Higher Nationals in Computing

Mobile Application Design and Development

Coursework 2

Term 1

Learner's name: *Nguyen Duc Hoang*

Assessor name: *Ho Nguyen Phu Bao*

Class: TCS2102A

Learner's ID: *GCS18383*

Subject's ID: *COMP1786*

Assignment due: *Oct 2021*

Assignment submitted: *Nov 2021*

Report

1. Basic information

1.1 Student name	**Nguyen Duc Hoang**
1.2 Who did you work with?	**Name:** **Login id:**
1.3 Which Exercise is this?	**Create a Flutter using the Notification API**
1.4 How well did you complete the exercise?	• I tried but couldn't finish it ☐ • I did it but I think I should have done better ☐ • I did everything that asked me to do ☒ • I did greater than turned into requested for ☐
1.5 Briefly explain your answer to question 1.4	This task is quite easy, but I need a time to improve utilizing the Notification API.

1.1 Student name	**Nguyen Duc Hoang**
1.2 Who did you work with?	**Name:** **Login id:**
1.3 Which Exercise is this?	**Create a Flutter data entry screen**
1.4 How well did you complete the exercise?	• I tried but couldn't finish it ☐ • I did it but I think I should have done better ☒ • I did everything that asked me to do ☐ • I did greater than turned into requested for ☐

1.5	Briefly explain your answer to question 1.4	To create form, I need a knowledge of Flutter, so I need to clean my code as well as possible.

1.1	Student name	**Nguyen Duc Hoang**
1.2	Who did you work with?	**Name:** **Login id:**
1.3	Which Exercise is this?	**Create Firebase to store the event details entered in the RentalZ App**
1.4	How well did you complete the exercise?	• I tried but couldn't finish it ☐ • I did it but I feel I should have done better ☒ • I did everything that asked me to do ☐ • I did greater than turned into requested for ☐
1.5	Briefly explain your answer to question 1.4	I have deeply learned Web database using Firebase to store the data. But the problem has slightly applied them to my app. But more project its help me to improve in the future.

1.1	Student name	**Nguyen Duc Hoang**
1.2	Who did you work with?	**Name: Bui Van Thieu** **Login id: GCS18545**
1.3	Which Exercise is this?	**Create Native Android data entry screen**

1.4	How well did you complete the exercise?	• I tried but couldn't finish it ☐ • I did it but I think I should have done better ☐ • I did everything that asked me to do ☒ • I did greater than turned into requested for ☐
1.5	Briefly explain your answer to question 1.4	I work with my friend and follow the guideline by my tutor. When I done the exercise, I understand Java coded how it how, how the android created in the past by using Java.

