

Project Documentation

by NP MTSWENI

Submission date: 31-Oct-2020 06:41PM (UTC+0200)

Submission ID: 1432191514

File name: Documentation_for_DSW_project.docx (2.34M)

Word count: 1200

Character count: 6085

LAUNDREE

We'll take care of your Load

Contents

1. Members	4
1.1. Project Description:	4
1.2. Problem:	4
1.3. Solution:	4
1.3.1. Objectives	4
2. Design	5
2.1. Use Case.....	5
2.2. Use Case design.....	6
.....	6
2.3. Database Tables	7
2.3.1. Main Tables	7
3. Screen layout mockups	8
3.1. User First Screen	8
3.2. Registration Page.....	9
3.2. Login Page	9
3.2.1. Location Page	11
3.2.2. Service Page	13
3.2.3. Checkout Page	14
3.2.4. Profile Page	16
4. Screen Flows.....	17
4.1. Admin First Screen.....	18
4.2. Registration Screen	19
4.3. Login Screen.....	21
4.4. Orders Screen	22
4.5. Categories	23
4.5.1. Accepted Screen	23

4.6. Location Screen	24
4.7. Prices screen	25
4.8. Profile Screen.....	26
5. Database Analysis.....	27
5.1. Debugging.....	27
5.2. Laundry Admin APK analysis	28
5.3. Laundry Admin profile analysis.....	28
5.4. Laundry Client APK analysis	29
5.5. Laundry Client Profile Analysis.....	29
6. Activities.....	30
6.1. OTP code for new users	30
6.2. Resend OTP Code to Admin and Resetting password	30
6.3. Resend OTP Code to user and Resetting password	31
6.4. Current Location.....	31
6.5. Distance between two points	32
6.6. Notification for an accepted order	32
6.7. Notification to verify delivery date	33
7. Database Authentication.....	33
7.1 Firebase connection	33
7.2. OTP and Location Authentication	34

1. Members

Sibusiso Mabena_219005273 (Did not participate)

Olwethu Mlimi_217020779

Tlou Ramotshela_201907197 (Did not participate)

Nozipho Mtsweni_218053022

Cedrick Zulu_201572538 (Did not participate)

1.1. Project Description:

You need choose one platform that you are comfortable with and use it to develop your Mobile Application.

1.2. Problem:

We have noticed that a lot of people struggle with time management, especially students. Having to worry about studying, Assignments and Projects then having a pile of laundry that needs to be attended to. The laundry is the issue here.

1.3. Solution:

This is where the app comes in, we have created an app that helps our users get connected to any laundry companies near them that can do their laundry and they don't have to worry about taking the laundry there because the app offers a service that collects the laundry at the specified location.

1.3.1. Objectives:

- To provide an application that makes our user's lives easier, by saving time.
- To provide the users with an application that is User friendly.
- To provide excellent service to our users.
- To have an application that helps ours users save a lot of money by paying less.

2. Design

2.1. Use Case

- Users and the admin register using their phone number
- Firebase is the internet-based platform used.
- Access the Geolocation (Integrate app with google maps)
- Search for the nearest Laundry company
- Select the Laundry of your choice and send request
- Request gets accepted
- Predefined prices are displayed
- Select the load size
- Select Laundry service (Wash, tumble dry, fold or ironing)
- Select payment method (cash only)
- Schedule a date for laundry to be delivered
- Get a notification on app to confirm the request

2.2. Use Case design

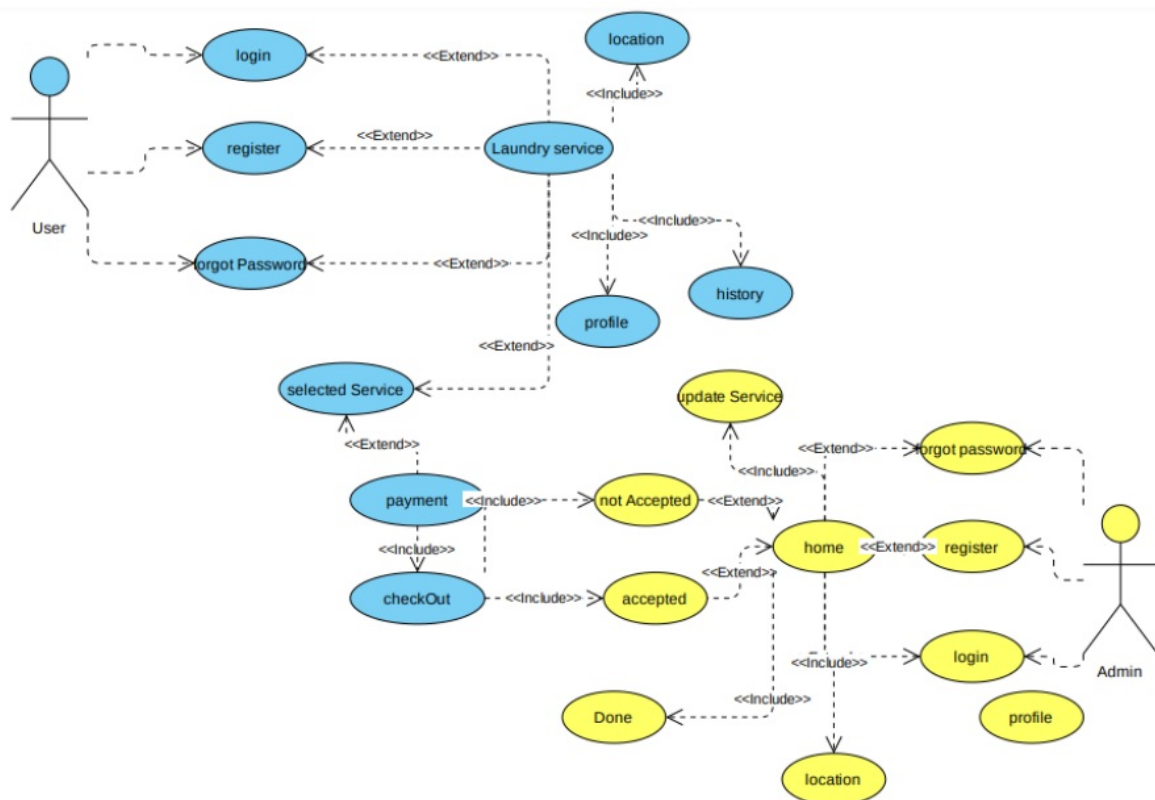


Figure 2.2: Above is the Use case which illustrates the user's interaction with the application.

2.3. Database Tables

2.3.1. Main Tables

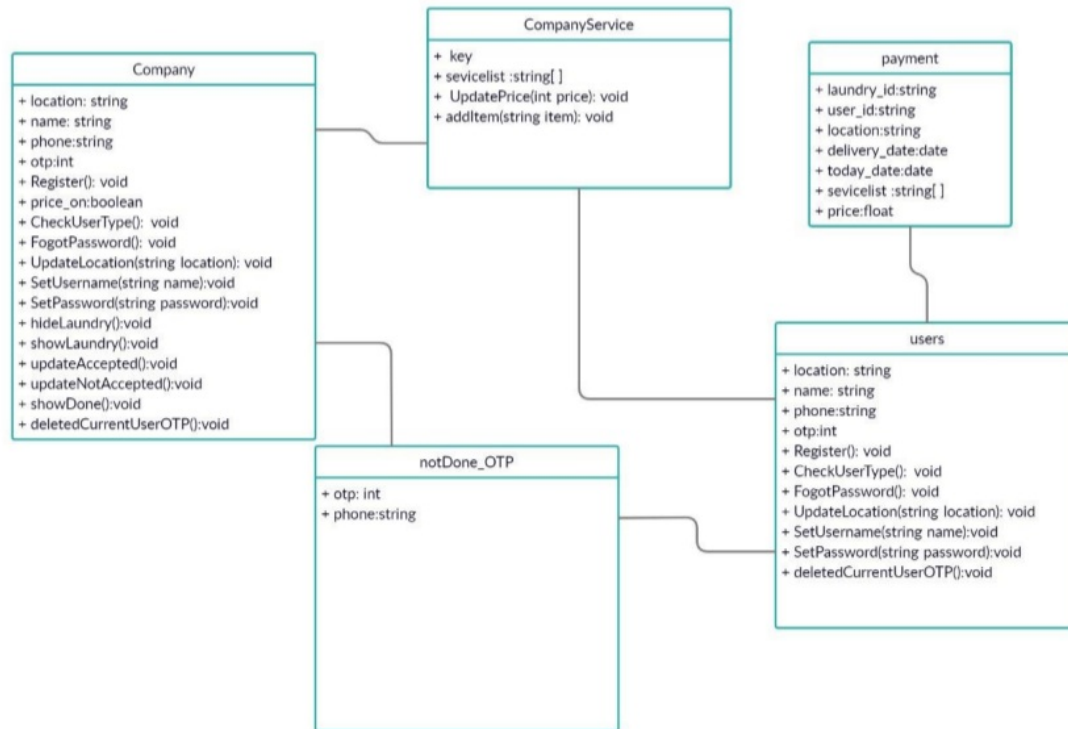


Figure 2.3.1. Above we have the UML diagrams which represent the database diagram of the table.

