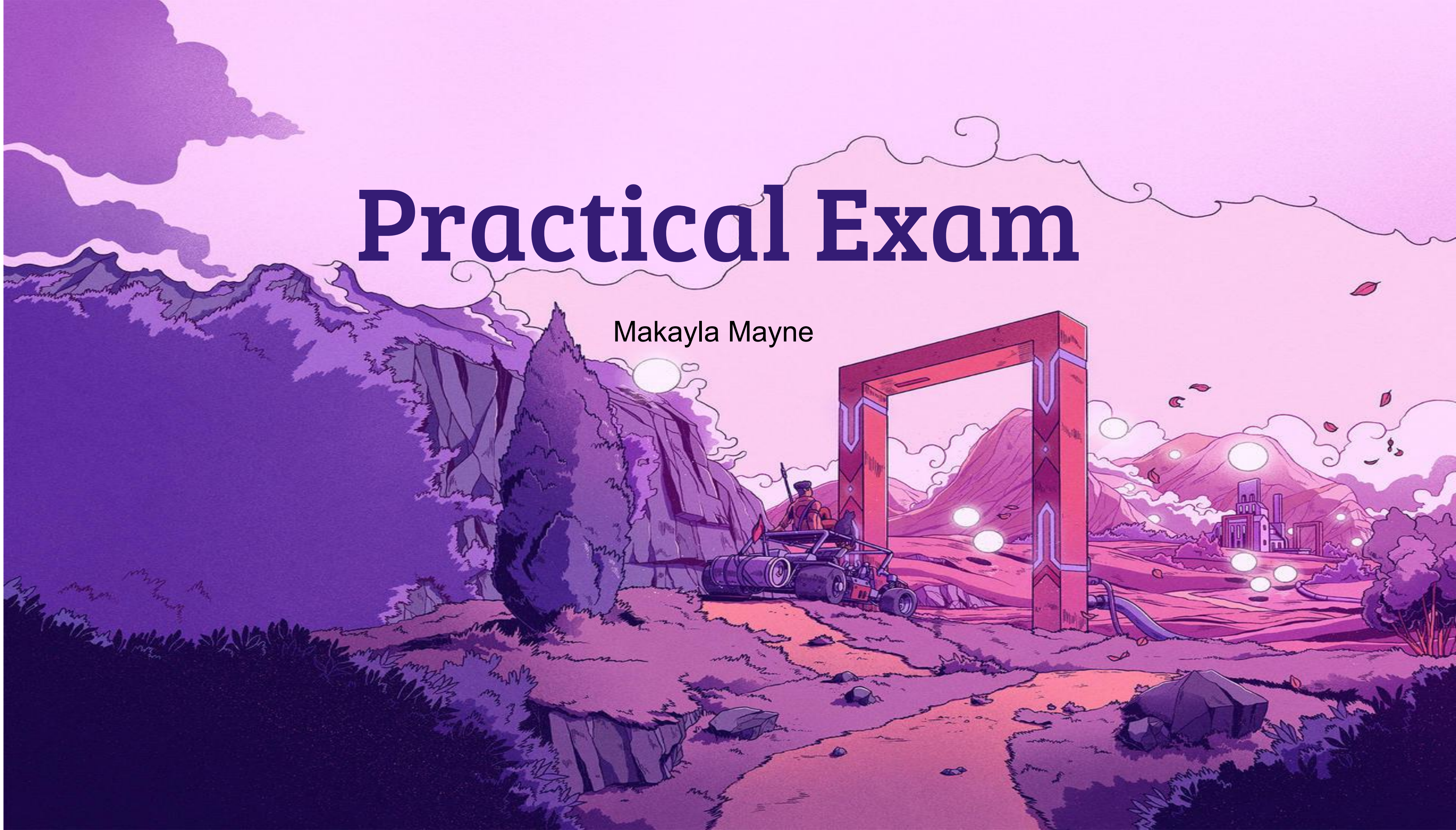
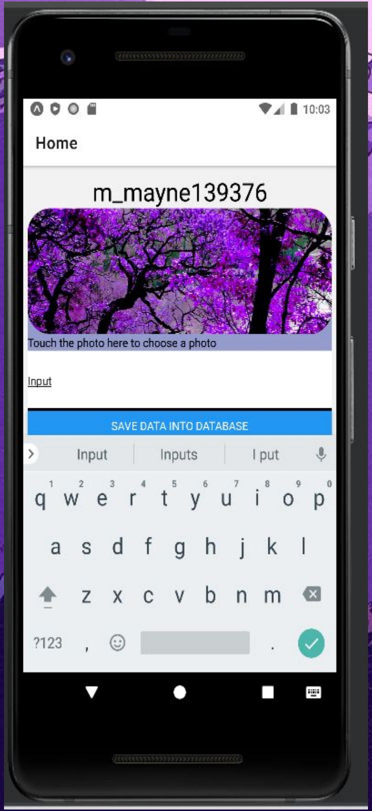
