COIT20269

**Mobile Web Apps**

Term -2, 2020

**Unit Coordinator Lecturer:**

Phillip Whiteley

**Tutor:**

Allan Browning

Prepared By:

Bishal Budhathoki

12116421

**Note: Assignment Done in Pair with Raken Shahi - 12119317**

Table of Contents

[App Design and Specification: 3](#_Toc54314159)

[Description: 3](#_Toc54314160)

[Motivation: 3](#_Toc54314161)

[Summary of task: 3](#_Toc54314162)

[Specification of variables and data structures: 4](#_Toc54314163)

[User Interface Prototypes: 7](#_Toc54314164)

[App Testing: 9](#_Toc54314165)

[Business Plan: 20](#_Toc54314167)

[Estimated costs: 20](#_Toc54314168)

[Time Costing: 20](#_Toc54314169)

[Database Costing: 20](#_Toc54314170)

[Revenue: 20](#_Toc54314171)

[Break-even Sales Analysis: 20](#_Toc54314172)

[Marketing: 20](#_Toc54314173)

[Research: 21](#_Toc54314174)

[References: 22](#_Toc54314175)

# App Design and Specification:

## Description:

The app that we are developing is called “Hamro Ghar” which literally Means “Our Home”. The idea behind the app comes from the problem general people faces when that time of the month comes to pay the landlord. Issues like past remaining amount or like unclear transaction that occurred. Now from this app that we are developing, our intention is to solve that issue providing landlord to manage tenant details along with all bills and past transaction record. Now this will provide clarity in every transaction that occurs, and every detail are backed up on the cloud, so no data is stored on the device.

Motivation: The people in Nepal faces issues in every month to month transaction when need to pay monthly rent, water bill, electricity bill, other bills to the landlord/owner. There can always be some past transaction issue or the issue in the current transaction. Now with the help of this every detail can be shared with the tenant so every transaction is seamless and clear. This issue is not always the same with other country people and for now this app is targeted to the people in Nepal and more generally towards the people in cities. And not every house will have this kind of issue but to make transaction clear between both parties is always a plus point.

Summary of task: The Major tasks to be completed in this app are as follow:

1. Add details of tenant in Mongo Database.

Every details of the tenant are stored in the Mongo database that include their Name, permanent residency, Citizenship Number and more. Local storage is also used in some constraints. So, the app is real-time.

1. Display data of Tenant from Mongo Database.

Stored data of the tenant is retrieved from the database only and displayed to the user.

1. Bill calculation for every user on per month basis.

Bill calculation include Room/unit/apartment rent, drinking water bill, miscellaneous bill, electricity bill with details like per unit charge and the date the bill is to be paid or paid.

1. Storing the bill calculation data in new Collection in database that includes user data and user ID.

Every bill calculated data is stored in the new collection database and managed using the user ID and their data.

Every transaction made using the app is recorded on the cloud and that can be retrieved to view.

1. Delete Tenant.

Tenant after leaving the unit can be deleted from the database so only the active tenants’ details can be accessed and managed. The data is not deleted, it will be on the database, only the status is changed and not displayed. If future dispute arises then we can settle the dispute by going back to the history records.

## Specification of variables and data structures:

Data types used:

Server Based:

|  |  |
| --- | --- |
| Data Type | Description |
| Integer | ObjectID, id are the integers used |
| Array | currentCollection, result |
| Constants | Express, app, cors, bodyParser, MongoClient, uri, client are all the imported files |

Data structure: The data is collected from the user and stored in the local storage and then stored in the Database. Then, when the data is to be displayed then it is retrieved from the database and presented in the display in their required format. Various methods and variables are used here.

The source code for the server is provided below along with the method and their intended work comment:

const ***express*** = ***require***('express');  
const app = ***express***();  
const cors=***require***('cors');  
const ***bodyParser***=***require***('body-parser');  
var config = ***require***('./config.js')  
const MongoClient = ***require***('mongodb').MongoClient;  
var ObjectID=***require***('mongodb').ObjectID;  
var currentCollection;  
  
app.use(cors());  
app.use(***bodyParser***.json());  
app.use(***bodyParser***.urlencoded({extended:true}));  
  
const uri=config.mongohq.uri; //getting mongodb url link  
const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true});  
client.connect(err => { //connecting to the database  
 currentCollection = client.db(config.mongohq.database\_name).collection(config.mongohq.collection\_name);  
 // perform actions on the collection object  
 ***console***.log("Database Connected!");  
}); //connection to mongodb  
  
//route to homepage  
app.get('/', function(req,res) {  
 res.send("Tenant Homepage / Run index.html via cordova");  
});  
  
//route to perform search required Tenant from database  
app.get('/search/:query', function (req,res){  
 client.db(config.mongohq.database\_name).collection(config.mongohq.collection\_name).find({\_id:ObjectID(req.params.query)}).toArray( function(err,docs) {  
 if(err) {  
 ***console***.log("Some error.. " + err);  
 } else {  
 res.send(docs);  
 }  
 });  
})  
  
//route to perform remove Tenant operation from database  
app.get('/remove/:query', function (req,res){  
 var id=req.params.query  
 ***console***.log("Removing.. "+id );  
 updateStatus(client,id);//Status field is updated from active to REMOVED  
})  
  
//route to perform search transaction from database  
app.get('/searchTransaction/:query', function (req,res){  
 ***console***.log("search transaction of id.. "+ req.params.query);  
 client.db(config.mongohq.database\_name).collection(config.mongohq.invoice\_collection).find({tenantID:req.params.query}).toArray( function(err, docs) {  
  
 if(err) {  
 ***console***.log("Some error.. " + err);  
 } else {  
 res.send(docs);  
 ***console***.log(docs);  
 }  
 });  
})  
  
//route to perform get all Tenant from database  
app.get('/getAll', function (req,res){  
 client.db(config.mongohq.database\_name).collection(config.mongohq.collection\_name).find({status:"Active"})  
 .toArray( function(err,docs) {  
  
 if(err) {  
 ***console***.log("Some error.. " + err);  
 } else {  
 res.send(docs);  
 }  
 });  
  
})  
  
//route to send data to server  
app.post('/postData', function(req,res) {  
 var data= req.body;  
 res.send(req.body);  
 addTenant(client,data);  
});  
  
//route to send Invoice to server  
app.post('/postInvoice/:query', function(req,res) {  
 var tenantID=req.params.query;  
 var data= req.body;  
  
 res.send(req.body);  
 addInvoice(client,data);  
});  
  
app.listen( 3000, () => {  
 ***console***.log("Web Server started on port 3000");  
});  
  
async function updateStatus(client,data){  
 ***console***.log("Removed "+ data);  
 result = await client.db(config.mongohq.database\_name).collection(config.mongohq.collection\_name).updateOne({\_id:ObjectID(data)},{$set:{status:"REMOVED"}});  
 ***console***.log(result);  
}  
  
//function to send list of data to server  
async function addTenant(client, newListings){  
 ***console***.log("SENDJSON:")  
 const result = await client.db(config.mongohq.database\_name).collection(config.mongohq.collection\_name).insertMany(newListings);  
 ***console***.log(`${result.insertedCount} new list(s) created :`);  
 ***console***.log(result);  
}  
  
//function to add invoice in database  
async function addInvoice(client, data) {  
 result = await client.db(config.mongohq.database\_name).collection(config.mongohq.invoice\_collection).insertMany(data);  
 ***console***.log(`${result.insertedCount} new list(s) created :`);  
 ***console***.log(result);  
}

## User Interface Prototypes:

The user interface prototype that we came up with contains a n app name at the top of the screen then have a 2 \* 3 grid that contains various options to deal with along with the icons/images for better user interface. So, the protype screenshots are as follow:

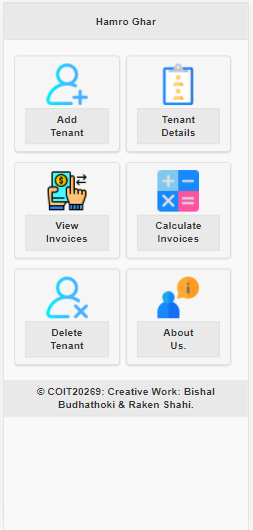
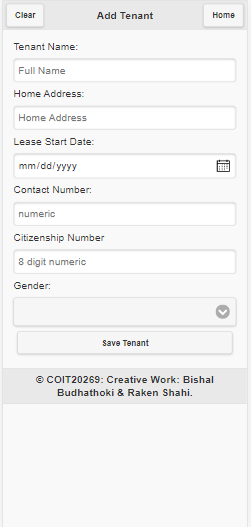


