextActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".RegisterActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".LoginActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".LaunchActivity"  
 android:theme="@style/AppThemeNo" />  
 <activity  
 android:name=".MainActivity"  
 android:theme="@style/AppThemeNo">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

#### activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <!--FrameLayout: show fragments-->  
 <FrameLayout  
 android:id="@+id/content"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1">  
  
 </FrameLayout>  
  
 <!--Bottom navigation: show menu-->  
 <com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/navigation"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom"  
 android:background="?android:windowBackground"  
 app:menu="@menu/menu\_nav" />  
 </LinearLayout>  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/fab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom|end"  
 android:layout\_marginBottom="64dp"  
 android:layout\_marginEnd="16dp"  
 android:src="@drawable/ic\_add"  
 android:background="@color/colorPrimary"  
 android:layout\_marginRight="16dp" />  
  
</androidx.coordinatorlayout.widget.CoordinatorLayout>

##### Main Activity

package com.keytech.roomatefdr;  
import android.annotation.SuppressLint;  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.os.Build;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.WindowManager;  
import android.widget.Toast;  
import androidx.annotation.NonNull;  
import androidx.annotation.RequiresApi;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.fragment.app.Fragment;  
  
import com.google.android.material.bottomnavigation.BottomNavigationView;  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.Query;  
import com.google.firebase.database.ValueEventListener;  
import com.keytech.roomatefdr.fragments.AllListingsFragment;  
import com.keytech.roomatefdr.fragments.InboxFragment;  
import com.keytech.roomatefdr.fragments.MyWishListFragment;  
import com.keytech.roomatefdr.fragments.ProfileFragment;  
  
import java.util.ArrayDeque;  
import java.util.Deque;  
  
public class MainActivity extends AppCompatActivity {  
  
 private Deque<Integer> integerDeque = new ArrayDeque<>(4);  
 private boolean flag = true;  
  
 private BottomNavigationView navigation;  
  
 private FloatingActionButton fab;  
 private String userType, uid;  
 private SharedPreferences preferences;  
  
 //firebase  
 private FirebaseAuth firebaseAuth;  
 private DatabaseReference userDbRef;  
  
 @SuppressLint({"ResourceAsColor", "WrongConstant"})  
 @RequiresApi(api = Build.VERSION\_CODES.*LOLLIPOP*)  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 //set status text dark  
 getWindow().getDecorView().setSystemUiVisibility(View

.*SYSTEM\_UI\_FLAG\_LIGHT\_STATUS\_BAR*);  
 //set status background white  
 getWindow().setStatusBarColor(android.R.color.*white*);  
 //hide key board  
 getWindow().setSoftInputMode(WindowManager.LayoutParams

.*SOFT\_INPUT\_STATE\_ALWAYS\_HIDDEN*);

//init views  
 fab = findViewById(R.id.*fab*);  
  
 //init firebase  
 firebaseAuth = FirebaseAuth.*getInstance*();  
 checkUserStatus();  
  
 Toast.*makeText*(this, "" + uid, Toast.*LENGTH\_SHORT*).show();  
  
 //onClick  
 fab.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 //get userType  
 getUserType();  
  
 }  
 });  
  
 //Bottom navigation  
 navigation = findViewById(R.id.*navigation*);  
 navigation.setOnNavigationItemSelectedListener(selectedListener);  
  
 //add profile fragment in dequeue list  
 integerDeque.push(R.id.*nav\_all*);  
 if (savedInstanceState == null) {  
 //load home fragment  
 loadFragment(new AllListingsFragment());  
 }  
 }  
  
 private void showNavigationDialog() {  
 String[] options = {"Add House", "Add Room mate request "};  
 //dialog  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Select:");  
 //set options to dialog  
 builder.setItems(options, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 //item click handle  
 if (which == 0) {  
 //add house clicked  
 startActivity(new Intent(MainActivity.this, AddHouseActivity.class));  
 }  
 if (which == 1) {  
 //add roomate clicked  
 startActivity(new Intent(MainActivity.this, AddRooMateActivity.class));  
 }  
 }  
 });  
 //create and show dialog  
 builder.create().show();  
 }  
  
 private void getUserType() {  
 userDbRef = FirebaseDatabase.*getInstance*().getReference("Users");  
 Query query = userDbRef.orderByChild("uid").equalTo(uid);  
 query.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 userType = "" + ds.child("userType").getValue();  
 }  
  
 if (userType.equals("student")) {  
 showNavigationDialog();  
 } else {  
 startActivity(new Intent(MainActivity.this, AddHouseActivity.class));  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 private void loadFragment(Fragment fragment) {  
 getSupportFragmentManager()  
 .beginTransaction()  
 .replace(R.id.*content*, fragment, fragment.getClass()