3. Screen layout mockups

3.1. User First Screen



Figure 3.1. First Screen

After opening the app, this is the first page that appears on your screen.

3.2. Registration Page

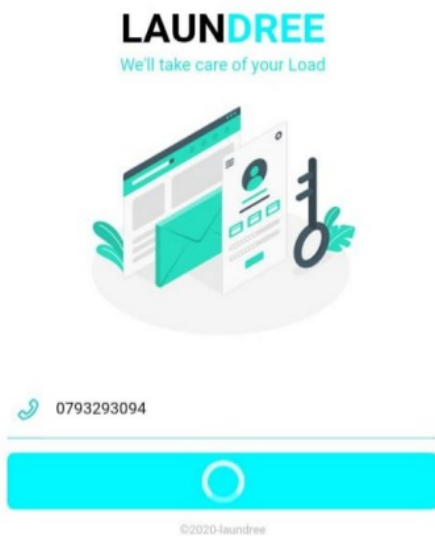


Figure 3.1. Registering the user number

Here the user is required to start by registering their phone number in order to get an OTP number.

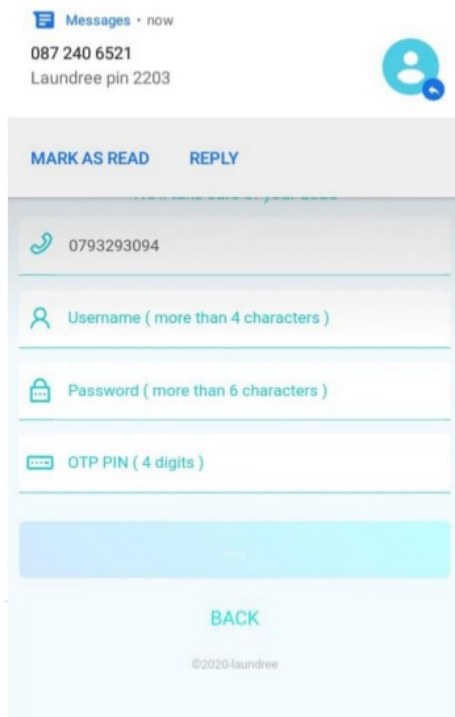
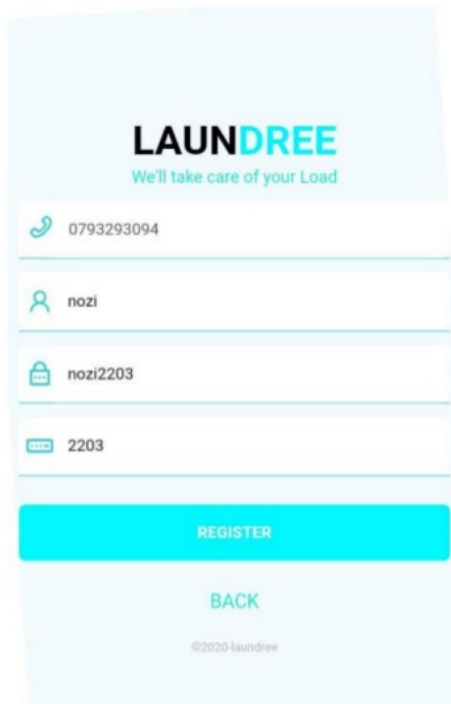


Figure 3.1.1. OTP is sent to the user

3.2. Login Page

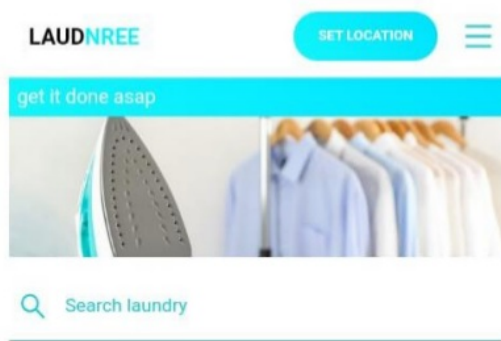


The image shows a mobile app registration screen for LAUNDREE. At the top, the logo 'LAUNDREE' is displayed in black and teal, with the tagline 'We'll take care of your Load' below it. The form consists of four input fields: a phone number field containing '0793293094', a username field containing 'nozi', a password field containing 'nozi2203', and a PIN field containing '2203'. Each field has a corresponding icon (phone, person, padlock, and PIN) on the left. Below the fields are two buttons: a teal 'REGISTER' button and a teal 'BACK' button. At the bottom, there is a small copyright notice '©2020-laundree'.

Figure 3.2. Entering the OTP and user detail

Once the user has entered the OTP and their details, the “Register” button will then be enabled and the user will be logged in.

3.2.1. Location Page



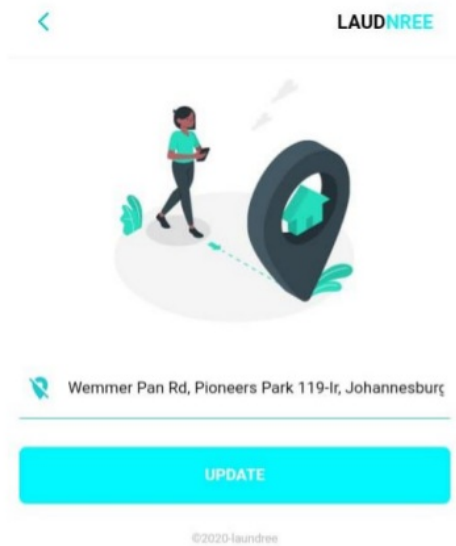


Figure 3.2.1. Setting the user location

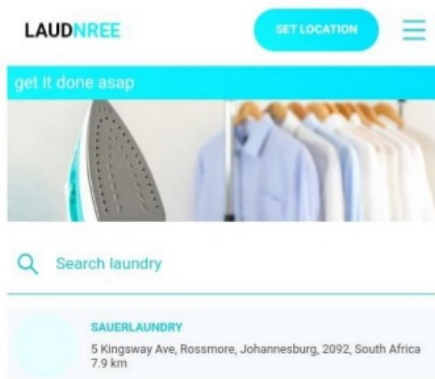


Figure 3.2.1. The user selects the laundry company

The user will then be required to set their location where their laundry will be collected from.

After updating their location, they are then provided with companies near them. The user will then select their laundry company of choice.

3.2.2. Service Page

The screenshot shows the LAUDNREE service selection interface. At the top, there is a back arrow and the company name 'LAUDNREE'. Below this, a list of services is presented, each with a checkbox and a price:

Service	Price
clothing	R0 :1 kg(s)
clothing	R0 :2 kg(s)
clothing	R0 :3 kg(s)
clothing	R0 :4 kg(s)
clothing	R0 :5 kg(s)
clothing	R0 :6 kg(s)
clothing	R0 :7 kg(s)
clothing	R0 :8 kg(s)
clothing	R0 :9 kg(s)
clothing	R0 :10 kg(s)

Below the list, there is a section for 'DUVET:ALL IN ONE' with two options:

Service	Price
king	<input checked="" type="checkbox"/> R 60
queen	<input checked="" type="checkbox"/> R 60

At the bottom, there is a 'delivery date' field showing '11/03/2020' with a dropdown arrow. A large orange 'CHECK OUT' button is positioned below the date field. To the right of the button, a blue badge displays 'Total: 150'.