Figure 1: Home Screen Figure 2: Add Tenant Detail Page

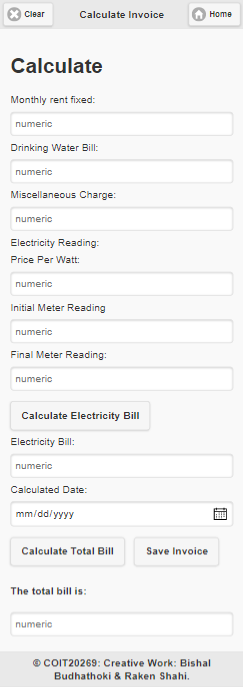


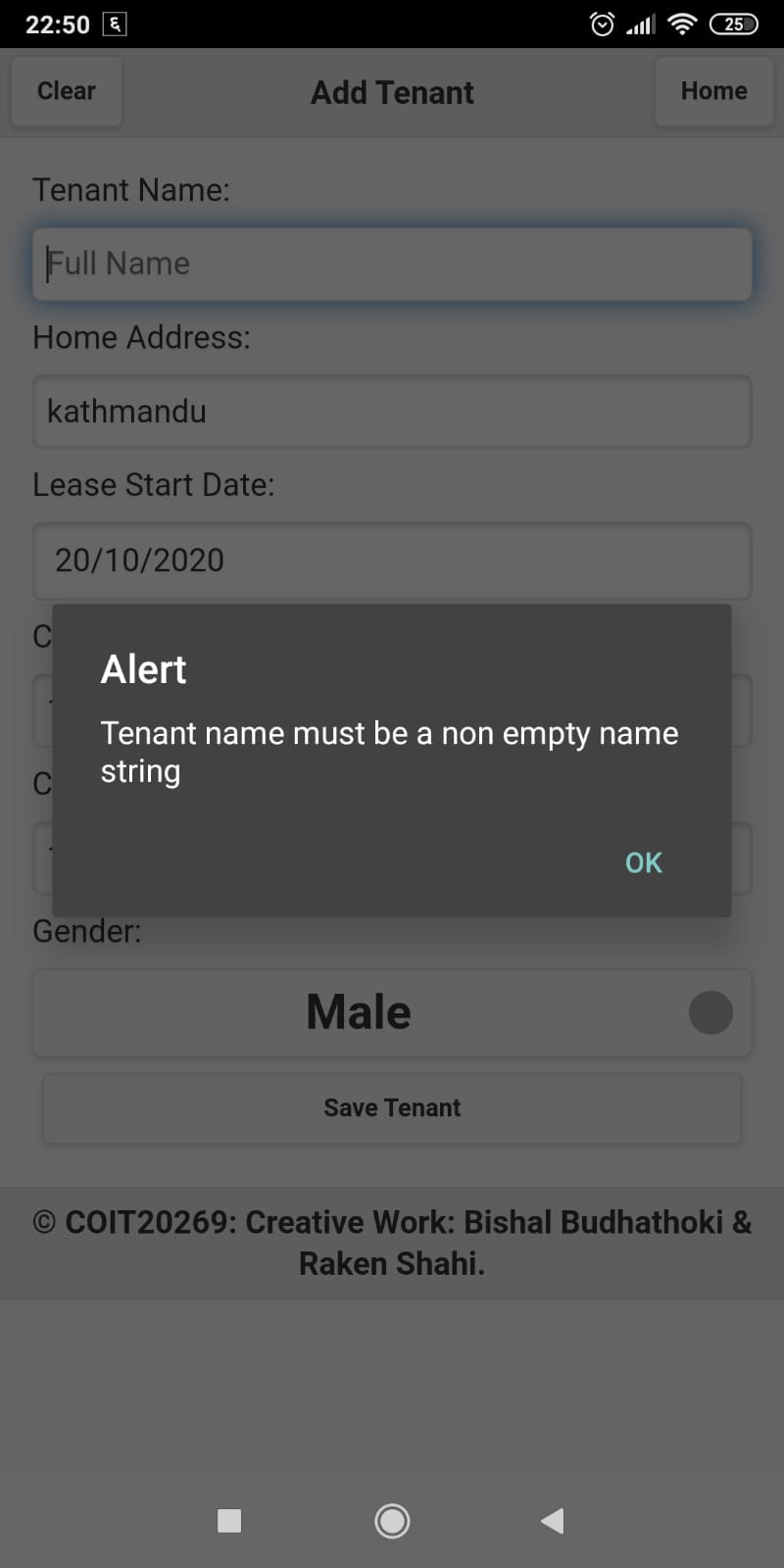
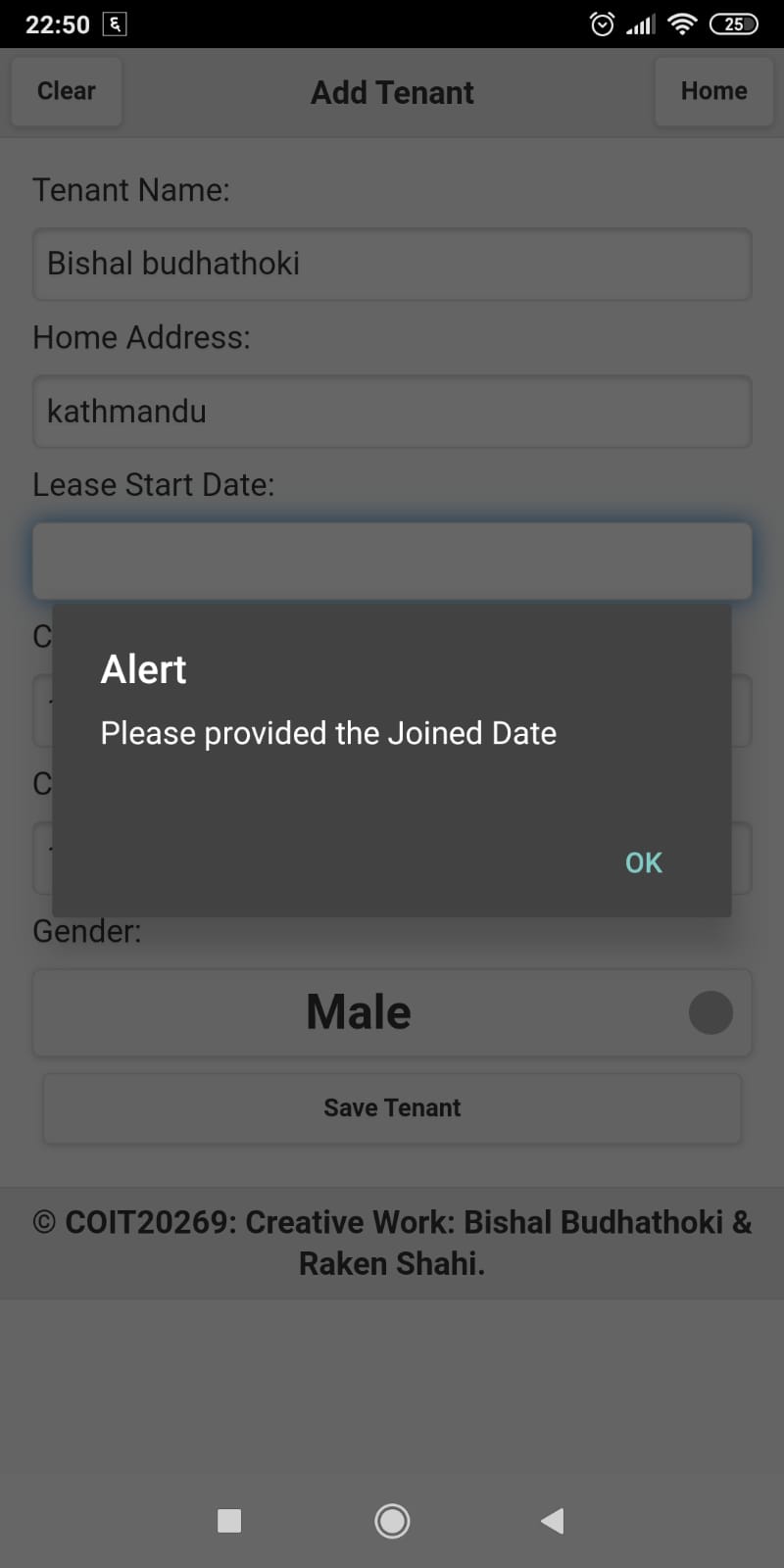
Figure 3: Calculate Invoice page prototype

For the prototype, we initially came up with just a button in a home screen then created the above form page for details. Then after working on and iterating every new methods and aspects we made more screens and work on them. It is provided further on app testing section.