.getSimpleName())  
 .commit();  
 }  
  
 private BottomNavigationView.OnNavigationItemSelectedListener selectedListener =  
 new BottomNavigationView.OnNavigationItemSelectedListener() {  
 @Override  
 public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {  
  
 //handle item clicks  
 int id = menuItem.getItemId();  
 if (integerDeque.contains(id)) {  
 //check condition  
 if (id == R.id.*nav\_all*) {  
 //when selected id is equal to profile fragment id  
 if (integerDeque.size() != 1) {  
 //when deque list size is not equal to one  
 if (flag) {  
 //when flag value is true add profile //fragment in deque list  
 integerDeque.addFirst(R.id.*nav\_all*);  
 //set flag is equal to false  
 flag = false;  
 }  
 }  
 }  
  
 //remove selected id from deque list  
 integerDeque.remove(id);  
 }  
 //push selected id in deque list  
 integerDeque.push(id);  
 //load fragment  
 loadFragment(getFragment(menuItem.getItemId()));  
  
 return true;  
 }  
 };  
  
 @SuppressLint("RestrictedApi")  
 private Fragment getFragment(int itemId) {  
 switch (itemId) {  
 case R.id.*nav\_all*:  
 //set checked  
 navigation.getMenu().getItem(0).setChecked(true);  
 fab.setVisibility(View.*VISIBLE*);  
 return new AllListingsFragment();  
  
 case R.id.*nav\_wishList*:  
 //set checked  
 navigation.getMenu().getItem(1).setChecked(true);  
 fab.setVisibility(View.*GONE*);  
 return new MyWishListFragment();  
  
 case R.id.*nav\_inbox*:  
 //set checked  
 navigation.getMenu().getItem(2).setChecked(true);  
 fab.setVisibility(View.*GONE*);  
 return new InboxFragment();  
  
 case R.id.*nav\_profile*:  
 //set checked  
 navigation.getMenu().getItem(3).setChecked(true);  
 fab.setVisibility(View.*GONE*);  
 return new ProfileFragment();  
 }  
  
 navigation.getMenu().getItem(0).setChecked(true);  
 //return profile fragment  
 return new AllListingsFragment();  
 }  
  
 private void checkUserStatus() {  
 //get current user  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
 if (user != null) {  
 //user is signed in stay here  
 uid = user.getUid();  
 } else {  
 //user not signed in  
 startActivity(new Intent(this, LaunchActivity.class));  
 finish();  
 }  
 }  
  
 @Override  
 public void onBackPressed() {  
 integerDeque.pop();  
 //check condition  
 if (!integerDeque.isEmpty()) {  
 //when not empty  
 loadFragment(getFragment(integerDeque.peek()));  
 } else {  
 //when empty  
 finish();  
 }  
 }  
}

#### item\_house

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_marginBottom="16dp">  
  
 <androidx.cardview.widget.CardView  
 android:id="@+id/image\_layout"  
 android:layout\_width="match\_parent"  
 app:cardBackgroundColor="@color/color\_white"  
 app:cardCornerRadius="3dp"  
 app:cardUseCompatPadding="true"  
 android:layout\_height="wrap\_content">  
  
 <ImageView  
 android:id="@+id/house\_main\_photoIv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="240dp"  
 android:scaleType="centerCrop"/>  
  
 <RelativeLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content">  
  
 <RelativeLayout  
 android:id="@+id/likeLayout"  
 android:layout\_width="wrap\_content"  
 android:layout\_alignParentRight="true"  
 android:layout\_height="wrap\_content">  
  
 <ImageView  
 android:id="@+id/likeBtn"  
 android:layout\_width="36dp"  
 android:layout\_height="42dp"  
 android:tint="@color/color\_black"  
 android:layout\_margin="8dp"  
 android:src="@drawable/ic\_unliked"/>  
  
