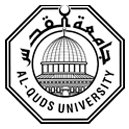
بسم الله الرحمن الرحيم



Faculty of Engineering

Computer Engineering Department

**Final Software Graduation Project Report**

SAKAN-University Housing Services

Final graduation project report submitted to the Faculty of Engineering as a partial fulfillment toward bachelor degree in Computer Engineering.

By:

**Huthyfah Wajeeh Amer**

**Yousef Hussain Hababbeh**

**Haitham Mohammad Ajaj**

Supervisor**:**

**Dr. Yacoub Sabatin**

**Table of contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subject | | | | Page Number |
| Abstract | | | | III |
| Table of contents | | | | IV |
| List of figures | | | | VI |
| List of Tables | | | | VIII |
| **Chapter One: Introduction** | | | | |
| Section Number | | Section Name | Page Number | |
| 1.1 | | Overview | 6 | |
| 1.2 | | Project Idea Description | 6 | |
| 1.3 | | Problem statement | 6 | |
| 1.4 | | Solution impact | 7 | |
| 1.5 | | Related work | 7 | |
| **Chapter Two: System Functional Requirements** | | | | |
| 2.1 | | Introduction | 8 | |
| 2.2 | | Use-Case Diagrams | 8 | |
| 2.3 | | Functional requirements description | 10 | |
| 2.4 | | Non-Functional requirements description | 12 | |
| **Chapter Three: Realization and Implementation** | | | | |
| 3.1 | | Architectural design | 13 | |
| 3.2 | | Class diagram | 17 | |
| 3.3 | | Sequence diagram | 18 | |
| 3.4 | | Entity relationship diagram | 19 | |
| 3.5 | | Database Normalization | 20 | |
| 3.6 | | Software evolution | 22 | |
| **Chapter Four: testing and results** | | | | |
| 4.1 | Introduction | | 24 | |
| 4.2 | System testing | | 24 | |
| 4.3 | Testing results | | 25 | |
| **Chapter Five :conclusion** | | | | |
| 5.1 | Conclusion | | | 26 |
| 5.2 | References | | 26 | |
| **Chapter Six :Appendices** | | | | |
| 6.1 | Views | | 27 | |
| 6.2 | Code | | 38 | |

**List of Figures**

|  |  |
| --- | --- |
| Figures name | page |
| Figure 2.2 Use Case Diagram | 9 |
| Figure 3.1 System Architecture | 13 |
| Figure .2 Layered Architecture | 15 |
| Figure 3.3 ER Diagram | 19 |
| Figure 3.4 Relational Model | 22 |
| Figure .1.1 Login Interface | 27 |
| Figure 6.1.2 Sign up Interface | 28 |
| Figure 6.1.3 Main Page | 29 |
| Figure 6.1.4 Homes List | 30 |
| Figure 6.1.5 Sort Options | 31 |
| Figure 6.1.6 Home details | 32 |
| Figure 6.1.7 Insert Home Details | 33 |
| Figure 6.1.8 User Profile | 34 |
| Figure 6.1.9 Profile Settings | 35 |

**Abstract**

It is a difficult task to find a home for the university students with their required specification especially for the new students, our project represent a solution by implementing android application and utilize the spread of smart phones and ease of using it, this application provide the user the ability to advertise their homes and lease the existing homes, and will provide the ability to receive notifications about the important actions this functionality include the creating account, managing their profile and storing some required information, all the required data will be stored in a remote database server which will communicate through the internet and we will PHP as a server side scripting language to manage the database transactions.

**Chapter 1: introduction and Background**

**1.1Overview**

University housing is very important subject for most of the university students especially those from far places, they have to find the appropriate and comfortable home, in this document we present a software project which will help the students to find residence for living and will help home owners find tenants and providing good marketing and advertising tool, we have chosen to implement this project as a mobile application which available tool for most of people in this time.

**1.2 Project idea description**

Our project is an application for android mobiles is intended to be used by the university students who looks for a home to live and homes owners to post their homes advertisements, the application provide the basic functionality that is required such as add/view homes, receive notifications, and profile management.

**1.3 problem statement**

It is difficult to university student to find appropriate residence to live during his university study period especially first-year students, in the traditional way the student have to ask people to know who has a house for rent, at the same time the house owner may have difficulty in finding tenants and marketing his house, on the other hand, a student may have vacant room in his house and need another student to live with him to reduce the rental cost, for this many reasons it must be a tool to put together each one who interested in university housing taking advantage of the prevalence of smart phones and mobile applications.

**1.4 Solution impact**

To solve this problem we have implemented a mobile application which is a good and easy tool to use instead of search manually by asking peoples or other ways, it will solve the problem of misunderstanding of home specification by provide a detailed description of each advertised home, and will help students who need a part of home, and help the problem of marketing, and using this application will not need any effort, and today internet is everywhere.

**1.5 Related Work**

Zillow Rentals

The Zillow Rentals app (Apple, Android) gives users access to more than 400,000 spaces for rent throughout the country. You have the ability to personalize a search with specific features such as pet policy, in-unit laundry or on-site parking, and then save the searches that interest you the most. [1]

**Chapter 2: System Requirements**

**2.1 Introduction**

If we talking about the system requirements then we have to describe the services that the system will provide, the features that included in the system and the possible constraints which we may encounter during system operation. The requirements must be taken in details and defined accurately because it will be the basis of the contract between the developer and the customer and will depend on it to evaluate the final product.

**2.2 Use case diagram**

Use case diagrams is one of the interaction models that depict how the system will interact and which services it provide to people or systems outside its boundaries, it is a good way for requirements elicitation the figure is the use case diagram of our system.

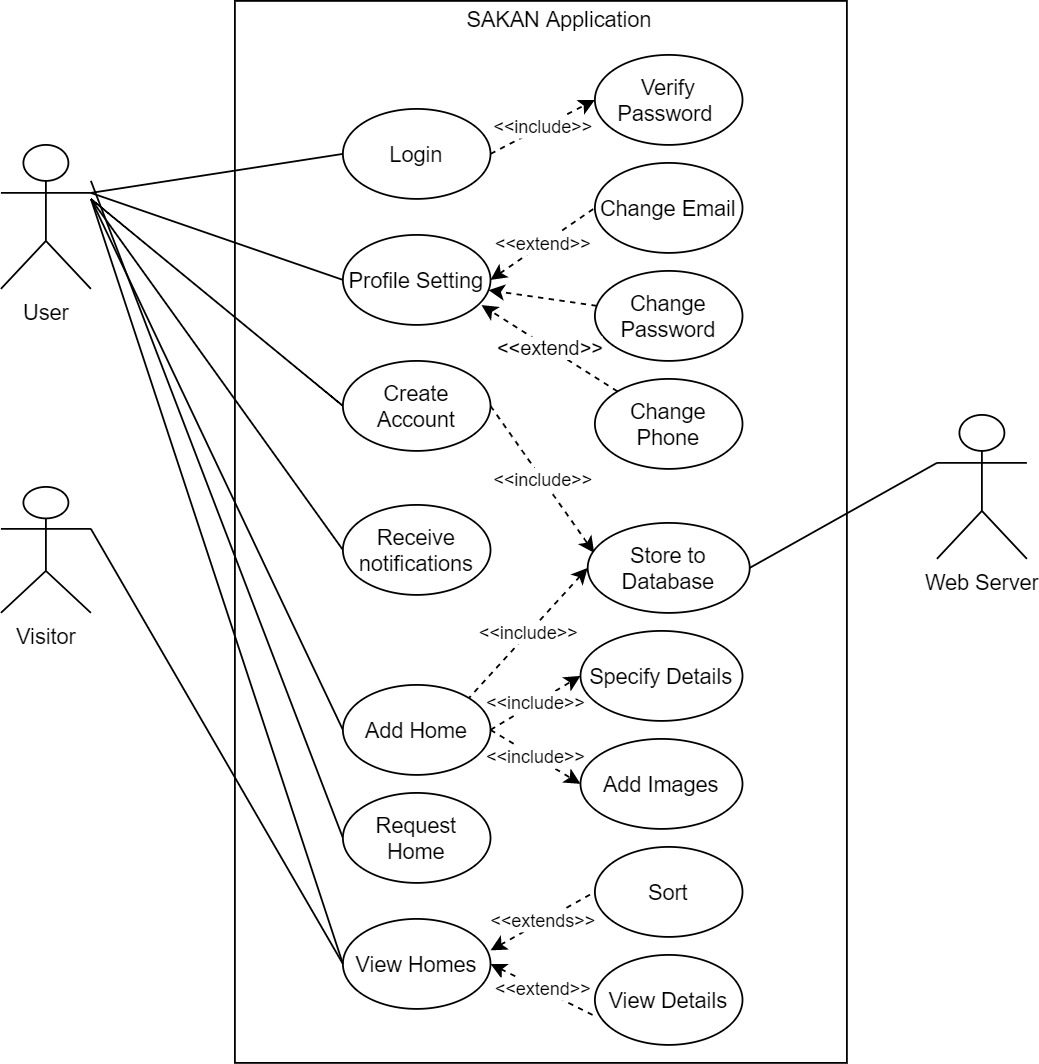


Figure 2.2 Use Case Diagram

In this diagram we represent system tasks by an ovals which interact directly with actors by an association relationship, some tasks interacts with each other include relationship which show that the behavior of the included use case is part of the including (base) use case, or extend relationship which extends the base use case and adds more functionality to the system.

In our project, *Login* use case show that the user can get more feature if he login with valid username and password, this use case include implicitly password verification process, to be able to login, the user has to *create account* by specify personal information which implicitly include the data on a remote database, once the user created account and login successfully he can *add home* for advertisement and specify its details and upload images about this home, all home specifications will be stored to database, and he can *view homes* in a list and sort this list according to price or the required gender or number of rooms, the user can *request home* through the application and can receive a notifications if someone requested his home or new home added to the application. There another actor for the system who is the visitor who can enter the application with the need for create account and login but he can only see the advertised homes, a third actor is the web server which we consider as a separate system that contain the database and manage the data retrieval process.