2. Exercise answer

2.1 Screenshots showing what I have achieved

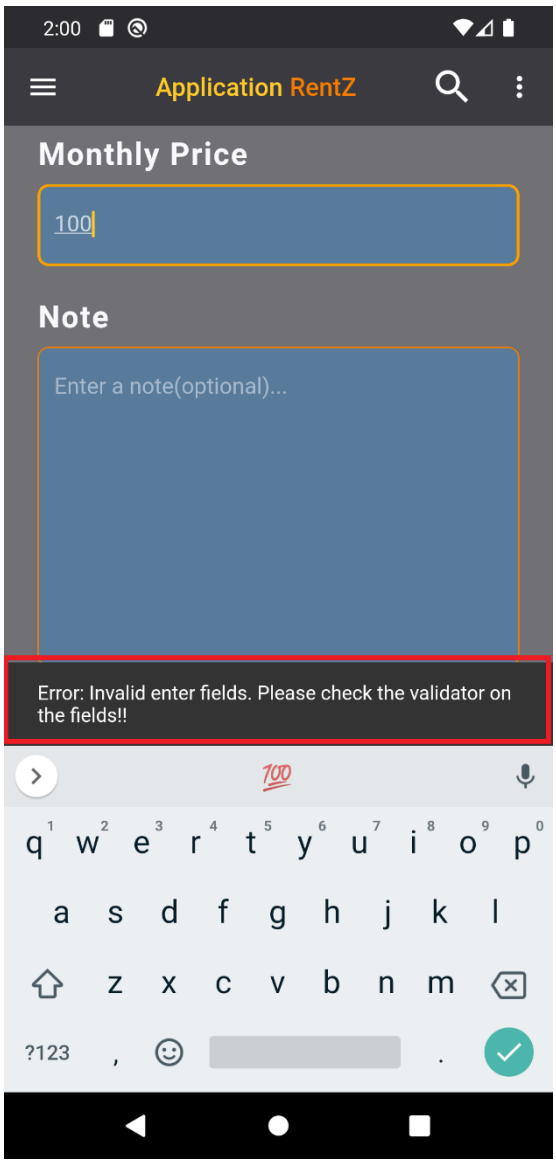

Figure 1: Error message

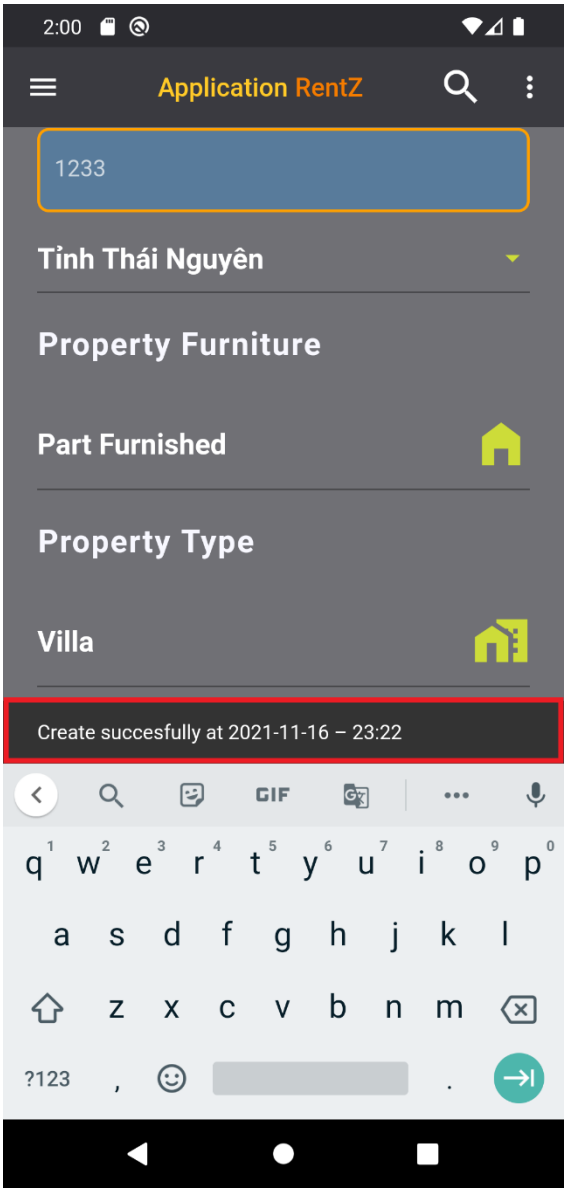

Figure 2: Successfully message

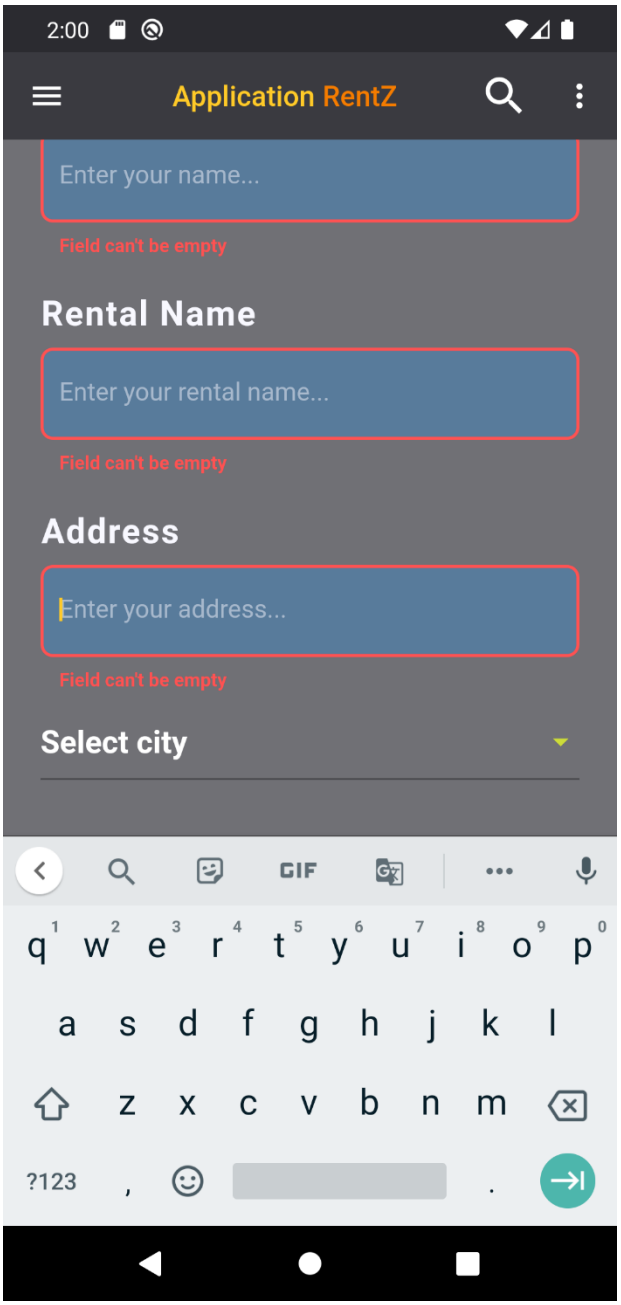

Figure 3: Form validator

After the user confirm to create new property if the fields receive of the error data so the notification should be displayed an error message (Figure 1).

The user needs to enter the fields, which is required “Field can’t be empty”. Also, the fields must be more than 3 than character that should be display “Field should be greater than 3 characters”.