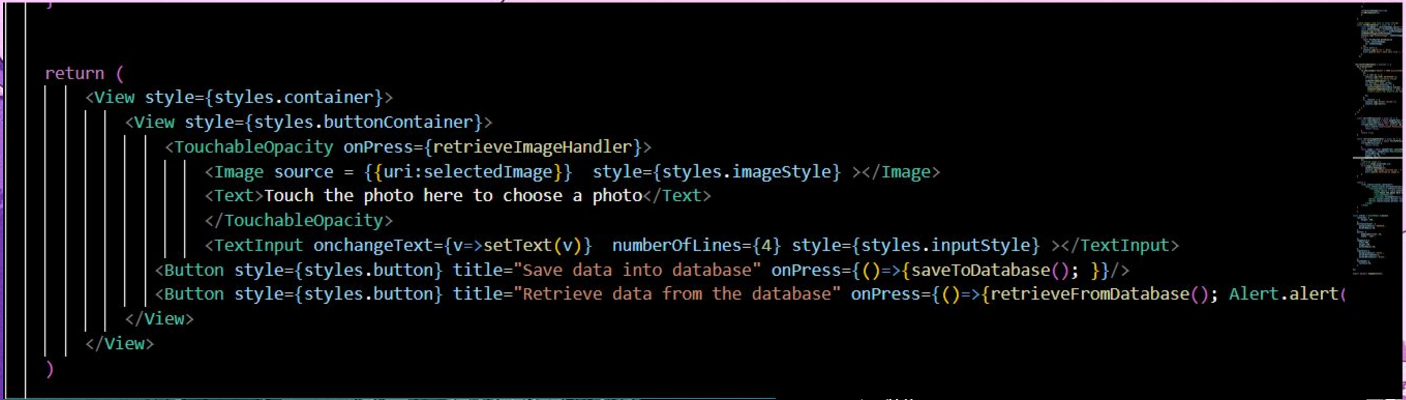
Practical Exam