Figure 3.2.1. Services screen

The services provided by the company will then be displayed with the prices set by the company. The user also needs to pick a delivery date for their laundry.

3.2.3. Checkout Page



Figure 3.2.3. Checkout screen

Once the user has checked out, all the items selected plus their prices will then be display on the checkout page.

3.2.4. History Screen

LAUDNREE

≡

pending(s) 1

delivered(all) no

FETCH DATE

MON OCT 26 2020

PRICE

110

DELIVERY DATE

Wed Oct 28 2020

DELIVERED ?

Yes

ACCEPTED ?

Yes

FETCH DATE

SUN OCT 25 2020

PRICE

150

DELIVERY DATE

Tue Oct 27 2020

DELIVERED ?

No

ACCEPTED ?

Yes

CLEAR

©2020-laundree

Figure 3.2.4. History screen

All the past and current orders appear on the history page and the past orders can be deleted from the page.



Figure 3.2.4. Offline History Page

The user can still access their history even when they are offline.

3.2.4. Profile Page

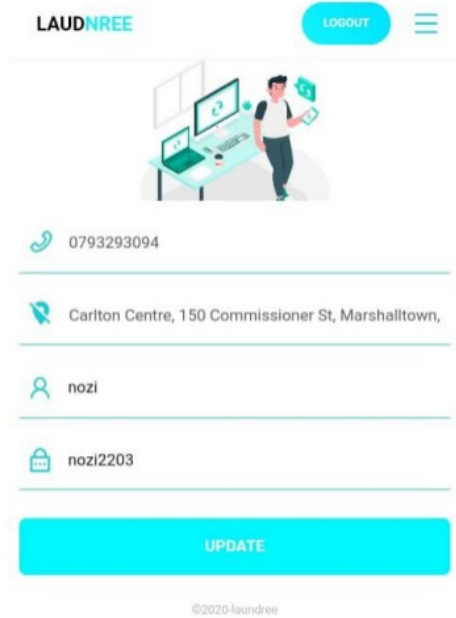


Figure 3.2.4. Profile Page

The user can logout or update their details on the profile page.

4. Screen Flows


4.1. Admin First Screen



Figure 4.1. Admin first Screen

Now we look at the system from the company's side

4.2. Registration Screen



Welcome **Laundree** Admin

Taking care of the load is your job

Phone Number

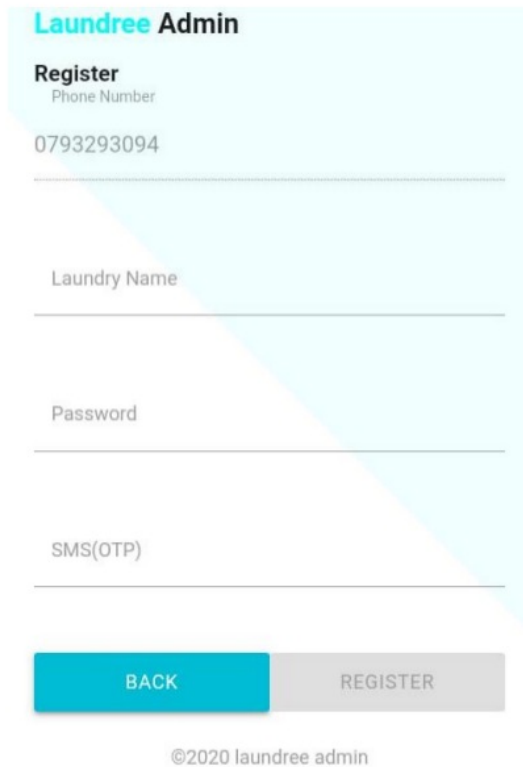
NEXT

[Forgot Password](#)

©2020 laundree admin

Figure 4.2. Registration screen

The company registers their telephone numbers, where their customers can get hold of them.



The registration screen for Laundree Admin features a light blue header with the title 'Laundree Admin' in teal. Below the header, the word 'Register' is displayed in bold, followed by the label 'Phone Number'. A text input field contains the number '0793293094'. Below this, there are three more text input fields labeled 'Laundry Name', 'Password', and 'SMS(OTP)'. At the bottom of the form, there are two buttons: a teal 'BACK' button and a grey 'REGISTER' button. The footer of the screen displays the copyright notice '©2020 laundree admin'.

Laundree Admin

Register
Phone Number

0793293094

Laundry Name

Password

SMS(OTP)

BACK REGISTER

©2020 laundree admin

Figure 4.2. Registration screen

The company then has to register themselves and they will receive an OTP for verification.

4.3. Login Screen

Laundree Admin

Register

Phone Number

0793293094

Laundry Name

SauerLaundry

correct

Password

19970716

correct

SMS(OTP)

3141

correct

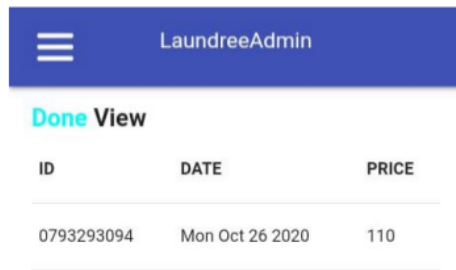
BACK

©2020 laundree admin

Figure 4.3. Admin Login Screen

Once the company has registered their number and name, they automatically login to their system.

4.4. Orders Screen

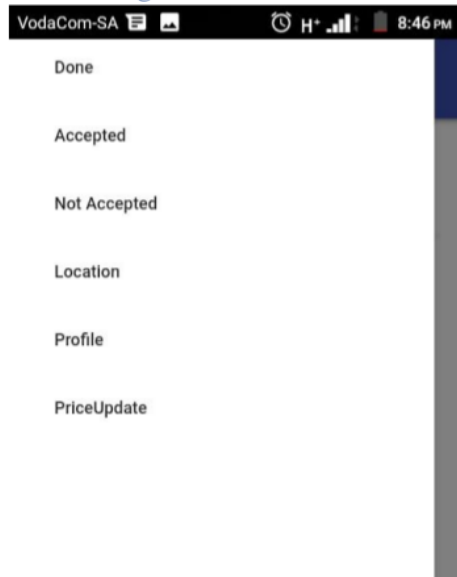


ID	DATE	PRICE
0793293094	Mon Oct 26 2020	110

Figure 4.4. Placed orders

All the orders placed by the user appear on this page, with the services, total prices and delivery date.

4.5. Categories



4.5.1. Accepted Screen

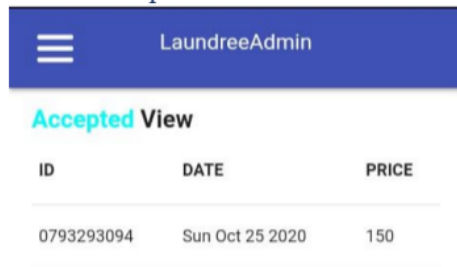



Figure 4.5. Accepted Orders

Once the order has been accepted by the company, the order will appear on this page.


4.6. Location Screen



LaundreeAdmin

SEARCH YOUR LOCATION

(Current)



5 Kingsway Ave, Rossmore, Johannesburg, 2092,
South Africa

Uj d

UPDATE

Figure 4.6. Setting Location

The admin can set the location of the company.

4.7. Prices screen

king

10

UPDATE

queen

20

UPDATE

clothing

clothing

100

UPDATE

HIDE SHOW

Figure 4.7. Setting prices

The company sets the prices of the services that they provide on this screen which will appear on the user's app.