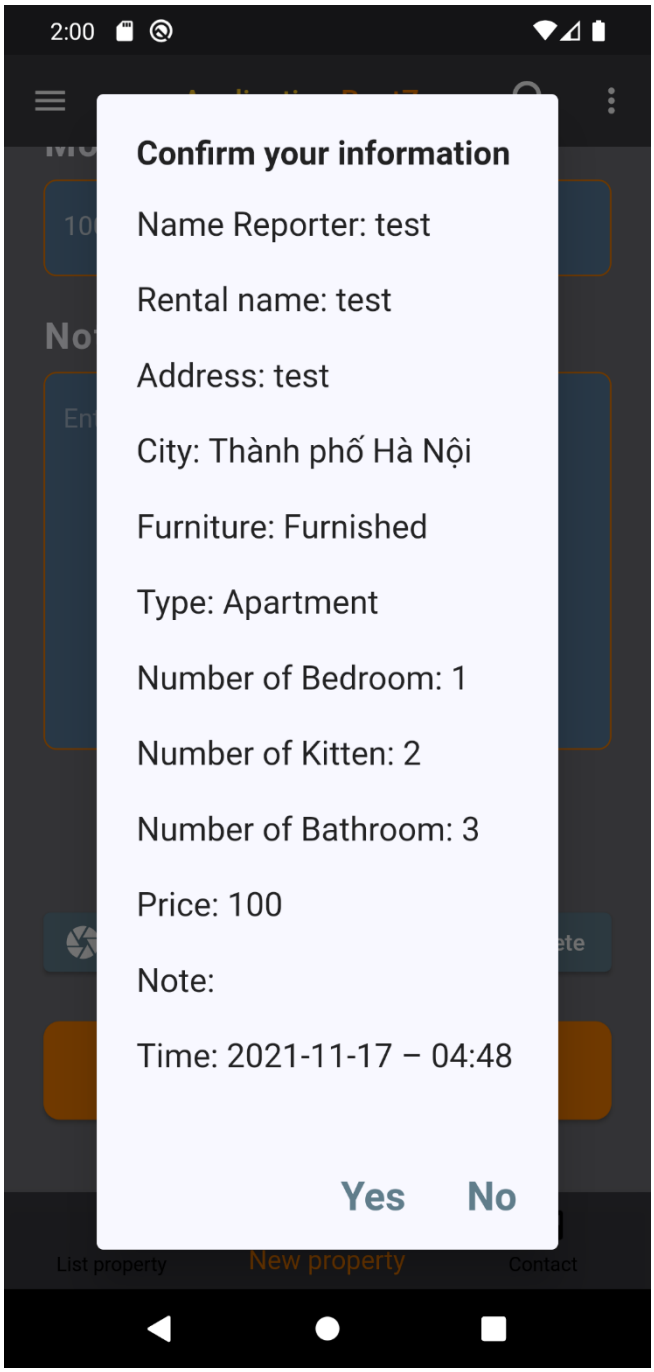

Figure 4: Show confirm dialog

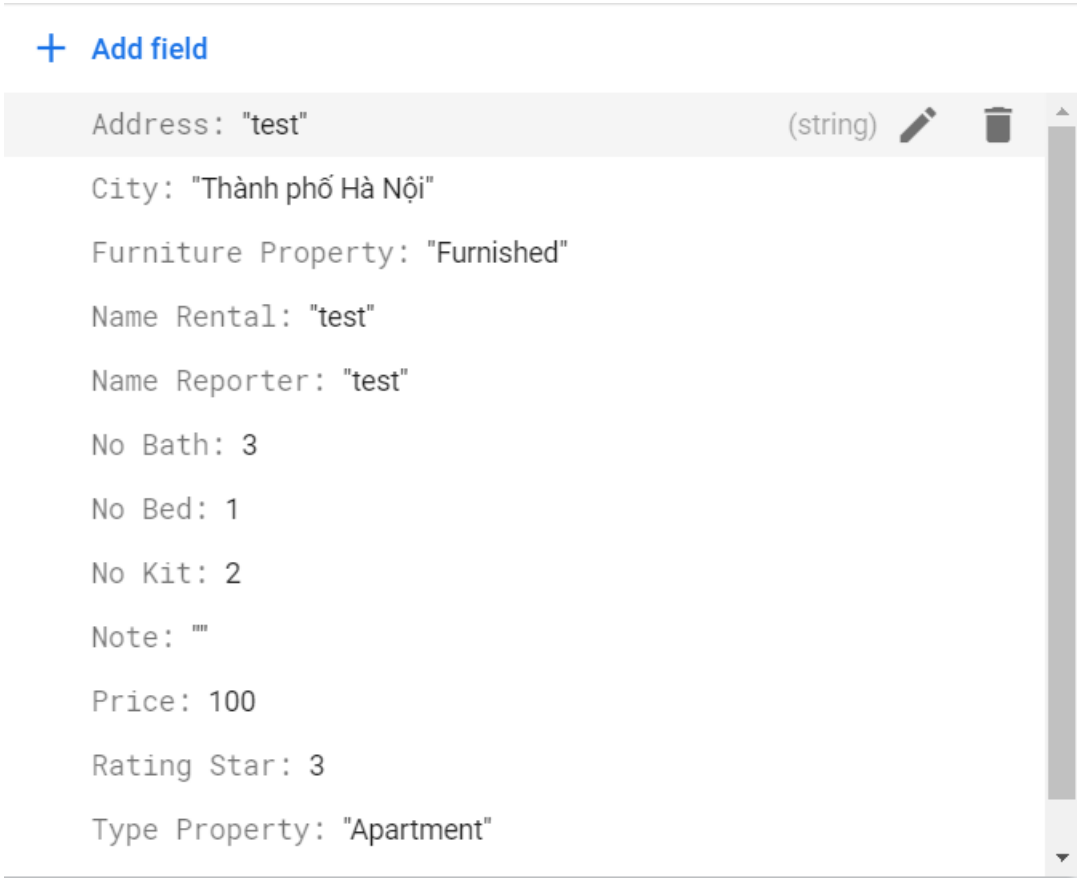