 <ImageView  
 android:id="@+id/likedBtn"  
 android:layout\_width="36dp"  
 android:layout\_height="42dp"  
 android:tint="@color/colorPrimary"  
 android:layout\_margin="8dp"  
 android:src="@drawable/ic\_favorite"/>  
 </RelativeLayout>  
  
 </RelativeLayout>  
  
 </androidx.cardview.widget.CardView>  
  
 <TextView  
 android:id="@+id/commentsTv"  
 android:layout\_below="@id/image\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:drawableLeft="@drawable/ic\_comment"  
 android:drawableTint="@color/colorPrimary"  
 android:padding="16dp"  
 android:text=" 2 Comments"/>  
  
 <TextView  
 android:id="@+id/block\_nameTv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/commentsTv"  
 android:text="Baringo House. Rafiki."  
 android:textColor="@color/color\_black"  
 android:textSize="16sp"  
 android:paddingLeft="16dp" />  
  
 <TextView  
 android:id="@+id/house\_descriptionTv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/block\_nameTv"  
 android:textSize="18sp"  
 android:text="Cheerful 2 bedroom residential apartment"  
 android:paddingLeft="16dp" />  
  
 <TextView  
 android:id="@+id/house\_priceTv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/house\_descriptionTv"  
 android:text="Ksh. 2500 / month"  
 android:paddingLeft="16dp"  
 android:textSize="16dp"  
 android:textStyle="bold"  
 android:textColor="@color/color\_black"/>  
  
</RelativeLayout>

#### fragment\_house

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.keytech.roomatefdr.fragments.AllListingsFragment">  
  
 <RelativeLayout  
 android:id="@+id/layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:id="@+id/searchLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_margin="8dp"  
 android:weightSum="10">  
  
 <EditText  
 android:id="@+id/searchET"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_weight="9"  
 android:padding="4dp"  
 android:lines="1"  
 android:hint="Search by price/location"  
 android:background="@drawable/search\_bar" />  
  
 <ImageView  
 android:id="@+id/filterBtn"  
 android:layout\_width="32dp"  
 android:layout\_height="match\_parent"  
 android:background="@null"  
 android:padding="4dp"  
 android:src="@drawable/emptyfilter"/>  
  
 </LinearLayout>  
  
 <!-- <!--houses recyclerView -->  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/housesRecyclerView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/searchLayout"  
 android:layout\_marginTop="2dp" app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  
 tools:listitem="@layout/item\_house" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="72dp"  
 android:layout\_below="@id/housesRecyclerView" />  
  
