A Project Report on

“CRM”

Submitted by [Group-11]

1791005 – Purav Uday Desai

1791226 – Aakash Arvind Parmar

1791048 – Jay Bhavesh Thadeshwar

Guided By

Mrs. K. V. Bhat

Technician

Mr. Sachin

Term – December’19 to April’20

Department of Information Technology

SVKM’s

Shri Bhagubhai Mafatlal Polytechnic,

Irla, N.R.G. Marg, Vile Parle (W), Mumbai – 400056


iVCRM

IV Consultancy Services Pvt. Ltd. www.ivgroup.in

PROJECT OFFER LETTER

Jay Thadeshwar, student of Shri Bhagubhai Mafatlal Polytechnic, Information Technology, has been offered a project by *IVConsultancy Services Pvt. Ltd* for their final year project.

He was selected by our company for making a new application for our enterprise.

He is offered to develop a CRM(Customer Relationship Management) application for our enterprise, which will be further governed and sold by us.

The project was started in implementation in December,2019 and technology stack for this project will be Flutter, Java and DHTML, this project will be guided by our company and will follow industry standards for coding and project development.

Hardish Patel

Co-Founder & Director
IV Consultancy Services Private Limited

Address : Gala No. 524, Prestige Industrial Estate, Bawadi Cross lane, Marve road, Malad(W), Mumbai - 400 064

Contact No : 9819804748 || www.ivgroup.in

Index

1. Acknowledgement

First and foremost, we would like to thank our supervisor/guide of this project, **Mrs. Krishna Bhatt**. For the valuable guidance and advice. She inspired us greatly to work in this project. Her willingness to motivate us contributed tremendously to our project.

Besides, we would like to thank the authority of **IV Consultancy Services Pvt. Ltd.** for providing us with a good environment and facilities to complete this project. Also we would like to take this opportunity to thank the IV Consultancy Services Pvt.Ltd. for offering this subject, CRM Application. It gave us an opportunity to participate and learn about the real-world software development with industry.

We are highly indebted to ***Mr. Hardish Patel*** *(Director and Co-Founder of IV Consultancy Services Pvt.Ltd.)* for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our special gratitude and thanks to industry persons for giving us such attention and time. Finally, an honourable mention goes to our **families and friends** for their understanding and support for us in completing this project. Without the help of the that mentioned above, we would face many difficulties while doing this.

Thanking you,

1791005 – Purav Uday Desai,

1791226 – Aakash Arvind Parmar,

1791048 – Jay Bhavesh Thadeshwar

2. Problem definition

Today in industry there is a requirement for many management tools, that could provide their organization better control over their resources and utilities, so there should be proper IT framework which should be provided to them for handling the required things, there are many softwares available in the market today but are very costly and does not provide full flexibility to their users, so our aim is to provide the same set of service with some betterments with low economic weight, so it can be usable for all types of organization whether it can be startup, a medium sized business or a huge enterprise.

Customer relationship management is an approach to managing a company's interaction with current and potential customers. It uses data analysis about customers' history with a company to improve business relationships with customers, specifically focusing on customer retention and ultimately driving sales growth.

3. Abstract

This application is a CRM (Customer Relationship Management) Software, which is an industry requirement by **IV Consultancy Services Pvt. Ltd.** to develop such an application which manages, monitors, tracks all the field resources and provides statistical reports about the performance of the same.

In this application, we have to build an API, by which a mobile application working on both iOS and Android platforms calls the API and performs their respective functionalities, which provides management a proper digital architecture to handle their resource fields. It as a business management tool in greater scale, it should be generic for all types of domain (Example: RealEstate, CallCenter/BPO, Travel, Trading, Telecom, etc) this means that all the organizational bodies of each domain should be able to set rules and structure according to their domain and use specifications.

4. Introduction

4.1 Concept of CRM:

Entrepreneurs started using the term Customer Relationship Management (CRM) in the early 1990s when the concept of business started to change from being transactional to relational. CRM directly contributes towards customer benefits and the growth of businesses.

Information Technology plays a very critical role in identifying, acquiring, and retaining the customers, and thereby managing a healthy relationship with them.

There can be multiple definitions of CRM from different perspectives −

- Management perspective: CRM can be defined as *an organized approach of developing, managing, and maintaining a profitable relationship with customers*.
- Technological perspective: CRM can be defined as *a software that assists marketing, merchandising, selling, and smooth service operations of a business*.
- As per Franics Buttle, World's first professor of CRM, it is the core business strategy that integrates internal processes and functions, and external networks, to create & deliver value to a target customer at profit. It is grounded on high quality customer data and information technology.

4.2 Objectives of CRM:

1. Improve Customer Satisfaction
2. Expand the Customer Base

3. Enhance Business Sales
4. Improve Workforce Productivity

The primary goal of CRM is to increase customer loyalty and in turn improve business profitability.

4.3 History of CRM:

1970s – Emergence of Material Resource Planning (MRP)
The complex and expensive MRP solutions emerged. They needed a large technical staff to execute MRP software on mainframe computers.

1972: Five German engineers started System Analysis and Programming (SAP) business to develop a software for providing business solutions.

1977: Larry Ellison started Oracle Corporation.

1978: Baan started The Baan Corporation for financial and administrative consultation.

1979: Oracle offered first commercial SQL Relational Database Management System (RDBMS).

The 1980s: Formation of Database Marketing

- Robert and Kate Kestnbaum pioneered database marketing, collected customer contacts, and analyzed customer information.
- **1986:** ACT! created Contact Management System (CMS) to store the data and organize customer information.
- **1987:** PeopleSoft created Human Resource Management System (HRMS).

The 1990s: Industrial Growth

- Brock Control Systems started Sales Force Automation (SFA), and added the features of database marketing, CMS, inventory control, and customer interaction tracking.
- **1993:** Tom Siebel started Siebel Systems, soon became leading provider of SFA.
- **1995:** SFA and CMS evolved as very similar modern CRM software, soon settled as Customer Relationship Management System (CRMS or simply, CRM). Enterprise Resource Planning (ERP) solution vendors Oracle and Baan entered the CRM market, and added more marketing, sales, and service applications to CRM.
- **1999:** ECRM vendors emerged. CRM also entered into the mobile market.

The 2000s: The modern CRM

- Microsoft entered CRM market with its product Dynamics CRM.
- Oracle acquired Siebel, and numerous other enterprise application vendors.
- **2007:** Salesforce introduced the world to cloud-based CRM.

5. Software requirements

- Spring Tools Suite 4
- Java Development Kit 1.8 or above
- Android/iOS Software Development kit
- Flutter (stable release)
- PostgreSQL (stable release)
- PgAdmin 14.07 or above
- VSCode (stable release)
- Tortoise SVN (stable release)
- Spring Boot 2.0 or above

6. Hardware requirements

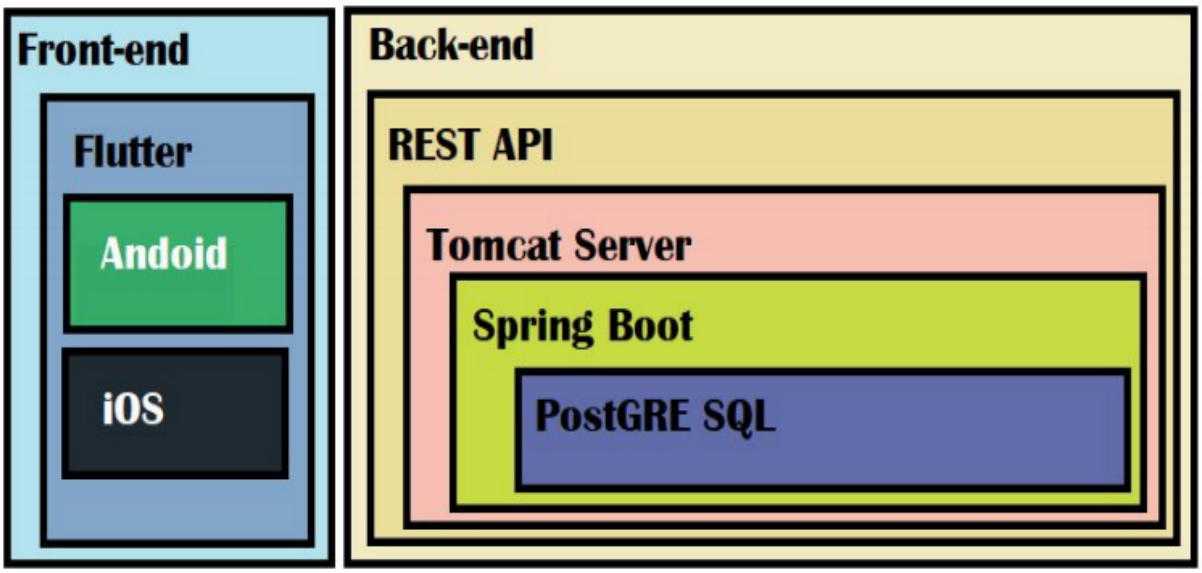
- Android Jelly Bean, v16, 4.1.x or newer, and iOS 8 or newer.
- iOS devices (iPhone 4S or newer) and ARM Android devices.
- Qualcomm Snapdragon 450 or above
- 1 GB RAM or above
- Minimum disk space : 50 MB