Figure 5: Field of cloud firebase firestore

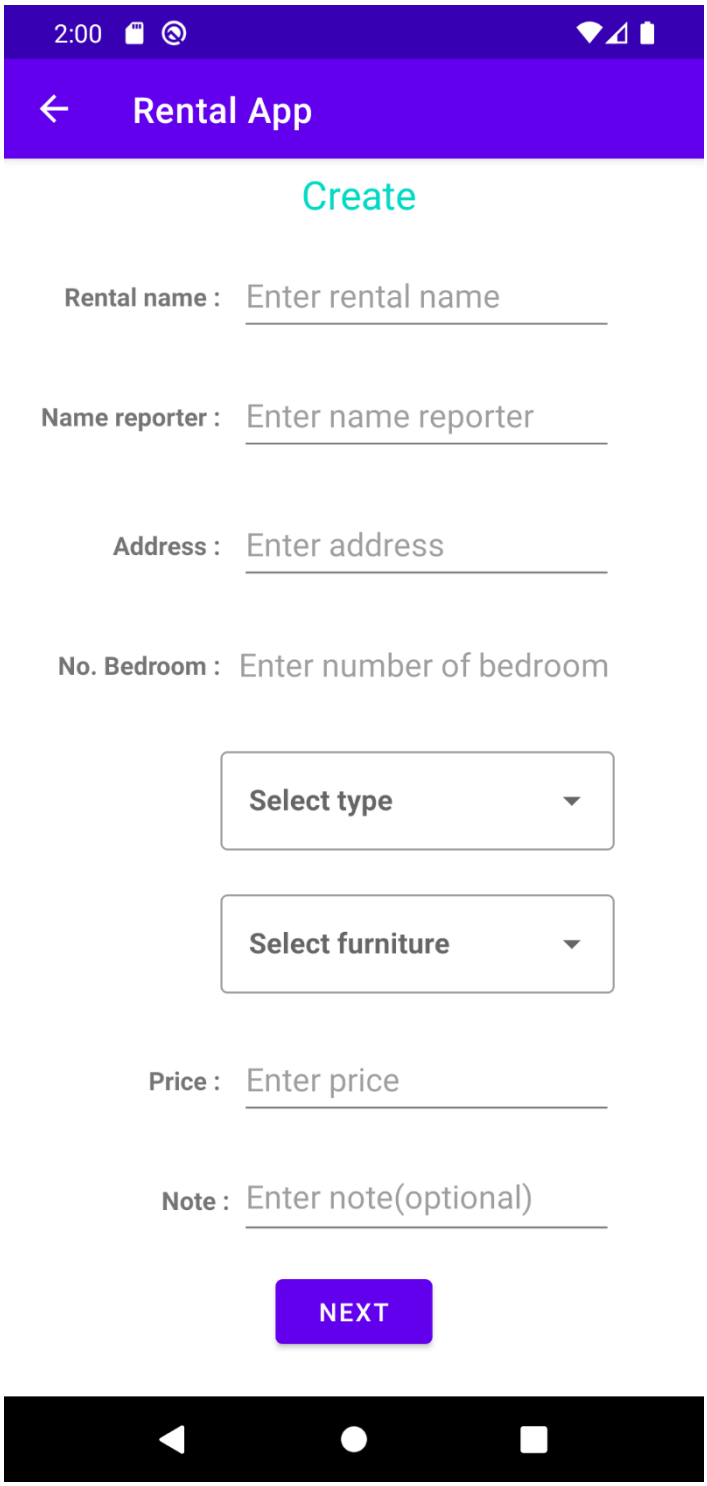

Figure 6: Form entry data (native)