**2.3 Functional Requirements**

The electronic leasing application will provide the user with the ability to perform some services in an organized and easy way without exerting much effort as soon as possible.

* **Sign in**: the user if he has an account (email and password) he’s can sign in directly but if not he’s need to create an account to can sign in to application.
* **Create an user account**: if he has face book or Gmail can connected with the application if no he can create throw create password and email and personal information’s and then he can sign in .

* **User added home**: the user can add a home to Leasing It throw Check box views the information to and fall this information and submit to show and leasing it.
* **Leasing a home**: the user can Leasing a home by application after viewed the home to Tenant.
* **User rent a home**: the user that want rent a home he can see all homes and rent what he wants.
* **User watch:** any user he can watch the houses shown if he can an owner or tenant from viewer list.
* **the user want Rent a home He’s can sort it**: when the tenant he rent a home he can sort the homes according specification it like the number of room or price or gender to see what he want quickly.
* **Change profile settings**: if a user he want change he’s information he can do it.
* **Notifications user**: the user receive a notification if he was tenant receive every added a new home and the owner he receive if any one rent the
* **Sign out**: the user sign out from the application.

**2.4 Non-Functional Requirements**

These requirements describe the main properties of the software such as reliability, security, availability, etc. it is a critical part of the requirements elicitation process and must be tested carefully, neglecting it may cause the system to be valueless, the following are non-functional requirements show be in the software:

* **Performance requirements:**

The system must respond the operation in less than 3 seconds for user, to increase the performance and resources utilization will use the multithreading in java code especially in the network connections.

* **Portability**

We will implement this application on API 16 which means that the application will run on 99.8% android devices.

* **Usability**

The system has user-friendly interface, organized in such a way that reduce the user errors.

* **Availability**

The system must be available 24 hours daily in week Downtime within normal working hours shall not exceed five seconds in any one day.

* **Security**

The password for the users will be encrypted before storing in the database using the *md5* which is non-reversible hashing algorithm.

**Chapter Three: Realization and Implementation**

**3.1 System Architecture:**

In early stage of the software development we have to design the system architecture to define the major components of the system and the communication between these components, the following diagram show our system components as a block linked together:

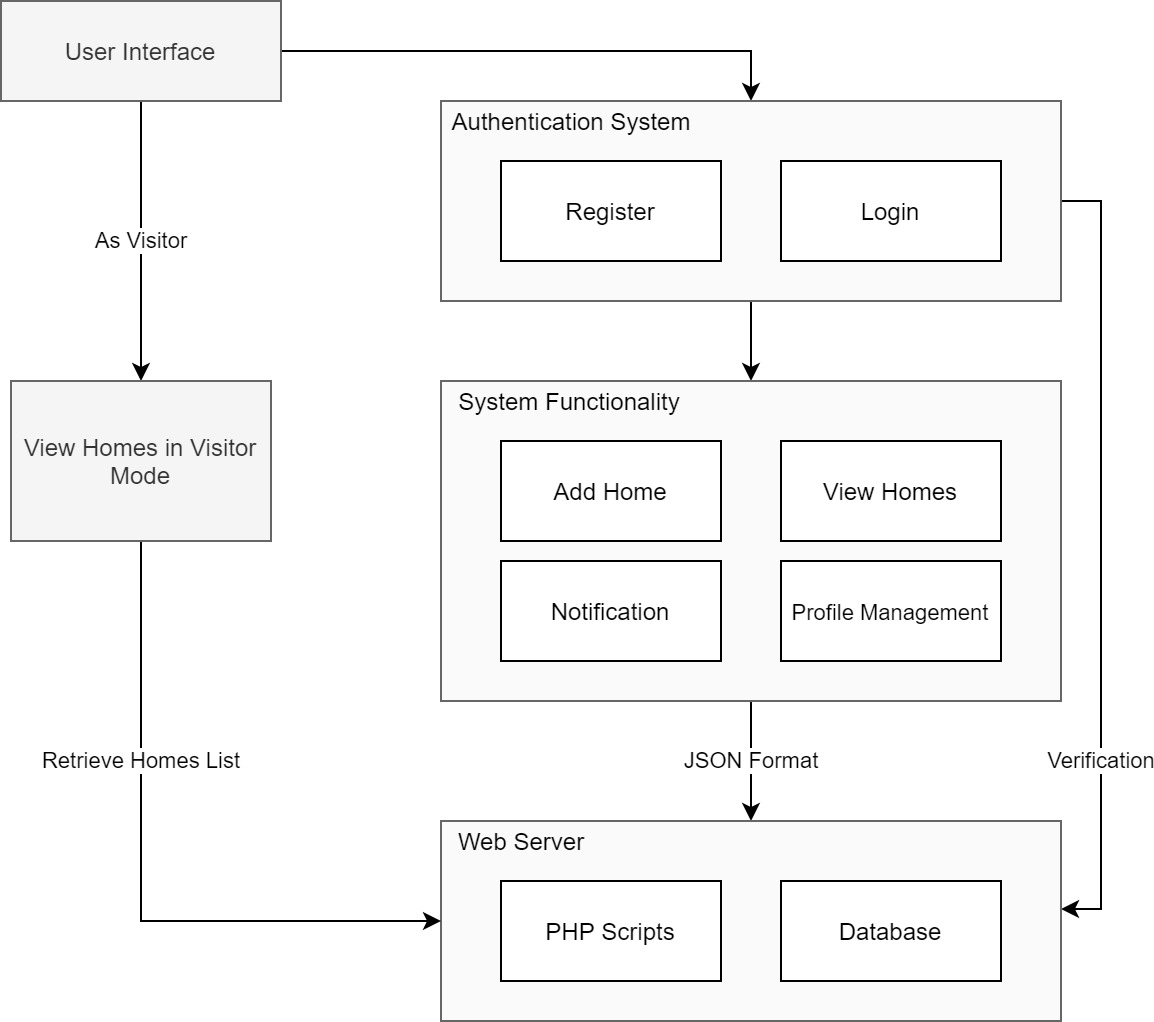


Figure 3.1 System Architecture

Our system consist several components such as user interfaces which the user actually interact with to get the system services and the authentication component which consist of the registration part to store required personal information and login interface that responsible about user verification process, however our system allow users to access the software in visitor mode which restricted mode allow the user to see list of available homes, we have grouped all the system functionality in one component which provide the user the ability to add home, view homes, receive notification, provide a profile for each user which he an set its settings, the last component of our system is the web server which consist of MySQL database and the PHP conde which manage all database queries and transactions.

**3.2 Layered Architecture:**

It is a model that describe the system as a set of layers, each layer provide a several services to the adjacent layers, this pattern show how these layered are interface and related to each other and illustrate the incremental software as if we decide to change on layer, only the adjacent layers will be affected.

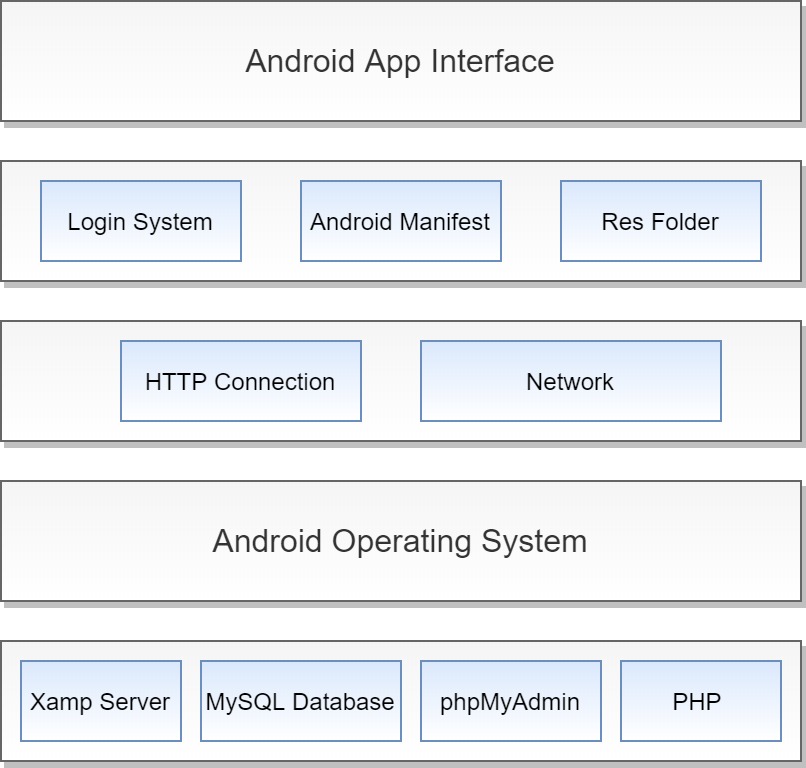
****

Figure .2 Layered Architecture

The figure represent a layered architecture diagram which include the layers of our system, first layer is the user interfaces layer which represent all of the application interfaces that we have implemented in XML language for each activity in the system, the next layer concerned with user interface management and providing user authentication and authorization through the login system, in this layer, the interfaces will be linked with its resources such as colors, strings, images which exist in the resources folder. The manifest file in this layer defines the structure and metadata of our application, its components, and its requirements [2]. The third layer we will use the classes and libraries that perform the internet connection task to communicate with web server such as HttpURLConnection class. We have put the operating system in the fourth layer because it manage all the above services. Finally we have the remote data storage layer which is Wamp server manage the MySQL database through phpMyAdmin interface and link backend PHP codes.