7. Gannt chart / Timeline chart

Months	December				January				Febuary				March				April			
Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Tasks																				
Planning/Requirements																				
Learning																				
Design																				
Development																				
Code Review																				
Testing																				

	Planning/Requirements
	Learning
	Design
	Development
	Code Review
	Testing

8. Literature survey



8.1 Flutter:

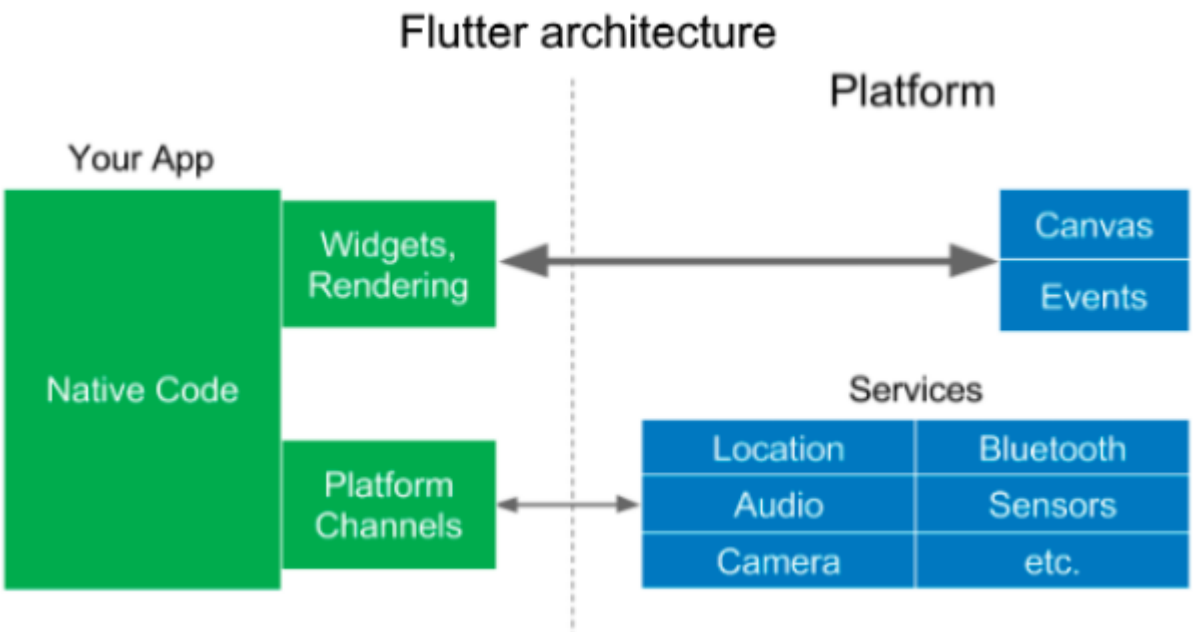
8.1.1 Introduction to Flutter:

Flutter is Google's new open source technology for creating native Android and iOS apps with a single codebase. Unlike other popular solutions, Flutter is not a framework; it's a complete SDK – software development kit – which already contains everything you will need to build cross-platform applications. This includes a rendering engine, ready-made widgets, testing and integration APIs, and command-line tools.

8.1.2 Why Flutter?

1. Same UI and business logic in all platforms.
2. Similar to native app performance.
3. Custom, animated UI of any complexity available.
4. Hot reloading feature facilitates designer developer cooperation.
5. High app responsiveness.

8.1.3 Flutter architecture:



Flutter solves the most challenging part of the other cross-platform frameworks, i.e. getting rid of the BRIDGE. Flutter does not use the OEM widgets, it provides its own widgets. Flutter moves the widgets and the renderer from the Platform into the app, which allows them to be customizable and extensible. This also helps Flutter to maintain the consistent 60 FPS.

8.2 Spring Boot:

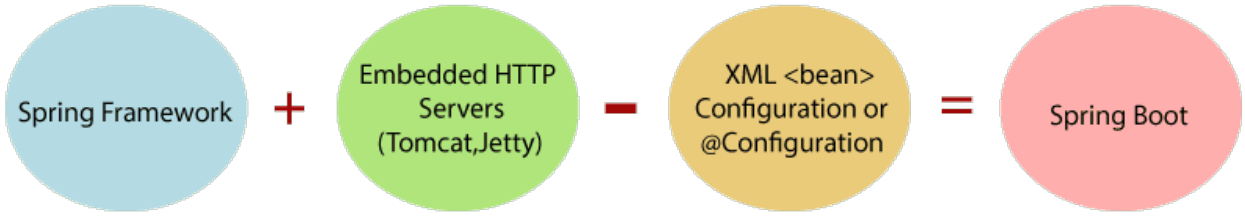
8.2.1 Introduction to Spring Boot:

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup. It is a Spring module that provides the RAD (*Rapid Application Development*) feature to the Spring Framework.

8.2.2 Why Spring Boot?

- It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions.
- It provides a powerful batch processing and manages REST endpoints.
- In Spring Boot, everything is auto configured; no manual configurations are needed.
- It offers annotation-based spring application
- Eases dependency management
- It includes Embedded Servlet Container

8.2.3 Spring Boot Architecture:



Handling dependency management is a difficult task for big projects. Spring Boot resolves this problem by providing a set of dependencies for developers convenience. Spring Boot automatically configures your application based on the dependencies you have added to the project by using *@EnableAutoConfiguration* annotation.

The entry point of the spring boot application is the class contains *@SpringBootApplication* annotation and the main method.
Spring Boot automatically scans all the components included in the project by using *@ComponentScan* annotation.

8.3 PostgreSQL:

8.3.1 Introduction to PostgreSQL:

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL has

earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all_major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender.

8.3.2 Why PostgreSQL?

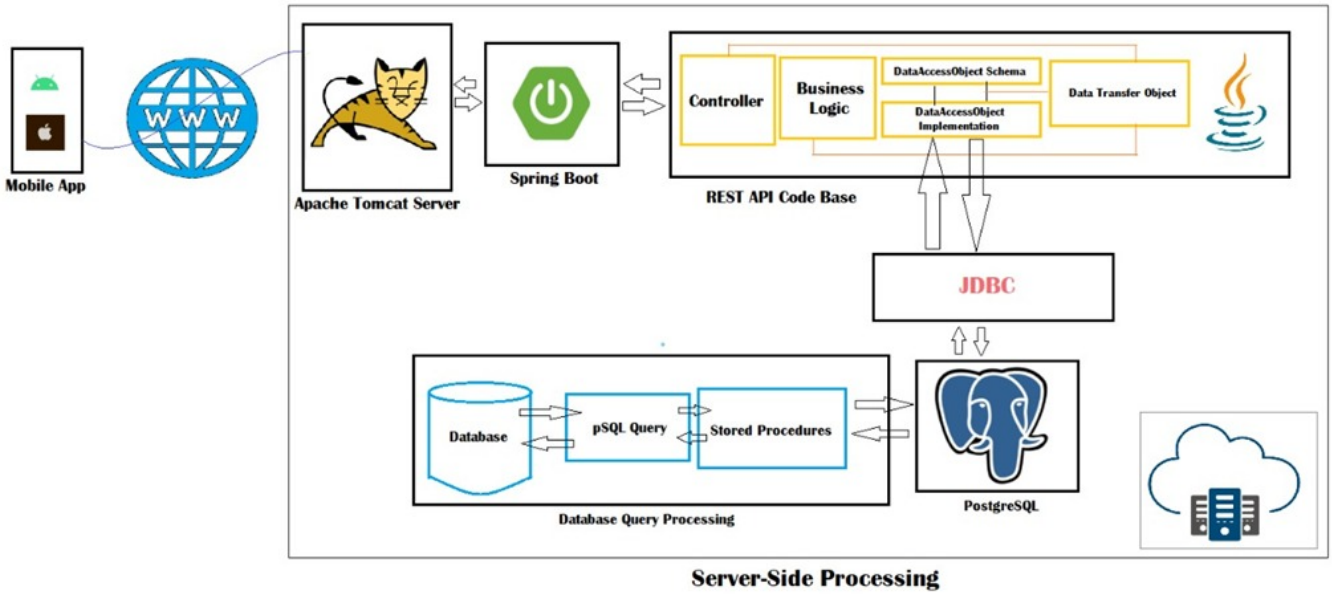
PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible.

For example, you can define your own data types, build out custom functions, even write code from different programming languages without recompiling your database.

8.3.3 PostgreSQL GUI:

The pgAdmin package is a free and open source graphical user interface administration tool for PostgreSQL, which is supported on many computer platforms. pgAdmin is a management tool for PostgreSQL and derivative relational databases such as Enterprise DB's EDB Advanced Server. It may be run either as a web or desktop application.