# App Testing:

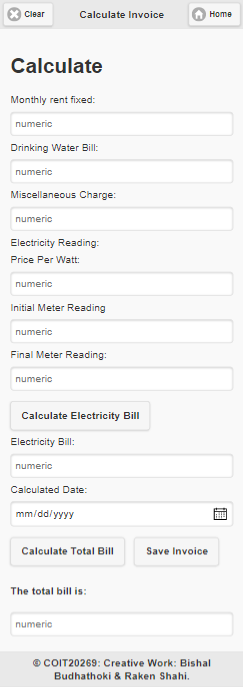
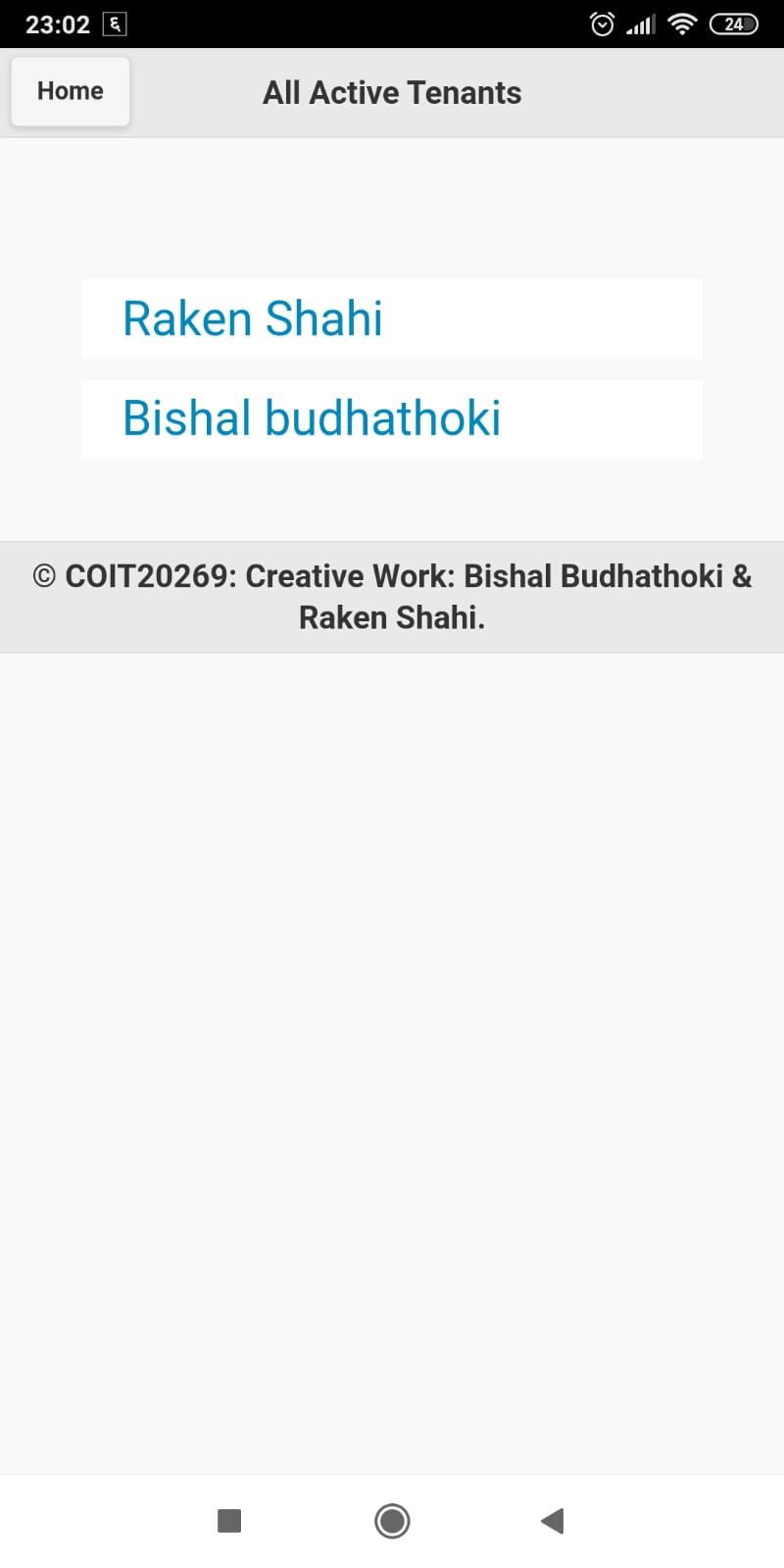
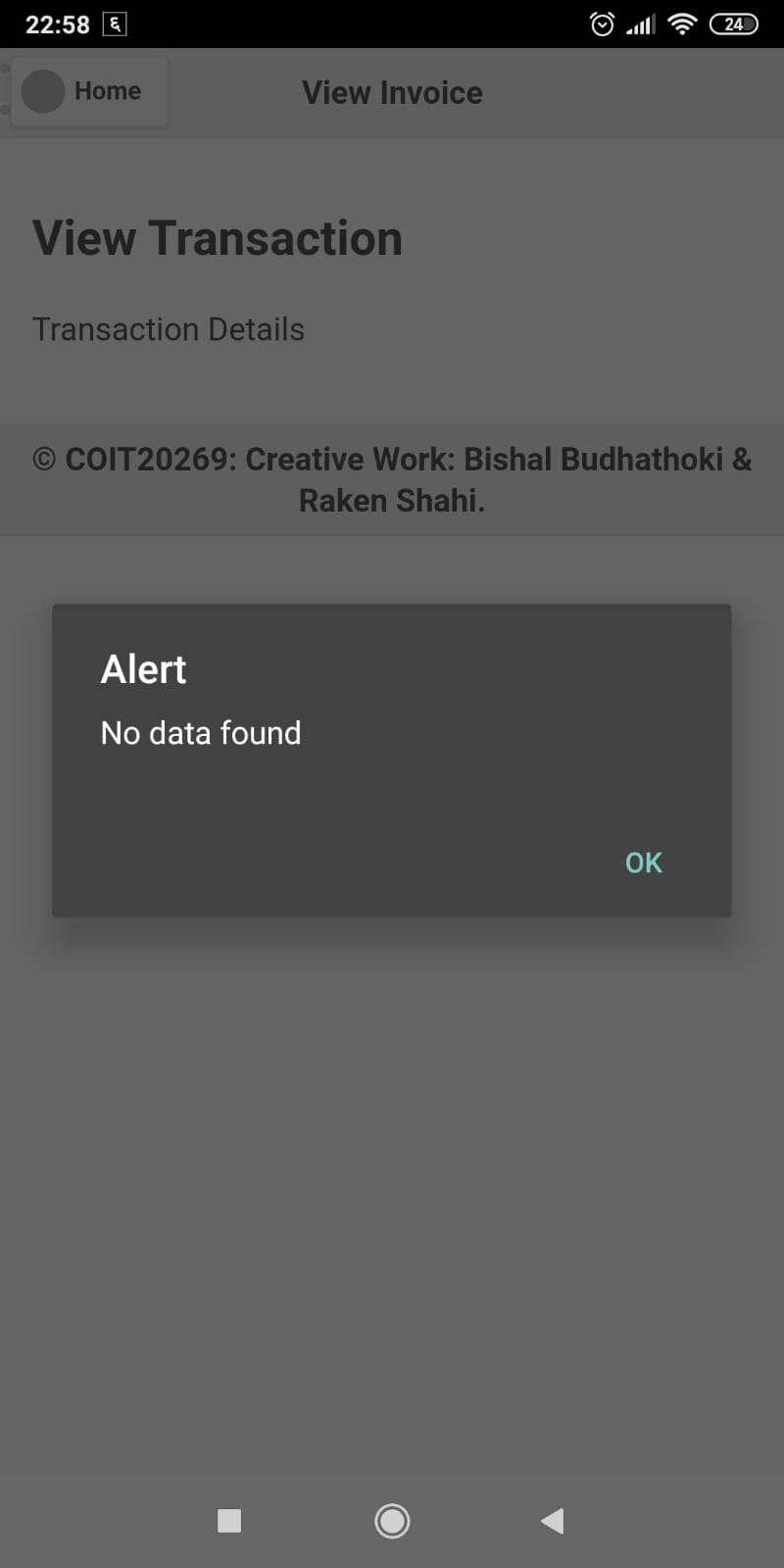
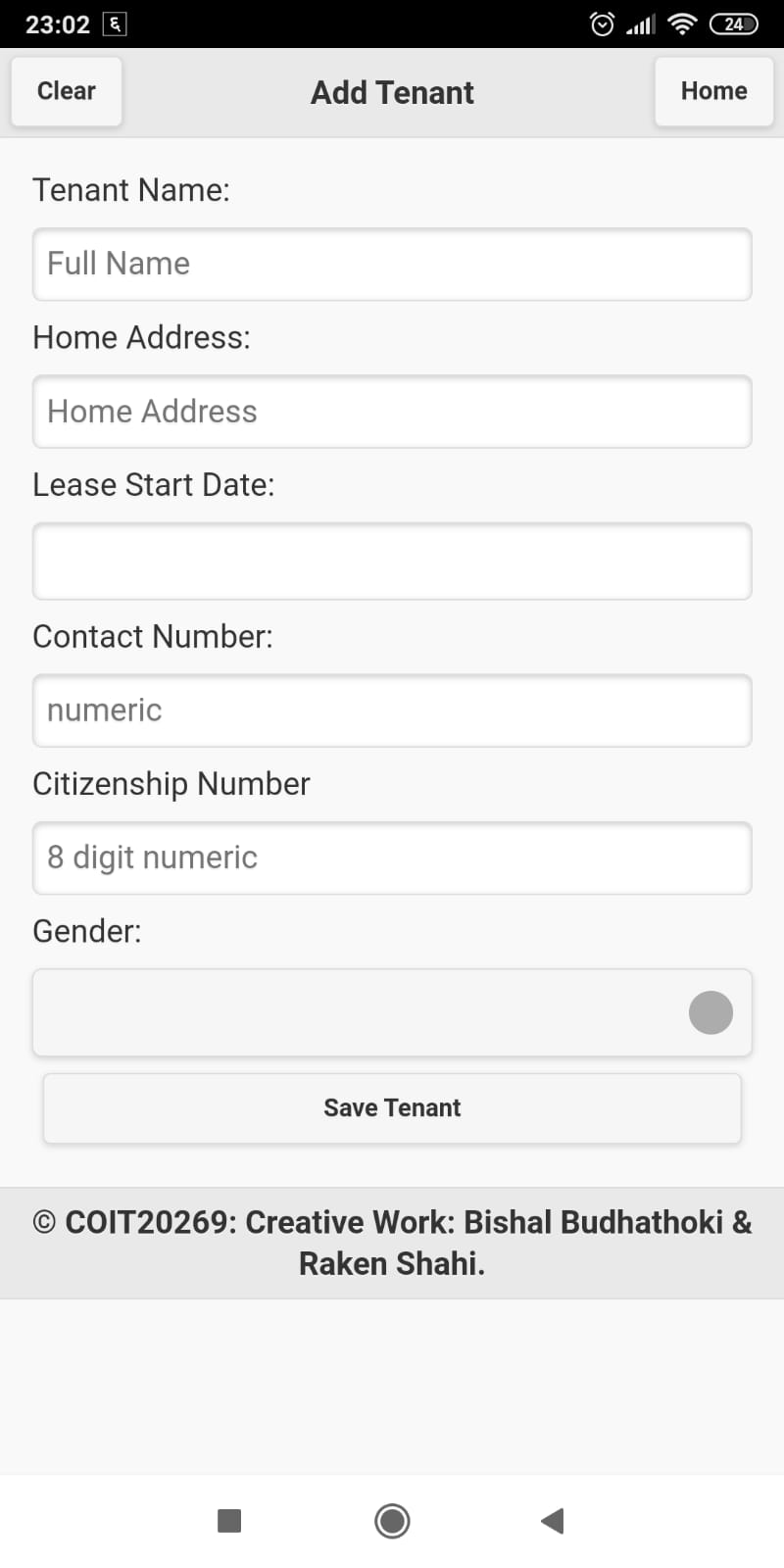