**3.3 Class Diagram**

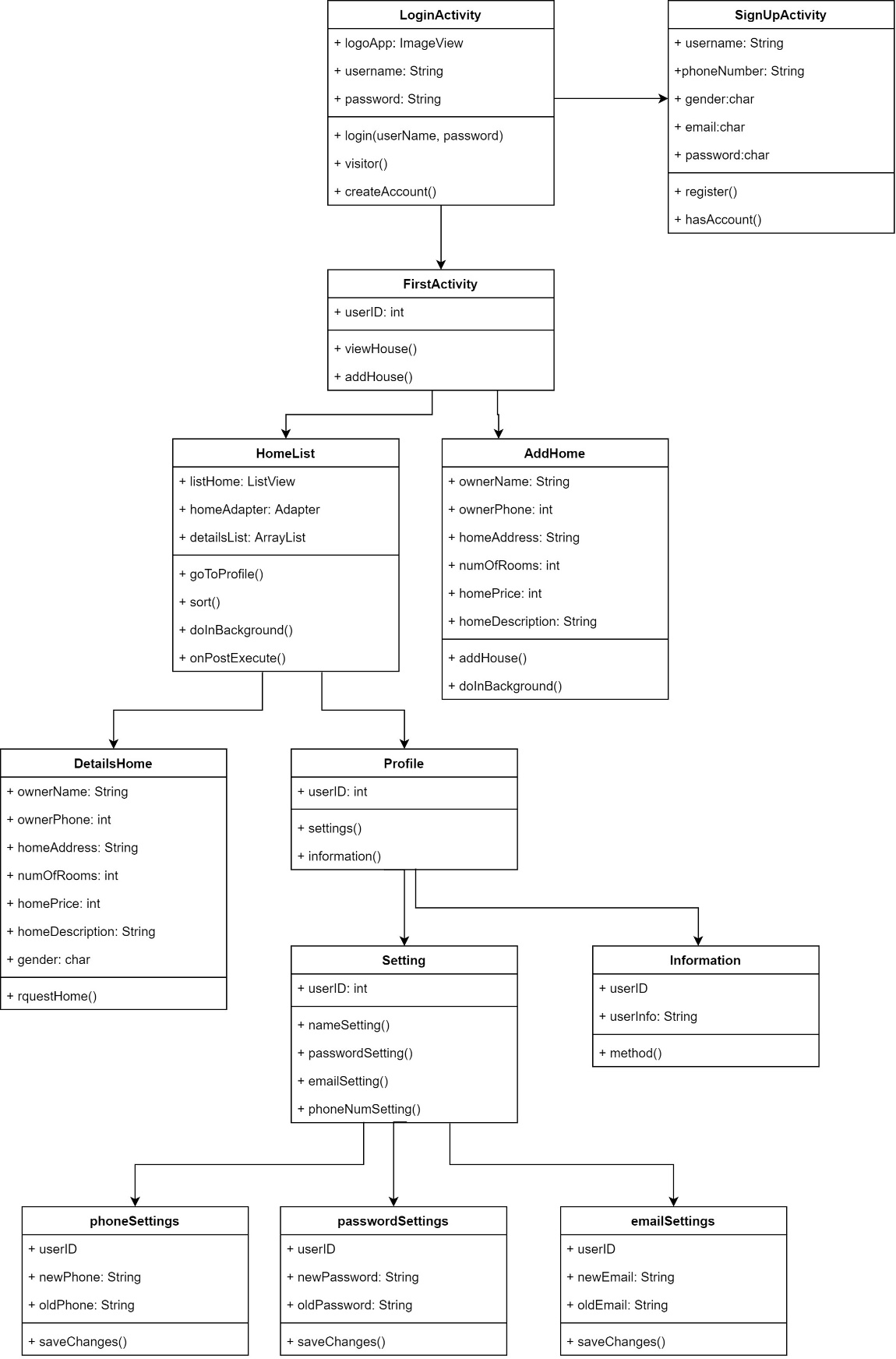
****

Figure 3.3 Class Diagram

**3.4 Sequence Diagram**

**Login Sequence Diagram**

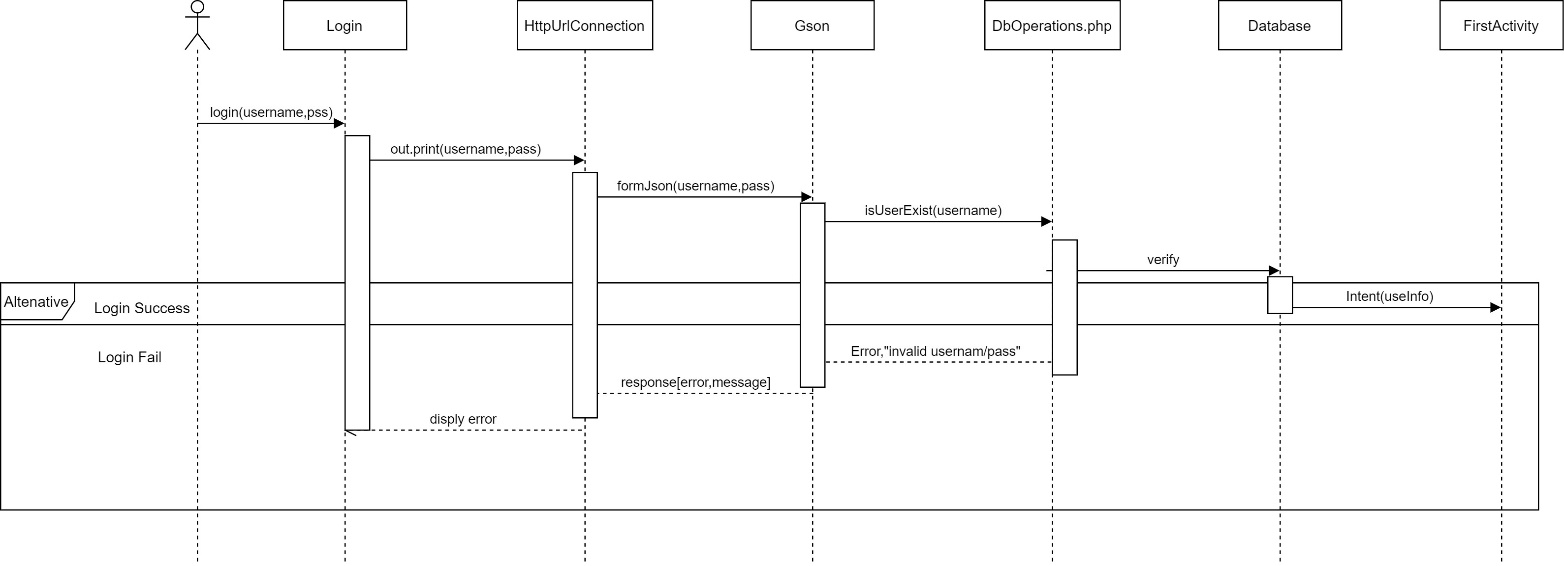
****

Figure 3.4.1 Login Sequence Diagram

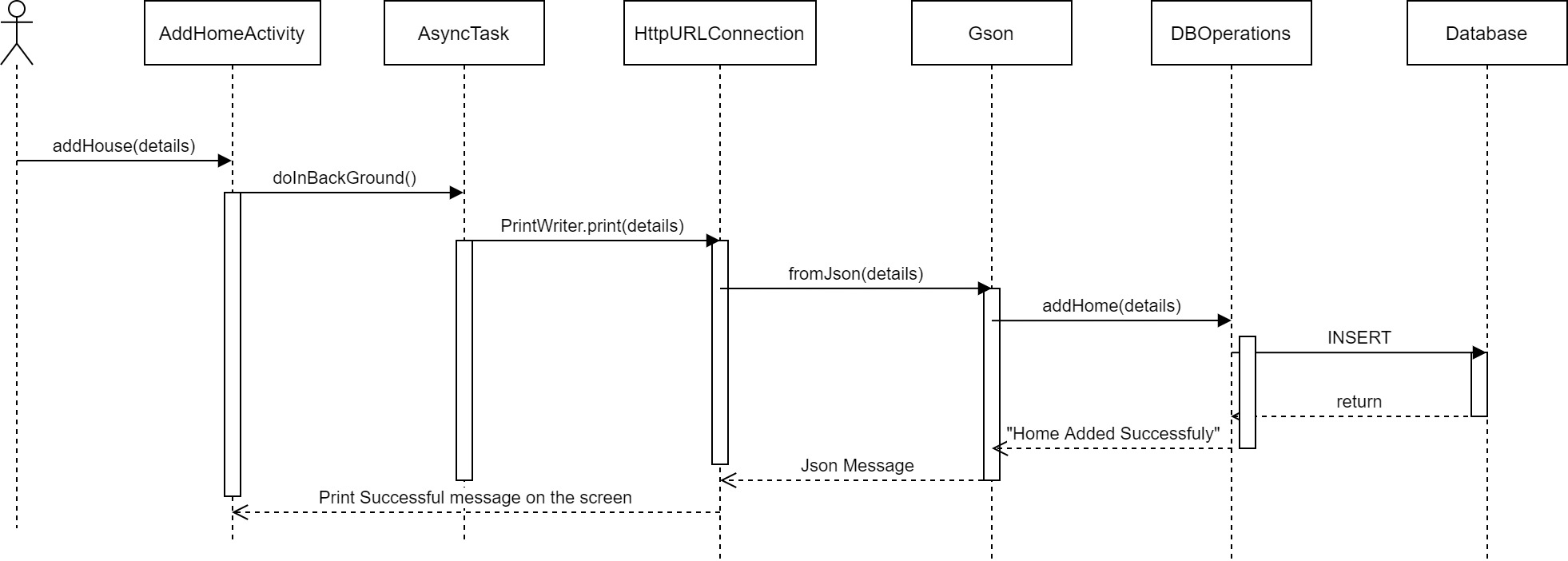
****

Figure 3.4.2 Add Home Sequence Diagram

**3.5 Entity-Relationship Diagram:**

To design the database of the system we have to create a graphical notation that describe all tables we need and it attributes and the relationship between these tables, figure show the ER diagram of our project, before doing normalization we see that we need a table user information such as name, gender, etc. and has a *Phone* attribute as a multi-valued attribute to store the phone number of the user and we must identify a *User\_ID* as a primary key to uniquely identify each user.