9. Workflow of application



- The mobile app sends a request to a web server which has our Spring Boot API deployed on it. Our API is designed by maintaining the industry standards.
- First, the request arrives to the controller, which appropriately transfers the control to business logic that gets executed.
- Then, the business logic calls the suitable function from the data access object (DAO) which performs the data operation with database.
- The DAO internally uses data transfer object (DTO) which carries data between processes.
- The database contains stored procedures which is used by DAO. Stored procedure uses plpgSQL queries internally.

10. IVCRM App

10.1 Executive App:

10.1.1 Dashboard:

The dashboard provides a stunning user interface with stats at glance. It provides information about the number of tickets for each status type and a graph which shows the change in employee's PL Rate according to their performance.

Code Snippet:

1. Frontend

```
class HomePageState extends State<HomePage> {
  List<charts.Series<ReportClass, DateTime>> _createSampleData(
      List<ReportClass> data) {
    return [
      charts.Series<ReportClass, DateTime>(
        id: 'time',
        colorFn: (_, __) => charts.Color.fromHex(code: '#ff6d00'),
        domainFn: (ReportClass sales, _) => sales.filterType,
        measureFn: (ReportClass sales, _) => sales.plrate,
          data: data,
      )
      ];
    }

    Future<List<ReportClass>> getChartData() async {
      var json = await ApiCall.getDataFromApi(Uri.GET_PL + "/${CurrentUser.id}");
      List<ReportClass> reports = [];
      if (json == "nothing") {
        return reports;
      }

      int ii = json.length > 10 ? 10 : json.length;
      for (int i = 0; i < ii; i++) {
        dynamic time = json[i]['dateOfEntry'];
        dynamic rate = json[i]['plrate'];

        ReportClass r1 = ReportClass.retrieve(DateTime.parse(time), rate * 1.0);
        reports.add(r1);
```

```
29.      }
30.      return reports;
31.    }
32.
33.    final TextStyle stats = TextStyle(
34.        fontWeight: FontWeight.bold, fontSize: 20.0, color: Colors.white);
35.    Widget build(BuildContext context) {
36.      return SafeArea(
37.        child: Scaffold(
38.            body: SingleChildScrollView(
39.          child: Column(
40.              crossAxisAlignment: CrossAxisAlignment.start,
41.              mainAxisSize: MainAxisSize.min,
42.              children: <Widget>[
43.                Card(
44.                  elevation: 2.0,
45.                  borderOnForeground: false,
46.                  color: Colors.white,
47.                  child: ListTile(
48.                    title: Text(CurrentUser.name,
49.                        style: TextStyle(
50.                          fontWeight: FontWeight.w800,
51.                          color: Theme.of(context).primaryColor,
52.                        )),
53.                    subtitle:
54.                        Consumer<ThemeNotifier>(builder: (context, data, child) {
55.                      return Text(
56.                        CurrentUser.companyName + ' , ' + CurrentUser.branchName,
57.                        style: TextStyle(
58.                          color: data.getTheme().brightness == Brightness.dark
59.                              ? Colors.black
60.                              : null,
61.                        ),
62.                      );
63.                    }),
64.                    trailing:
65.                        Consumer<ThemeNotifier>(builder: (context, data, child) {
66.                      return Stack(children: <Widget>[
67.                        IconButton(
68.                          icon: Icon(
69.                            FontAwesomeIcons.bell,
70.                            color: data.getTheme().brightness == Brightness.dark
71.                                ? Colors.black
72.                                : null,
73.                          ),
74.                          onPressed: () {
75.                            Navigator.push(
76.                                context,
77.                                MaterialPageRoute(
78.                                    builder: (context) => Notifications()));
79.                          },
80.                        ),
81.                        CurrentUser.notificationCount != 0
82.                            ? Positioned(
83.                                right: 5.0,
84.                                top: 5.0,
```

```
85.                     child: CircleAvatar(
86.                       backgroundColor: Theme.of(context).primaryColor,
87.                       foregroundColor: data.getTheme().brightness ==
88.                           Brightness.dark
89.                         ? Colors.black
90.                         : null,
91.                       radius: 10.0,
92.                       child: ClipOval(
93.                         child: Text(
94.                           '${CurrentUser.notificationCount}',
95.                         ),
96.                       ),
97.                     ),
98.                   )
99.                 : Text(''),
100.            ]);
101.          }),
102.        ),
103.      ),
104.      Row(children: <Widget>[
105.        Expanded(
106.          child: Consumer<BottomNavigationNotifier>(
107.            builder: (context, data, child) {
108.              return GestureDetector(
109.                onTap: () {
110.                  data.setData(3, name: 'Immediate');
111.                },
112.                child: Container(
113.                  padding: const EdgeInsets.all(8.0),
114.                  margin: const EdgeInsets.all(8.0),
115.                  height: 80.0,
116.                  decoration: BoxDecoration(
117.                    borderRadius: BorderRadius.circular(10.0),
118.                    color: Color(0xffd41e1e),
119.                  ),
120.                  child: Column(
121.                    mainAxisAlignment: MainAxisAlignment.center,
122.                    children: <Widget>[
123.                      Text(
124.                        '+${CurrentUser.immediateTicketCount}',
125.                        style: stats,
126.                      ),
127.                      const SizedBox(height: 5.0),
128.                      Text('Immediate'),
129.                    ]),
130.              ),
131.            );
132.          }),
133.        ),
134.        Expanded(
135.          child: Consumer<BottomNavigationNotifier>(
136.            builder: (context, data, child) {
137.              return GestureDetector(
138.                onTap: () {
139.                  data.setData(3, name: 'High');
140.                },
```

```
141.                 child: Container(
142.                   padding: const EdgeInsets.all(8.0),
143.                   margin: const EdgeInsets.all(8.0),
144.                   height: 80.0,
145.                   decoration: BoxDecoration(
146.                     borderRadius: BorderRadius.circular(10.0),
147.                     color: Color(0xffd9b11e),
148.                   ),
149.                   child: Column(
150.                     mainAxisAlignment: MainAxisAlignment.center,
151.                     children: <Widget>[
152.                       Text(
153.                         '+${CurrentUser.highTicketCount}',
154.                         style: stats,
155.                       ),
156.                       const SizedBox(height: 5.0),
157.                       Text('High'),
158.                     ]),
159.                 ),
160.               );
161.             }),
162.           ),
163.         ]),
164.         Row(children: <Widget>[
165.           Expanded(
166.             child: Consumer<BottomNavigationNotifier>(
167.               builder: (context, data, child) {
168.                 return GestureDetector(
169.                   onTap: () {
170.                     data.setData(3, name: 'Normal');
171.                   },
172.                   child: Container(
173.                     padding: const EdgeInsets.all(8.0),
174.                     margin: const EdgeInsets.all(8.0),
175.                     height: 80.0,
176.                     decoration: BoxDecoration(
177.                       borderRadius: BorderRadius.circular(10.0),
178.                       color: Color(0xff6db1e8),
179.                     ),
180.                     child: Column(
181.                       mainAxisAlignment: MainAxisAlignment.center,
182.                       children: <Widget>[
183.                         Text(
184.                           '+${CurrentUser.normalTicketCount}',
185.                           style: stats,
186.                         ),
187.                         const SizedBox(height: 5.0),
188.                         Text('Normal'),
189.                       ]),
190.                   ),
191.                 );
192.               }),
193.           ),
194.           Expanded(
195.             child: Consumer<BottomNavigationNotifier>(
196.               builder: (context, data, child) {
```

```
197.              return GestureDetector(
198.                onTap: () {
199.                  data.setData(3, name: 'Low');
200.                },
201.                child: Container(
202.                  padding: const EdgeInsets.all(8.0),
203.                  margin: const EdgeInsets.all(8.0),
204.                  height: 80.0,
205.                  decoration: BoxDecoration(
206.                    borderRadius: BorderRadius.circular(10.0),
207.                    color: Color(0xff4bbd6c),
208.                  ),
209.                  child: Column(
210.                    mainAxisAlignment: MainAxisAlignment.center,
211.                    children: <Widget>[
212.                      Text(
213.                        '+${CurrentUser.lowTicketCount}',
214.                        style: stats,
215.                      ),
216.                      const SizedBox(height: 5.0),
217.                      Text('Low'),
218.                    ]),
219.                ),
220.              );
221.            }),
222.          ),
223.        ]),
224.        Container(
225.          margin: const EdgeInsets.all(8.0),
226.          padding: const EdgeInsets.all(8.0),
227.          width: double.infinity,
228.          height: 350.0,
229.          decoration: BoxDecoration(
230.            borderRadius: BorderRadius.circular(5.0),
231.            color: Colors.white,
232.          ),
233.          child: FutureBuilder(
234.              future: getChartData(),
235.              builder: (c, s) {
236.                if (s.data == null) {
237.                  return Center(
238.                    child: CircularProgressIndicator(),
239.                  );
240.                } else {
241.                  return charts.TimeSeriesChart(
242.                    _createSampleData(s.data),
243.                    domainAxis: charts.DateTimeAxisSpec(
244.                      showAxisLine: true,
245.                    ),
246.                    animate: true,
247.                    animationDuration: Duration(seconds: 2),
248.                    behaviors: [
249.                      charts.SlidingViewport(),
250.                      charts.PanAndZoomBehavior(),
251.                      charts.LinePointHighlighter(),
252.                      charts.ChartTitle(
```

```
                    'Performance',
                    subTitle: 'Sep,2020',
                    titleStyleSpec: charts.TextStyleSpec(
                      color: charts.Color.fromHex(code: '#429dbe'),
                    ),
                  ),
                  new charts.ChartTitle(
                    'Dates',
                    titleStyleSpec: charts.TextStyleSpec(
                      color: charts.Color.fromHex(code: '#429dbe'),
                    ),
                    behaviorPosition: charts.BehaviorPosition.bottom,
                  ),
                  new charts.ChartTitle('PL Rate',
                      titleStyleSpec: charts.TextStyleSpec(
                        color: charts.Color.fromHex(code: '#429dbe'),
                      ),
                      behaviorPosition: charts.BehaviorPosition.start,
                      titleOutsideJustification:
                          charts.OutsideJustification.middleDrawArea),
                ],
                dateTimeFactory: const charts.LocalDateTimeFactory(),
                defaultRenderer: charts.LineRendererConfig(
                  includePoints: true,
                ),
              );
            }
          }),
        ),
      ]),
    )),
  );
 }
}
```