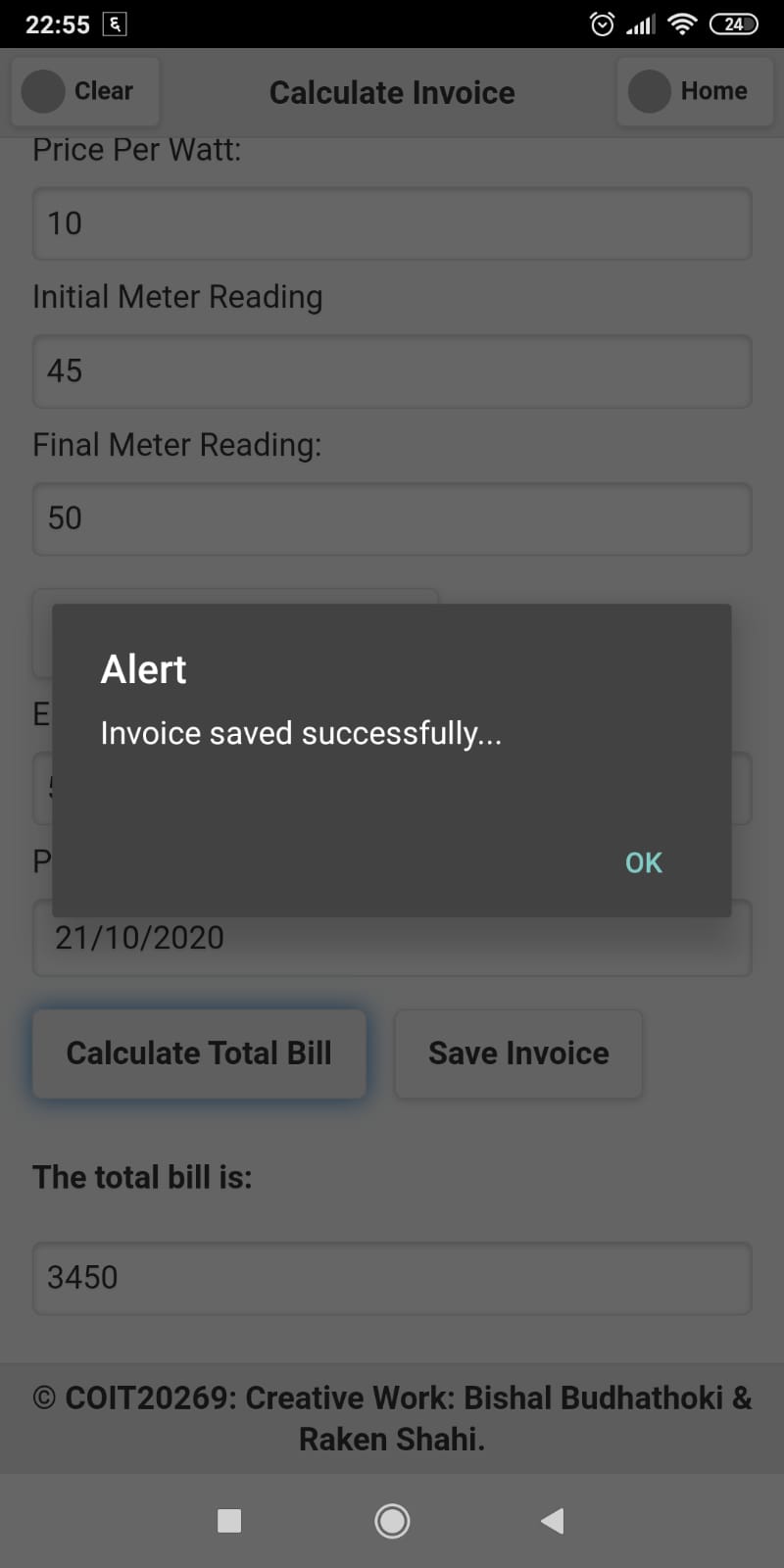
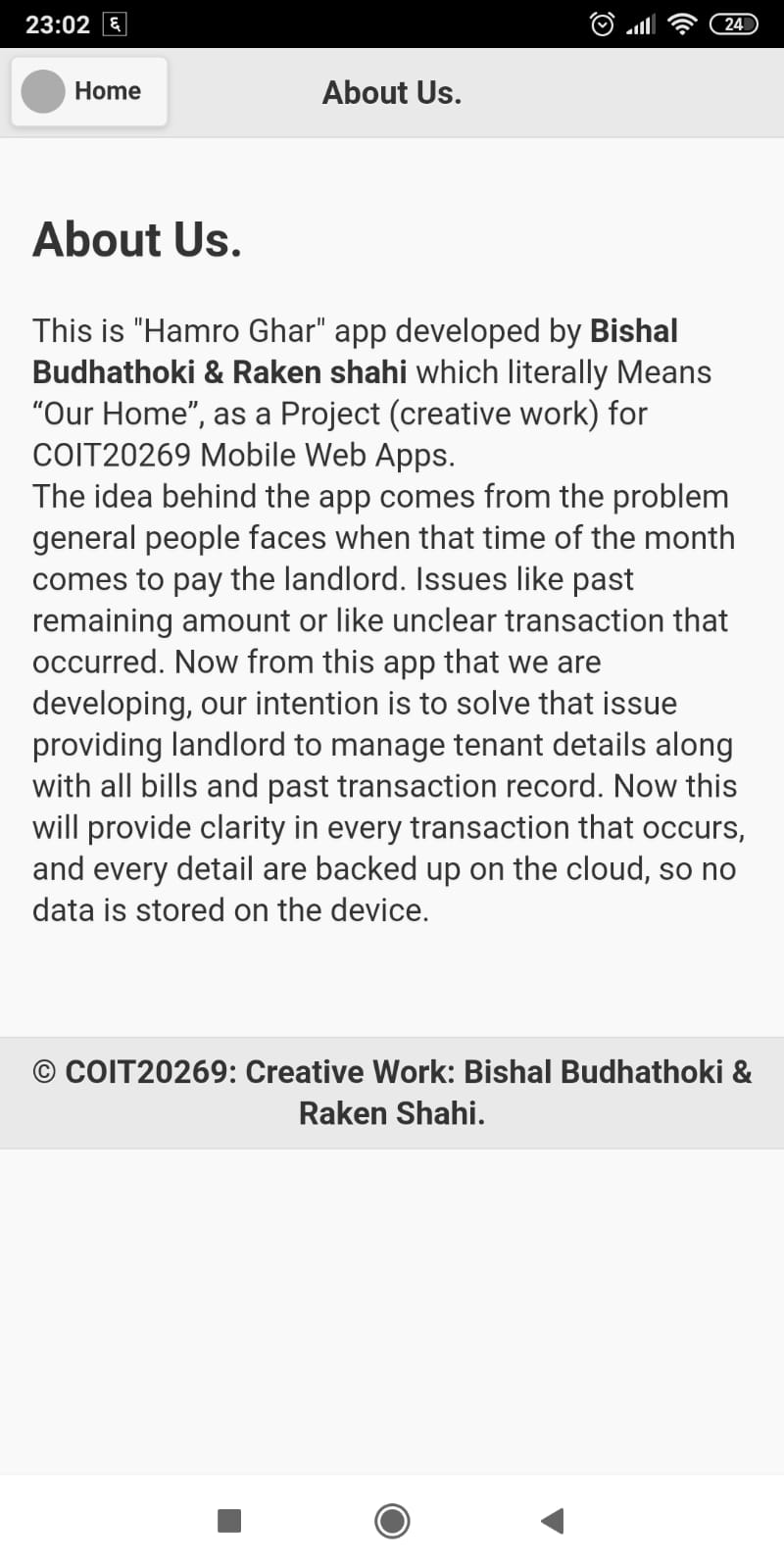
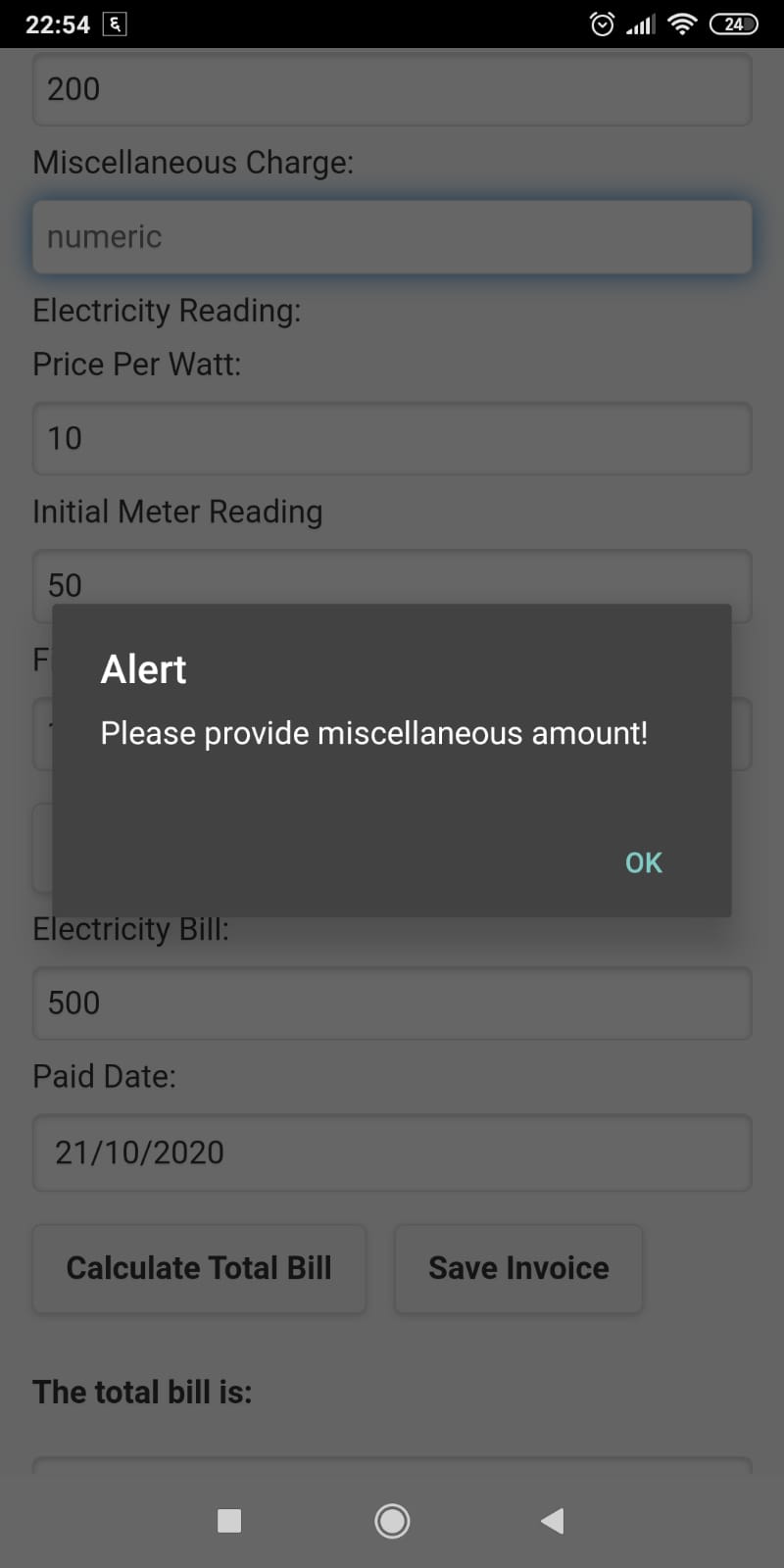
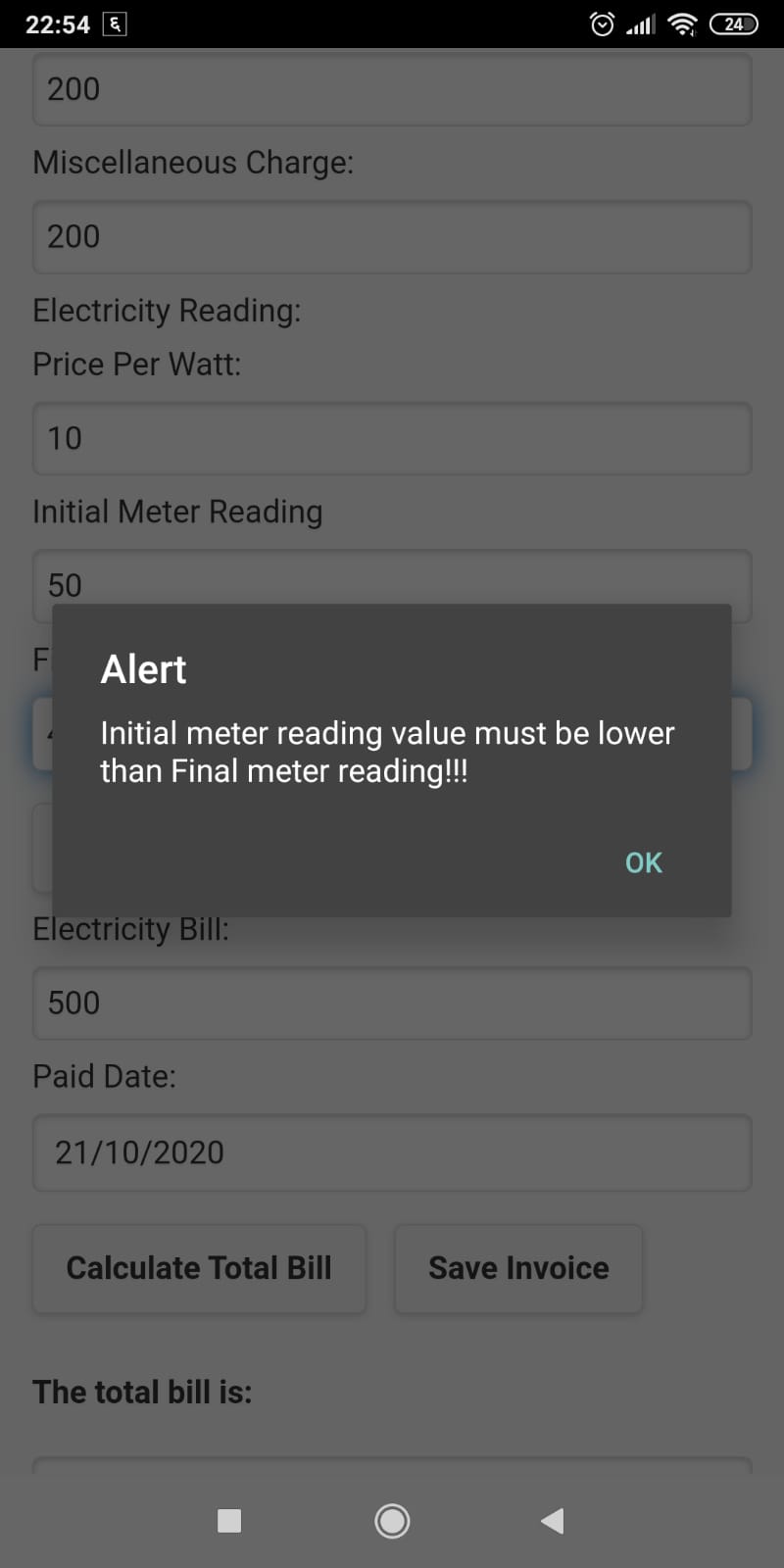
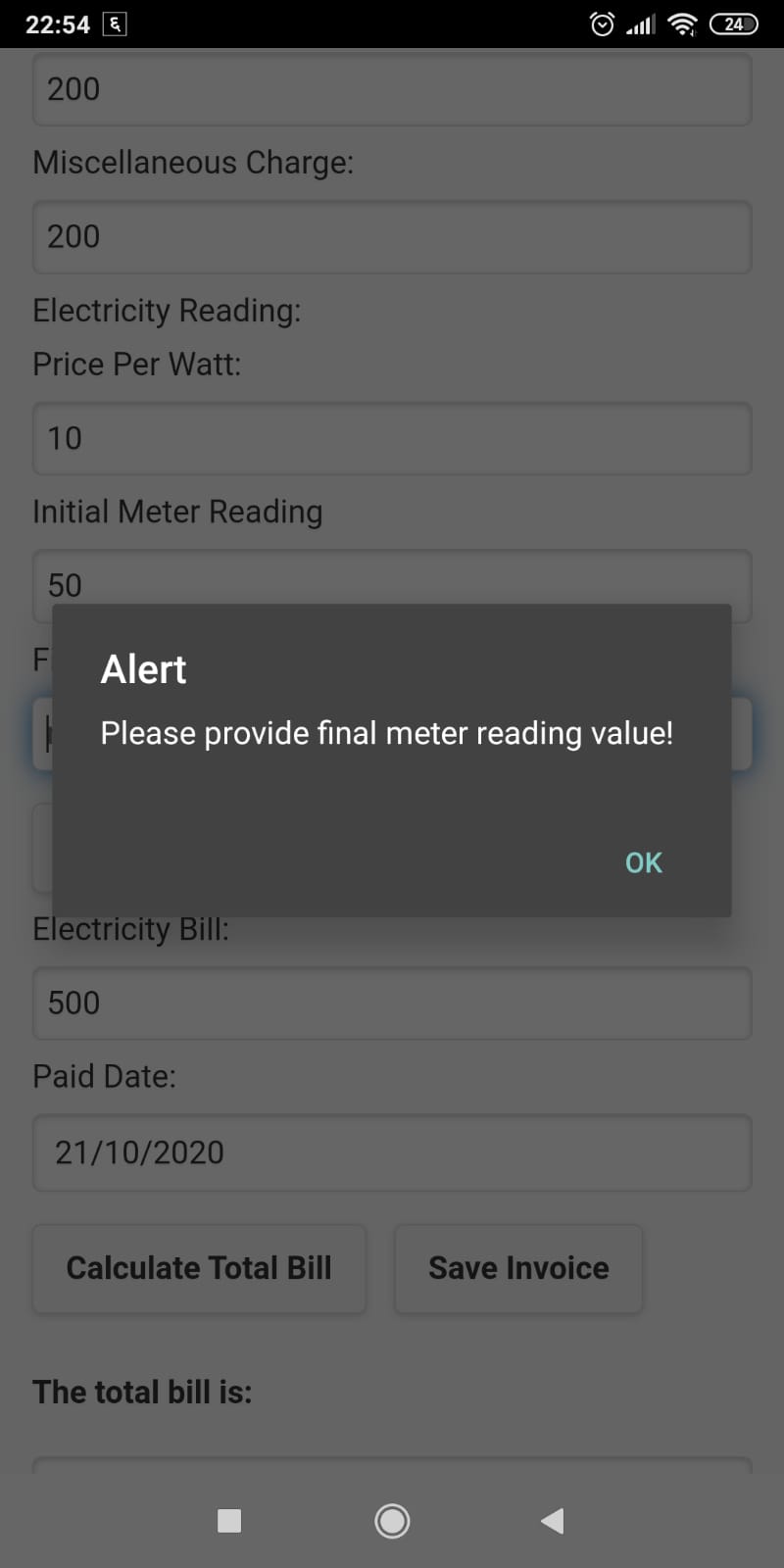