4.8. Profile Screen

The screenshot shows the 'Profile' screen of the 'LaundreeAdmin' application. At the top, a dark blue header bar contains a hamburger menu icon and the text 'LaundreeAdmin'. Below this, a light blue card displays the user's name 'Laundree Admin' and a 'LOGOUT' button. The card is titled 'Profile' and contains four input fields: 'Phone Number' (0793293094), 'Location', 'Laundry Name' (SauerLaundry), and 'Password' (19970716). A large cyan 'UPDATE' button is positioned at the bottom of the card.

Field	Value
Phone Number	0793293094
Location	
Laundry Name	SauerLaundry
Password	19970716

Figure 4.8. Profile Screen

The profile page is used by the admin to update the details of the company.

5. Database Analysis

5.1. Debugging

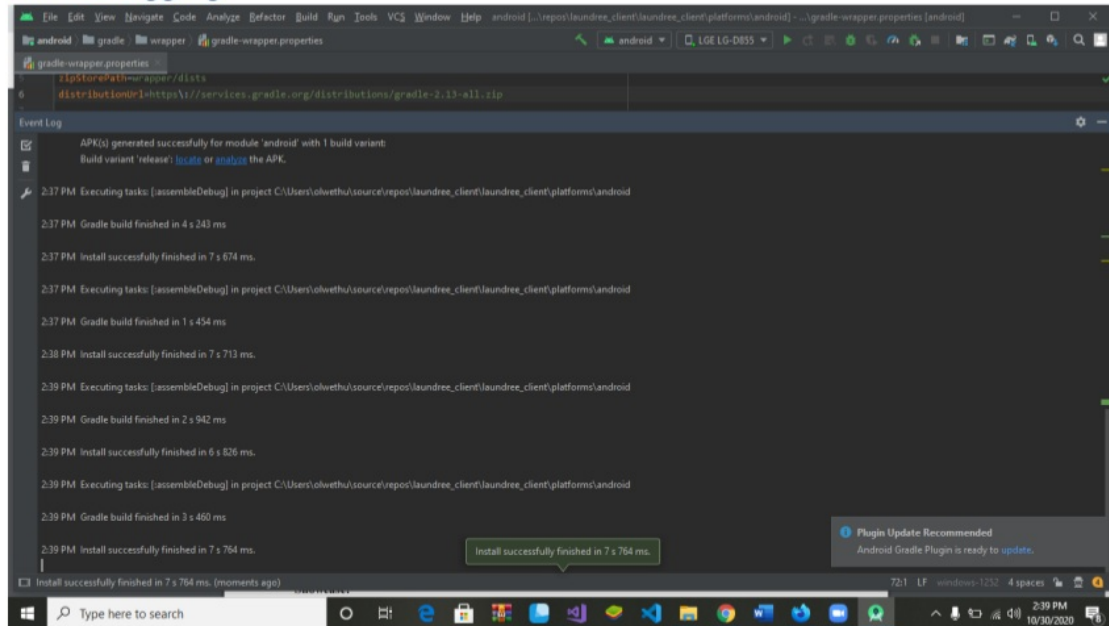


Figure 5.1. Debugging

By debugging the app, we are building the app to make sure that we don't encounter any problems such as any errors, or the app crashing.