Also, we need a *Home* table to all needed information about the homes such as address, number of rooms, etc. and we assigned an *Image* attribute to the table to visually describe the home and it will be a multivalued attribute

The relationship between the two table is *Has* relationship which is one-to-many relationship because each user can more than one home, and the user participate partially in the relationship and home has full participation in the relationship.

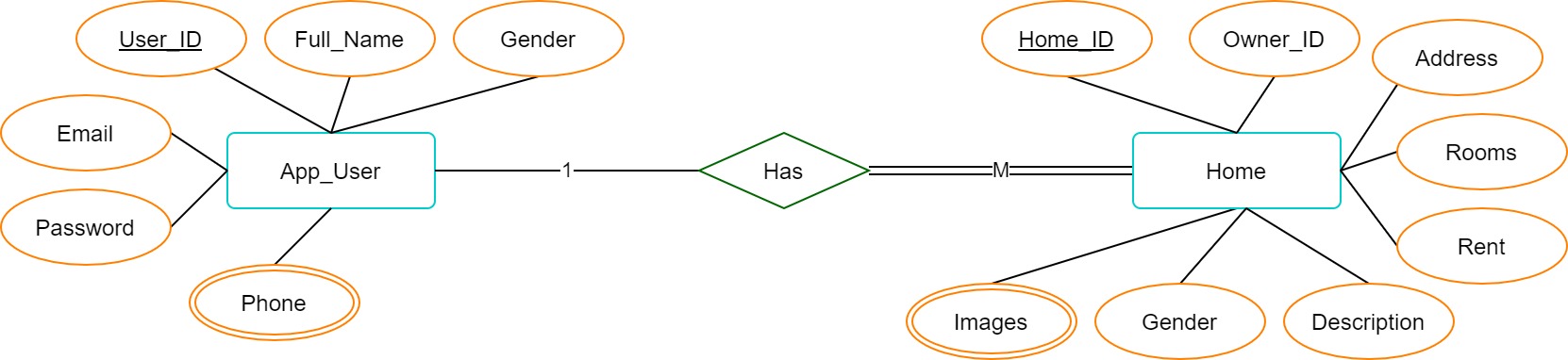
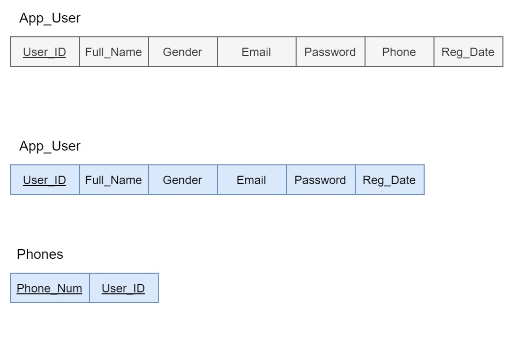


Figure 3.3 ER Diagram

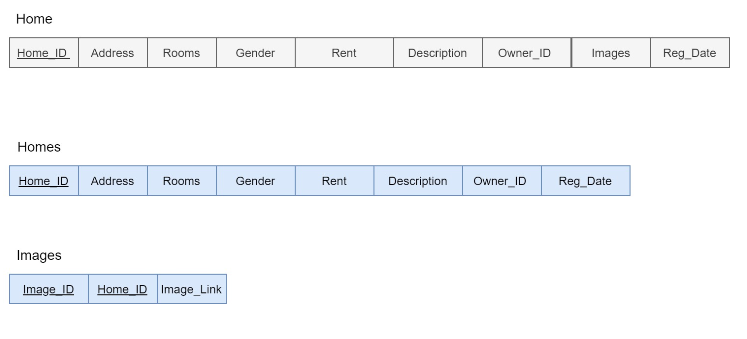
**3.4 Database Normalization:**

In this stage of the database design we will enhance the structure of the database to avoid data anomalies such as insertion, deletion, and modification anomalies to do this check each table in accordance to which known normal forms

The table App\_User is not the first normal form because it has *Phone* which is multi-valued attribute, so we will make a distinct table contain *Phone\_Num* and *User\_ID* as a composite primary key, figure show this step.



The table *Homes* is not the first normal form because it has *Images* which is multi-valued attribute, so we will make a distinct table contain *Images* and *Home\_ID* as a composite primary key, and add a link attribute to describe the URL cloud storage of the image, figure show this step.



The resulted for tables in the first normal form, also they are in the second normal form since there is no partial dependencies, and in the third normal form since there is no transitive dependencies, so we have four tables ready to implement on phpMyAdmin, figure show the relational model and how the tables will be related to each other.

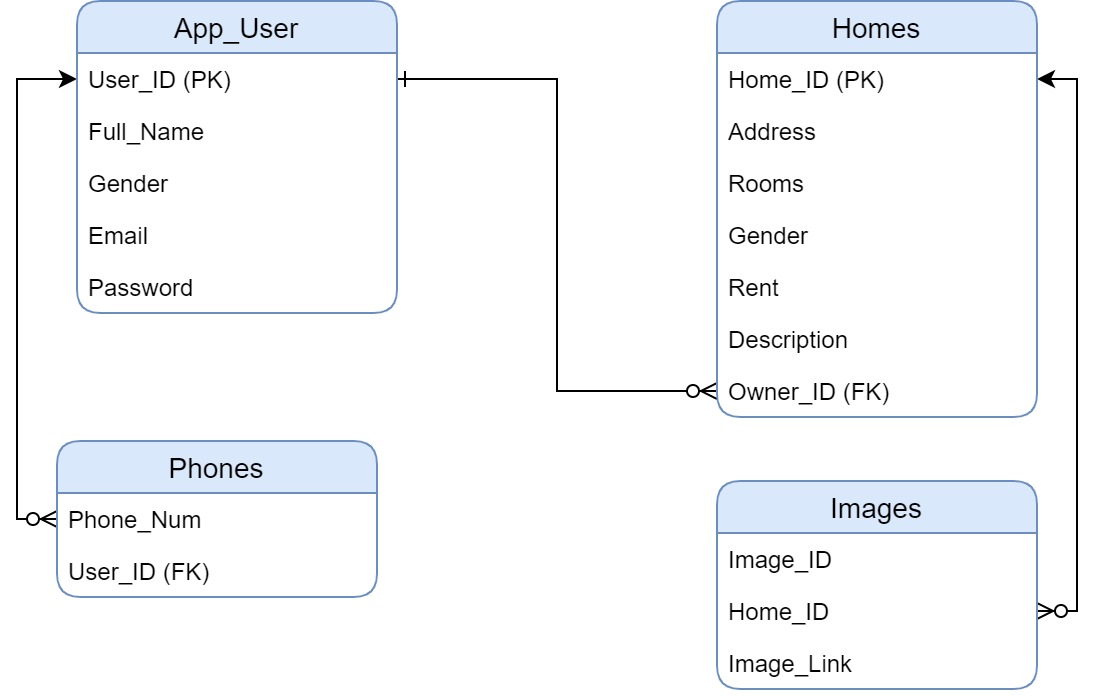


Figure 3.4 Relational Model

**3.5 Software Evolution**

When we started building our project, it was built on the web, but after a consultation with Dr. Yacoub, we decided to build it in the form of an Android application. We realized that if we build it using Android, it will be better, more efficient, easier, and use in the medium surrounding us.

So this is our first Android app or let's say that our first code was written to create an app, so we made a lot of effort to reach this point as it took us a lot of time to build it and to modify and fix the problems that we faced. I hope you appreciate our work and give us good feedback to give you more in the future

We got some mistakes during the construction of the project, so we had to change many parts of it. The reason for these mistakes was the lack of experience and the current circumstances that forced us to work remotely and we had started to start us

Part after another, and we installed these parts as the first part was building system interfaces, then building

Database on the web is a SQL browser on the PHP language and then we have implemented the Java code and the backend and then we have completed the report based on the project data, but we faced some problems that prevented the project from running.

So we made some changes and those changes were that we modified the data base and deleted some features from the project such as the messaging feature inside the project and we changed the server from Wamp to XAMPP and we changed the PHP code from OOP to PDO but we also faced some problems in the Update and the delete These are some errors That we could not solve but this is what happened, and this is what happened, and we have done everything in our power to reach this point, so please accept our work as it is.

**Chapter Four: testing and results**

**4.1 Unit testing**

It is the process of testing individual components in isolation, very line we wrote we checked it directly, the Android studio IDE helped us to do this testing by showing error messages when there is a syntax error, while writing PHP code, the Atom IDE helps us to detect each syntax error and we have used *echo* command to see the output of each code section.

**4.2 Component Testing**

In this testing, several individual units are integrated together to test it as a whole component, we applied this in interfaces design when we combine several screen elements in XML code and run the program to see how it looks on the screen, also when creating the PHP-database connection and executing at least one operation then we need to link files together and test the result.

**4.3 System Testing**

This done when all the system components are integrated and tested as a whole system, it focus on the interaction between the components, we have checked the program after we finished programming it to test the interaction between java and PHP and SQL codes and there is several problems.

**4.4 Testing results**

As a result for unit testing there is no problems appear, the problems appeared when doing component testing for design when run the application on different screen sizes, this problem solved by using linear layout, another problem appeared when doing connection the database using JDBC driver this problem we cant solve because the android studio wasn’t support the driver version (OJDBC6) so we convert the application to connect using web services and PHP, after doing system testing the web-based application we had a lot of errors so we performed some changes to the PHP code, finally some features of the application work properly such adding home and viewing homes and there is problems with other parts such as login and profile settings.

**Chapter Five: conclusion**

In conclusion, we want to talk about our project and that we have overcome all the problems that we faced with having some things that we could not accomplish and we hope the project will meet the requirements of it in the labor market and that we will keep pace with the needs of the labor market with thanks and appreciation.