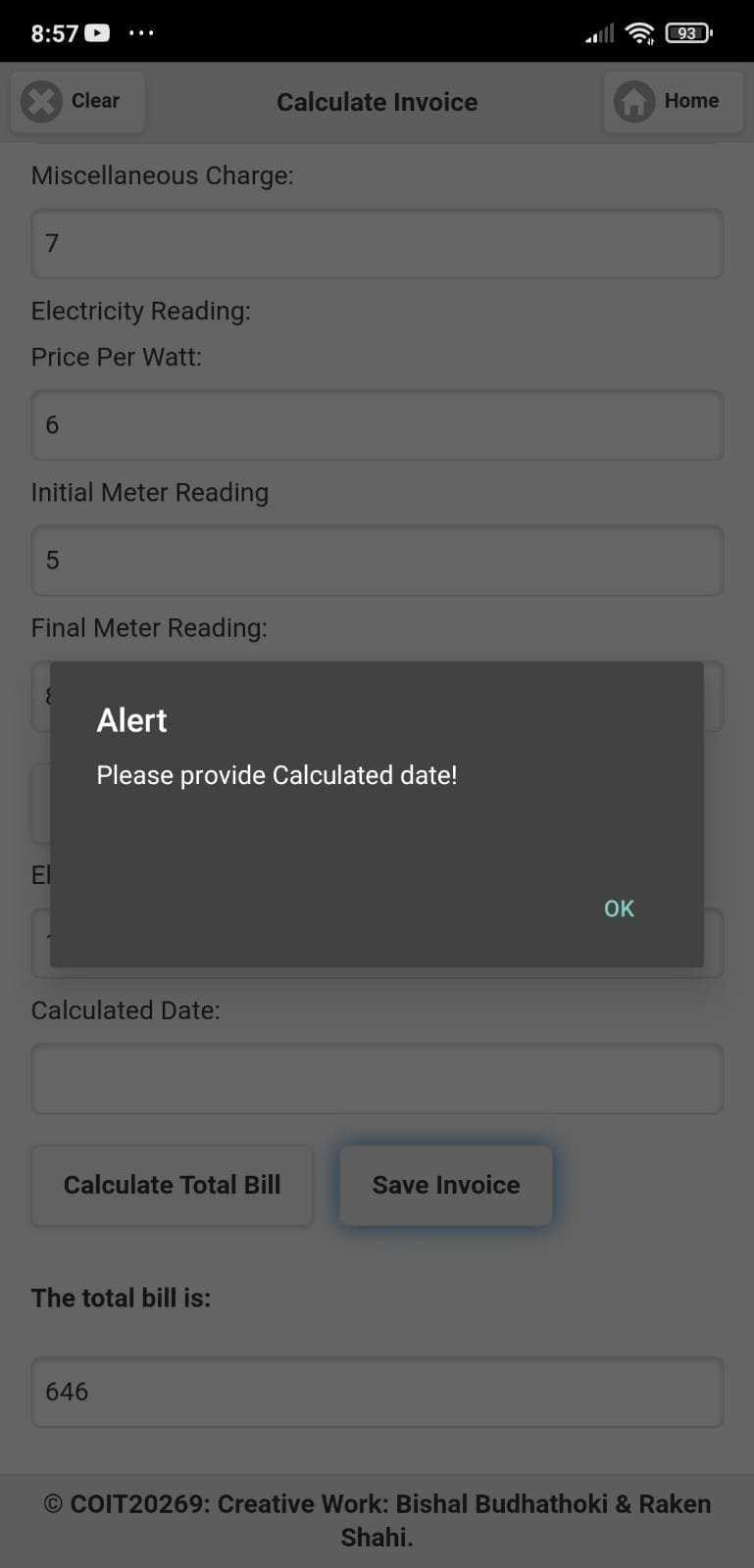
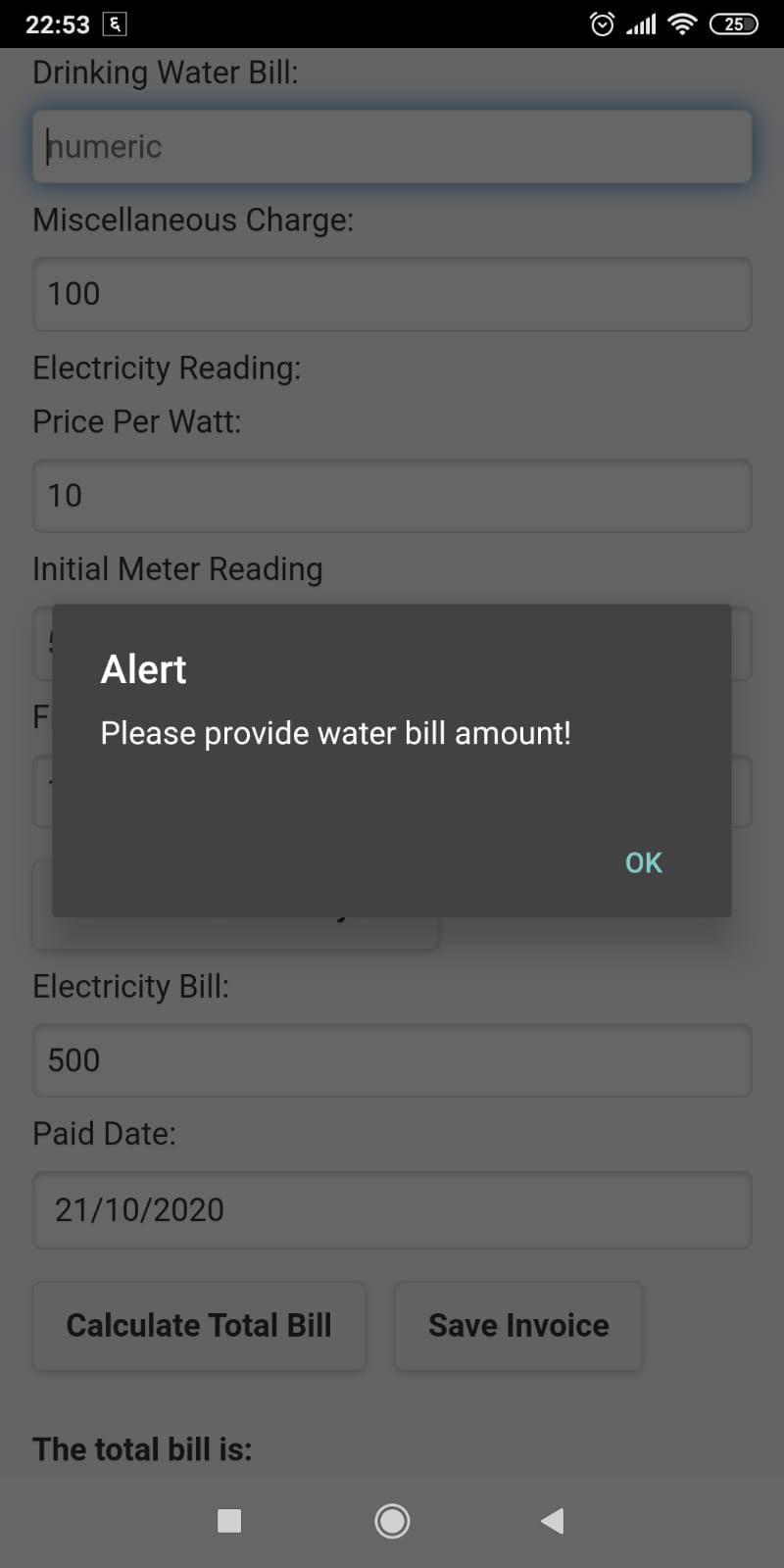
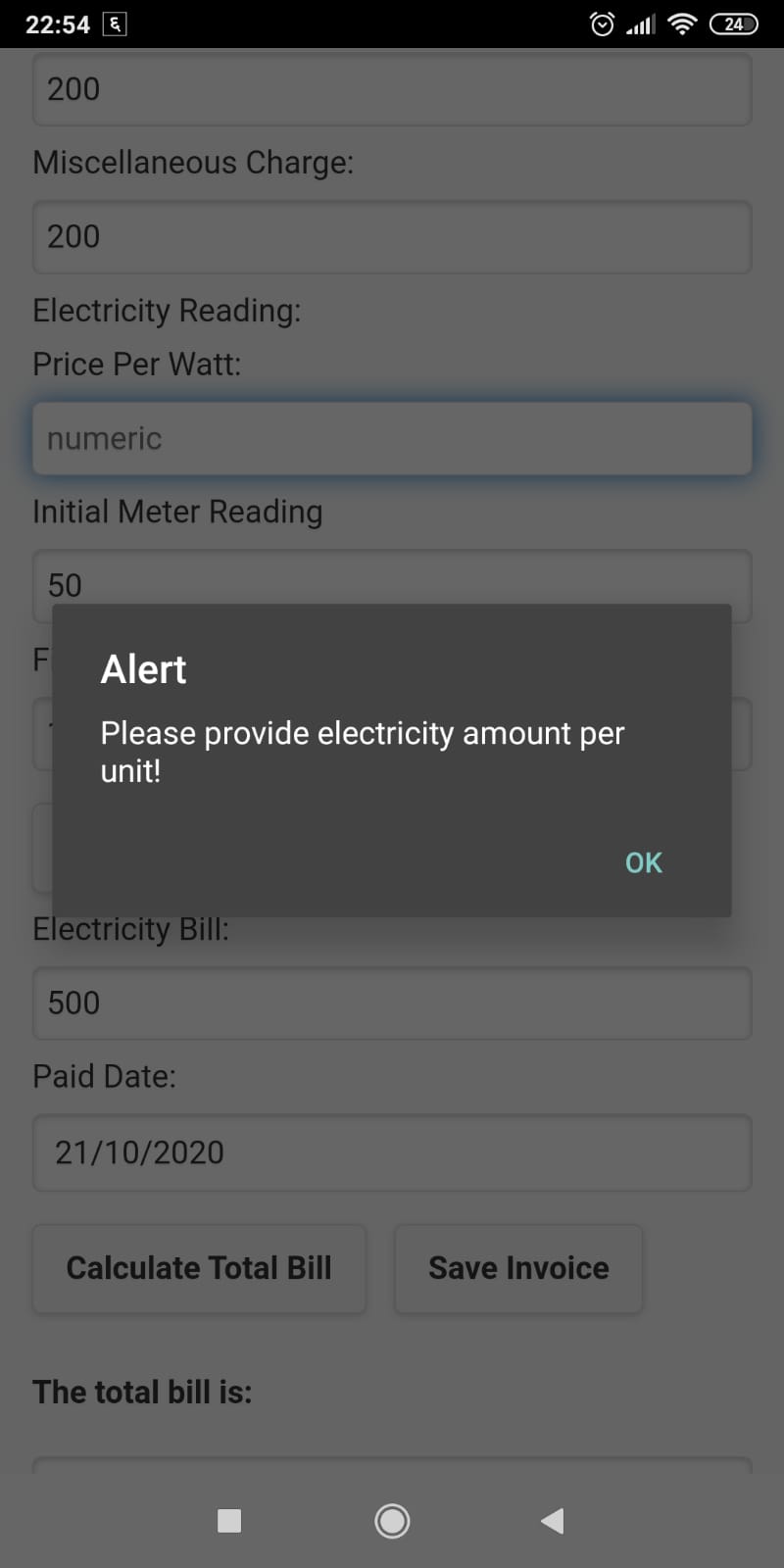
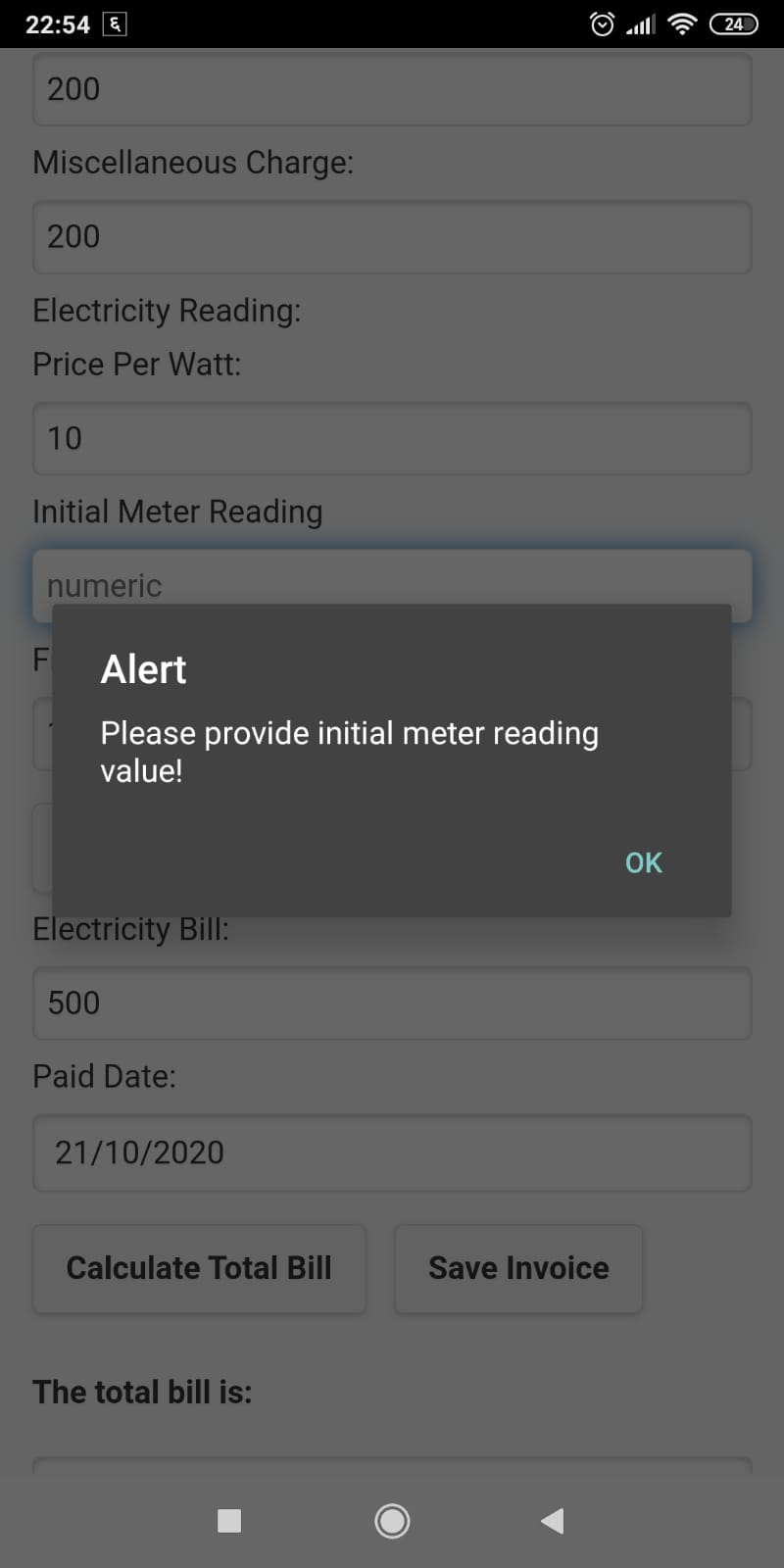
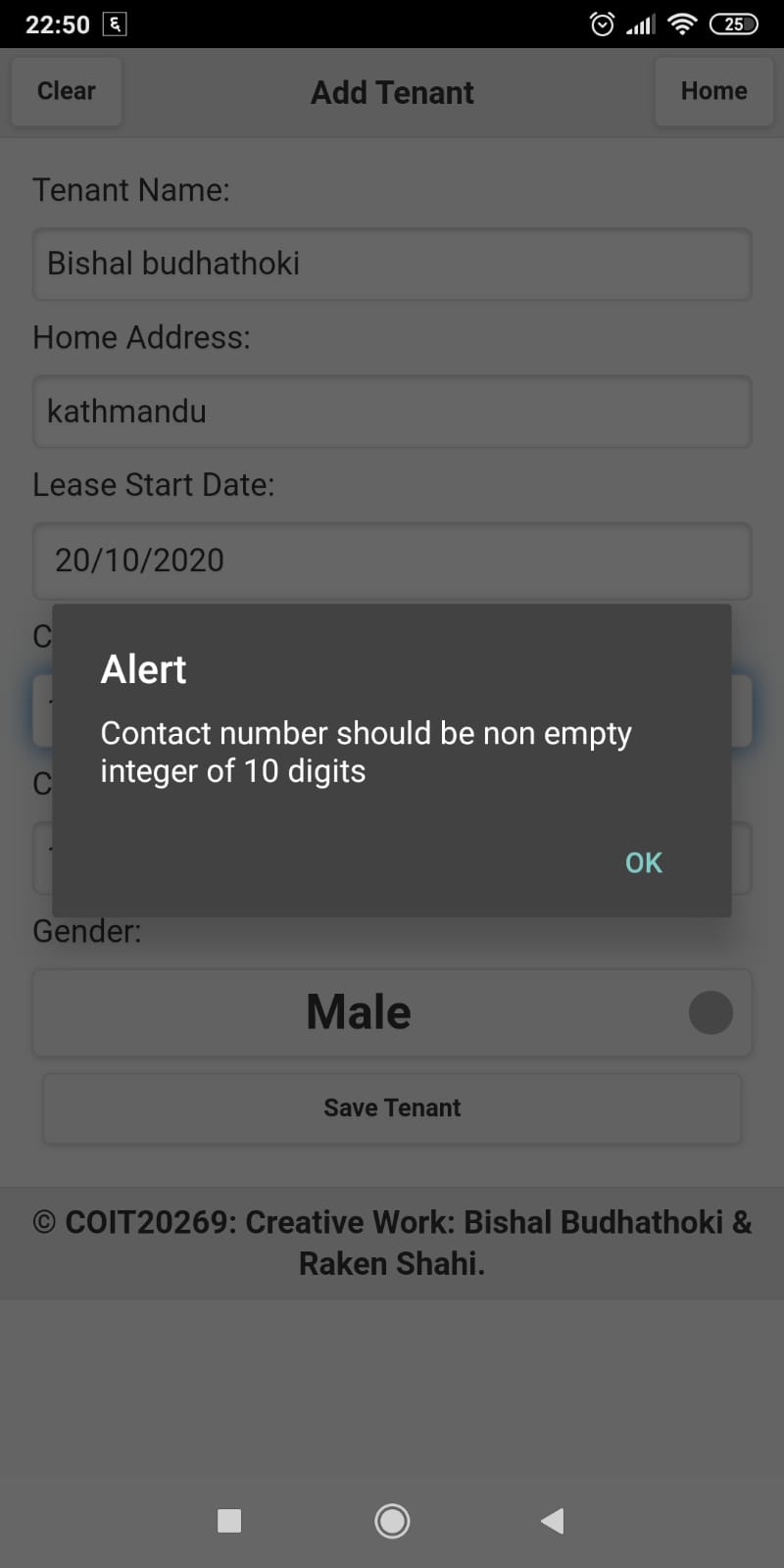