5.2. Laundry Admin APK analysis

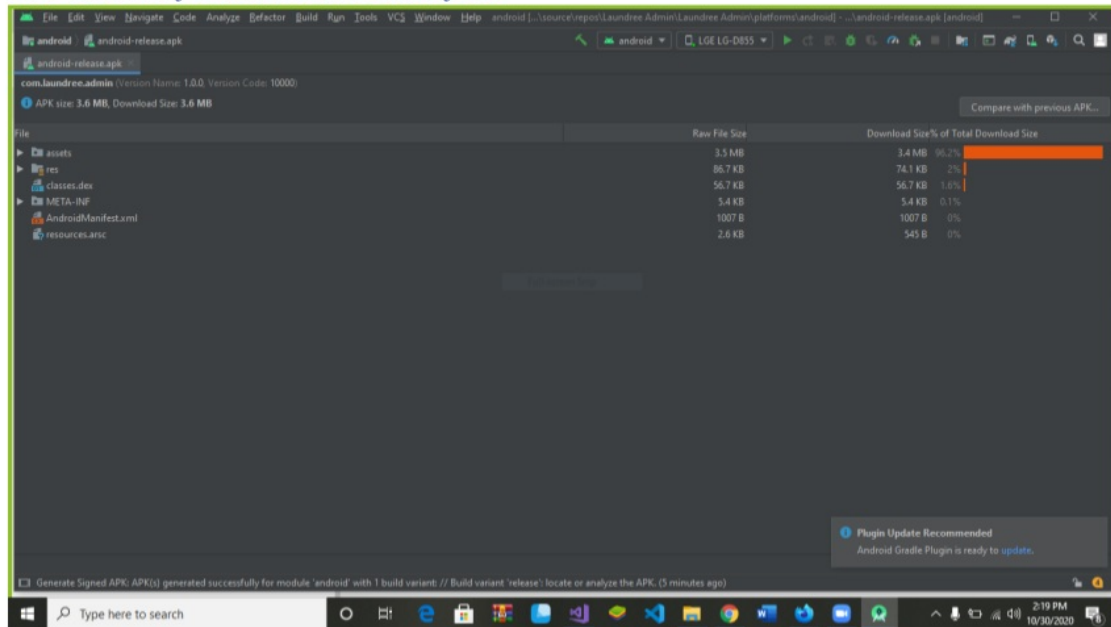


Figure 5.2. Laundry admin apk analysis

5.3. Laundry Admin profile analysis

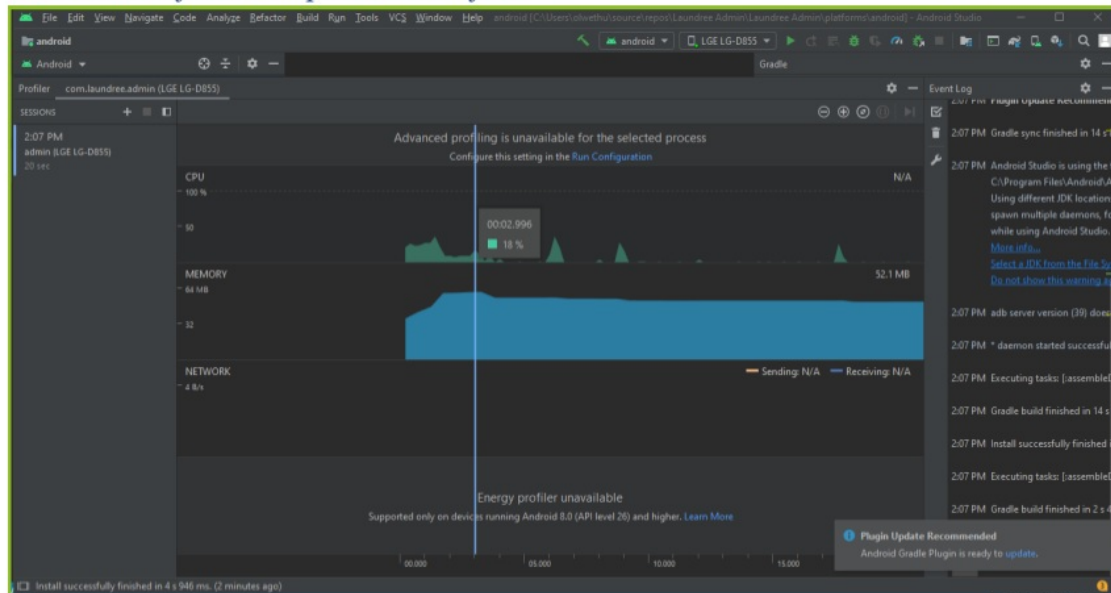


Figure 5.3. Laundry Admin profile analysis

5.4. Laundry Client APK analysis

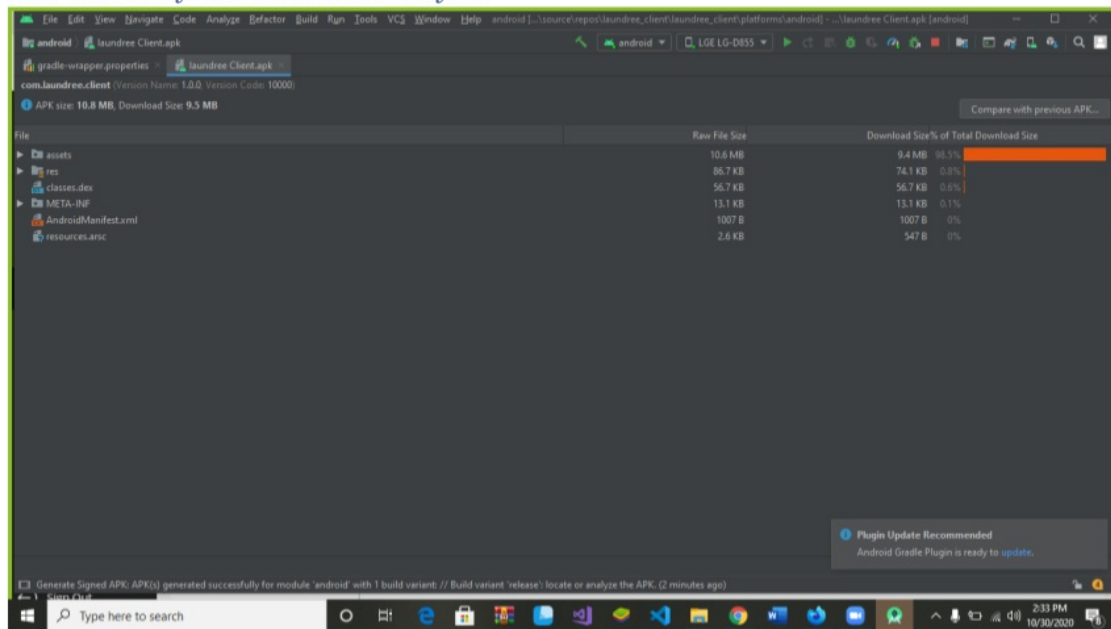


Figure 5.4. Laundry client apk analysis

5.5. Laundry Client Profile Analysis

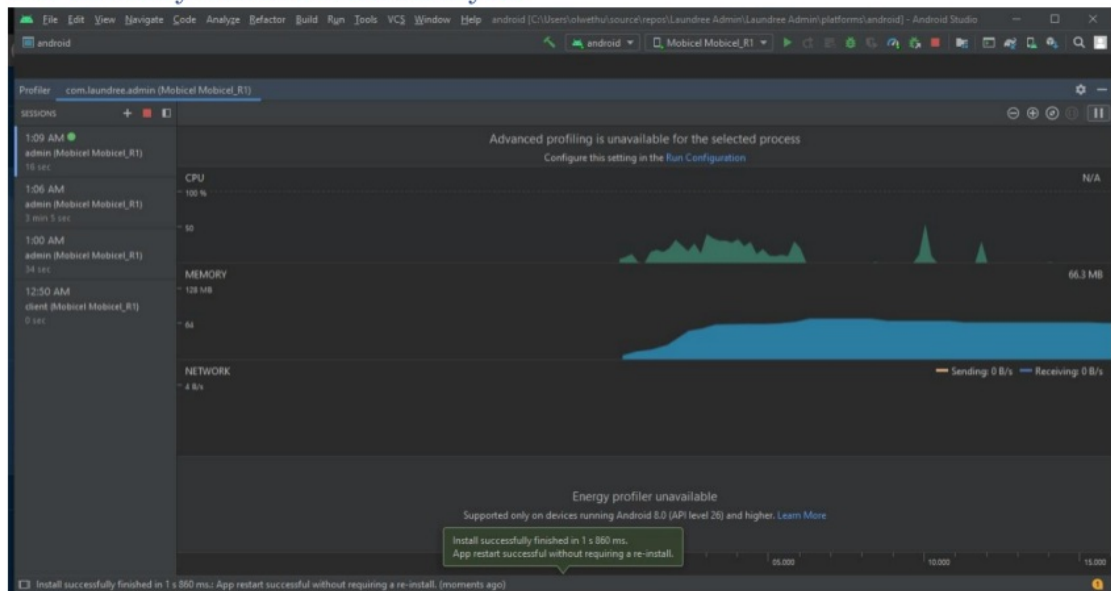
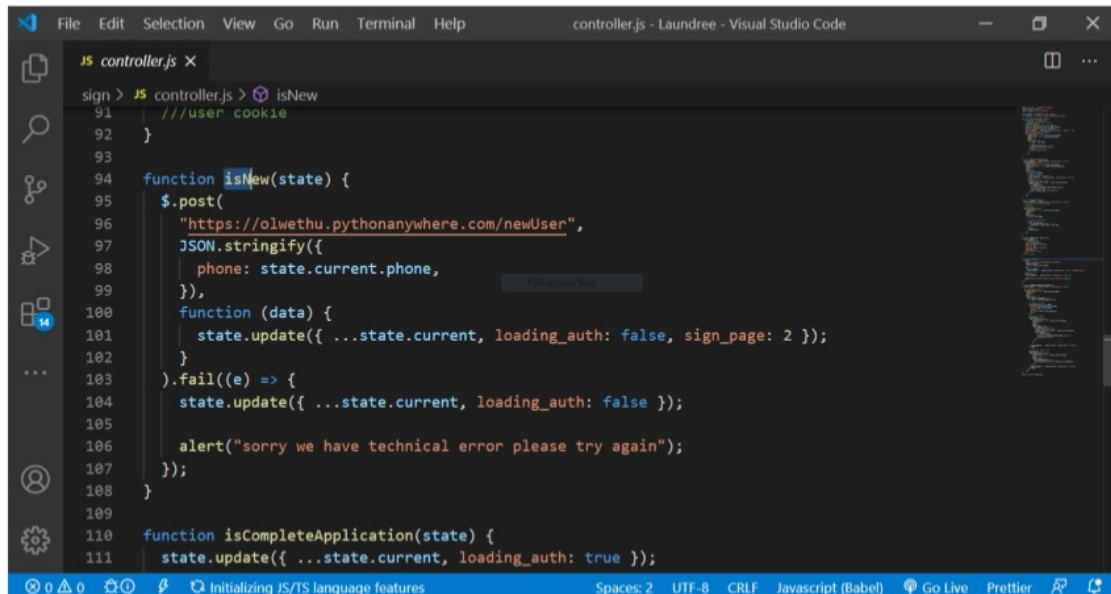


Figure 5.5. Laundry client profile analysis

6. Activities

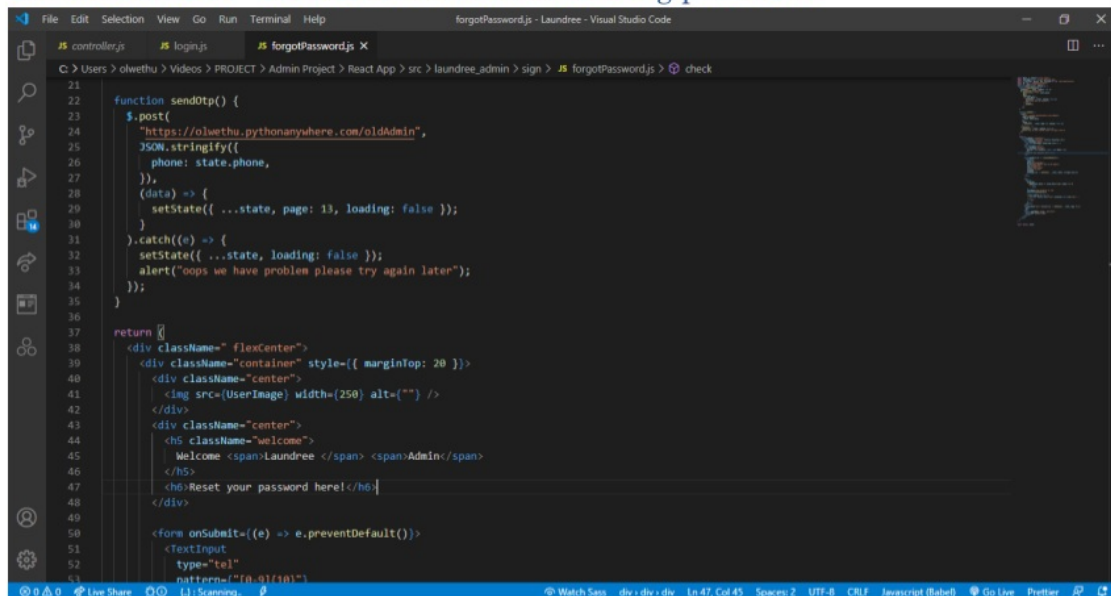
6.1. OTP code for new users



```
sign > JS controller.js > isNew
91  //user cookie
92  }
93
94  function isNew(state) {
95    $.post(
96      "https://olwethu.pythonanywhere.com/newUser",
97      JSON.stringify({
98        phone: state.current.phone,
99      }),
100    function (data) {
101      state.update({ ...state.current, loading_auth: false, sign_page: 2 });
102    }
103  ).fail((e) => {
104    state.update({ ...state.current, loading_auth: false });
105
106    alert("sorry we have technical error please try again");
107  });
108  }
109
110  function isCompleteApplication(state) {
111    state.update({ ...state.current, loading_auth: true });
```