2. API

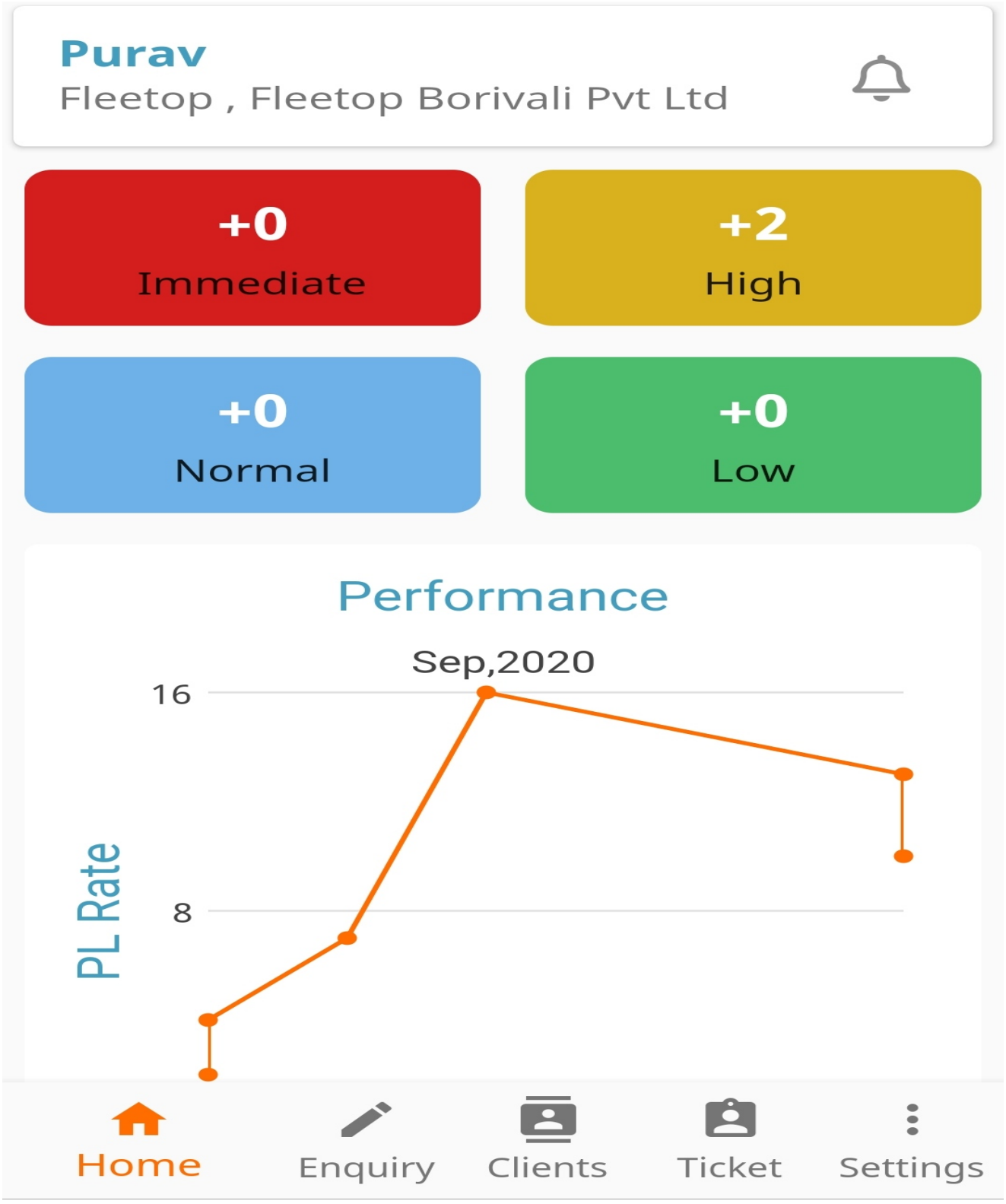
```
@RestController
@RequestMapping("/companyExecutivePL")
@CrossOrigin(origins = Constants.URL_TO_REQUEST)
public class CompanyExecutivePLController
{
@Autowired
CompanyExecutivePLBusinessLogic cebl;

@ApiImplicitParams({
@ApiImplicitParam(name="Authorization",value="${ivclient.request.authorization.description}",paramType="header")
})
@GetMapping(path="/{companyExecutiveId}",produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
public ResponseEntity<List<@Valid CompanyExecutivePLSelect>>
selectCompanyExecutivePLByExecutiveId(@PathVariable @NotNull Long companyExecutiveId)
{
return cebl.selectCompanyExecutivePLByExecutiveId(companyExecutiveId);
}
}
```

```
@Service
public class CompanyExecutivePLDAOImpl implements CompanyExecutivePLDAO
{
@Override
public List<CompanyExecutivePLSelect> selectCompanyExecutivePLByExecutiveId(Long companyExecutiveId)throws
SQLException, ClassNotFoundException
{
Connection c=ConnectionProvider.getConnection();
CallableStatement stmt=c.prepareCall("SELECT * FROM company.\"fn_selectCompanyExecutivePLByExecutiveId\"(?)
ORDER BY \"dateOfEntry\";");
stmt.setLong(1, companyExecutiveId);
ResultSet rs=stmt.executeQuery();
List<CompanyExecutivePLSelect> lcpl=new ArrayList<CompanyExecutivePLSelect>();
while(rs.next())
{
lcpl.add(new CompanyExecutivePLSelect(
rs.getLong("id"),
rs.getLong("executiveId"),
rs.getInt("PLRate"),
rs.getTimestamp("dateOfEntry")
));
}
rs.close();
stmt.close();
c.close();
return lcpl;
}
}
```

3. Database Scripts

```
1.  CREATE OR REPLACE FUNCTION company."fn_selectCompanyExecutivePLByExecutiveId"(
2.      "CompanyExecutiveIdIn" bigint)
3.       RETURNS TABLE(id bigint, "executiveId" bigint, "PLRate" integer, "dateOfEntry" timestamp without time
    zone)
4.      LANGUAGE 'sql'
5.
6.      COST 100
7.      VOLATILE
8.      ROWS 1000
9.  AS $BODY$
10.     SELECT
11.                             "id",
12.                             "executiveId",
13.                             "PLRate",
14.                             "dateOfEntry"
15.     FROM "company"."CompanyExecutivePL"
16.     WHERE "company"."CompanyExecutivePL"."executiveId"="CompanyExecutiveIdIn"
17.     $BODY$;
```

Screenshots:



10.1.2 Generating enquiry:

All the executives can generate enquiries readily based on their interaction with the customer. Enquires are logged, tracked, and managed systematically so that sales teams can act. This way, you can properly handle and follow up with everyone who requests more information, giving you more chances to sell and grow your business.