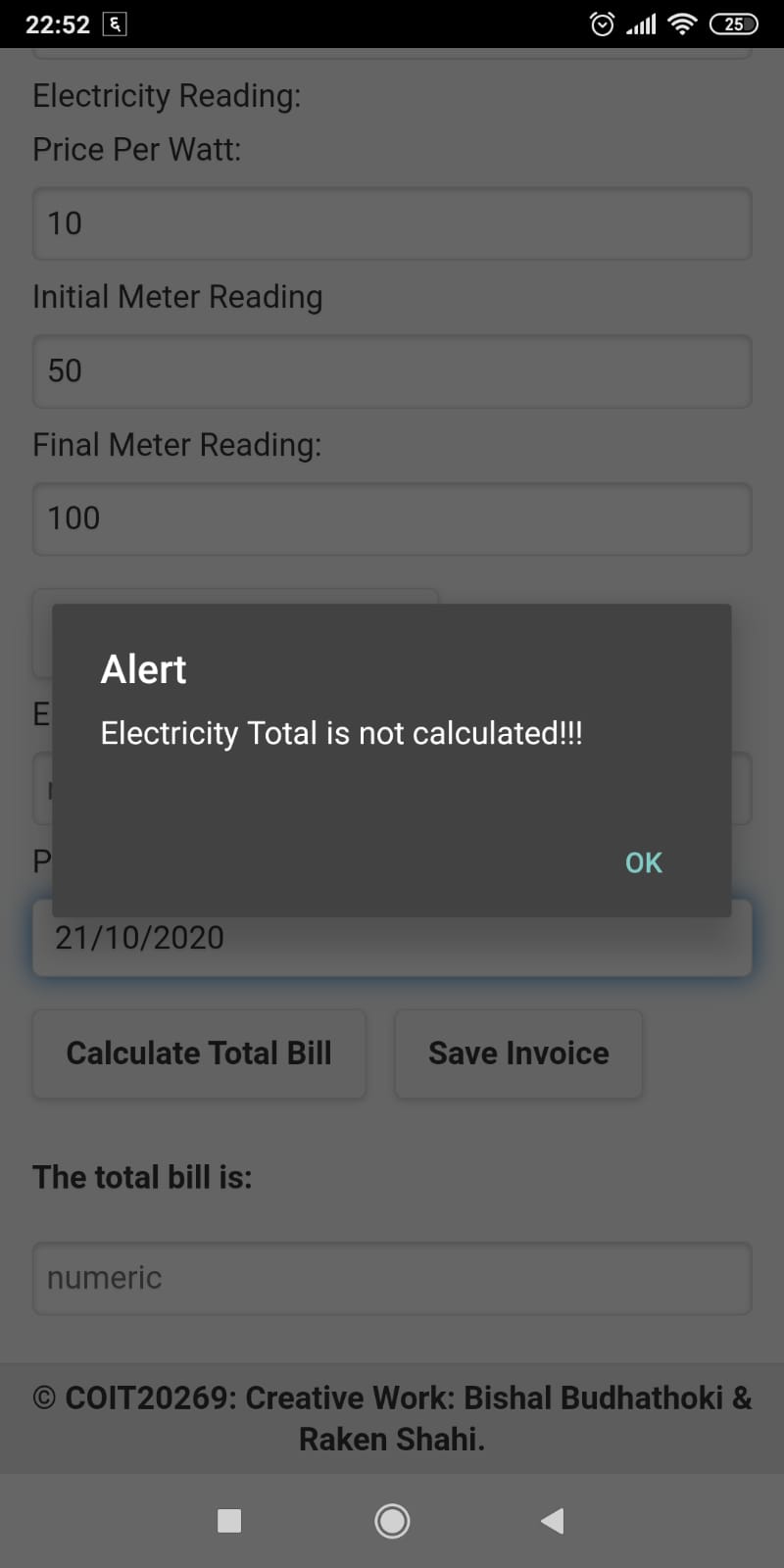
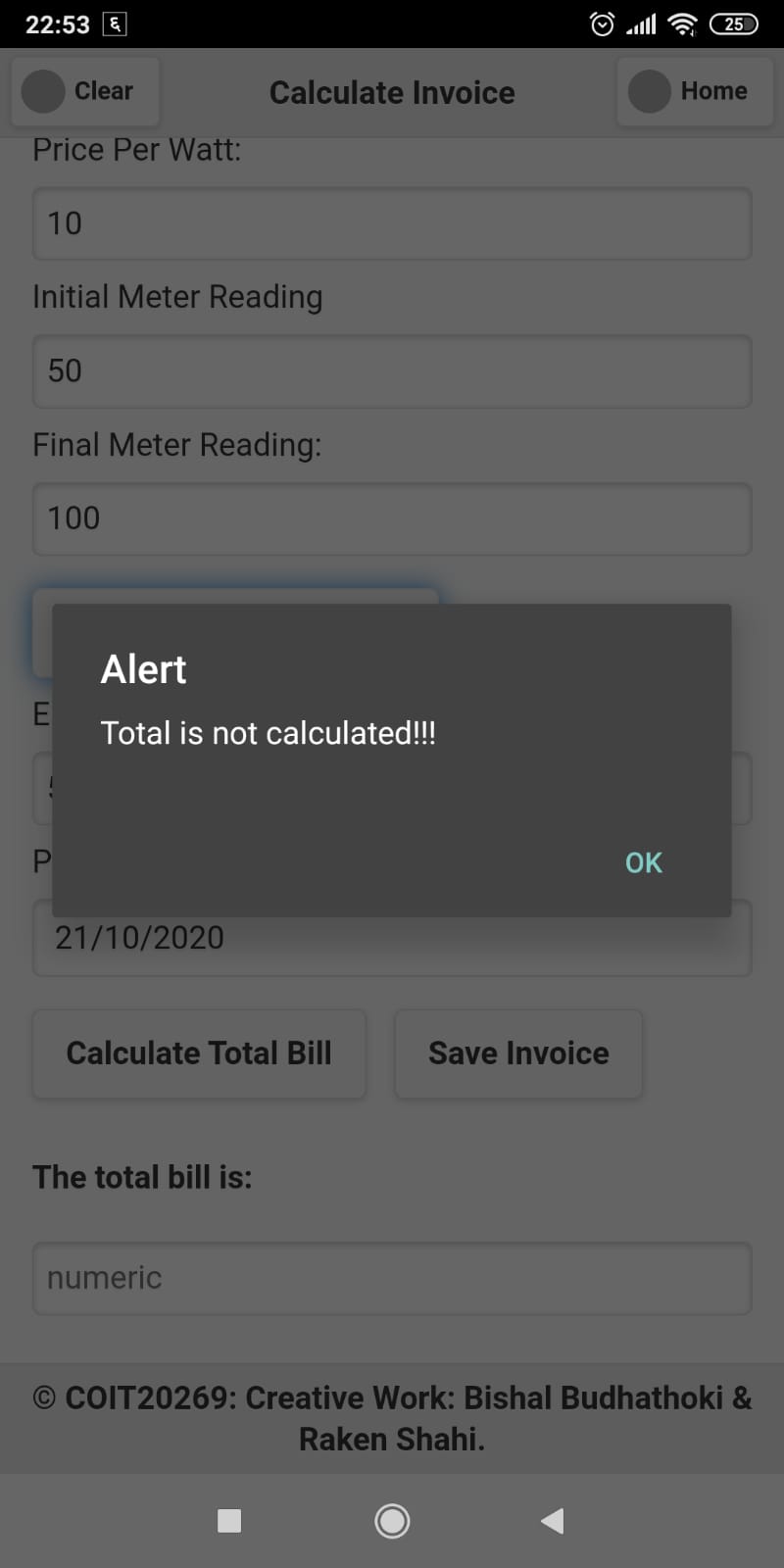
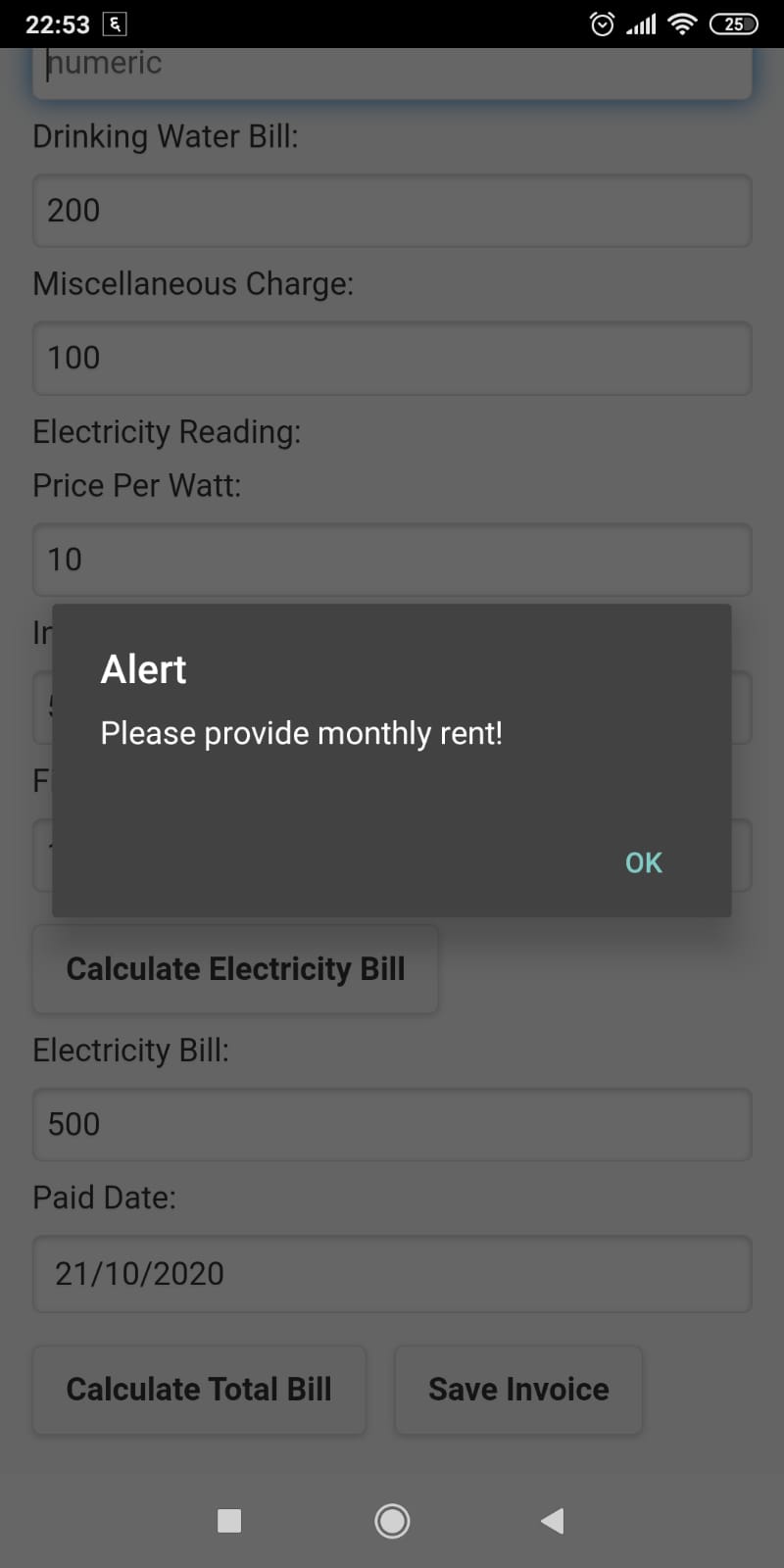
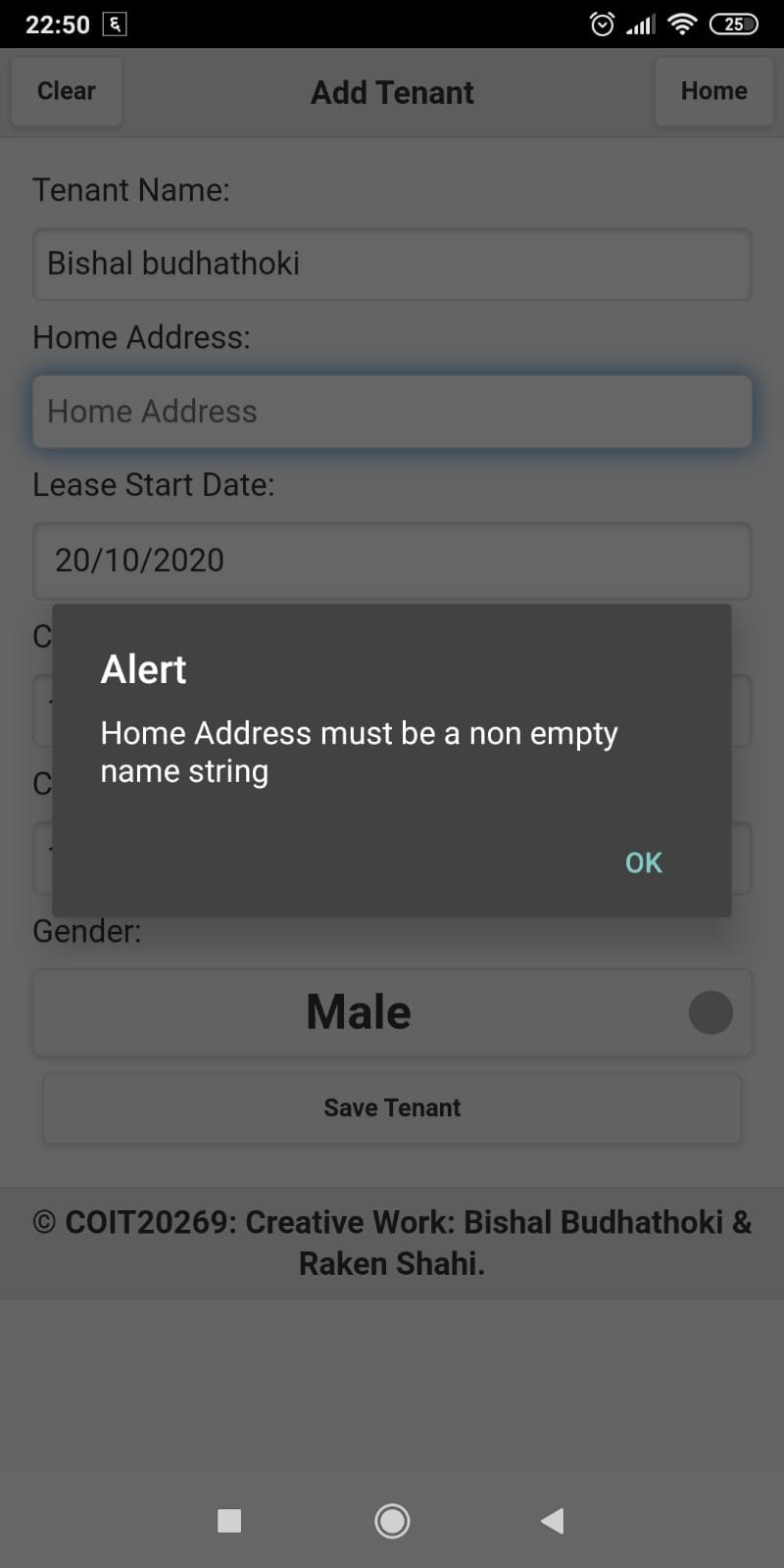
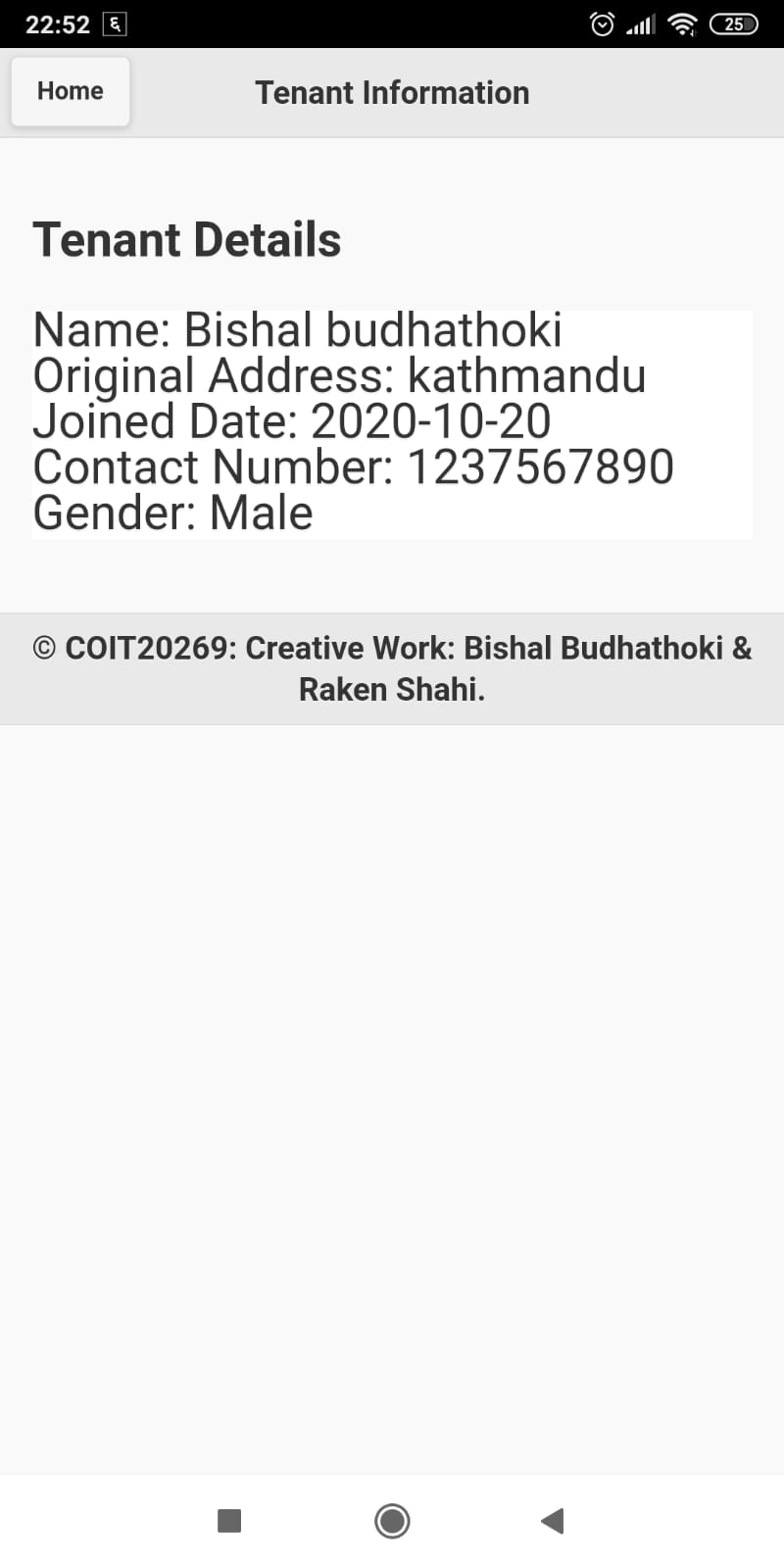