Figure 6.1. OTP for new user

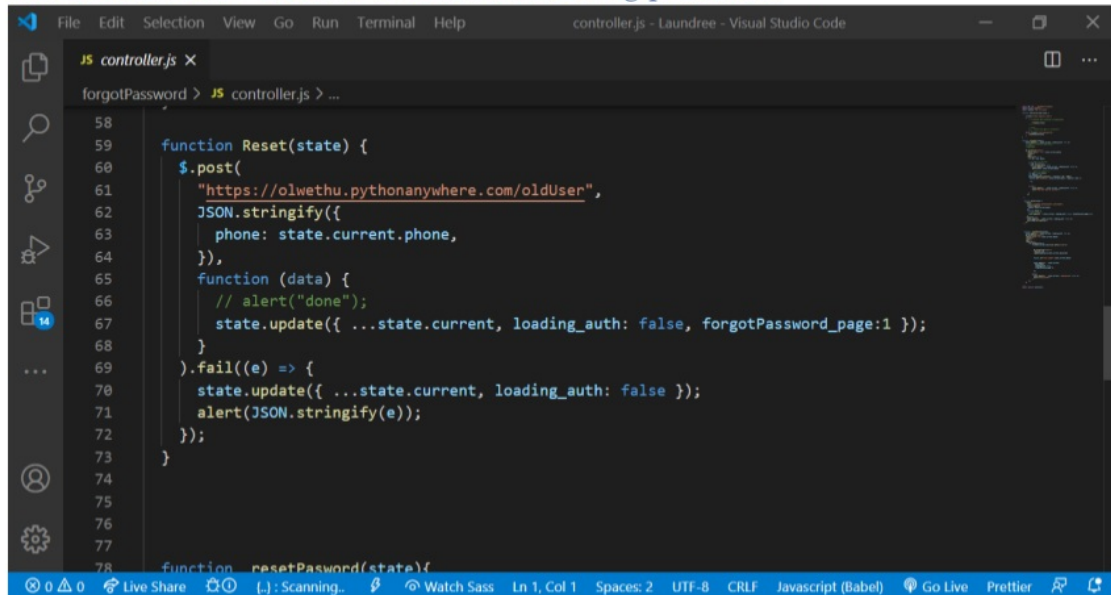
6.2. Resend OTP Code to Admin and Resetting password



```
21
22
23  function sendOtp() {
24    $.post(
25      "https://olwethu.pythonanywhere.com/oldAdmin",
26      JSON.stringify({
27        phone: state.phone,
28      }),
29    (data) => {
30      setState({ ...state, page: 13, loading: false });
31    }
32  ).catch((e) => {
33    setState({ ...state, loading: false });
34    alert("oops we have problem please try again later");
35  });
36  }
37
38  return (
39    <div className="flexCenter">
40      <div className="container" style={{ marginTop: 20 }}>
41        <img src={UserImage} width={250} alt="" />
42      </div>
43      <div className="center">
44        <h5 className="welcome">
45          Welcome <span>Laundree </span> <span>Admin</span>
46        </h5>
47        <h6>Reset your password here!</h6>
48      </div>
49    </div>
50
51    <form onSubmit={e => e.preventDefault()}>
52      <textInput
53        type="tel"
54        pattern="(09-01101)"
```

Figure 6.2. Resend OTP and reset password

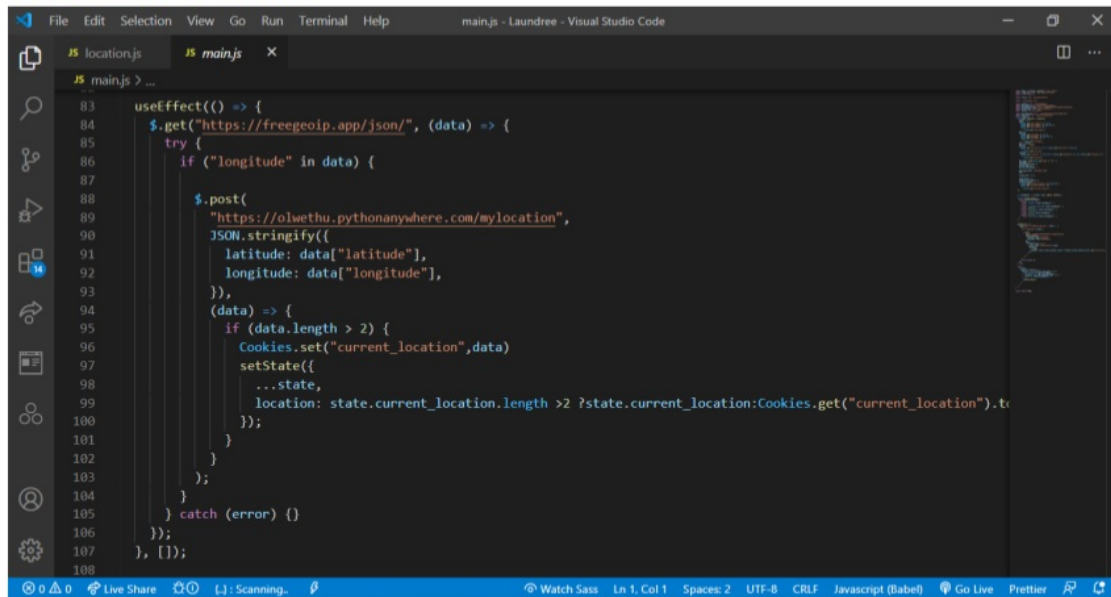
6.3. Resend OTP Code to user and Resetting password



```
58
59 function Reset(state) {
60   $.post(
61     "https://olwethu.pythonanywhere.com/oldUser",
62     JSON.stringify({
63       phone: state.current.phone,
64     }),
65     function (data) {
66       // alert("done");
67       state.update({ ...state.current, loading_auth: false, forgotPassword_page:1 });
68     }
69   ).fail((e) => {
70     state.update({ ...state.current, loading_auth: false });
71     alert(JSON.stringify(e));
72   });
73 }
74
75
76
77
78 function resetPasword(state){
```