Code Snippet:

1. Frontend

```
@override
Widget build(BuildContext context) {
 return Scaffold(
  key: _scaffoldKey,
  appBar: AppBar(
   title: Text("Create Enquiry"),
   bottom: TabBar(
    controller: tabController,
    indicatorColor: Colors.white,
    tabs: <Widget>[
     Tab(icon: firstIcon),
     Tab(icon: secondIcon),
     Tab(icon: thirdIcon),
    ],
   ),
  ),
  body: TabBarView(
   controller: tabController,
   children: <Widget>[
    FutureBuilder(
      future: getCountries(),
      builder: (BuildContext context, AsyncSnapshot snapshot) {
       if (snapshot.data == null) {
        return Center(
         child: CircularProgressIndicator(),
        );
       } else {
        return Container(
         margin: const EdgeInsets.all(15.0),
         padding: const EdgeInsets.all(10.0),
         decoration: myBoxDecoration(),
         child: Form(
          key: _formKey,
          autovalidate: _autoValidate,
          child: ListView(
           children: <Widget>[
            Center(
             child: Text(
```

```
          "Enquiry Details",
          style: TextStyle(
            fontSize: 25.0,
            fontWeight: FontWeight.bold,
          ),
        ),
      ),
      Padding(
        padding: const EdgeInsets.only(top: 20.0),
        child: TextFormField(
          controller: enquiryRemarks,
          decoration: InputDecoration(
            labelText: "Enquiry Remarks",
            hintText: "Enter enquiry remarks",
            border: OutlineInputBorder(
                borderSide:
                    BorderSide(color: Colors.black)),
            focusedBorder: OutlineInputBorder(
              borderRadius:
                  BorderRadius.all(Radius.circular(5.0)),
              borderSide: BorderSide(
                  color: Theme.of(context).primaryColor),
            ),
          ),
          validator: (val) => val.isEmpty
              ? 'enquiry remarks is required'
              : null,
        )),
      Padding(
        padding: const EdgeInsets.only(top: 20.0),
        child: DropdownButtonHideUnderline(
          child: DropdownButtonFormField<String>(
            validator: (v) => v == null
                ? 'Please Select Enquiry Type'
                : null,
            decoration: InputDecoration(
              labelText: 'Enquiry Type',
              hintText: 'Select Enquiry Type',
            ),
            isDense: true,
            autovalidate: _autoValidate,
            items: enquiryTypeMenuItems,
            onChanged: (v) {
              FocusManager.instance.primaryFocus.unfocus();
              setState(() {
                enquiryValue = v;
              });
            },
            value: enquiryValue == '' ? null : enquiryValue,
          ),
        ),
      ),
      Padding(
        padding: const EdgeInsets.only(top: 20.0),
        child: DropdownButtonHideUnderline(
          child: DropdownButtonFormField<String>(
            validator: (v) =>
                v == null ? 'Please Select Client' : null,
            decoration: InputDecoration(
              labelText: 'Client',
```

```
                hintText: 'Select Client',
              ),
              isDense: true,
              autovalidate: _autoValidate,
              items: clientMenuItems,
              onChanged: (v) {
                FocusManager.instance.primaryFocus.unfocus();
                setState(() {
                  clientValue = v;
                });
              },
              value: clientValue == '' ? null : clientValue,
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(top: 20.0),
          child: DropdownButtonHideUnderline(
            child: DropdownButtonFormField<String>(
              validator: (v) =>
                  v == null ? 'Please Select Status' : null,
              decoration: InputDecoration(
                labelText: 'Status',
                hintText: 'Select Status',
              ),
              isDense: true,
              autovalidate: _autoValidate,
              items: statusMenuItems,
              onChanged: (v) {
                FocusManager.instance.primaryFocus.unfocus();
                setState(() {
                  statusValue = v;
                });
              },
              value: statusValue == '' ? null : statusValue,
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(top: 20.0),
          child: DropdownButtonHideUnderline(
            child: DropdownButtonFormField<String>(
              validator: (v) =>
                  v == null ? 'Please Select Priority' : null,
              decoration: InputDecoration(
                labelText: 'Priority',
                hintText: 'Select Priority',
              ),
              isDense: true,
              autovalidate: _autoValidate,
              items: priorityMenuItems,
              onChanged: (v) {
                FocusManager.instance.primaryFocus.unfocus();
                setState(() {
                  priorityValue = v;
                });
              },
              value:
                  priorityValue == '' ? null : priorityValue,
            ),
```

```
                  ),
                ),
                Container(
                  height: 60.0,
                ),
                Align(
                    alignment: Alignment.bottomRight,
                    child: RaisedButton(
                      onPressed: () {
                        FocusManager.instance.primaryFocus.unfocus();
                        FormState form = _formKey.currentState;
                        if (form.validate()) {
                          setState(() {
                            firstIcon = Icon(Icons.check_circle);
                            completed[0] = true;
                          });

                          tabController.animateTo(1);
                        } else {
                          setState(() {
                            _autoValidate = true;
                            firstIcon = Icon(Icons.error_outline);
                          });
                        }
                      },
                      color: Theme.of(context).accentColor,
                      child: Padding(
                        padding: EdgeInsets.fromLTRB(
                            0.5 * 5, 0, 0.5 * 5, 0),
                        child: Row(
                          mainAxisAlignment: MainAxisAlignment.end,
                          mainAxisSize: MainAxisSize.min,
                          children: <Widget>[
                            Text(
                              'Submit (1/3) ',
                              style: TextStyle(
                                fontSize: 20,
                                fontWeight: FontWeight.w700,
                                color: Colors.white,
                              ),
                            ),
                            Container(width: 10.0),
                            Icon(
                              Icons.arrow_forward,
                              color: Colors.white,
                            ),
                          ],
                        ),
                      ),
                    )),
              ],
            ),
          ),
        ); //AddComapnyAddress(),
      }
    }),

Container(
  margin: const EdgeInsets.all(15.0),
  padding: const EdgeInsets.all(10.0),
```

```
        decoration: myBoxDecoration(),
        child: Form(
          key: _formKey2,
          autovalidate: _autoValidate2,
          child: ListView(
            children: <Widget>[
              Center(
                child: Text(
                  "Enquiry Address",
                  style: TextStyle(
                    fontSize: 25.0,
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ),
              Padding(
                  padding: const EdgeInsets.only(top: 20.0),
                  child: TextFormField(
                    controller: addressLine1,
                    decoration: InputDecoration(
                      labelText: "Address Line 1",
                      hintText: "Enter address line 1",
                      border: OutlineInputBorder(
                          borderSide: BorderSide(color: Colors.black)),
                      focusedBorder: OutlineInputBorder(
                        borderRadius:
                            BorderRadius.all(Radius.circular(5.0)),
                        borderSide: BorderSide(
                            color: Theme.of(context).primaryColor),
                      ),
                    ),
                    validator: (val) =>
                        val.isEmpty ? 'Address line 1 is required' : null,
                  )),
              Padding(
                  padding: const EdgeInsets.only(top: 20.0),
                  child: TextFormField(
                    controller: addressLine2,
                    decoration: InputDecoration(
                      labelText: "Address Line 2",
                      hintText: "Enter address Line 2 ",
                      border: OutlineInputBorder(),
                      focusedBorder: OutlineInputBorder(
                        borderRadius:
                            BorderRadius.all(Radius.circular(5.0)),
                        borderSide: BorderSide(
                            color: Theme.of(context).primaryColor),
                      ),
                    ),
                    validator: (val) =>
                        val.isEmpty ? 'address Line 2 is required' : null,
                  )),
              Padding(
                  padding: const EdgeInsets.only(top: 20.0),
                  child: TextFormField(
                    controller: addressLine3,
                    decoration: InputDecoration(
                      labelText: "Address Line 3",
                      hintText: "Enter address Line 3 ",
                      border: OutlineInputBorder(),
```

```
            focusedBorder: OutlineInputBorder(
              borderRadius:
                  BorderRadius.all(Radius.circular(5.0)),
              borderSide: BorderSide(
                  color: Theme.of(context).primaryColor),
            ),
          ),
          validator: (val) =>
              val.isEmpty ? 'address Line 3 is required' : null,
        )),
    Padding(
        padding: const EdgeInsets.only(top: 20.0),
        child: TextFormField(
          controller: pincode,
          decoration: InputDecoration(
            labelText: "Pincode",
            hintText: "Enter pincode ",
            border: OutlineInputBorder(),
            focusedBorder: OutlineInputBorder(
              borderRadius:
                  BorderRadius.all(Radius.circular(5.0)),
              borderSide: BorderSide(
                  color: Theme.of(context).primaryColor),
            ),
          ),
          keyboardType: TextInputType.phone,
          inputFormatters: [
            LengthLimitingTextInputFormatter(6),
            WhitelistingTextInputFormatter(
                new RegExp(r'^[()\d -]{1,15}$')),
          ],
          validator: (val) =>
              val.isEmpty ? 'Pincode is required' : null,
        )),
    FutureBuilder(
        future: getLocationData(),
        builder: (c, s) {
          if (s.data == null) {
            return Center(
              child: CircularProgressIndicator(),
            );
          } else {
            return StatefulBuilder(builder: (c, st) {
              return Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: <Widget>[
                    DropdownButtonHideUnderline(
                      child: DropdownButtonFormField<int>(
                        validator: (v) => v == null
                            ? 'Please Select Country'
                            : null,
                        isDense: true,
                        decoration: InputDecoration(
                          labelText: 'Country',
                          hintText: 'Select Country',
                        ),
                        autovalidate: _autoValidate2,
                        items: s.data,
                        onChanged: (v) async {
                          FocusManager.instance.primaryFocus
```

```
                .unfocus();
            stateMenuItems.clear();
            cityMenuItems.clear();
            areaMenuItems.clear();
            var r = await ApiCall.getDataFromApi(
                Uri.GET_STATE_FROM_COUNTRY +
                    '/${v.toString()}');
            for (int i = 0; i < r.length; i++) {
              stateMenuItems.add(
                DropdownMenuItem<int>(
                  value: r[i]['stateID'],