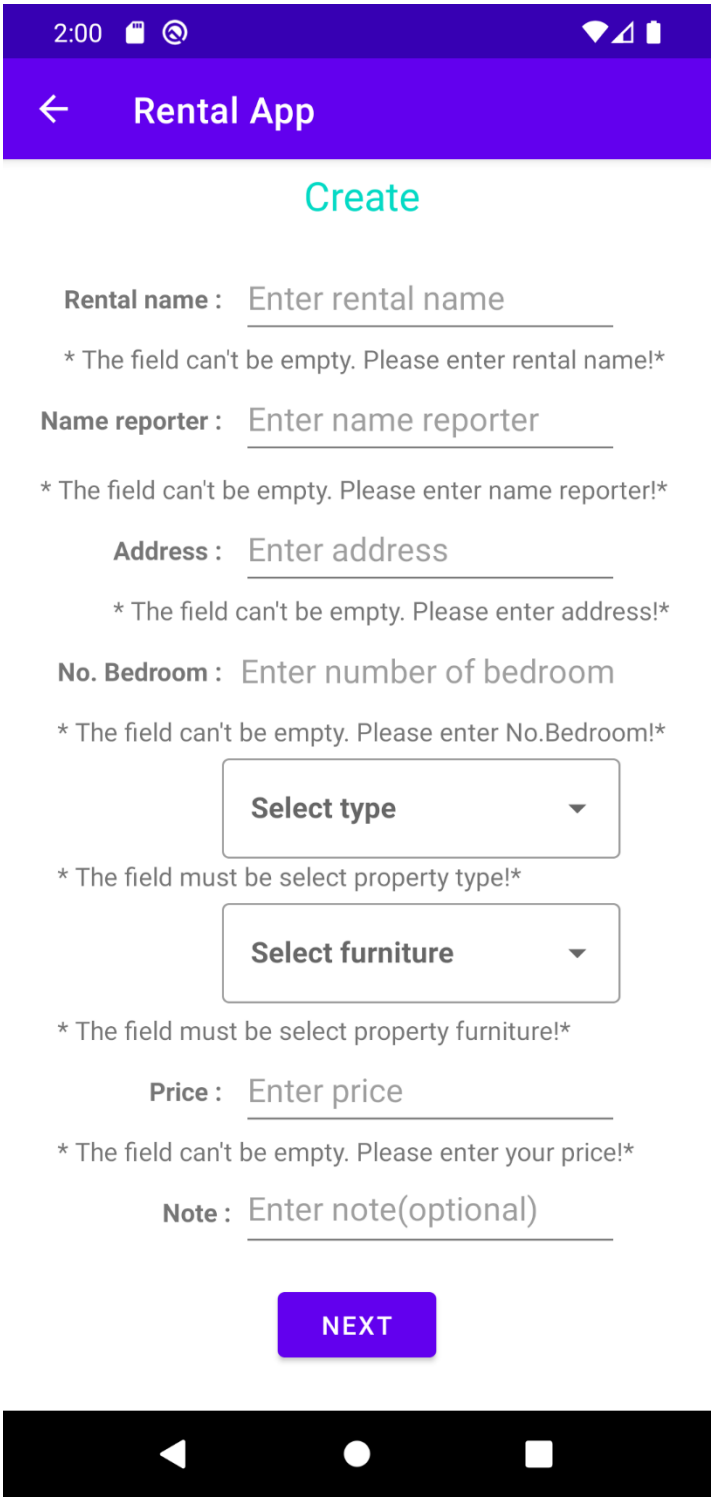

Figure 7: Validate form

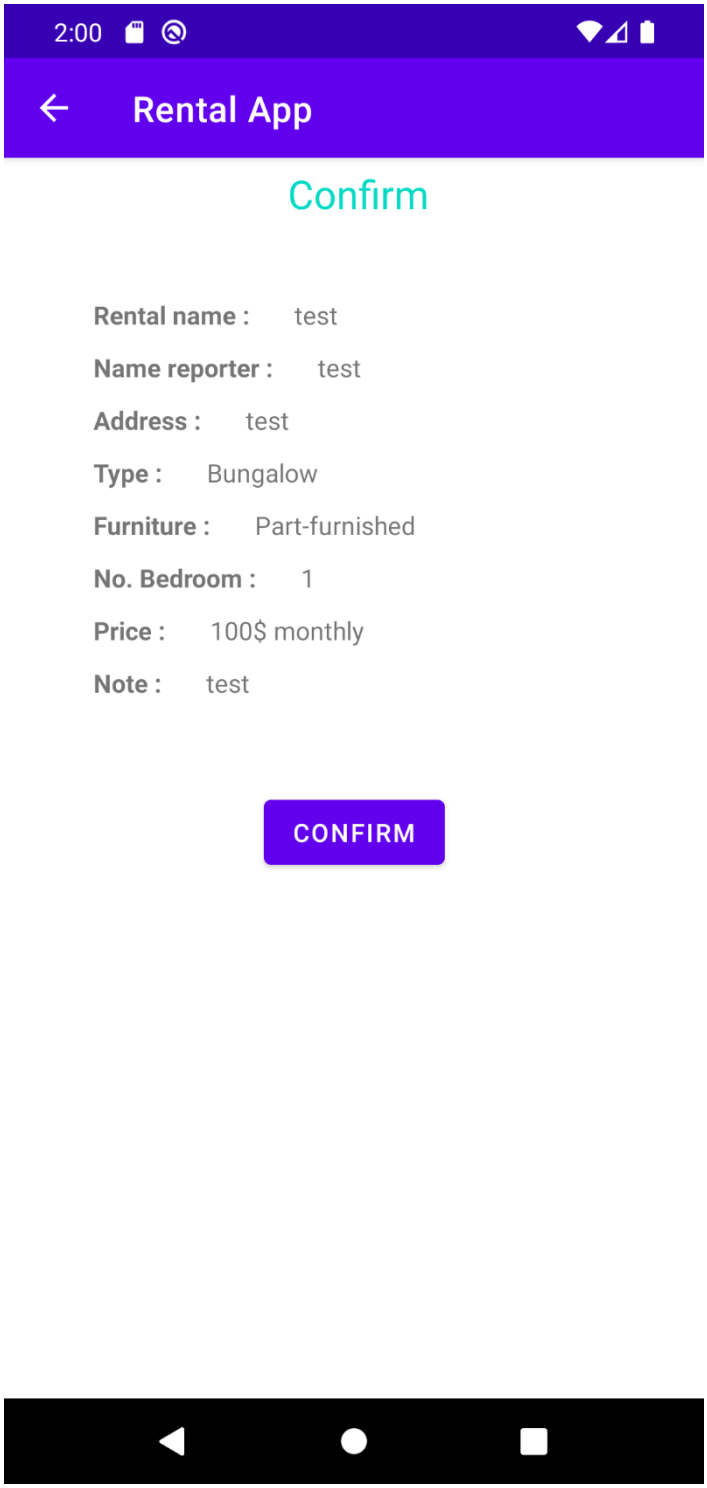

Figure 8: Confirm form

Figure 9: Created successfully

2.2 Code that you wrote

To display notification API, first the user needs to confirm the data entry before.

If the data entry at a currentState is valid, the Notification display “Create successfully ${at current time}”.

Else if invalid, the Notification display “"Error: Invalid enter fields. Please check the validator on the fields!!”

```
if (_addItemFormKey.currentState!.validate()) {
    setState(() {
      ScaffoldMessenger.of(_context).showSnackBar(SnackBar(
          content:
              Text('Create succesfully at $formatDate')));
      Navigator.of(_context).pop();
      _isProcessing = true;
    });
    await Databases.addData(
        rentalName: _rentalNameController.text,
        address: _addressController.text,
        city: _cityController.toString(),
        furniture: _furnitureController.toString(),
        type: _typeController.toString(),
        numBath: int.parse(_numBathController.text),
        numBed: int.parse(_numBedController.text),
        numKit: int.parse(_numKitController.text),
        star: rating,
        price: int.parse(_priceController.text),
        note: _noteController.text,
        nameOwn: _nameReporterController.text,
        createdTime: formatDate);
    Navigator.of(_context).pop();
  }


Navigator.of(_context).pop();ScaffoldMessenger.of(_context).showSnackBar(SnackBar
(
      content: Text(
          'Error: Invalid enter fields. Please check the validator on the
fields!!')));
});
```

In flutter, we can design custom form easy for better UI. I also created custom form for each property it, it will make clear for my code.