Figure 6.3. Resend OTP and reset password

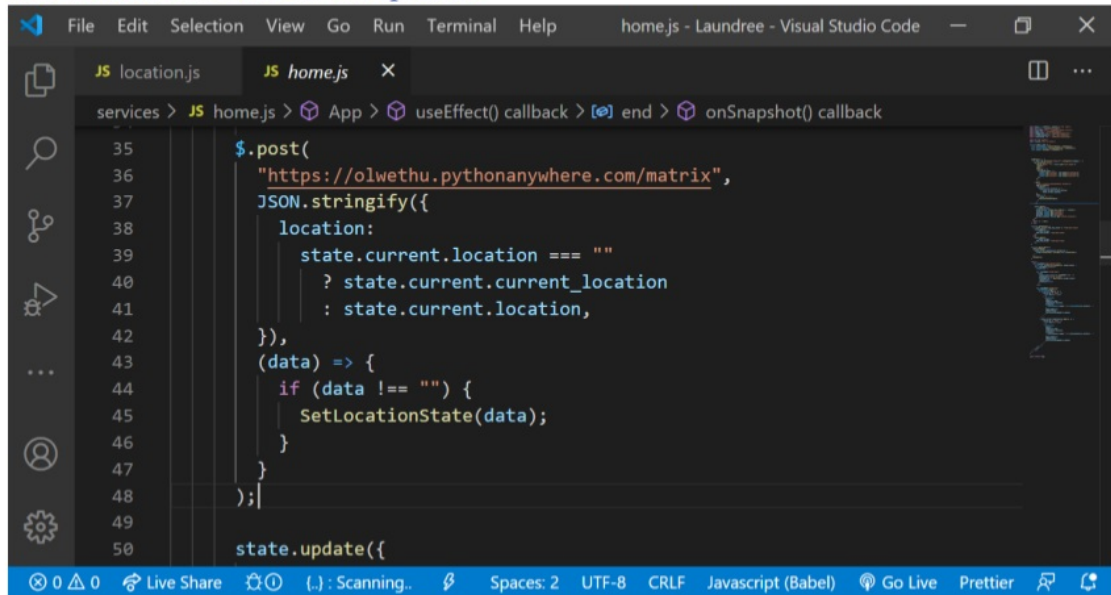
6.4. Current Location



```
83 useEffect(() => {
84   $.get("https://freegeoip.app/json/", (data) => {
85     try {
86       if ("longitude" in data) {
87
88         $.post(
89           "https://olwethu.pythonanywhere.com/mylocation",
90           JSON.stringify({
91             latitude: data["latitude"],
92             longitude: data["longitude"],
93           }),
94           (data) => {
95             if (data.length > 2) {
96               Cookies.set("current_location", data)
97               setState({
98                 ...state,
99                 location: state.current_location.length > 2 ? state.current_location : Cookies.get("current_location").t
100             });
101           }
102         });
103       }
104     } catch (error) {}
105   });
106 }, []);
```

Figure 6.4. Current Location of user or admin

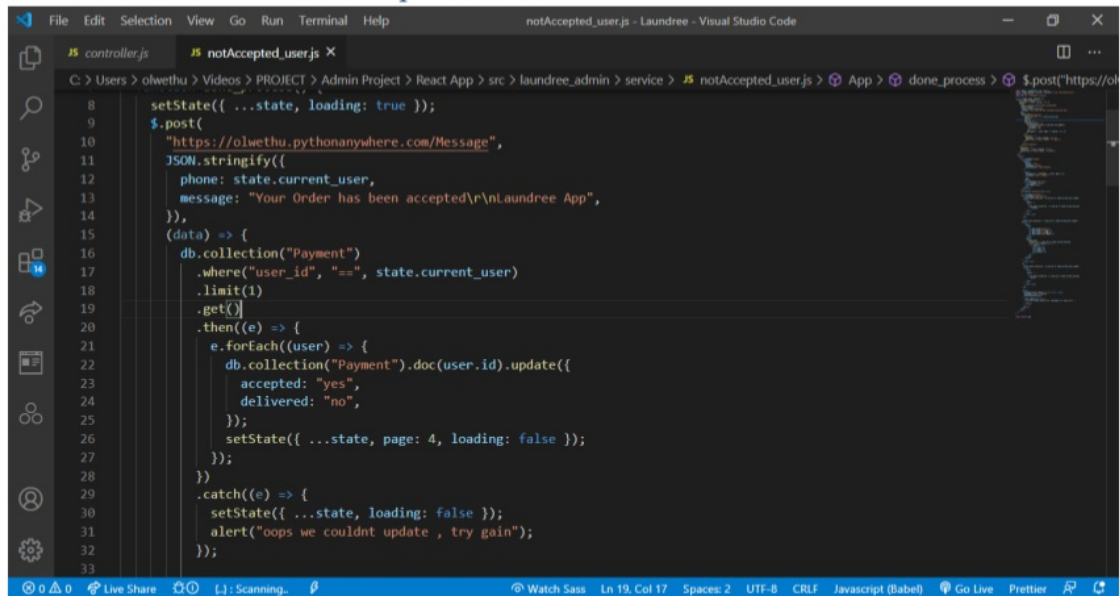
6.5. Distance between two points



```
35     $.post(
36       "https://olwethu.pythonanywhere.com/matrix",
37       JSON.stringify({
38         location:
39           state.current.location === ""
40             ? state.current.current_location
41             : state.current.location,
42       }),
43       (data) => {
44         if (data !== "") {
45           setLocationState(data);
46         }
47       }
48     );
49
50     state.update({
```

Figure 6.5. Distance between two points

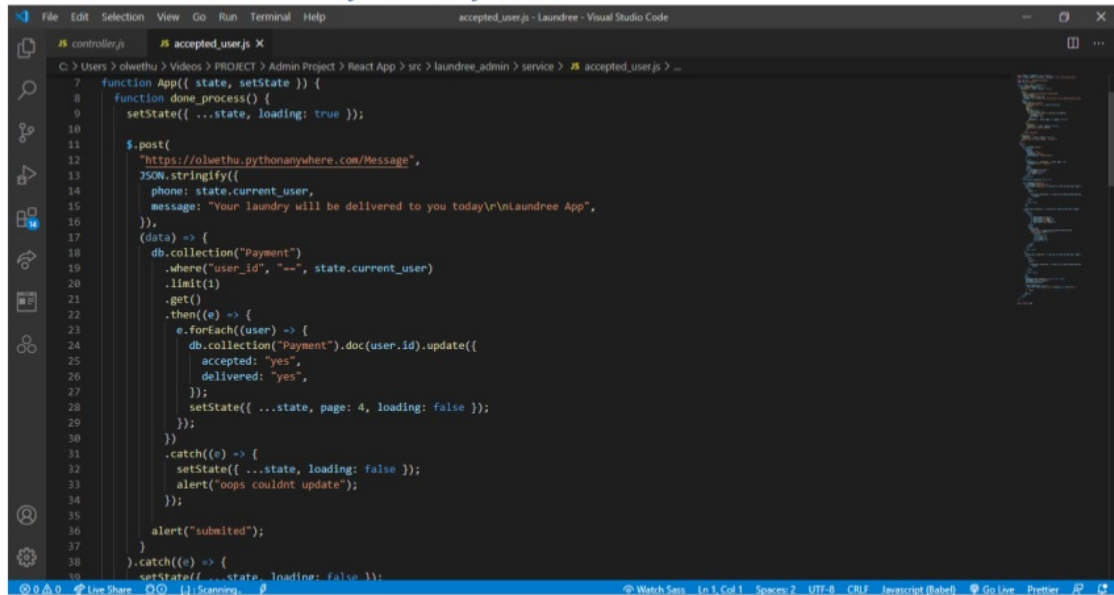
6.6. Notification for an accepted order



```
8     setState({ ...state, loading: true });
9     $.post(
10       "https://olwethu.pythonanywhere.com/Message",
11       JSON.stringify({
12         phone: state.current_user,
13         message: "Your Order has been accepted\r\nLaundree App",
14       }),
15       (data) => {
16         db.collection("Payment")
17           .where("user_id", "=", state.current_user)
18           .limit(1)
19           .get()
20           .then((e) => {
21             e.forEach((user) => {
22               db.collection("Payment").doc(user.id).update({
23                 accepted: "yes",
24                 delivered: "no",
25               });
26               setState({ ...state, page: 4, loading: false });
27             });
28           })
29           .catch((e) => {
30             setState({ ...state, loading: false });
31             alert("oops we couldnt update , try gain");
32           });
33     });
```