 </RelativeLayout>  
  
</RelativeLayout>

#### AdapterHouse

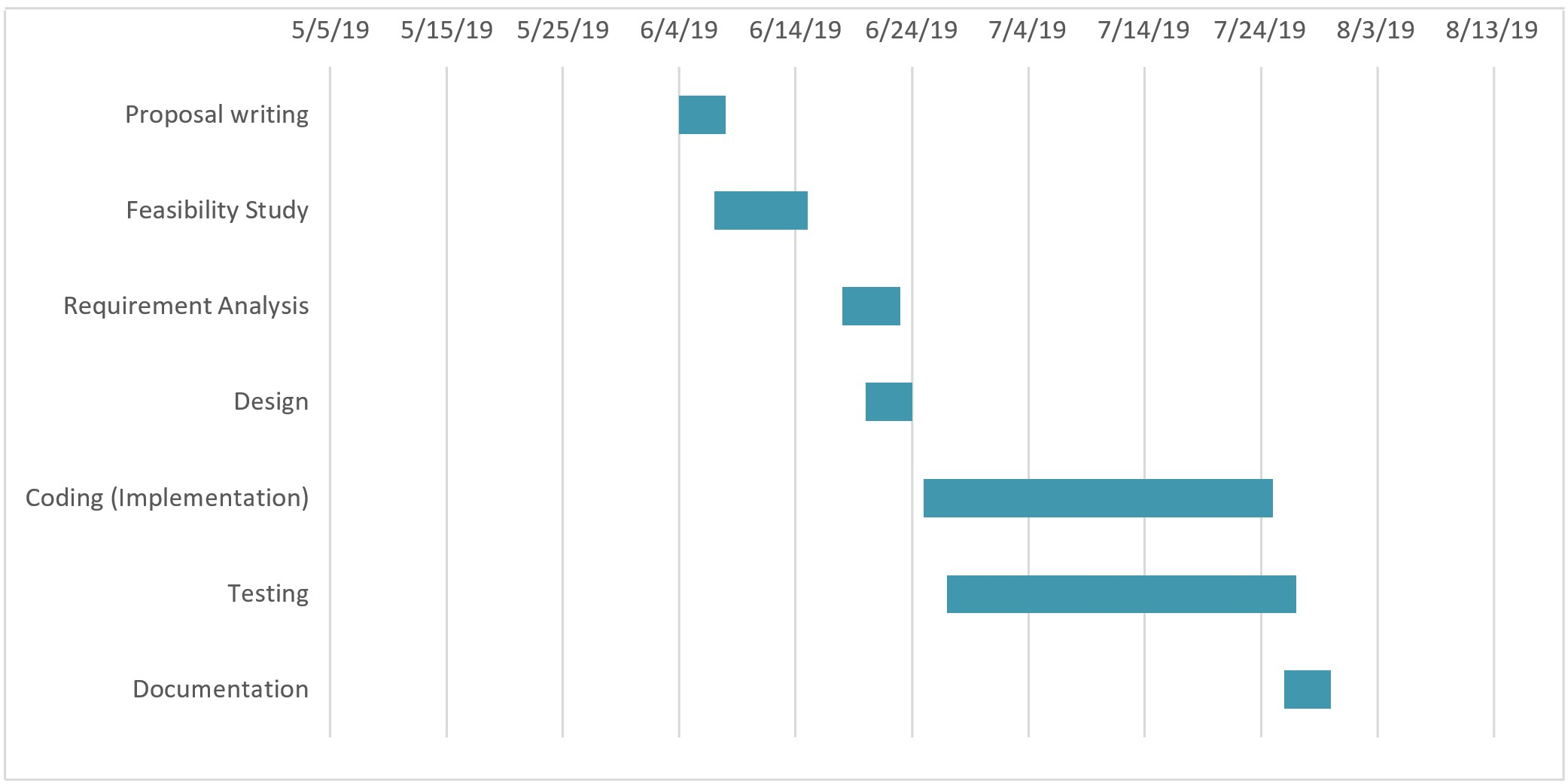
package com.keytech.roomatefdr.adapters;  
import android.app.AlertDialog;  
import android.app.ProgressDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import android.widget.RelativeLayout;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.cardview.widget.CardView;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.google.android.gms.tasks.OnFailureListener;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.ValueEventListener;  
import com.keytech.roomatefdr.HouseDetailActivity;  
import com.keytech.roomatefdr.R;  
import com.keytech.roomatefdr.models.House;  
import com.squareup.picasso.Picasso;  
  
import java.util.List;  
  
public class AdapterHouse extends RecyclerView.Adapter<AdapterHouse.MyHolder> {  
  
 private List<House> houseList;  
 private Context context;  
  
 String myUid, owner;  
 private ProgressDialog pd;  
  
 private DatabaseReference reference;  
 private DatabaseReference likesRef;  
 private FirebaseAuth firebaseAuth;  
  
 private boolean mProcessLike = false;  
  
 public AdapterHouse(List<House> houseList, Context context, String owner) {  
 this.houseList = houseList;  
 this.context = context;  
 this.owner = owner;  
  
 myUid = FirebaseAuth.*getInstance*().getCurrentUser().getUid();  
 likesRef = FirebaseDatabase.*getInstance*().getReference().child("Likes");  
 reference = FirebaseDatabase.*getInstance*().getReference("Houses");  
 firebaseAuth = FirebaseAuth.*getInstance*();  
 }  
  
 @NonNull  
 @Override  
 public MyHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int viewType) {  
 View view = LayoutInflater.*from*(context).inflate(R.layout.*item\_house*, viewGroup, false);  
 return new MyHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull MyHolder holder, final int i) {  
 House house = houseList.get(i);  
 //get data  
 final String id = house.getId();  
 final String uid = house.getUid();  
 String blockName = house.getBlockName();  
 String location = house.getLocation();  
 final String image = house.getImg();  
 String description = house.getDescription();  
 String rent = house.getRent();  
  
 //set image  
 try {  
 Picasso.*get*().load(image).into(holder.houseImage);  
 } catch (Exception e) {  
  