Custom Form in Flutter:

```
class CustomFormField extends StatelessWidget {
  const CustomFormField({
    Key? key,
```

```
    required TextEditingController controller,
    required FocusNode focusNode,
    required TextInputType keyboardType,
    required TextInputAction inputAction,
    required String label,
    required String hint,
    required Function(String value) validator,
    this.isObscure = false,
    this.isCapitalized = false,
    this.maxLines = 1,
    this.isLabelEnabled = true,
  })  : _emailController = controller,
        _emailFocusNode = focusNode,
        _keyboardtype = keyboardType,
        _inputAction = inputAction,
        _label = label,
        _hint = hint,
        _validator = validator,
        super(key: key);

  final TextEditingController _emailController;
  final FocusNode _emailFocusNode;
  final TextInputType _keyboardtype;
  final TextInputAction _inputAction;
  final String _label;
  final String _hint;
  final bool isObscure;
  final bool isCapitalized;
  final int maxLines;
  final bool isLabelEnabled;
  final Function(String) _validator;

  @override
  Widget build(BuildContext context) {
    return TextFormField(
      maxLines: maxLines,
      controller: _emailController,
      focusNode: _emailFocusNode,
      keyboardType: _keyboardtype,
      obscureText: isObscure,
      textCapitalization:
      isCapitalized ? TextCapitalization.words : TextCapitalization.none,
      textInputAction: _inputAction,
      cursorColor: CustomColors.firebaseYellow,
      validator: (value) => _validator(value!),
      style: TextStyle(
        color: Colors.white70
      ),
      decoration: InputDecoration(
        labelText: isLabelEnabled ? _label : null,
        labelStyle: TextStyle(color: CustomColors.firebaseYellow),
        hintText: _hint,
        hintStyle: TextStyle(
          color: CustomColors.firebaseWhite.withOpacity(0.5),
        ),
        fillColor: Colors.blue.withOpacity(0.3),
        filled: true,
        errorStyle: TextStyle(
```

```
          color: Colors.redAccent,
          fontWeight: FontWeight.bold,
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(8.0),
          borderSide: BorderSide(
            color: CustomColors.firebaseAmber,
            width: 2,
          ),
        ),
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(8.0),
          borderSide: BorderSide(
            color: CustomColors.firebaseOrange,
          ),
        ),
        errorBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(8.0),
          borderSide: BorderSide(
            color: Colors.redAccent,
            width: 2,
          ),
        ),
        focusedErrorBorder: OutlineInputBorder(
          borderRadius: BorderRadius.circular(8.0),
          borderSide: BorderSide(
            color: Colors.redAccent,
            width: 2,
          ),
        ),
      ),
    );
  }
}
```

SizeBox() is distance between widgets, we can choose height or width as horizontal or vertical distance.

Text() widget display a text with style we can modify the text style as below.

```
SizedBox(height: 24.0),
Text(
  'Your name',
  style: TextStyle(
    color: CustomColors.firebaseWhite,
    fontSize: 22.0,
    letterSpacing: 1,
    fontWeight: FontWeight.bold,
  ),
),
SizedBox(height: 8.0),
CustomFormField(
  isLabelEnabled: false,
  controller: _nameReporterController,
  focusNode: widget.nameReporterFocusNode,
  keyboardType: TextInputType.text,
```

```
  inputAction: TextInputAction.done,
  validator: (value) => Validator.validateField(
    value: value,
  ),
  label: 'Name',
  hint: 'Enter your name...',
),
```

In the CustomFormField() The controller which control the text user enter and validator => we valiate at valiateField in FormState.

To validate the data entry, we need to define a GlobalKey<Class> to validate the value. Here I create a class call FormState in dart.

```
final _addItemFormKey = GlobalKey<FormState>();
```

Class FormState:

```
class Validator {
  static String? validateField({required String value}) {
    if (value.isEmpty) {
      return 'Field can\'t be empty';
    }
    else if(value.length <= 3){
      return 'Field should be greater than 3 character';
    }
    return null;
  }

  static String? validateEmail({required String email}) {
    Pattern pattern =
        r"^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]"
        r"{0,253}[a-zA-Z0-9])?(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]"
        r"{0,253}[a-zA-Z0-9])?)*$";
    RegExp regex = new RegExp('$pattern');
    if (email.isEmpty) {
      return 'User ID can\'t be empty';
    } else if (email.length <= 5) {
      return 'User ID should be greater than 5 characters';
    }
    if(!regex.hasMatch(email)){
      return 'Enter a valid email address';
    }

    return null;
  }


  static String? validatePassword({required String password}) {
    if (password.isEmpty) {
      return 'Password can\'t be empty';
    }
```

```
    else if (password.length <= 8) {
      return 'Password should be greater than 8 characters';
    }

    return null;
  }

  static String? validateNumber({required String value}){
    if(value.isEmpty)
      {
        return 'Field can\'t be empty';
      }
    final number = num.tryParse(value);
    if(number!.isNaN){
      return 'Field can\'t be string. Please enter a number !';
    }
    if(number >= 10 ) {
      return 'Field can\'t more than 10 numbers. Please enter again!';
    }
  }
  static String? validatePrice({required String value}){
    if(value.isEmpty)
    {
      return 'Field can\'t be empty';
    }
    final number = num.tryParse(value);
    if(number!.isNaN){
      return 'Field can\'t be string. Please enter a number !';
    }
    if(number <= 10) {
      return 'Field can\'t less than 10\$. Please enter again!';
    }
    if(number >= 1000) {
      return 'Field can\'t more than 1000\$. Please enter again!';
    }
  }
  static String? validateOptional({required String value}){
    if(value.isEmpty){
      return null;
    }
  }
}
```

To connect to Firebase, we need add packages in pubspect.yami to use them.