                  child: Text(r[i]['stateName']),
                ),
              );
            }
            st(() {
              stateAvail = true;
              cityAvail = false;
              areaAvail = false;

              stateValue = -1;
              cityValue = -1;
              areaValue = -1;
              countryValue = v;
            });
          },
          value: countryValue == -1
              ? null
              : countryValue,
        ),
      ),
      DropdownButtonHideUnderline(
        child: DropdownButtonFormField<int>(
          validator: (v) {
            if (stateAvail) {
              if (v == null) {
                return 'Please Select State';
              } else {
                return null;
              }
            } else {
              return null;
            }
          },
          decoration: InputDecoration(
            labelText: 'State',
            hintText: 'Please Select State',
          ),
          isDense: true,
          autovalidate: _autoValidate2,
          items: stateMenuItems,
          onChanged: stateAvail
              ? (v) async {
                  FocusManager.instance.primaryFocus
                      .unfocus();
                  cityMenuItems.clear();
                  areaMenuItems.clear();
                  var r = await ApiCall.getDataFromApi(
                      Uri.GET_BUSINESS_CITY_FROM_STATE +
                          '/${v.toString()}?ownerID=${CurrentUser.ownerId}');
```

```
                for (int i = 0;
                    i < r.length;
                    i++) {
                  cityMenuItems.add(
                    DropdownMenuItem<int>(
                      value: r[i][
                          'businessCityForCompanyID'],
                      child: Text(r[i][
                          'businessCityForCompanyName']),
                    ),
                  );
                }
                st(() {
                  cityAvail = true;
                  areaAvail = false;

                  cityValue = -1;
                  areaValue = -1;
                  stateValue = v;
                });
              }
            : null,
        value:
            stateValue == -1 ? null : stateValue,
      ),
    ),
    DropdownButtonHideUnderline(
      child: DropdownButtonFormField<int>(
        validator: (v) {
          if (stateAvail && cityAvail) {
            if (v == null) {
              return 'Please Select City';
            } else {
              return null;
            }
          } else {
            return null;
          }
        },
        decoration: InputDecoration(
          labelText: 'City',
          hintText: 'Select City',
        ),
        isDense: true,
        autovalidate: _autoValidate2,
        items: cityMenuItems,
        onChanged: cityAvail
            ? (v) async {
                FocusManager.instance.primaryFocus
                    .unfocus();
                areaMenuItems.clear();
                var r = await ApiCall.getDataFromApi(
                    Uri.GET_BUSINESS_AREA_FROM_BUSINESS_CITY +
                        '/${v.toString()}?ownerID=${CurrentUser.ownerId}');
                for (int i = 0;
                    i < r.length;
                    i++) {
                  areaMenuItems.add(
                    DropdownMenuItem<int>(
                      value: r[i][
```

```
                                'businessAreaForCompanyID'],
                            child: Text(r[i][
                                'businessAreaForCompanyName'])),
                        );
                      }
                      st(() {
                        areaAvail = true;

                        areaValue = -1;
                        cityValue = v;
                      });
                    }
                  : null,
              value: cityValue == -1 ? null : cityValue,
            ),
          ),
          DropdownButtonHideUnderline(
            child: DropdownButtonFormField<int>(
              validator: (v) {
                if (stateAvail &&
                    cityAvail &&
                    areaAvail) {
                  if (v == null) {
                    return 'Please Select Area.';
                  } else {
                    return null;
                  }
                } else {
                  return null;
                }
              },
              decoration: InputDecoration(
                labelText: 'Area',
                hintText: 'Please Select Area',
              ),
              autovalidate: _autoValidate2,
              isDense: true,
              items: areaMenuItems,
              onChanged: (v) {
                FocusManager.instance.primaryFocus
                    .unfocus();
                st(() {
                  areaValue = v;
                });
              },
              value: areaValue == -1 ? null : areaValue,
            ),
          ),
        ]);
      });
    }
  }),
Container(
  height: 60.0,
),
Align(
  alignment: Alignment.bottomRight,
  child: RaisedButton(
    onPressed: () {
      FormState form = _formKey2.currentState;
```

```
                if (form.validate()) {
                  _formKey2.currentState.save();

                  setState(() {
                    secondIcon = Icon(Icons.check_circle);
                    completed[1] = true;
                  });

                  tabController.animateTo(2);
                } else {
                  setState(() {
                    _autoValidate2 = true;
                    secondIcon = Icon(Icons.error_outline);
                  });
                }
              },
              color: Theme.of(context).accentColor,
              child: Padding(
                padding: EdgeInsets.fromLTRB(0.5 * 5, 0, 0.5 * 5, 0),
                child: Row(
                  mainAxisAlignment: MainAxisAlignment.end,
                  mainAxisSize: MainAxisSize.min,
                  children: <Widget>[
                    Text(
                      'Submit (2/3) ',
                      style: TextStyle(
                        fontSize: 20,
                        fontWeight: FontWeight.w700,
                        color: Colors.white,
                      ),
                    ),
                    Container(width: 10.0),
                    Icon(
                      Icons.arrow_forward,
                      color: Colors.white,
                    ),
                  ],
                ),
              ),
            )),
      ],
    ),
  ),
), //AddCompanyRegionArea(),

FutureBuilder(
    future: getProducts(),
    builder: (c, s) {
      if (s.data == null) {
        return Center(
          child: CircularProgressIndicator(),
        );
      } else {
        return Container(
          margin: const EdgeInsets.all(15.0),
          padding: const EdgeInsets.all(10.0),
          decoration: myBoxDecoration(),
          child: Form(
            key: _formKey3,
            autovalidate: _autoValidate3,
```

```
child: ListView(
  children: <Widget>[
    Center(
      child: Text(
        "Product And Time",
        style: TextStyle(
          fontSize: 25.0,
          fontWeight: FontWeight.bold,
        ),
      ),
    ),
    widget.companyOptions
        ? Padding(
            padding: const EdgeInsets.only(
                left: 16.0, right: 16.0),
            child: DropdownButtonHideUnderline(
              child: DropdownButtonFormField(
                validator: (v) => v == null
                    ? 'Please Select Company'
                    : null,
                isDense: true,
                decoration: InputDecoration(
                  labelText: 'Company',
                  hintText: 'Select Company',
                ),
                autovalidate: _autoValidate,
                items: companyMenuItems,
                onChanged: (v) {
                  FocusManager.instance.primaryFocus
                      .unfocus();
                  setState(() {
                    companyValue = v;
                  });
                },
                value: companyValue == -1
                    ? null
                    : companyValue,
              ),
            ))
        : Padding(
            padding:
                EdgeInsets.only(left: 10.0, right: 16.0),
            child: TextFormField(
              enabled: false,
              decoration: InputDecoration(
                icon: Icon(FontAwesomeIcons.building),
                labelText: 'Company Name',
              ),
              controller: companyName,
            ),
          ),
    Padding(
      padding: const EdgeInsets.only(top: 20.0),
      child: MultiSelectFormField(
          autovalidate: false,
          titleText: 'Products',
          validator: (value) {
            if (value == null || value.length == 0) {
              return 'Please select one or more options';
            }
```

```
            return null;
          },
          dataSource: productMenuItems.map((s) {
            return {'display': '$s', 'value': s};
          }).toList(),
          textField: 'display',
          valueField: 'value',
          okButtonLabel: 'OK',
          cancelButtonLabel: 'CANCEL',
          hintText: 'Please choose one or more',
          onSaved: (value) {
            setState(() {
              selectedProducts = value;
            });
          }),
      ),
      Row(children: <Widget>[
        new Expanded(
            child: new TextFormField(
          decoration: new InputDecoration(
            icon: const Icon(Icons.calendar_today),
            hintText: 'Enter Start Date',
            labelText: 'Start date',
          ),
          controller: startDate,
          keyboardType: TextInputType.datetime,
          validator: (val) => isValidStartDate(val)
              ? null
              : 'Not a valid date',
        )),
        new IconButton(
          icon: new Icon(Icons.more_horiz),
          tooltip: 'Choose date',
          onPressed: (() {
            chooseStartDate(context);
          }),
        )
      ]),
      Row(children: <Widget>[
        new Expanded(
            child: new TextFormField(
          decoration: new InputDecoration(
            icon: const Icon(Icons.calendar_today),
            hintText: 'Enter End Date',
            labelText: 'End date',
          ),
          controller: endDate,
          keyboardType: TextInputType.datetime,
          validator: (val) => isValidEndDate(val)
              ? null
              : 'Not a valid date',
        )),
        new IconButton(
          icon: new Icon(Icons.more_horiz),
          tooltip: 'Choose date',
          onPressed: (() {
            chooseEndDate(context);
          }),
        )
      ]),
```

```
Container(
  height: 60.0,
),
Align(
   alignment: Alignment.bottomRight,
   child: RaisedButton(
    onPressed: () {
     FormState form = _formKey3.currentState;
     if (form.validate()) {
      setState(() {
       thirdIcon = Icon(Icons.check_circle);
       completed[2] = true;
       for (int i = 0;
          i < completed.length;
          i++) {
        if (!completed[i]) {
         int ii = i + 1;
         String ii2 = ii.toString();
         if (page == '') {
          page = page + ii2;
         } else {
          page = page + ' , ' + ii2;
         }

         index = false;
        }
       }
       if (index) {
        addData();
       } else {
        showMessage(
           "Looks like some fields are missing please review page no. ${page}",
           Theme.of(context).primaryColor);
        page = "";
        index = true;
       }
      });
     } else {
      setState(() {
       _autoValidate3 = true;
       thirdIcon = Icon(Icons.error_outline);
      });
     }
    },
    color: Theme.of(context).accentColor,
    child: Padding(
     padding: EdgeInsets.fromLTRB(
        0.5 * 5, 0, 0.5 * 5, 0),
     child: Row(
      mainAxisAlignment: MainAxisAlignment.end,
      mainAxisSize: MainAxisSize.min,
      children: <Widget>[
       Text(
        'Submit (3/3) ',
        style: TextStyle(
         fontSize: 20,
         fontWeight: FontWeight.w700,
         color: Colors.white,
        ),
       ),
```

```
                        Container(width: 10.0),
                        Icon(
                          Icons.arrow_forward,
                          color: Colors.white,
                        ),
                      ],
                    ),
                  ),
                )),
              ],
            ),
          ),
        );
      }
    }),
  ],
 ),
);
}
```