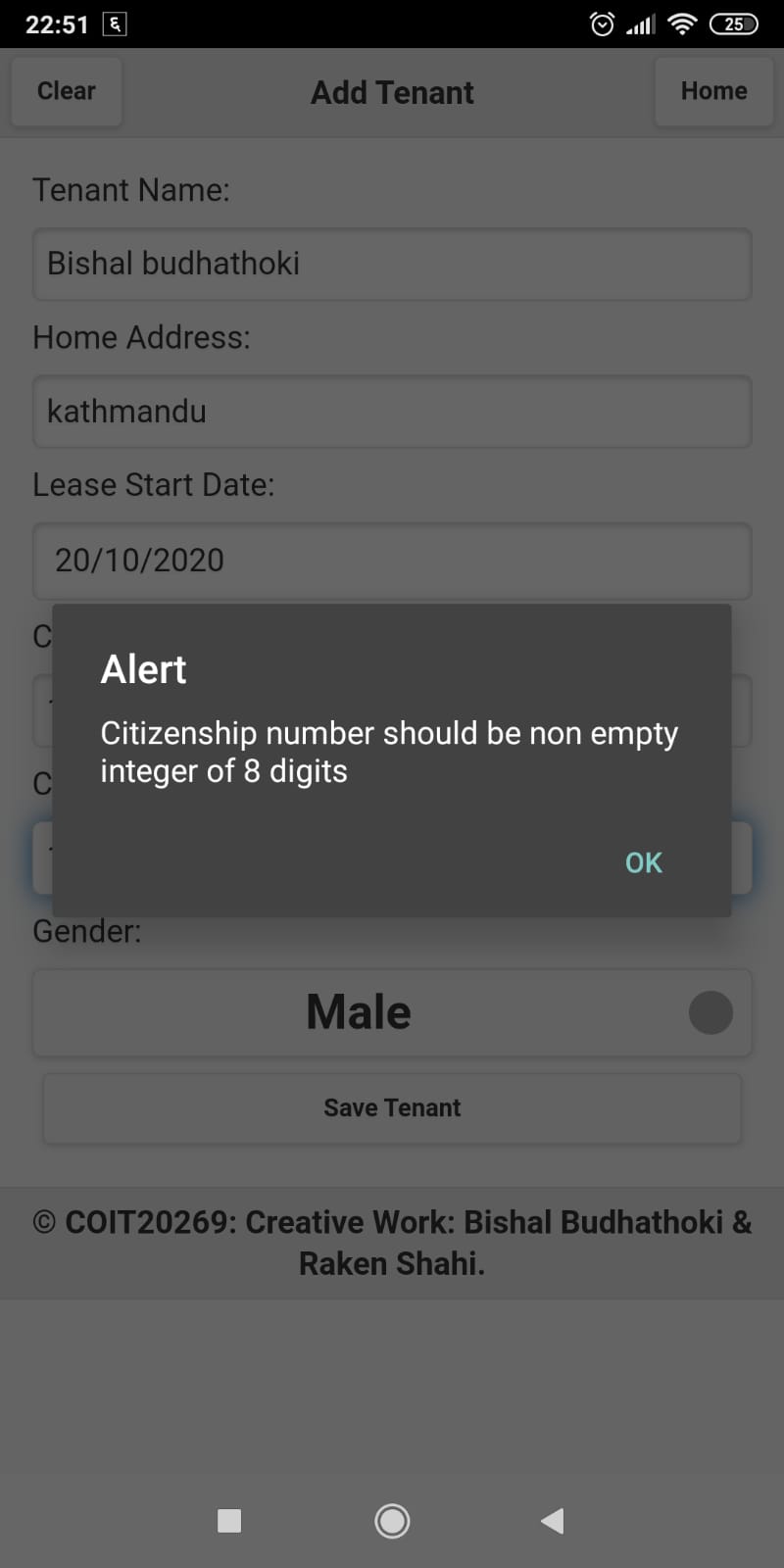
For this application, we have made a testing plan. For this due to time constraint, we have limited testing strategy to do. And here is the testing plan.