Makayla Mayne

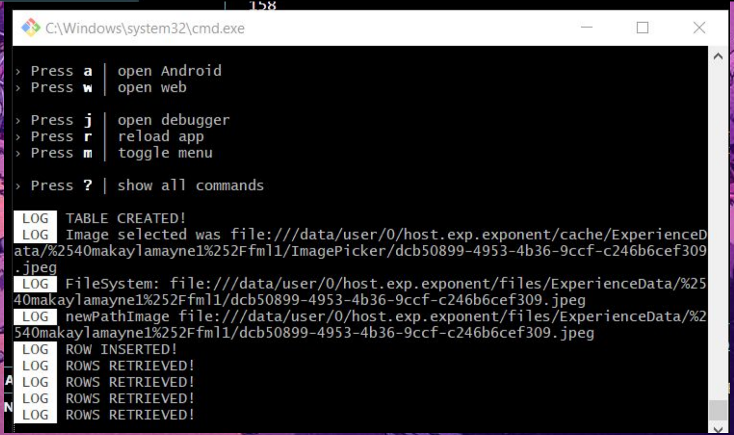


An area for the picture to be displayed



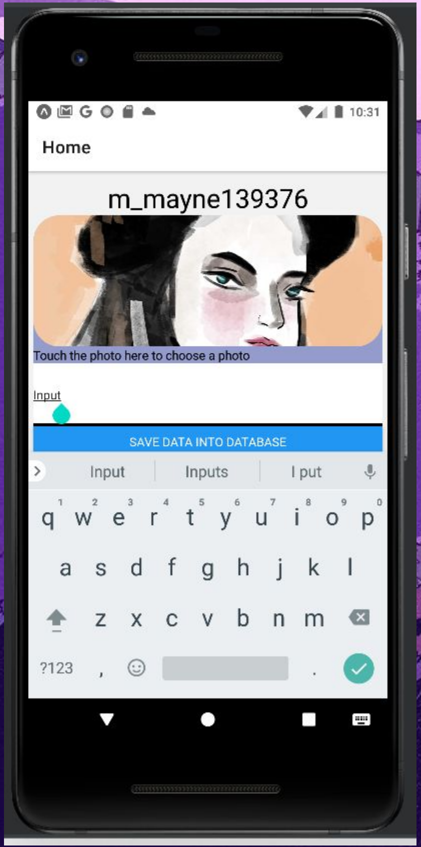


```
return (
    <View style={styles.container}>
        <View style={styles.buttonContainer}>
            <TouchableOpacity onPress={retrieveImageHandler}>
                <Image source = {{uri:selectedImage}}  style={styles.imageStyle} ></Image>
                <Text>Touch the photo here to choose a photo</Text>
                </TouchableOpacity>
                <TextInput onchangeText={v=>setText(v)}  numberOfLines={4} style={styles.inputStyle} ></TextInput>
          <Button style={styles.button} title="Save data into database" onPress={()=>{saveToDatabase(); }}/>
          <Button style={styles.button} title="Retrieve data from the database" onPress={()=>{retrieveFromDatabase(); Alert.alert(
        </View>
    </View>
)
```

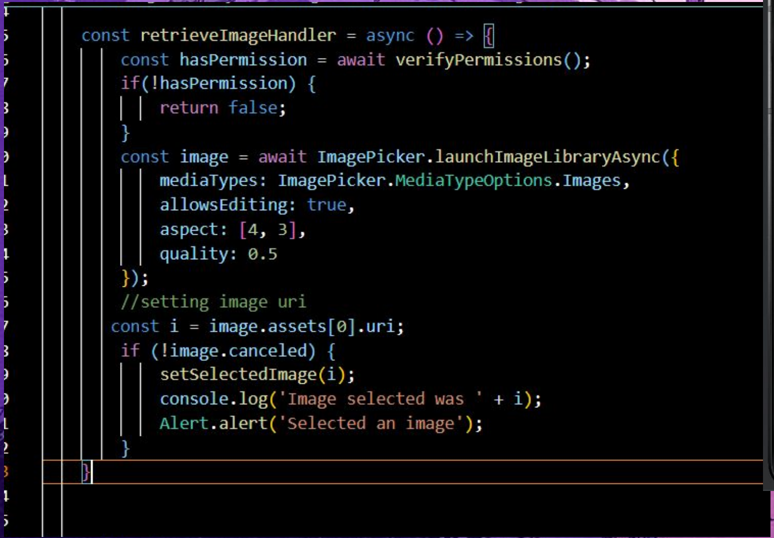


```
C:\Windows\system32\cmd.exe

› Press a │ open Android
› Press w │ open web

› Press j │ open debugger
› Press r │ reload app
› Press m │ toggle menu

› Press ? │ show all commands

 LOG  TABLE CREATED!
 LOG  Image selected was file:///data/user/0/host.exp.exponent/cache/ExperienceD
ata/%2540makaylamayne1%252Ffml1/ImagePicker/dcb50899-4953-4b36-9ccf-c246b6cef309
.jpeg
 LOG  FileSystem: file:///data/user/0/host.exp.exponent/files/ExperienceData/%25
40makaylamayne1%252Ffml1/dcb50899-4953-4b36-9ccf-c246b6cef309.jpeg
 LOG  newPathImage file:///data/user/0/host.exp.exponent/files/ExperienceData/%2
540makaylamayne1%252Ffml1/dcb50899-4953-4b36-9ccf-c246b6cef309.jpeg
 LOG  ROW INSERTED!
 LOG  ROWS RETRIEVED!
 LOG  ROWS RETRIEVED!
 LOG  ROWS RETRIEVED!
 LOG  ROWS RETRIEVED!
```