2. API

```
1.  @RestController
2.  @RequestMapping("/enquiry")
3.  @CrossOrigin(origins = Constants.URL_TO_REQUEST)
4.  public class EnquiryController
5.  {
6.  @Autowired
7.  EnquiryBusinessLogic ebl;
8.
9.
10.   @ApiImplicitParams({
11.   @ApiImplicitParam(name="Authorization",value="${ivclient.request.authorization.description}",paramType="header")
12.   })
13.   @PostMapping(consumes = { MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE },
14.   produces = { MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE })
15.   public ResponseEntity<Void> addEnquiry(@Valid @RequestBody EnquiryInsert ei)
16.   {
17.   return ebl.addEnquiry(ei);
18.   }
19.   }
20.
21.   @Service
22.   public class EnquiryDAOImpl implements EnquiryDAO
23.   {
24.   @Override
25.   public Long addEnquiry(EnquiryInsert ei) throws SQLException, ClassNotFoundException
26.   {
27.   Connection c=ConnectionProvider.getConnection();
28.   CallableStatement stmt=c.prepareCall("SELECT * FROM enquiry.\"fn_insertEnquiry\"(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);");
29.   stmt.setLong(1, ei.getCompanyId());
30.   stmt.setString(2, ei.getEnquiryRemarks());
31.   stmt.setLong(3, ei.getEnquiryType());
```

```
32.  stmt.setLong(4, ei.getClientId());
33.  stmt.setLong(5, ei.getCountryId());
34.  stmt.setLong(6, ei.getStateId());
35.  stmt.setLong(7, ei.getCityId());
36.  stmt.setLong(8, ei.getAreaId());
37.  stmt.setString(9, ei.getAddressLine1());
38.  stmt.setString(10, ei.getAddressLine2());
39.  stmt.setString(11, ei.getAddressLine3());
40.  stmt.setString(12, ei.getPincode());
41.  stmt.setString(13, ei.getLatitude());
42.  stmt.setString(14, ei.getLongitude());
43.  stmt.setTimestamp(15, ei.getStartDateAndTime());
44.  stmt.setTimestamp(16, ei.getDeadlineDateAndTime());
45.  stmt.setLong(17, ei.getCreatedBy());
46.  stmt.setTimestamp(18, ei.getCreatedOn());
47.  ResultSet rs=stmt.executeQuery();
48.  Long enquiryId=null;
49.  c.commit();
50.  if(rs.next())
51.  {
52.  enquiryId=rs.getLong("EnquiryId");
53.  }
54.  rs.close();
55.  stmt.close();
56.  c.close();
57.  return enquiryId;
58.  }
59.  }
```

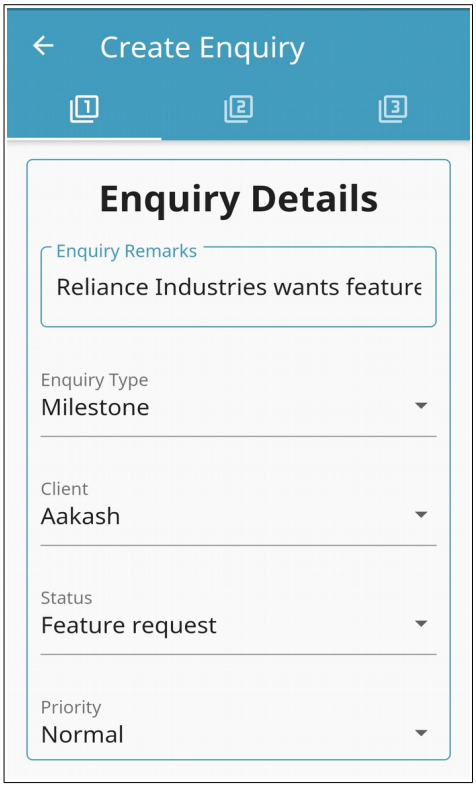
3. Database Scripts

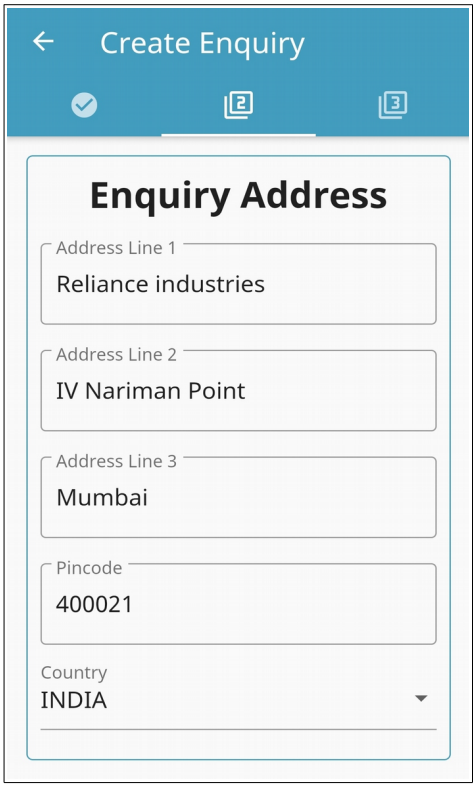
```
1.  CREATE FUNCTION enquiry."fn_insertEnquiry"(
2.  "CompanyIdIn" bigint, "EnquiryRemarksIn" text, "EnquiryTypeIn" bigint, "ClientIdIn" bigint, "CountryIdIn"
    bigint, "StateIdIn" bigint, "CityIdIn" bigint, "AreaIdIn" bigint, "AddressLine1In" text, "AddressLine2In" text,
    "AddressLine3In" text, "PincodeIn" text, "LatitudeIn" text, "LongitudeIn" text, "startDateAndTimeIn" timestamp
    without time zone, "deadlineDateAndTimeIn" timestamp without time zone, "CreatedByIn" bigint,
    "CreatedOnIn" timestamp without time zone
3.  ) RETURNS TABLE("EnquiryId" bigint)
4.     LANGUAGE sql
5.     AS $$
6.  WITH "MainEnquiryRegion" AS(
7.  INSERT INTO "enquiry"."MainEnquiry"
8.     (
9.                  "CompanyId",
10.                          "CompanyName",
11.                          "EnquiryRemarks",
12.                          "CreatedBy",
13.                          "CreatedOn",
14.                          "LastEditBy",
15.                          "LastEditOn",
16.                          "EnquiryType",
17.                          "EnquiryTypeName"
18.              )
```