1. Testing the UI: This include grid width and position, forms and buttons position and functioning (do the required function or take to the required page layout), screen size and the widget position and sizes.
2. Validation: Every input validation is done. Connection to the database, response from the database is done here.
3. Exception handling: Sometimes errors occurs while working

Overall, these are the testing plan to perform and the result obtained from the testing is displayed through screenshots provided below:

First figure is validation alert for Joined Date not selected or provided.

Another figure is validation alert for tenant name.



First figure is validation alert for Citizenship Number

Another figure is the tenant’s details after selecting the tenants name from the list.

First figure is validation alert for home address left empty or string not provided.

Another figure is validation alert for monthly rent not provided.

First figure is validation alert for Total bill not calculated due to input wrong or empty.

Another figure is validation alert for total electricity not calculated.

First figure is validation alert for Contact number either being empty or not 10 digits.

Another figure is validation alert for initial meter reading value.

First figure is validation alert for per unit price not given.

Another figure is validation alert for Drinking water bill/price not provided.

First figure is validation alert for Calculate invoice date if not provided.

Another figure is validation alert for final meter reading value.

First figure is validation alert for initial meter reading being higher than final meter reading.

Another figure is validation alert for miscellaneous amount not provided.

First figure is About Us page.

Another figure is validation alert for Invoice saved successfully in the local storage and the database.

First figure is the add tenant page.

Another figure is validation alert for no transaction data found for a particular person.

This figure shows the current active tenants in the list.

This figure shows the calculate Invoice page.

# Business Plan:

## Estimated costs:

### Time Costing:

Staff Development Cost = $80 per hour

Total hours spend to develop this application = 14 hour

Total cost = Total hours spend to develop this application \* Staff Development Cost

= 14 \* 80

= $1120 for 14 hours of work

### Database Costing:

For this application there are two different collection named Invoices\_Info and Tenant\_Collection. Size of 10 entries is 2.43 KB and 1.97 KB respectively as per document size and 36 KB as per index size for both collections. For this total database size is 4.4 KB. And considering a house can have 10 entries and there are 6708 households/houses ("Ward 09", 2020). Considering least 50% of the houses will use the app that means 3354 houses are going to use it and for that total size of database to be consumed is 14.41 MB in a month. And for a year it can cost 172.94 MB for a year. And for this the current free tier data base is enough as it is 2 GB.

Revenue:

Let consider that the app monthly subscription fee is $5 per month and 10 entries are given and for more than 10 entries fee increases by a dollar per entries. Then from 3354 houses total revenue per month can be $5 \* 3354 houses = $27,670 from 3354 houses and considering only entries are done.

### Break-even Sales Analysis:

Now knowing the total cost for the app development, database costing and the earned revenue, break-even sales can be analysed. Total app development cost is $1120, Database Cost is null and if 5 GB is taken with $9 per month that is also acceptable. And the Revenue is $2250 per month. Marketing cost is around $2090. So, then the break-even point is around 2nd month.

## Marketing:

Demographic Identification: Demographic is a part of a population that determine certain aspect like gender, ethnicity and more (Davis, 2020). And as per the writer Market Demographic is a part of the population that may use or buy the product, or the software and these people are the one that the product is/will be targeted, and marketing is directed towards. So, here for the people of Nepal residing mostly on city areas and have their house on rent for the people to use and the expected users are around 6708.

Description of tactics: Now to promote the app to every targeted user, we can promote the app TV Ads, Radio Ads, Social Media Ads and Few Social Media Influencers. The cost for TV Ads and radio Ads can be around $150 and Social media ads can around $440 if 1000 users install from the ads. Social Media Influencers Can Cost around $50 to $150 per content and if 10 influencers are used then it can cost around $1500 considering the max cost for the influencers.

We can initially start from radio ads and then promote to through TV. When the people start to learn the app name and when talk about the app starts then we can use social media ads to cover more of the talk and rise the popularity. Now after that we can then use social influencers who owns a house, demonstrate the usage and talks about the app in brief. If continue this trend, then we might be able to capture the market after all there is always a place to learn and improve.

## Research:

On intensive researching, similar app was found in the Apple App Store namely, RentPal. There after loading the splash screen, list of tenant is displayed along with the details like Address, Tenant name, room/house/unit taken number, rent amount along with Paid/Unpaid Statues and Remaining credit if any remaining and + icon button to add tenant if any. On selecting the list, individual tenant property page is displayed with property details like monthly rent, number of tenants, lease type and start date and documents provided if any with an option to add Credit if required. There is an option to delete along with edit option that opens tenant detail page where detail of tenant can be edited and deleted. It is also based on PayPal’s servers. Rent is also deducted on first of every month. Leases documents are provided in app. Pet fee and other miscellaneous charges are also there. Complaints can also be done via app. Any late notices are provided to the tenant. Even if tenant does not want to use the app or own a smartphone, tenants payment details can be done manually along with their payment done in cash, cheque or other means. ("‎RentPal – Property Manager, Rent Manager, Rental Management, Apartment Managemen", 2020)

# References:

Davis, O. (2020). What are Market Demographics? - Definition & Overview. Retrieved 21 October 2020, from <https://study.com/academy/lesson/what-are-market-demographics-definition-lesson-quiz.html>

‎RentPal – Property Manager, Rent Manager, Rental Management, Apartment Managemen. (2020). Retrieved 21 October 2020, from <https://apps.apple.com/us/app/rentpal-property-manager-rent-manager-rental-management/id1028943100>

Ward 09. (2020). Retrieved 21 October 2020, from <https://www.kathmandu.gov.np/sites/kathmandu.gov.np/files/ward%209.pdf>