**5.1 References:**

[1]Getproofed <https://getproofed.com/writing-tips/cite-website-ieee-referencing/>

[2]GeeksforGeeks <https://www.geeksforgeeks.org/application-manifest-file-android/>

**Chapter Six: Appendices**

**6.1 Views**

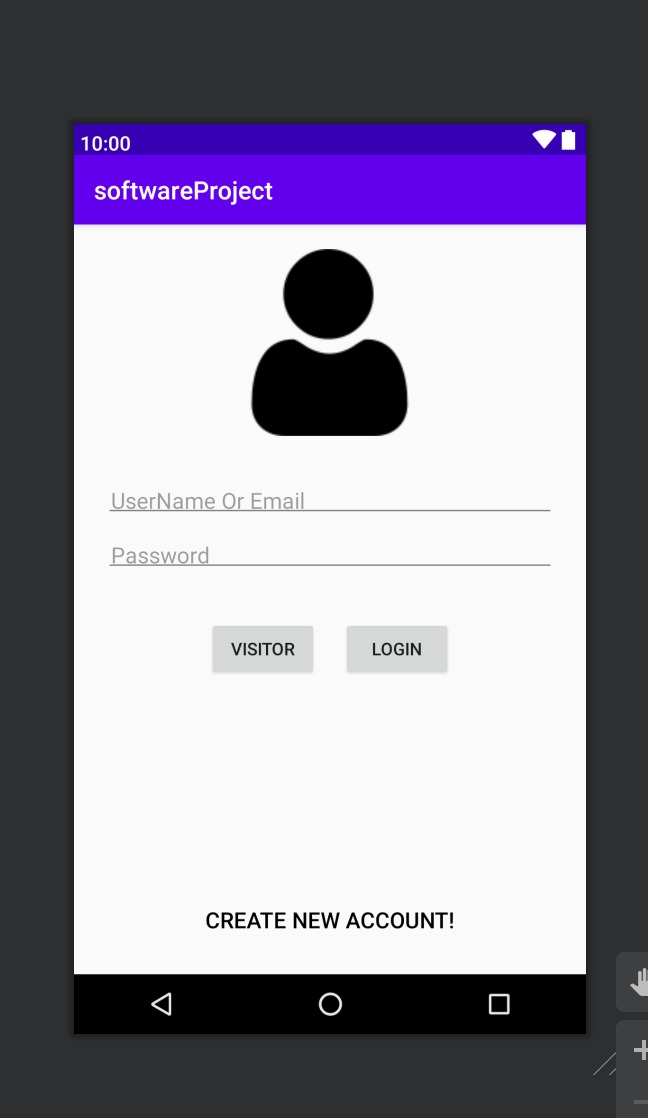


Figure .1.1 Login Interface

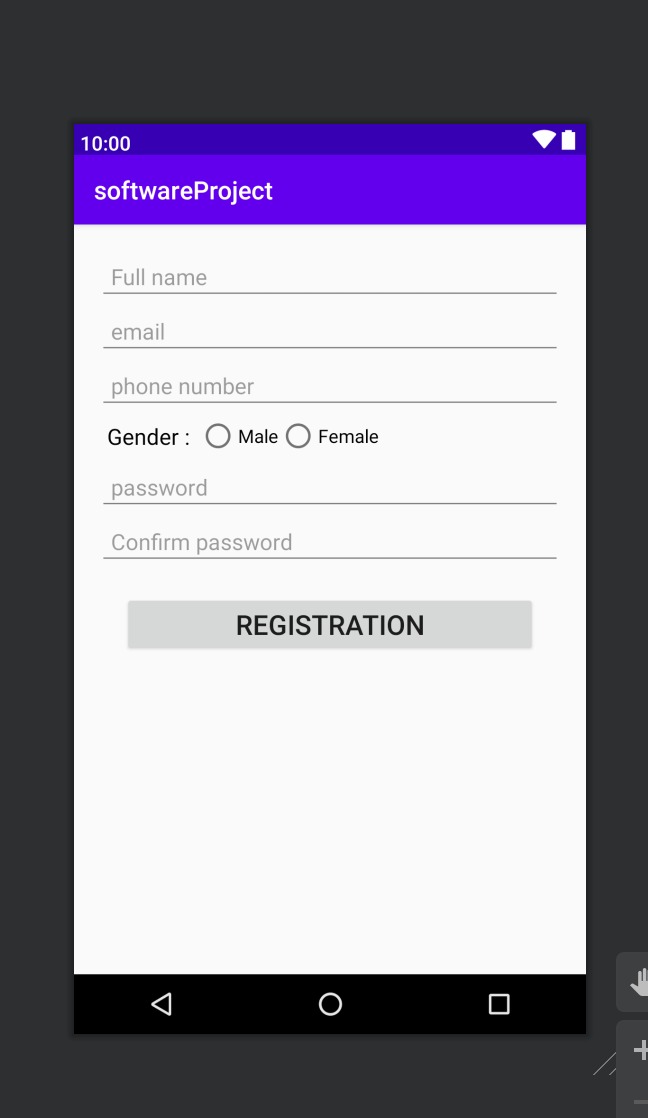


Figure 6.1.2 Sign up Interface

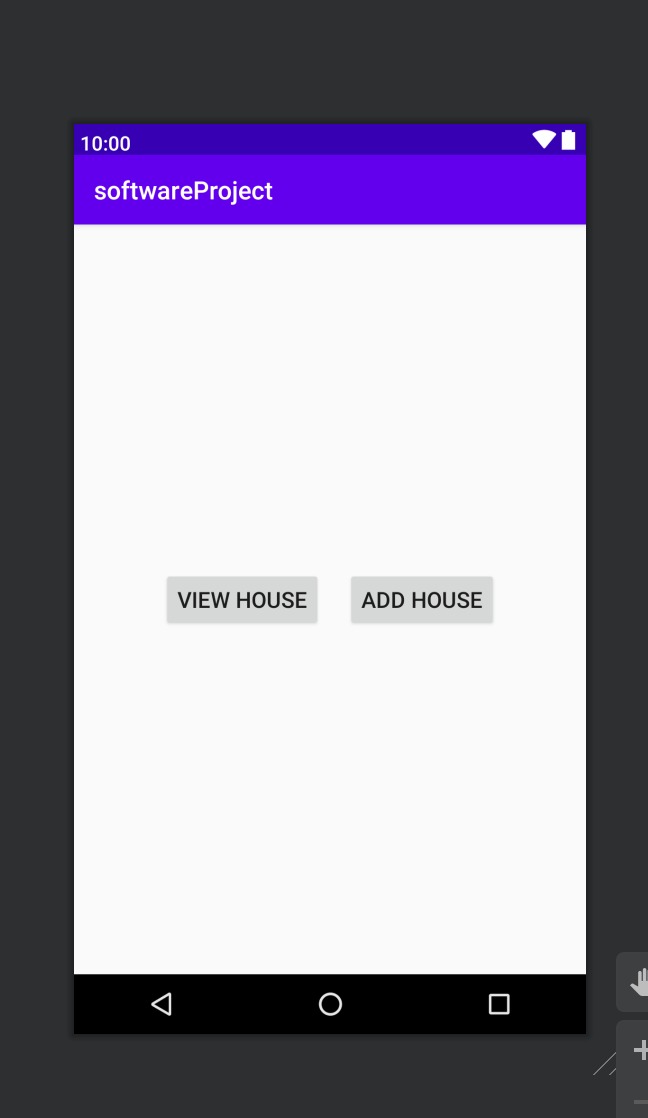


Figure 6.1.3 Main Page

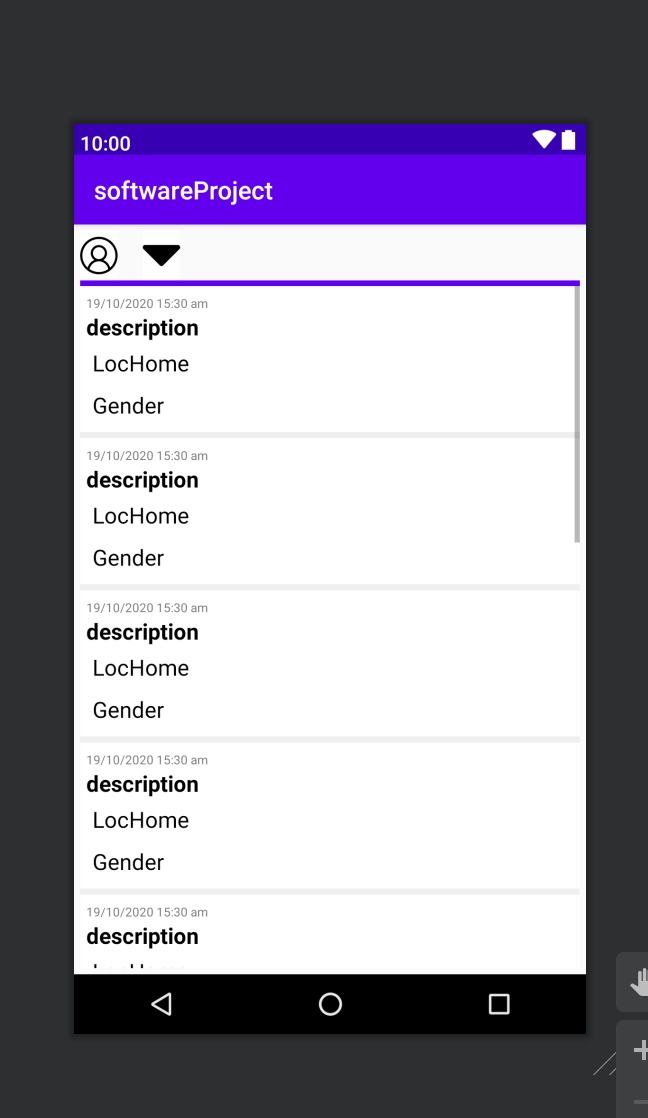


Figure 6.1.4 Homes List

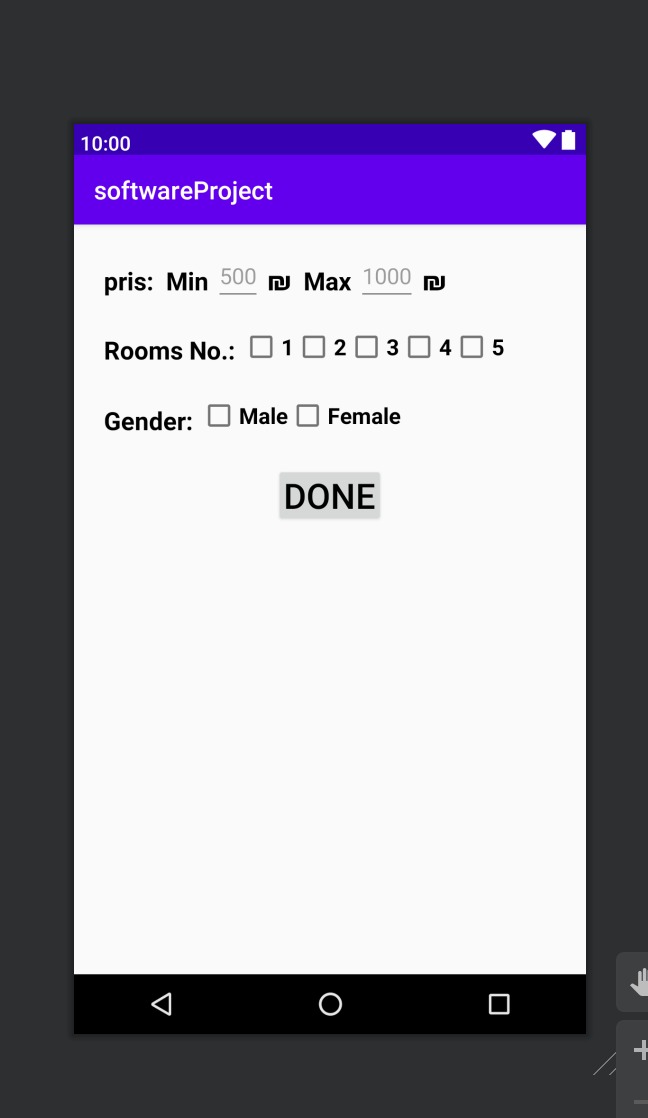


Figure 6.1.5 Sort Options

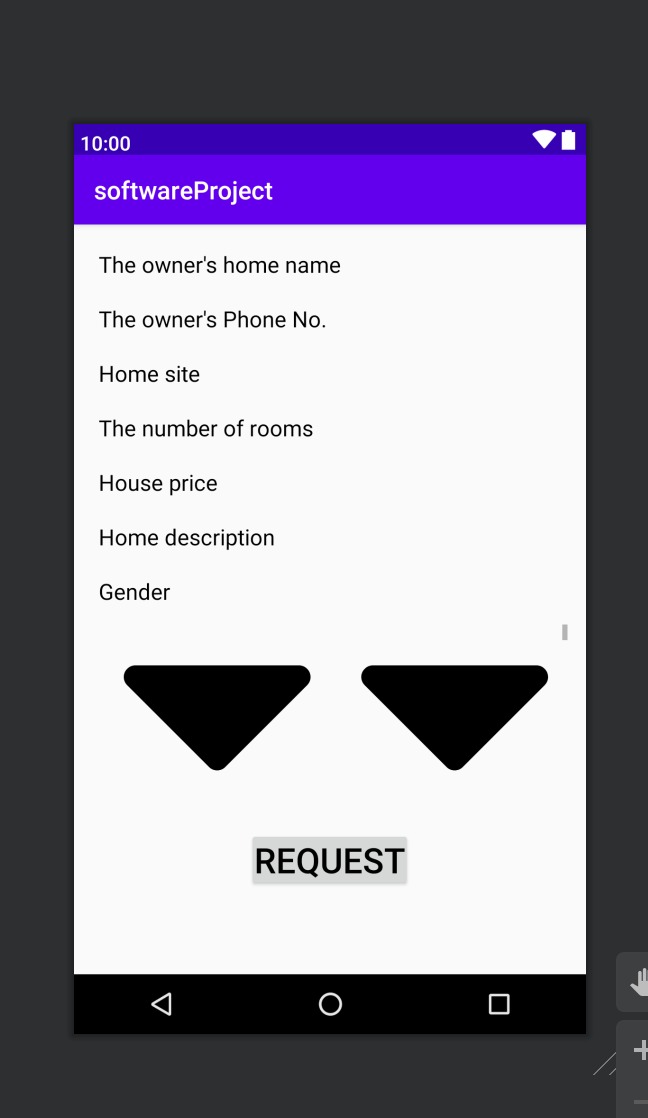


Figure 6.1.6 Home details

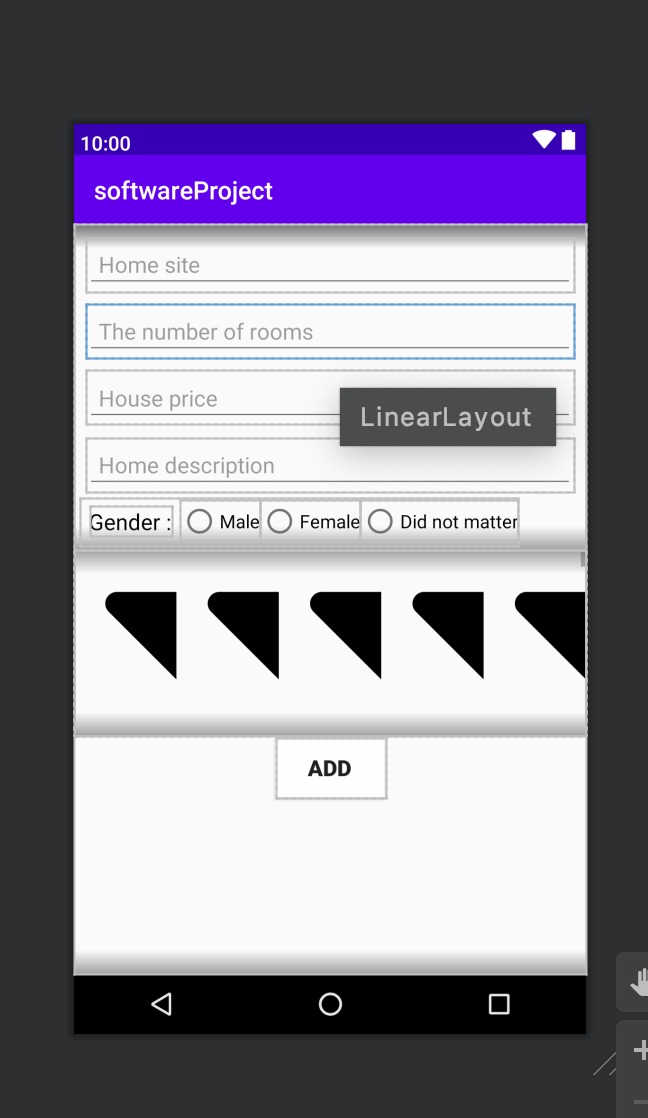


Figure 6.1.7 Insert Home Details

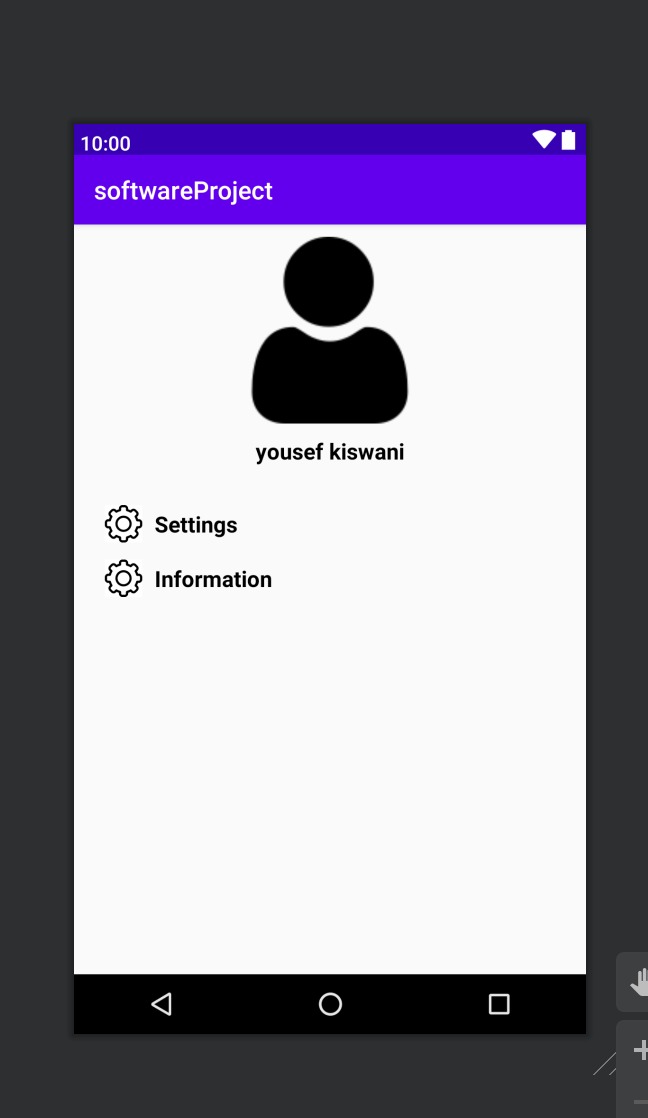


Figure 6.1.8 User Profile

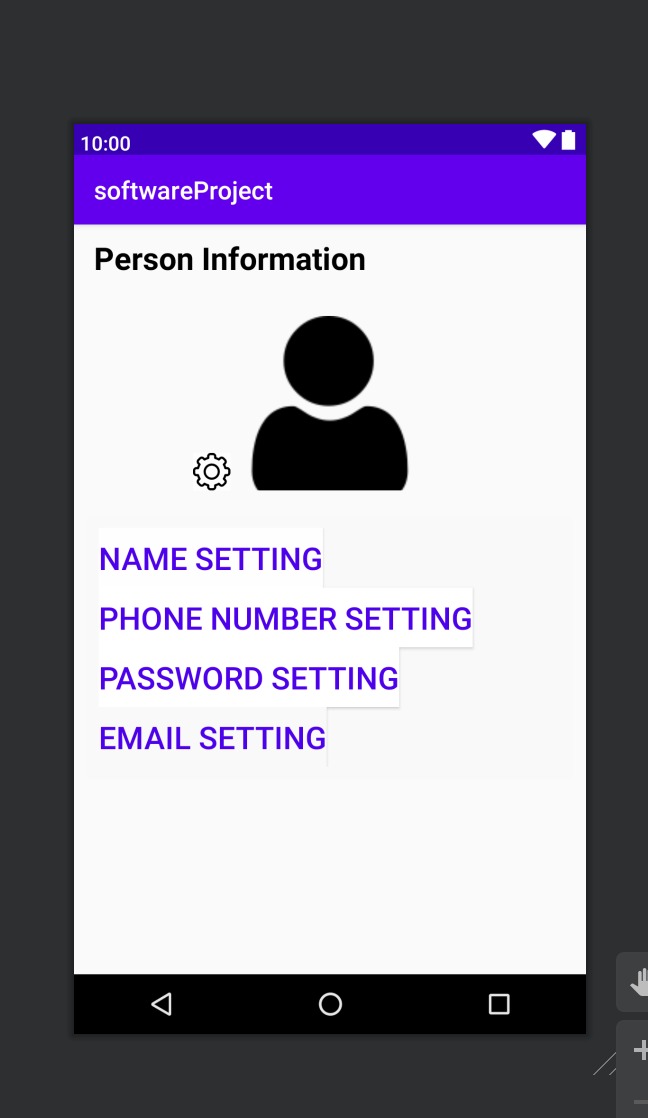


Figure 6.1.9 Profile Settings

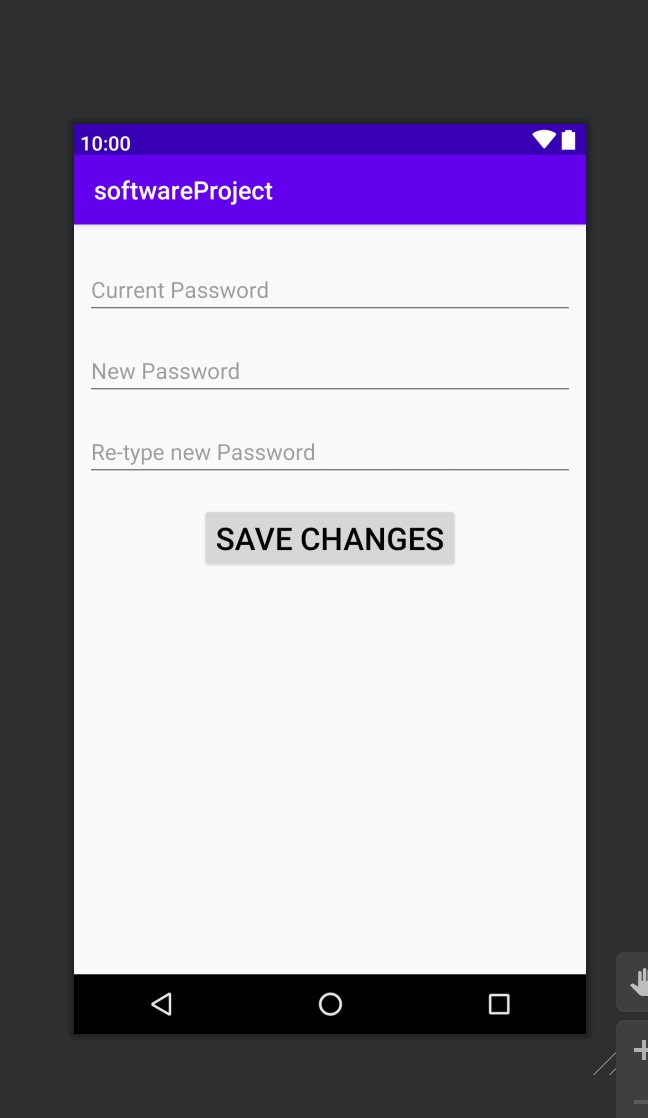


Figure 6.1.10 Change Password

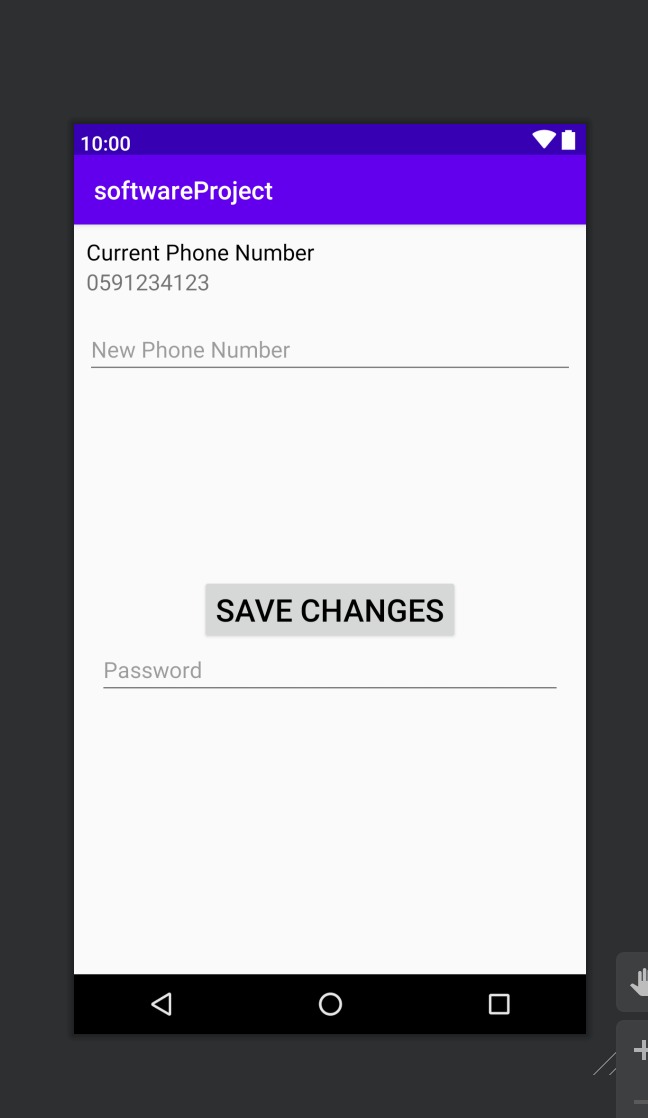


Figure 6.1.11 Change Phone Number

**6.1 Code:**

*Login Code:*