 }  
 //set data  
 holder.block\_nameTv.setText(blockName);  
 holder.house\_descriptionTv.setText(description);  
 holder.house\_priceTv.setText("Ksh." + rent + " /month");  
  
 getComments(holder, id);  
  
 if (owner.equals("myHouse")) {  
 holder.likeLayout.setVisibility(View.*GONE*);  
 //onClick  
 holder.itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 showDeleteDialog(id);  
 }  
 });  
 } else {  
 //onClick  
 holder.itemView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intent = new Intent(context, HouseDetailActivity.class);  
 intent.putExtra("id", id);  
 intent.putExtra("uid", uid);  
 intent.putExtra("image", image);  
 context.startActivity(intent);  
 }  
 });  
 }  
 //set likes  
 setLikes(holder, id);  
  
 holder.likeLayout.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 likePost(id);  
 }  
 });  
  
 }  
  
 private void showDeleteDialog(final String id) {  
 //dialog  
 AlertDialog.Builder builder = new AlertDialog.Builder(context);  
 builder.setTitle("Delete House?");  
 //set options to dialog  
 builder.setMessage("This action is irreversible!")  
 .setPositiveButton("Ok", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.dismiss();  
 deleteHouse(id);  
 }  
 }).setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.dismiss();  
 }  
 });  
 //create and show dialog  
 builder.create().show();  
 }  
  
 private void deleteHouse(String id) {  
 //remove uid from the list of houses  
 pd.setMessage("deleting...");  
 pd.show();  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Houses");  
 ref.orderByChild("id").equalTo(id)  
 .addListenerForSingleValueEvent(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 for (DataSnapshot ds : snapshot.getChildren()) {  
 if (ds.exists()) {  
 ds.getRef().removeValue()  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void aVoid) {  
 //deleted  
 pd.dismiss();  
 Toast.*makeText*(context, "Deleted", Toast.*LENGTH\_SHORT*).show();  
 }  
 })  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 //failed  
 pd.dismiss();  
 Toast.*makeText*(context, "failed:" + e.getMessage(), Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void setLikes(final MyHolder holder, final String id) {  
 likesRef.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
 if (snapshot.child(myUid).hasChild(id)) {  
 //user has liked, show colored heart  
 holder.likedBtn.setVisibility(View.*VISIBLE*);  
 holder.likeBtn.setVisibility(View.*INVISIBLE*);  
 holder.likeBtn.setTag("Liked");  
 } else {  
 //user has not liked  
 holder.likeBtn.setVisibility(View.*VISIBLE*);  
 holder.likedBtn.setVisibility(View.*INVISIBLE*);  
 holder.likeBtn.setTag("Like");  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void likePost(final String id) {  
 mProcessLike = true;  
 likesRef.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot snapshot) {  
  
 if (mProcessLike) {  
 if (snapshot.child(myUid).hasChild(id)) {  
 //already liked so remove like  
 // dislikeHeart(holder.like);  
 likesRef.child(myUid).child(id).removeValue();  
 mProcessLike = false;  
 } else {  
 //not liked, like it  
 //animateHeart(holder.liked);  
 likesRef.child(myUid).child(id).setValue("Liked");  
 mProcessLike = false;  
 }  
 }  
  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError error) {  
  
 }  
 });  
 }  
  
 private void getComments(final MyHolder holder, String id) {  
 DatabaseReference reference = FirebaseDatabase.*getInstance*().getReference("Comments").child(id);  
 reference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 holder.commentsTv.setText(dataSnapshot.getChildrenCount() + " Comments");  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 @Override  
 public int getItemCount() {  
 return houseList.size();  
 }  
  
 public class MyHolder extends RecyclerView.ViewHolder {  
  
 //xml views  
 private ImageView likeBtn, likedBtn, houseImage;  
 private TextView commentsTv, block\_nameTv, house\_descriptionTv, house\_priceTv;  
 private CardView cardView;  
 private RelativeLayout likeLayout;  
  
 public MyHolder(@NonNull View itemView) {  
 super(itemView);  
  