- Cloud_firestor
- Firebase_core
- Firebase_auth

```
dependencies:
  flutter:
    sdk: flutter
```

```
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.2
  cloud_firestore: ^2.5.3
  firebase_core: ^1.7.0
  provider: ^6.0.1
  image_picker: ^0.8.4+2
  path_provider: ^2.0.5
  firebase_storage: ^10.0.5
  firebase_auth: ^3.1.3
  animations: ^2.0.2
  http: ^0.13.4
  flutter_rating_bar: ^4.0.0
  equatable: ^2.0.0
  flutter_bloc: ^7.3.3
  get: any
  filter_list: any
```

First, I reference an instance of Firebase Firestore call _mainConllection and instance of Authentication call _auth.

```
final CollectionReference _mainCollection =
FirebaseFirestore.instance.collection('data');
final FirebaseAuth _auth = FirebaseAuth.instance;
```

```
class Databases{

  static Future<void> addData({
    required String nameOwn,
    required String rentalName,
    required String address,
    required String city,
    required String furniture,
    required String type,
    required int numBed,
    required int numKit,
    required int numBath,
    required int price,
    required double star,
    required String note,
    required String createdTime
  }) async{
    DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('rent').doc();

    Map<String, dynamic> apartmentData = <String,dynamic>{
        'Name Reporter' : nameOwn,
        'Name Rental' : rentalName,
        'Address': address,
        'City': city,
        'Furniture Property': furniture,
        'Type Property' : type,
        'No Bed' : numBed,
```

```
        'No Kit': numKit,
        'No Bath' : numBath,
        'Price' : price,
        'Note': note,
        'Rating Star': star,
        'createdTime': createdTime
    };
    await docApartment
        .set(apartmentData).whenComplete(() => print("The rental update
$rentalName"))
        .catchError((error) => print(error));
  }
  static Stream<QuerySnapshot> readData() {
    CollectionReference docApartment =
    _mainCollection.doc(_auth.currentUser!.uid).collection('rent');
    return docApartment.snapshots();
  }


  static Future<void> updateData({
    required String docId,
    required String nameOwn,
    required String rentalName,
    required String address,
    required String city,
    required String furniture,
    required String type,
    required int numBed,
    required int numKit,
    required int numBath,
    required int price,
    required String note,
    required double star,
    required String updatedTime
  }) async{
    DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('rent').doc(docId);

    Map<String, dynamic> apartmentData = <String,dynamic>{
      'Name Reporter' : nameOwn,
      'Name Rental' : rentalName,
      'Address': address,
      'City': city,
      'Furniture Property': furniture,
      'Type Property' : type,
      'No Bed' : numBed,
      'No Kit': numKit,
      'No Bath' : numBath,
      'Price' : price,
      'Note': note,
      'Rating Star': star,
      'createdTime': updatedTime
    };
    await docApartment
        .set(apartmentData).whenComplete(() => print("The rental added
$rentalName"))
        .catchError((error) => print(error));
  }
```

```
  static Future<void> deleteData({
    required String docId
    }) async{
    DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('rent').doc(docId);

    await docApartment
        .delete().whenComplete(() => print("The rental deleted"))
        .catchError((error) => print(error));
  }

  static Future<void> addNote({
    required String note,
    required String createdTime
  }) async{
    DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('note').doc();

    Map<String, dynamic> apartmentData = <String,dynamic>{
      'NoteRef': note,
      'createdTime': createdTime
    };
    await docApartment
        .set(apartmentData).whenComplete(() => print("The Note update $note"))
        .catchError((error) => print(error));
  }

  static Stream<QuerySnapshot> readNote() {
    CollectionReference docApartment =
    _mainCollection.doc(_auth.currentUser!.uid).collection('note');
    return docApartment.snapshots();
  }
}
```

Adding the data to the snapshot:

```
static Future<void> addData({
  …
}) async{
  DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('rent').doc();

  Map<String, dynamic> apartmentData = <String,dynamic>{
      'Name Reporter' : nameOwn,
      'Name Rental' : rentalName,
      'Address': address,
      'City': city,
      'Furniture Property': furniture,
      'Type Property' : type,
      'No Bed' : numBed,
```

```
      'No Kit': numKit,
      'No Bath' : numBath,
      'Price' : price,
      'Note': note,
      'Rating Star': star,
      'createdTime': createdTime
  };
  await docApartment
      .set(apartmentData).whenComplete(() => print("The rental update
$rentalName"))
      .catchError((error) => print(error));
```

Update data form a snapshot:

```
static Future<void> updateData({
  …
}) async{
  DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('rent').doc(docId);

  Map<String, dynamic> apartmentData = <String,dynamic>{
    'Name Reporter' : nameOwn,
    'Name Rental' : rentalName,
    'Address': address,
    'City': city,
    'Furniture Property': furniture,
    'Type Property' : type,
    'No Bed' : numBed,
    'No Kit': numKit,
    'No Bath' : numBath,
    'Price' : price,
    'Note': note,
    'Rating Star': star,
    'createdTime': updatedTime
  };
  await docApartment
      .set(apartmentData).whenComplete(() => print("The rental added
$rentalName"))
      .catchError((error) => print(error));
}
```

Deleting data form snapshot:

```
static Future<void> deleteData({
  required String docId
  }) async{
  DocumentReference docApartment =
_mainCollection.doc(_auth.currentUser!.uid).collection('rent').doc(docId);

  await docApartment
      .delete().whenComplete(() => print("The rental deleted"))
      .catchError((error) => print(error));
}
```

Reading data form snapshot:

```
static Stream<QuerySnapshot> readData() {
  CollectionReference docApartment =
  _mainCollection.doc(_auth.currentUser!.uid).collection('rent');
  return docApartment.snapshots();
}
```

Finally, after we add the property, we apply the static Future to the SteamProvider to read the data:

To List Property:

```
return StreamBuilder<QuerySnapshot>(
  stream: Databases.readData(),
  builder: (context, snapshot) {
    if (snapshot.hasError) {
      return Text('Something went wrong');
    } else if (snapshot.hasData || snapshot.data != null) {
      return ListView.separated(
        separatorBuilder: (context, index) => SizedBox(height: 16.0),
        itemCount: snapshot.data!.docs.length,
        itemBuilder: (context, index) {
          var noteInfo = snapshot.data!.docs[index];
          String docID = snapshot.data!.docs[index].id;
          String nameApm = noteInfo.get('Name Rental');
          String address = noteInfo.get('Address');
          String city = noteInfo.get('City');
          String furniture = noteInfo.get('Furniture Property');
          String type = noteInfo.get('Type Property');
          int numBed = noteInfo.get('No Bed');
          int numKit = noteInfo.get('No Kit');
          int numBath = noteInfo.get('No Bath');
          int price = noteInfo.get('Price');
          String nameReporter = noteInfo.get('Name Reporter');
          String note = noteInfo.get('Note');
          double ratingStar = noteInfo.get('Rating Star');
          String createdTime = noteInfo.get('createdTime');
```