**public class** LoginActivity **extends** AppCompatActivity {  
  
 ImageView **logoAppIV**;  
 **private** EditText **userNameET**, **userPasswordET**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_login***);  
  
 **if**(SharedPrefManager.*getInstance*(**this**).isLoggedIn()){  
 finish();  
 startActivity(**new** Intent(**this**, FirstActivity.**class**));  
 **return**;  
 }  
  
 **userNameET** = (EditText)findViewById(R.id.***userName\_et***);  
 **userPasswordET** = (EditText)findViewById(R.id.***userPassword\_et***);  
  
  
 }  
  
 **public void** Login(View view) {  
  
 **final** String username = **userNameET**.getText().toString();  
 **final** String password = **userPasswordET**.getText().toString();  
  
 StringRequest stringRequest = **new** StringRequest(  
 Request.Method.***POST***,  
 Constants.***URL\_LOGIN***,  
 **new** Response.Listener<String>() {  
 @Override  
 **public void** onResponse(String response) {  
 **try** {  
 JSONObject obj = **new** JSONObject(response);  
 **if**(!obj.getBoolean(**"error"**)){  
 SharedPrefManager.*getInstance*(getApplicationContext())  
 .userLogin(  
 obj.getInt(**"id"**),  
 obj.getString(**"username"**),  
 obj.getString(**"email"**)  
 );  
 startActivity(**new** Intent(getApplicationContext(), FirstActivity.**class**));  
 finish();  
 }**else**{  
 Toast.*makeText*(  
 getApplicationContext(),  
 obj.getString(**"message"**),  
 Toast.***LENGTH\_LONG*** ).show();  
 }  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 },  
 **new** Response.ErrorListener() {  
 @Override  
 **public void** onErrorResponse(VolleyError error) {  
  
 Toast.*makeText*(  
 getApplicationContext(),  
 error.getMessage(),  
 Toast.***LENGTH\_LONG*** ).show();  
 }  
 }  
 ){  
 @Override  
 **protected** Map<String, String> getParams() **throws** AuthFailureError {  
 Map<String, String> params = **new** HashMap<>();  
 params.put(**"username"**, username);  
 params.put(**"password"**, password);  
 **return** params;  
 }  
  
 };  
  
 RequestHandler.*getInstance*(**this**).addToRequestQueue(stringRequest);  
 }  
  
  
 **public void** GoToSignUp(View view) {  
  
 startActivity(**new** Intent(getApplicationContext(), SignUpActivity.**class**));  
  