 //init views  
 cardView = itemView.findViewById(R.id.*image\_layout*);  
 likeLayout = itemView.findViewById(R.id.*likeLayout*);  
 likeBtn = itemView.findViewById(R.id.*likeBtn*);  
 likedBtn = itemView.findViewById(R.id.*likedBtn*);  
 houseImage = itemView.findViewById(R.id.*house\_main\_photoIv*);  
 commentsTv = itemView.findViewById(R.id.*commentsTv*);  
 block\_nameTv = itemView.findViewById(R.id.*block\_nameTv*);  
 house\_descriptionTv = itemView.findViewById(R.id.*house\_descriptionTv*);  
 house\_priceTv = itemView.findViewById(R.id.*house\_priceTv*);  
  
 }  
 }  
}

#### House Fragment

package com.keytech.roomatefdr.fragments;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.Gravity;  
import android.view.LayoutInflater;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.widget.PopupMenu;  
import androidx.fragment.app.Fragment;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.Query;  
import com.google.firebase.database.ValueEventListener;  
import com.keytech.roomatefdr.MainActivity;  
import com.keytech.roomatefdr.MyListingsActivity;  
import com.keytech.roomatefdr.R;  
import com.keytech.roomatefdr.adapters.AdapterHouse;  
import com.keytech.roomatefdr.adapters.AdapterRooMates;  
import com.keytech.roomatefdr.models.House;  
import com.keytech.roomatefdr.models.RooMate;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
*/\*\**  
 *\* A simple {****@link*** *Fragment} subclass.*  
 *\*/*  
public class AllListingsFragment extends Fragment {  
  
 private EditText searchET;  
 private ImageView filterBtn;  
  
 private RecyclerView houseRecyclerView;  
 private AdapterHouse adapterHouse;  
 private List<House> houseList;  
  
 private AdapterRooMates adapterRooMates;  
 private List<RooMate> rooMateList;  
  
 //firebase  
 private FirebaseAuth firebaseAuth;  
 private FirebaseUser user;  
 private String myuid;  
  
 private String name, uid, dp;  
  
 public AllListingsFragment() {  
 // Required empty public constructor  
 }  
  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 // Inflate the layout for this fragment  
 View view = inflater.inflate(R.layout.*fragment\_all\_listings*, container, false);  
  
 //initialize recyclerView  
 searchET = view.findViewById(R.id.*searchET*);  
 filterBtn = view.findViewById(R.id.*filterBtn*);  
 houseRecyclerView = view.findViewById(R.id.*housesRecyclerView*);  
  
 //initialize comradeList  
 houseList = new ArrayList<>();  
 rooMateList = new ArrayList<>();  
  
 //init firebase  
 firebaseAuth = FirebaseAuth.*getInstance*();  
 user = FirebaseAuth.*getInstance*().getCurrentUser();  
  
 if(getActivity() instanceof MyListingsActivity){  
 MyListingsActivity activity = (MyListingsActivity) getActivity();  
 String myId = activity.getMyData();  
 getMyHouses(myId);  
 filterBtn.setVisibility(View.*GONE*);  
 }else {  
 myuid = user.getUid();  
 getUserData();  
 getAllHouses(myuid);  
 }  
  
 searchET.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
  
 }  
  
 @Override  
 public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
  
 }  
  
 @Override  
 public void afterTextChanged(Editable s) {  
 searchItem(s.toString(), 1);  
 }  
 });  
  
 filterBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 showPopUp();  
 }  
 });  
  
 return view;  
 }  
  
 private void getMyHouses(final String myId) {  
 //get path of database named Comrades  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Houses");  
  
 //get all data from path  
 ref.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 houseList.clear();  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 House house = ds.getValue(House.class);  
 if (house.getId() != null && house.getUid().equals(myId)) {  
 houseList.add(house);  
 }  
  
 //adapter

adapterHouse = new AdapterHouse(houseList, getActivity(), "myHouse");  
 //set adapter to recyclerView  
 houseRecyclerView.setAdapter(adapterHouse);  
  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 private void showPopUp() {  
 //pop up  
 PopupMenu popupMenu = new PopupMenu(getActivity(), filterBtn, Gravity.*END*);  
  