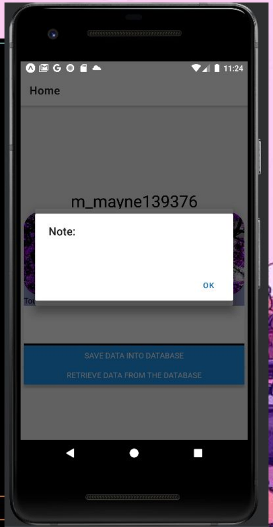
A text input to input a note



Alert to show the notes made previously saved in the URI and the database URI



```
const retrieveImageHandler = async () => {
    const hasPermission = await verifyPermissions();
    if(!hasPermission) {
        return false;
    }
    const image = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.Images,
        allowsEditing: true,
        aspect: [4, 3],
        quality: 0.5
    });
    //setting image uri
   const i = image.assets[0].uri;
    if (!image.canceled) {
        setSelectedImage(i);
        console.log('Image selected was ' + i);
        Alert.alert('Selected an image');
    }
}
```

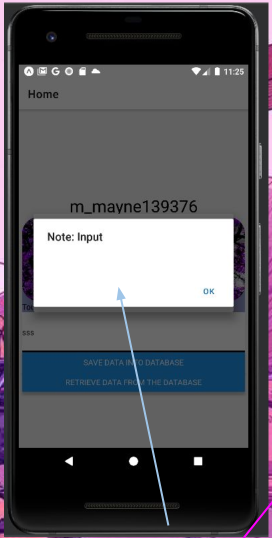


Home

m_mayne139376

Note:

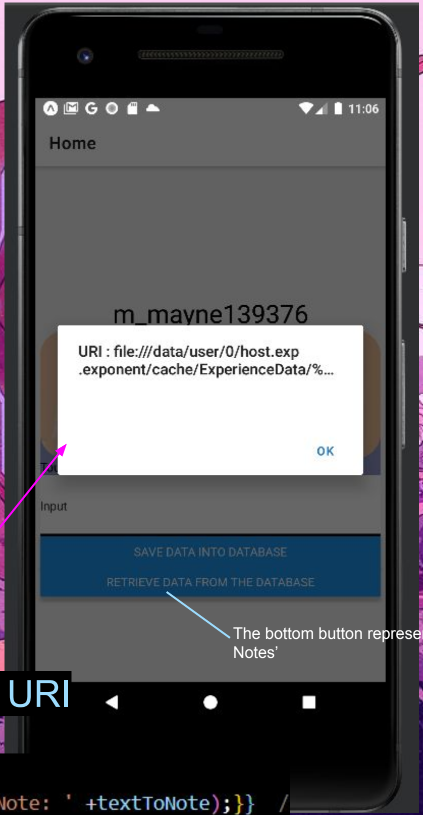
Then after saving more inputs…



Home

m_mayne139376

Note: Input



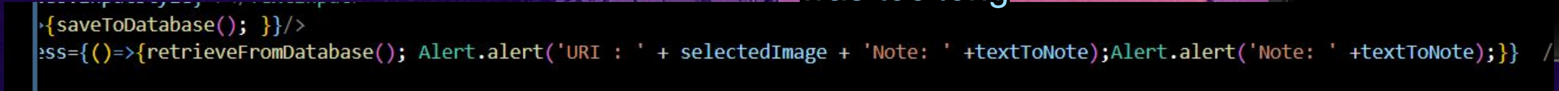
Home

m_mayne139376

URI : file:///data/user/0/host.exp
.exponent/cache/ExperienceData/%...