Figure 6.6. Accepted order notification

6.7. Notification to verify delivery date

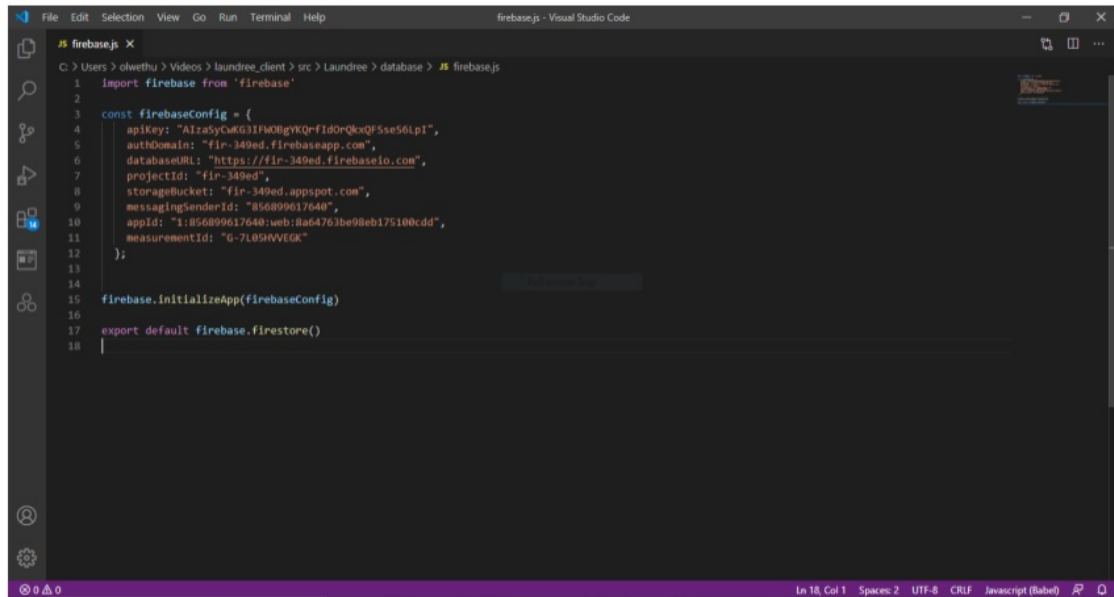


```
7 function App({ state, setState }) {  
8   function done_process() {  
9     setState({ ...state, loading: true });  
10  
11  
12     $.post(  
13       "https://oluwthi.pythonanywhere.com/Message",  
14       JSON.stringify({  
15         phone: state.current_user,  
16         message: "Your laundry will be delivered to you today\r\nlaundree App",  
17       })),  
18       (data) => {  
19         db.collection("Payment")  
20           .where("user_id", "==", state.current_user)  
21           .limit(1)  
22           .get()  
23           .then((e) => {  
24             e.forEach((user) => {  
25               db.collection("Payment").doc(user.id).update({  
26                 accepted: "yes",  
27                 delivered: "yes",  
28               });  
29               setState({ ...state, page: 4, loading: false });  
30             });  
31           })  
32           .catch((e) => {  
33             setState({ ...state, loading: false });  
34             alert("oops couldnt update");  
35           });  
36           alert("submitted");  
37         }  
38       ).catch((e) => {  
39         setState({ ...state, loading: false });  
40       })  
41     }  
42   }  
43 }  
44
```

Figure 6.7. Verifying delivery date

7. Database Authentication

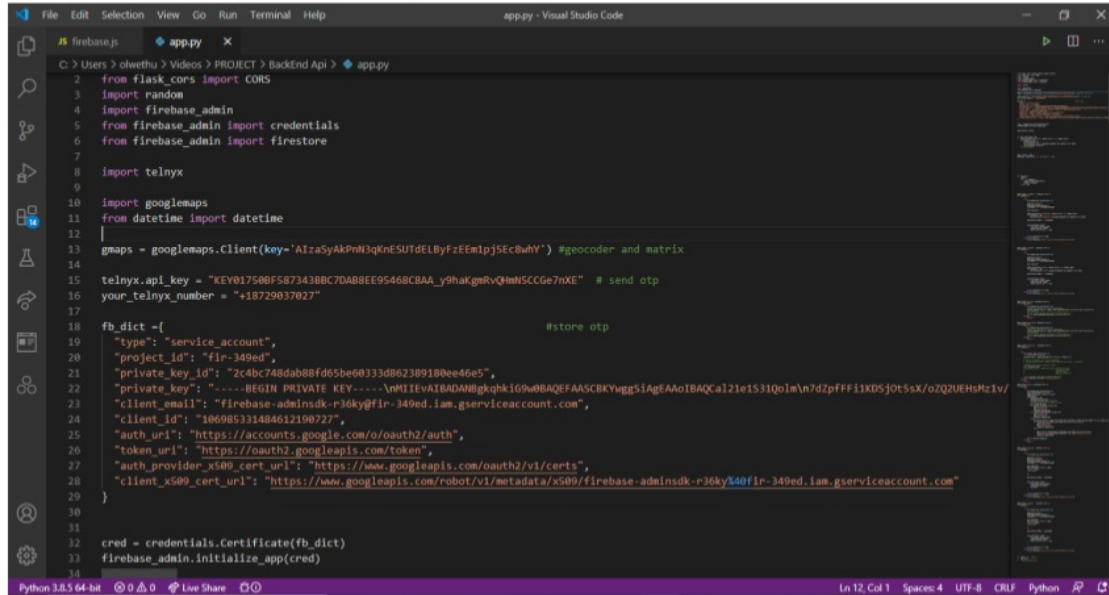
7.1 Firebase connection



```
1 import firebase from 'firebase'  
2  
3 const firebaseConfig = {  
4   apiKey: "AIzaSyCwG3IFACRgVQrFIdOrQkxQF5se56lpI",  
5   authDomain: "fir-349ed.firebaseio.com",  
6   databaseURL: "https://fir-349ed.firebaseio.com",  
7   projectId: "fir-349ed",  
8   storageBucket: "fir-349ed.appspot.com",  
9   messagingSenderId: "856899617640",  
10  appId: "1:856899617640:web:8a64763be98eb175100cdd",  
11  measurementId: "G-7L0SHVVEGK"  
12 };  
13  
14  
15 firebase.initializeApp(firebaseConfig)  
16  
17 export default firebase.firestore()  
18
```

Figure 7.1. Firebase connected to app

7.2. OTP and Location Authentication

A screenshot of a Visual Studio Code editor window titled 'app.py - Visual Studio Code'. The editor shows a Python script for OTP and location authentication. The script imports necessary modules like flask_cors, random, firebase_admin, telnyx, googlemaps, and datetime. It initializes a telnyx client and a googlemaps client. A dictionary 'fb_dict' is created with service account details. The script then creates a credential object and initializes the firebase_admin app.

```
1 from flask_cors import CORS
2 import random
3 import firebase_admin
4 from firebase_admin import credentials
5 from firebase_admin import firestore
6
7
8 import telnyx
9
10 import googlemaps
11 from datetime import datetime
12
13 gmaps = googlemaps.Client(key='AIzaSyakPmN3qKneSUTdELByfzEEwlpj5ecBwHY') #geocoder and matrix
14
15 telnyx.api_key = "KEY01750BF587343BBC7DABEE95468CBAA_y9haKgmRvQhMNSCCGe7nXE" # send otp
16 your_telnyx_number = "+18729037827"
17
18 fb_dict = {
19     "type": "service_account",
20     "project_id": "fir-349ed",
21     "private_key_id": "2c4bc748dab88fd65be60333d862389180ee46e5",
22     "private_key": "-----BEGIN PRIVATE KEY-----\nmMIEvAI8ADANBgkqhkiG9w0BAQEFAASCBKYYggSIAgEAAoIBAQA21e1S31QolmVn7dzpFFf1KDSj0tssX/oZQ2UEHsHz1v/
23     "client_email": "firebase-adminsdk-r36ky@fir-349ed.iam.gserviceaccount.com",
24     "client_id": "106985331484612198727",
25     "auth_uri": "https://accounts.google.com/o/oauth2/auth",
26     "token_uri": "https://oauth2.googleapis.com/token",
27     "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
28     "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-r36ky@fir-349ed.iam.gserviceaccount.com"
29 }
30
31 cred = credentials.Certificate(fb_dict)
32 firebase_admin.initialize_app(cred)
```

Figure 7.2. OTP and location authentication

Project Documentation

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35