 //add items  
 //add items in menu  
 popupMenu.getMenu().add(Menu.*NONE*, 0, 0, "All houses");  
 popupMenu.getMenu().add(Menu.*NONE*, 1, 0, "Owner houses");  
 popupMenu.getMenu().add(Menu.*NONE*, 2, 0, "Student houses");  
 popupMenu.getMenu().add(Menu.*NONE*, 3, 0, "Room mates finders");  
  
 popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 int id = item.getItemId();  
 if (id == 0) {  
 //house by owners clicked  
 getAllHouses(myuid);  
 } else if (id == 1) {  
 //house by students clicked  
 searchItem("owner", 2);  
 } else if (id == 2) {  
 //house by students clicked  
 searchItem("student", 2);  
 } else if (id == 3) {  
 //room mate requests clicked  
 getRooMateRequests();  
 }  
 return false;  
 }  
 });  
  
 popupMenu.show();  
 }  
  
 private void getRooMateRequests() {  
 //get path of database named Comrades  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("RooMates");  
  
 //get all data from path  
 ref.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 rooMateList.clear();  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 RooMate rooMate = ds.getValue(RooMate.class);  
 if (rooMate.getId() != null) {  
 rooMateList.add(rooMate);  
 }  
  
 //adapter  
 adapterRooMates = new AdapterRooMates(rooMateList, getActivity());  
 //set adapter to recyclerView  
 houseRecyclerView.setAdapter(adapterRooMates);  
  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 private void searchItem(final String s, final int num) {  
 //get path of database  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Houses");  
 //get all data from path  
 ref.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 houseList.clear();  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 House house = ds.getValue(House.class);  
  
 if(num == 1){  
 if (house.getRent().toLowerCase().contains(s.toLowerCase()) || house.getLocation().toLowerCase().contains(s.toLowerCase())  
 || house.getBlockName().toLowerCase().contains(s.toLowerCase())) {  
 if(!(house.getUid().equals(myuid)))  
 houseList.add(house);  
 }  
 }  
 if(num == 2){  
 if (house.getUserType().toLowerCase().contains(s.toLowerCase())) {  
 if(!(house.getUid().equals(myuid)))  
 houseList.add(house);  
 }  
 }  
  
 //adapter  
 adapterHouse = new AdapterHouse(houseList, getActivity(), "");  
 //refresh adapter  
 adapterHouse.notifyDataSetChanged();  
 //set adapter to recyclerView  
 houseRecyclerView.setAdapter(adapterHouse);  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 private void getAllHouses(final String myuid) {  
 //get path of database named Comrades  
 DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Houses");  
  
 //get all data from path  
 ref.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 houseList.clear();  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 House house = ds.getValue(House.class);  
 if (house.getId() != null && ! (house.getUid().equals(myuid))) {  
 houseList.add(house);  
 }  
  
 //adapter  
 adapterHouse = new AdapterHouse(houseList, getActivity(), "");  
 //set adapter to recyclerView  
 houseRecyclerView.setAdapter(adapterHouse);  
 adapterHouse.notifyDataSetChanged();  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 private void getUserData() {  
 DatabaseReference userDbRef = FirebaseDatabase.*getInstance*().getReference("Users");  
 Query query = userDbRef.orderByChild("uid").equalTo(uid);  
 query.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
 for (DataSnapshot ds : dataSnapshot.getChildren()) {  
 name = "" + ds.child("name").getValue();  
 dp = "" + ds.child("dp").getValue();  
 }  
 }  
  
 @Override  
 public void onCancelled(@NonNull DatabaseError databaseError) {  
  
 }  
 });  
 }  
  
 @Override  
 public void onStart() {  
 super.onStart();  
 }  
  
 private void checkUserStatus() {  
 //get current user  
 FirebaseUser user = firebaseAuth.getCurrentUser();  
 if (user != null) {  
 //user is signed in stay here  
 uid = user.getUid();  
 } else {  
 //user not signed in  
 startActivity(new Intent(getActivity(), MainActivity.class));  
 getActivity().finish();  
 }  
 }  
}

## APPENDIX C: WORK PLAN

Figure 21

## APPENDIX D: RESEARCH BUDGET

|  |  |  |
| --- | --- | --- |
| ITEM | QUANTITY | COST IN Ksh. |
| Laptop | 1 | 47,000 |
| Printing expenses |  | 3,000 |
| Windows 10 OS |  | 15,000 |
| MS office | 1 | 10,000 |
| Traveling expenses | 1 | 1,000 |
| Internet expenses |  | 10,000 |
| Others |  | 10,000 |
| **TOTAL** | **7** | **96,000** |

Table 1: research budget