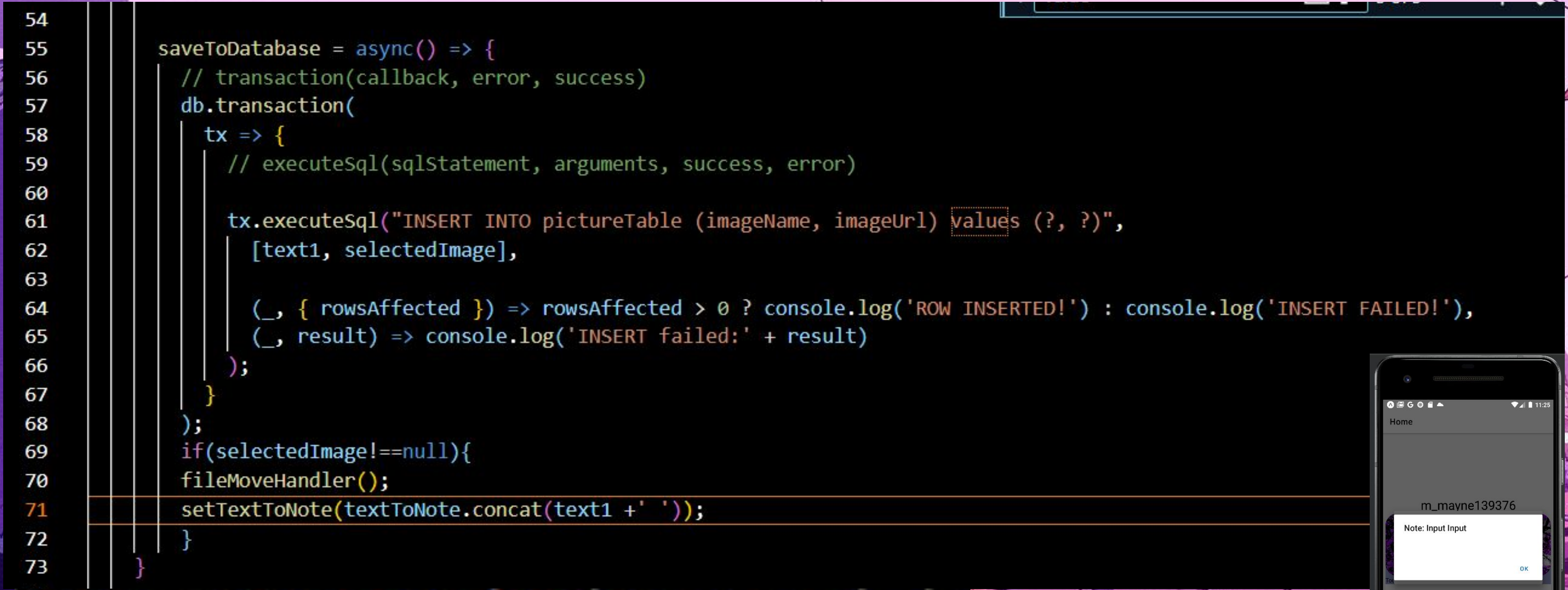
The bottom button represents 'View Notes'

I had to separate into two separate alerts because the URI was too long

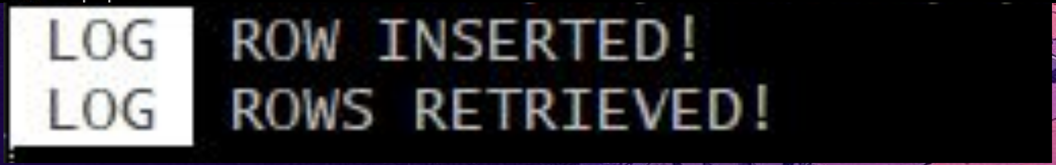


```
{saveToDatabase(); }}/>
ss={()=>{retrieveFromDatabase(); Alert.alert('URI : ' + selectedImage + 'Note: ' +textToNote);Alert.alert('Note: ' +textToNote);}}  /
```

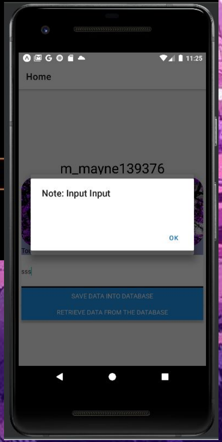
A button that will save the image uri and note



```
    saveToDatabase = async() => {
      // transaction(callback, error, success)
      db.transaction(
        tx => {
          // executeSql(sqlStatement, arguments, success, error)

          tx.executeSql("INSERT INTO pictureTable (imageName, imageUrl) values (?, ?)",
            [text1, selectedImage],

            (_, { rowsAffected }) => rowsAffected > 0 ? console.log('ROW INSERTED!') : console.log('INSERT FAILED!'),
            (_, result) => console.log('INSERT failed:' + result)
          );
        }
      );
      if(selectedImage!==null){
      fileMoveHandler();
      setTextToNote(textToNote.concat(text1 +' '));
      }
  }
```