 }  
}

*Add Home Code:*

**public class** AddHomesActivity **extends** AppCompatActivity {  
 **private** EditText **addressET**, **numberOfRoomsET**,**housePriceET**,**homeDescriptionET**;  
 **private** RadioButton **maleRB**, **femaleRB**;  
 GridView **photoListHomeGV**;  
 ImageView **image1IV**, **image2IV**, **image3IV**;  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_add\_homes***);  
  
 **image1IV** = findViewById(R.id.***imageView3***);  
 **image2IV** = findViewById(R.id.***imageView2***);  
 **image3IV** = findViewById(R.id.***imageView***);  
  
  
 **addressET** = findViewById(R.id.***address\_et***);  
 **numberOfRoomsET** = findViewById(R.id.***numberOfRooms\_et***);  
 **housePriceET** = findViewById(R.id.***housePrice\_et***);  
 **homeDescriptionET** = findViewById(R.id.***homeDescription\_et***);  
 **photoListHomeGV** = findViewById(R.id.***photoListHome\_gv***);  
 **maleRB** = (RadioButton) findViewById(R.id.***genderHomeMale\_rb***);  
 **femaleRB** = (RadioButton) findViewById(R.id.***genderHomeFemale\_rb***);  
  
  
  
  
 }  
  
 **public** String getGender(){  
 **if**(**maleRB**.isChecked())  
 **return "m"**;  
 **else if** (**femaleRB**.isChecked())  
 **return "f"**;  
 **else  
 return "d"**;  
 }  
  
  
  
 **public void** add\_house\_btn(View view) {  
  
 **final** String address = **addressET**.getText().toString();  
 **final** String rooms = **numberOfRoomsET**.getText().toString();  
 **final** String gender = getGender();  
 **final** String rent = **housePriceET**.getText().toString();  
 **final** String description = **homeDescriptionET**.getText().toString();  
  
 **final** String currentUserId = **""**+SharedPrefManager.*getInstance*(**this**).getUserId();  
  
  
  
  
  
 StringRequest stringRequest = **new** StringRequest(Request.Method.***POST***, Constants.***URL\_ADD\_HOME***,  
 **new** Response.Listener<String>() {  
 @Override  
 **public void** onResponse(String response) {  
  
 **try** {  
 JSONObject jsonObject = **new** JSONObject(response);  
  
 Toast.*makeText*(getApplicationContext(), jsonObject.getString(**"message"**), Toast.***LENGTH\_LONG***).show();  
  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 },  
 **new** Response.ErrorListener() {  
 @Override  
 **public void** onErrorResponse(VolleyError error) {  
  
 Toast.*makeText*(getApplicationContext(), error.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 }) {  
 @Override  
 **protected** Map<String, String> getParams() **throws** AuthFailureError {  
 Map<String, String> params = **new** HashMap<>();  
 params.put(**"address"**, address);  
 params.put(**"rooms"**, rooms);  
 params.put(**"gender"**, gender);  
 params.put(**"rent"**, rent);  
 params.put(**"description"**, description );  
 params.put(**"ownerid"**, currentUserId);  
 **return** params;  
 }  
 };  
  
  
 **if**(address.isEmpty()||rooms.isEmpty()||rent.isEmpty()||description.isEmpty()||gender.isEmpty())  
 Toast.*makeText*(getApplicationContext(), **"Some Fields Are Empty "**, Toast.***LENGTH\_LONG***).show();  
 **else** {  
 RequestHandler.*getInstance*(**this**).addToRequestQueue(stringRequest);  
 }  
  
  
  
 }  
  
  
 **public void** choosePhoto(View view) {  
  
 Intent intent = **new** Intent(Intent.***ACTION\_PICK***, MediaStore.Images.Media.***INTERNAL\_CONTENT\_URI***);  
 startActivityForResult(intent, 100);  
  
 }  
  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode, @Nullable Intent data) {  
 **super**.onActivityResult(requestCode, resultCode, data);  
  
 **if**(requestCode == 100 && resultCode == ***RESULT\_OK***){  
  
 Uri uri = data.getData();  
 **image1IV**.setImageURI(uri);  
   
  
 }  
 }  
}

*SQL Code:*

CREATE TABLE app\_user (

User\_ID INT(5) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

Full\_Name VARCHAR(30) NOT NULL,

Gender char(1),

phone\_num INT(20),

email VARCHAR(50),

pass\_word VARCHAR(30),

reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

)

CREATE TABLE Homes (

Home\_ID INT(5) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

Address VARCHAR(70) NOT NULL,

Rooms INT(2) NOT NULL,

Gender char(1),

Rent int(4),

Description VARCHAR(100),

Owner\_ID INT(5),

reg\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (Owner\_ID) REFERENCES app\_user(User\_ID)

)

CREATE TABLE Phones (

Phone\_Num INT(10) ,

User\_ID INT(5),

PRIMARY KEY (Phone\_Num, User\_ID),

FOREIGN KEY (User\_ID) REFERENCES app\_user(User\_ID)

)

CREATE TABLE Images (

Image\_ID INT(10) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

Image\_Link VARCHAR(100) NOT NULL,

Home\_ID INT(5) NOT NULL,

FOREIGN KEY (Home\_ID) REFERENCES homes(Home\_ID)

)

*PHP DBoperations Code:*

<?php

class DbOperations{

private $con;

function \_\_construct(){

require\_once dirname(\_\_FILE\_\_).'/DbConnect.php';

$db = new DbConnect();

$this->con = $db->connect();

}

/\*CRUD -> C -> CREATE \*/

public function createUser($username, $pass, $email, $phone, $gender){

if($this->isUserExist($username,$email)){

return 0;

}else{

$password = md5($pass);

$stmt = $this->con->prepare("INSERT INTO `users` (`id`, `username`, `password`, `email`, phone\_no, gender) VALUES (NULL, ?, ?, ?, ?, ?);");

$stmt->bind\_param("sssis",$username,$password,$email, $phone, $gender);

if($stmt->execute()){

return 1;

}else{

return 2;

}

}

}

public function userLogin($username, $pass){

$password = md5($pass);

$stmt = $this->con->prepare("SELECT id FROM users WHERE username = ? AND password = ?");

$stmt->bind\_param("ss",$username,$password);

$stmt->execute();

$stmt->store\_result();

return $stmt->num\_rows > 0;

}

public function getUserByUsername($username){

$stmt = $this->con->prepare("SELECT \* FROM users WHERE username = ?");

$stmt->bind\_param("s",$username);

$stmt->execute();

return $stmt->get\_result()->fetch\_assoc();

}

private function isUserExist($username, $email){

$stmt = $this->con->prepare("SELECT id FROM users WHERE username = ? OR email = ?");

$stmt->bind\_param("ss", $username, $email);

$stmt->execute();

$stmt->store\_result();

return $stmt->num\_rows > 0;

}

public function addHome($Address, $Rooms, $Gender, $Rent, $Description, $Owner\_ID){

$stmt = $this->con->prepare("INSERT INTO `homes` (`Home\_ID`, `Address`, `Rooms`, `Gender`, `Rent`, `Description`, `Owner\_ID`, `reg\_date`)

VALUES (NULL, ?, ?, ?, ?, ?, ?, current\_timestamp());");

$stmt->bind\_param("sisisi",$Address,$Rooms,$Gender,$Rent,$Description,$Owner\_ID);

if($stmt->execute()){

return 1;

}else{

return 2;

}

}

public function showHomes(){

$stmt = $this->con->query("SELECT \* FROM homes;");

$homeArray = array();

while($row = $stmt->fetch\_assoc())

{

$homeArray[] = $row;

}

return $homeArray;

}

public function reserve($Home\_ID){

$stmt = $this->con->prepare("UPDATE homes SET is\_reserved = 'y' WHERE Home\_ID = ? ");

$stmt->bind\_param("s",$Home\_ID);

if($stmt->execute()){

return 1;

}else{

return 2;

}

}

public function addPhone($userID, $phone){

$stmt = $this->con->prepare("INSERT INTO `phones` (`Phone\_Num`, `User\_ID`) VALUES (?, ?); ");

$stmt->bind\_param("ii",$phone, $userID );

if($stmt->execute()){

return 1;

}else{

return 2;

}

}

public function showSpecifiedHomes($minRent, $maxRent, $numRooms, $requiredGender){

$stmt = $this->con->prepare("SELECT \* FROM homes Where Rent >= ? AND Rent <= ? AND Gender = ? AND Rooms = ?;");

$stmt->bind\_param("iisi", $minRent, $maxRent, $requiredGender, $numRooms);

$stmt->execute();

$result = $stmt->get\_result();

$homeSpecifiedArray = array();

while($row = $result->fetch\_assoc())

{

$homeSpecifiedArray[] = $row;

}

return $homeSpecifiedArray;

}

public function changePassword($userID, $newPassword){

$password = md5($newPassword);

$stmt = $this->con->prepare("UPDATE users SET password = ? where id = ?;");

$stmt->bind\_param("si", $password, $userID);

if($stmt->execute()){

return 1;

}else{

return 2;

}

}

}

*PHP User Registration Code:*

<?php

require\_once '../includes/DbOperations.php';

$response = array();

if($\_SERVER['REQUEST\_METHOD']=='POST'){

if(

isset($\_POST['username']) and

isset($\_POST['email']) and

isset($\_POST['password']) and isset($\_POST['phone']) and isset( $\_POST['gender']))

{

//operate the data further

$db = new DbOperations();

$result = $db->createUser( $\_POST['username'],

$\_POST['password'],

$\_POST['email'],

$\_POST['phone'],

$\_POST['gender']

);

if($result == 1){

$response['error'] = false;

$response['message'] = "User registered successfully";

}elseif($result == 2){

$response['error'] = true;

$response['message'] = "Some error occurred please try again";

}elseif($result == 0){

$response['error'] = true;

$response['message'] = "It seems you are already registered, please choose a different email and username";

}

}else{

$response['error'] = true;

$response['message'] = "Required fields are missing";

}

}else{

$response['error'] = true;

$response['message'] = "Invalid Request";

}

echo json\_encode($response);