To detail:

```
return StreamBuilder<QuerySnapshot>(
    stream: Databases.readData(),
    builder: (context, snapshot) {
      final address = widget.data.get('Address');
      final city = widget.data.get('City');
      final nameApm = widget.data.get('Name Rental');
      final numBed = widget.data.get('No Bed');
      final numKit = widget.data.get('No Kit');
      final numBath = widget.data.get('No Bath');
      final price = widget.data.get('Price');
```

```
      final note = widget.data.get('Note');
      final furniture = widget.data.get('Furniture Property');
      final type = widget.data.get('Type Property');
      final nameReporter = widget.data.get('Name Reporter');
      final createdTime = widget.data.get('createdTime');
      final ratingStar = widget.data.get('Rating Star');
```

First, I created FormActivity have functions as below:

To set the text to the dropdown, I created array String “type” and “furniture” and refer to the AutoCompleTextView, after that they define in the ArrayAdapter<String>

```
    String[] types = {"Flat", "House", "Pen House", "Bungalow"};
    String[] furniture = {"Furnished", "Unfurnished", "Part-furnished"};
//    getResources().getStringArray(R.array.types)

    AutoCompleteTextView autoCompleteTypeTextView;
    AutoCompleteTextView autoCompleteFurnitureTextView;

    ArrayAdapter<String> adapterTypeItems;
    ArrayAdapter<String> adapterFurnitureItems;
```

On the onCreate function:

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_property);


        autoCompleteTypeTextView = findViewById(R.id.selectType);
        autoCompleteFurnitureTextView = findViewById(R.id.selectFurniture);
        adapterTypeItems = new ArrayAdapter<String>(this, R.layout.list_item,
types);
        adapterFurnitureItems = new ArrayAdapter<String>(this,
R.layout.list_item, furniture);
        autoCompleteTypeTextView.setAdapter(adapterTypeItems);
        autoCompleteFurnitureTextView.setAdapter(adapterFurnitureItems);
        autoCompleteTypeTextView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                String type = parent.getItemAtPosition(position).toString();
                Toast.makeText(getApplicationContext(), "Type: "+type,
Toast.LENGTH_SHORT).show();
            }
        });
        autoCompleteFurnitureTextView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                String furniture = parent.getItemAtPosition(position).toString();
                Toast.makeText(getApplicationContext(), "Furniture: "+furniture,
Toast.LENGTH_SHORT).show();
```

```
            }
        });

//          String btnNextName = getResources().getString(R.string.tv_nextBtn);
        Button btnNextForm = findViewById(R.id.btnNextForm);
        btnNextForm.setOnClickListener(_btnNextClick);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

    }
```

To create button to next Form, I created _btnNextClick = new View.OnClickListener().

```
private View.OnClickListener _btnNextClick = new View.OnClickListener() {};
```

We need to define isValid that is Boolean type make it's always true.

```
boolean isValid = true;
```

If the input is equal to empty, so "isValid" become is false and showing an error as below:

If isValid = true, input data to the TextView by creating Bundle that we can put string. New Intent to set this text to new Activity.

```
if(TextUtils.isEmpty(rentalName)){
    isValid                                 =                                false;
    errorRentalName += "* The field can't be empty. Please enter rental name!*";
}
if(TextUtils.isEmpty(nameReporter)){
    isValid                                 =                                false;
    errorNameReporter += "* The field can't be empty. Please enter name reporter!*";
}
if(TextUtils.isEmpty(address)){
    isValid                                 =                                false;
    errorAddress += "* The field can't be empty. Please enter address!*";
}
            if(TextUtils.isEmpty(type)){
    isValid                                 =                                false;
    errorSelectType += "* The field must be select property type!*";
}
if(TextUtils.isEmpty(furniture)){
    isValid                                 =                                false;
    errorSelectFurniture += "* The field must be select property furniture!*";
}
if(TextUtils.isEmpty(numBed)){
    isValid                                 =                                false;
    errorNumBed += "* The field can't be empty. Please enter No.Bedroom!*";
}
if(TextUtils.isEmpty(price)){
    isValid                                 =                                false;
```

```
    errorPrice += "* The field can't be empty. Please enter your price!*";
}
```

We use the Intent and Bundle method to get Extras String from FromActivity.

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
```

```
if(bundle != null){
    rentalName = bundle.getString("rentalName");
    nameReporter = bundle.getString("nameReporter");
    address = bundle.getString("address");
    type = bundle.getString("type");
    furniture = bundle.getString("furniture");
    numBed = bundle.getString("numBed");
    price = bundle.getString("price");
    note = bundle.getString("note");
}
```

If the bundle is equal to empty put the String to empty.

```
tvInfoRentalName.setText(rentalName);
tvInfoNameReporter.setText(nameReporter);
tvInfoAddress.setText(address);
tvInfoType.setText(type);
tvInfoFurniture.setText(furniture);
tvInfoNumBed.setText(numBed);
tvInfoPrice.setText(price);
tvInfoNote.setText(note);
```

To Put string to the empty String:

```
String rentalName = "", nameReporter = "", address = "";
String type = "", furniture = "", numBed ="", price = "", note = "";
```

The function is called Back to the Activity

```
public boolean onSupportNavigateUp(){
    finish();
    return true;
}
```