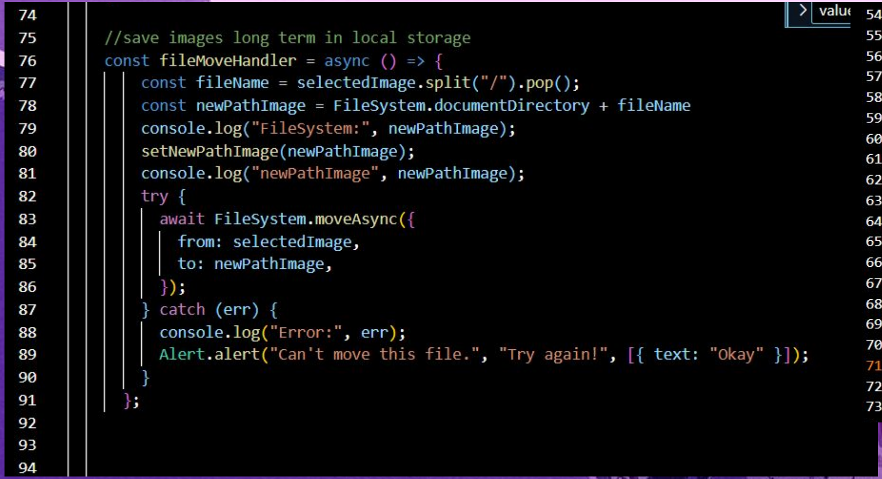


```
LOG  ROW INSERTED!
LOG  ROWS RETRIEVED!
```

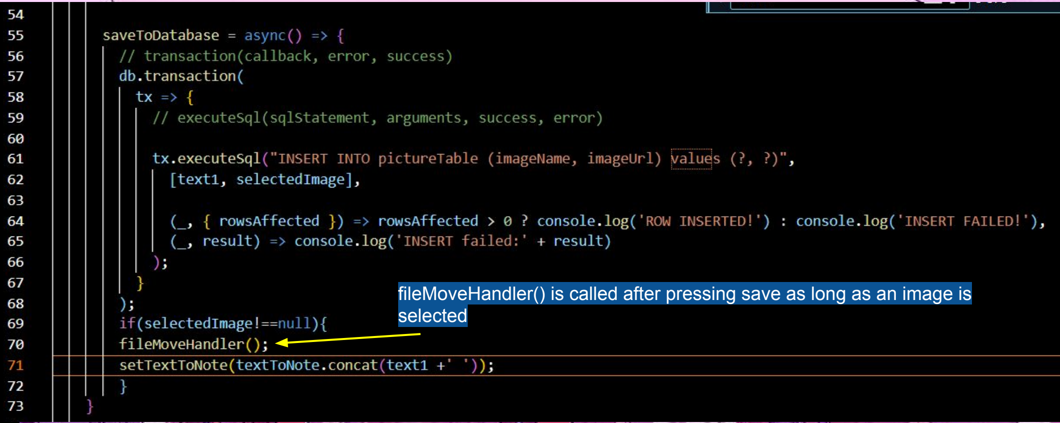


Note: Input Input

FileSystem



```
//save images long term in local storage
const fileMoveHandler = async () => {
    const fileName = selectedImage.split("/").pop();
    const newPathImage = FileSystem.documentDirectory + fileName
    console.log("FileSystem:", newPathImage);
    setNewPathImage(newPathImage);
    console.log("newPathImage", newPathImage);
    try {
      await FileSystem.moveAsync({
        from: selectedImage,
        to: newPathImage,
      });
    } catch (err) {
      console.log("Error:", err);
      Alert.alert("Can't move this file.", "Try again!", [{ text: "Okay" }]);
    }
  };
```



```
54
55    saveToDatabase = async() => {
56      // transaction(callback, error, success)
57      db.transaction(
58        tx => {
59          // executeSql(sqlStatement, arguments, success, error)
60
61          tx.executeSql("INSERT INTO pictureTable (imageName, imageUrl) values (?, ?)",
62            [text1, selectedImage],
63
64            (_, { rowsAffected }) => rowsAffected > 0 ? console.log('ROW INSERTED!') : console.log('INSERT FAILED!'),
65            (_, result) => console.log('INSERT failed:' + result)
66          );
67        }
68      );
69      if(selectedImage!==null){
70      fileMoveHandler();
71      setTextToNote(textToNote.concat(text1 +' '));
72      }
73    }
```

fileMoveHandler() is called after pressing save as long as an image is selected

Then we concatenate the notes that are in the input box (line 71)
To later display in the alert box