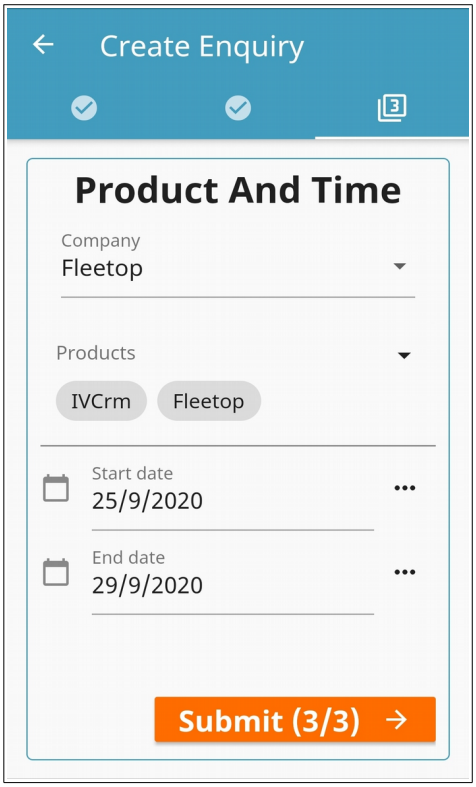
```
19.  VALUES
20.             (
21.                                     "CompanyIdIn",
22.                                     (SELECT "CompanyName" FROM company."Company" WHERE
    "CompanyId"="CompanyIdIn" AND "MarkForDelete"=false LIMIT 1),
23.                                     "EnquiryRemarksIn",
24.                                     "CreatedByIn",
25.                                     "CreatedOnIn",
26.                                     "CreatedByIn",
27.                                     "CreatedOnIn",
28.                                     "EnquiryTypeIn",
29.                                     (SELECT "EnquiryTypeName" FROM enquiry."EnquiryType" WHERE
    "EnquiryTypeId"="EnquiryTypeIn" AND "MarkForDelete"=false LIMIT 1)
30.             )RETURNING "EnquiryId"
31.  ),
32.  "SubEnquiryRegion1" AS(
33.  INSERT INTO "enquiry"."EnquiryLocation"
34.             (
35.                             "EnquiryId",
36.                             "CountryId",
37.                             "CountryName",
38.                             "StateId",
39.                             "StateName",
40.                             "CityId",
41.                             "CityName",
42.                             "AreaId",
43.                             "AreaName",
44.                             "AddressLine1",
45.                             "AddressLine2",
46.                             "AddressLine3",
47.                             "Pincode",
48.                             "Latitude",
49.                             "Longitude"
50.             )
51.  VALUES
52.             (
53.                                     (SELECT "EnquiryId" FROM "MainEnquiryRegion"),
54.                                     "CountryIdIn",
55.                                     (SELECT "CountryName" FROM location."countryinfo" WHERE
    "CountryID"="CountryIdIn" AND "MarkForDelete"=false LIMIT 1),
56.                                     "StateIdIn",
57.                                     (SELECT "StateName" FROM location."stateinfo" WHERE
    "StateID"="StateIdIn" AND "MarkForDelete"=false LIMIT 1),
58.                                     "CityIdIn",
59.                                     (SELECT "CityName" FROM location."cityinfo" WHERE "CityID"
    IN(SELECT "CityID"
60.
                                         FROM location."businessCityForCompany"
61.
                                         WHERE "BusinessCityForCompanyID"="CityIdIn"
62.
                                         LIMIT 1) AND "MarkForDelete"=false LIMIT 1),
63.                                     "AreaIdIn",
64.                                     (SELECT "AreaName" FROM location."areainfo" WHERE "AreaID" IN
    (SELECT "AreaID"
65.
                                     FROM location."businessAreaForCompany"
```

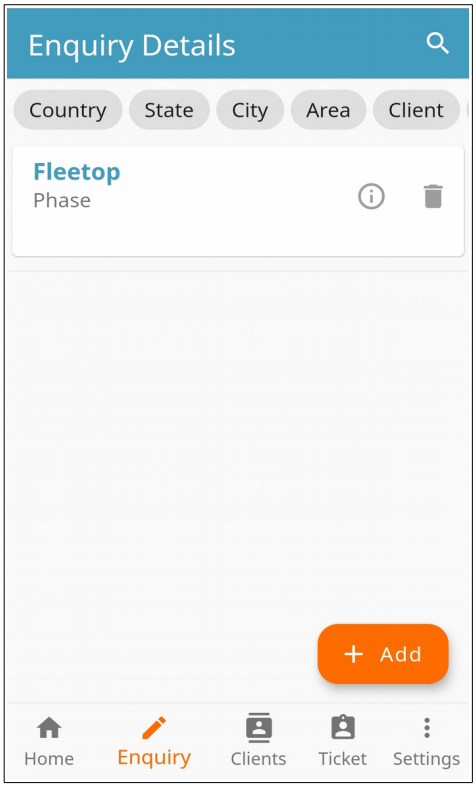
```
66.
                                        WHERE "BusinessAreaForCompanyID"="AreaIdIn"
67.
                                        LIMIT 1) AND "MarkForDelete"=false LIMIT 1),
68.                                     "AddressLine1In",
69.                                     "AddressLine2In",
70.                                     "AddressLine3In",
71.                                     "PincodeIn",
72.                                     "LatitudeIn",
73.                                     "LongitudeIn"
74.             )RETURNING "EnquiryId"
75.   ),
76.   "SubEnquiryRegion2" AS(
77.   INSERT INTO "enquiry"."EnquiryClient"
78.             (
79.                     "EnquiryId",
80.                     "ClientName",
81.                     "ContactPerson",
82.                     "EmailId",
83.                     "ContactNumber",
84.                     "ClientId"
85.             )
86.   VALUES
87.             (
88.                                     (SELECT "EnquiryId" FROM "SubEnquiryRegion1"),
89.                                     (SELECT "ContactName" FROM client."Client" WHERE
    "ClientId"="ClientIdIn" AND "MarkForDelete"=false LIMIT 1),
90.                                     (SELECT "ContactPerson" FROM client."Client" WHERE
    "ClientId"="ClientIdIn" AND "MarkForDelete"=false LIMIT 1),
91.                                     (SELECT "EmailId" FROM client."Client" WHERE "ClientId"="ClientIdIn"
    AND "MarkForDelete"=false LIMIT 1),
92.                                     (SELECT "ContactNumber" FROM client."Client" WHERE
    "ClientId"="ClientIdIn" AND "MarkForDelete"=false LIMIT 1),
93.                                     "ClientIdIn"
94.             )RETURNING "EnquiryId"
95.   ),
96.   "SubEnquiryRegion3" AS(
97.   INSERT INTO "enquiry"."EnquiryDateAndTimelines"
98.             (
99.                     "EnquiryId",
100.                    "startDateAndTime",
101.                    "deadlineDateAndTime"
102.            )
103.  VALUES
104.            (
105.                    (SELECT "EnquiryId" FROM "SubEnquiryRegion2"),
106.                    "startDateAndTimeIn",
107.                    "deadlineDateAndTimeIn"
108.            )RETURNING "EnquiryId"
109.  ) $$
```

Screenshots:









10.1.3 Adding new clients:

Executives can add new clients after understanding their requirements to generate new enquires. This helps in generating a large database of potential user information.

Code Snippet:

1. Frontend

```
class AddClientState extends State<AddClient> {
  makeForm(context1) {
    return IgnorePointer(
      ignoring: isLoading,
      child: Stack(children: <Widget>[
        FutureBuilder(
          future: getAllData(),
          builder: (c, s) {
            if (s.data == null) {
              return Center(
                child: CircularProgressIndicator(),
              );
            } else {
              return Form(
                key: _fbKey,
                autovalidate: _autoValidate,
                child: ListView(children: <Widget>[
                  Padding(
                    padding: const EdgeInsets.only(left: 10.0, right: 16.0),
                    child: TextFormField(
                      controller: nameController,
                      decoration: InputDecoration(
                        icon: Icon(Icons.person_add),
                        hintText: 'Enter name',
                        labelText: 'Name',
                      ),
                      validator: (v) =>
                          v.isEmpty ? 'Name is required .' : null,
                    ),
                  ),
                  Padding(
                    padding: const EdgeInsets.only(left: 10.0, right: 16.0),
                    child: TextFormField(
                      controller: contactPersonController,
                      decoration: InputDecoration(
                        icon: Icon(Icons.account_circle),
                        hintText: 'Enter contact person',
                        labelText: 'Contact Person',
                      ),
                      validator: (v) =>
```

```
41.                     v.isEmpty ? 'Contact person is required .' : null,
42.                 ),
43.               ),
44.               Padding(
45.                 padding: const EdgeInsets.only(left: 10.0, right: 16.0),
46.                 child: TextFormField(
47.                   controller: emailController,
48.                   decoration: InputDecoration(
49.                     icon: Icon(Icons.email),
50.                     hintText: 'Enter Email ID',
51.                     labelText: 'Email ID',
52.                   ),
53.                   keyboardType: TextInputType.emailAddress,
54.                   validator: (value) => isValidEmail(value)
55.                       ? (isEmailContains(value)
56.                           ? null
57.                           : 'Email must contain @ and .')
58.                       : 'Please enter a valid email address',
59.                 ),
60.               ),
61.               Padding(
62.                 padding: const EdgeInsets.only(left: 10.0, right: 16.0),
63.                 child: TextFormField(
64.                   controller: contactNumberController,
65.                   decoration: InputDecoration(
66.                     icon: Icon(Icons.phone),
67.                     hintText: 'Enter a phone number',
68.                     labelText: 'Phone',
69.                   ),
70.                   keyboardType: TextInputType.phone,
71.                   inputFormatters: [
72.                     LengthLimitingTextInputFormatter(10),
73.                     WhitelistingTextInputFormatter(
74.                         new RegExp(r'^[()\d -]{1,15}$')),
75.                   ],
76.                   validator: (value) => isValidPhoneNumber(value)
77.                       ? null
78.                       : 'Phone number must be of 10 numbers',
79.                 ),
80.               ),
81.               widget.companyOptions
82.                   ? Padding(
83.                       padding:
84.                           const EdgeInsets.only(left: 16.0, right: 16.0),
85.                       child: DropdownButtonHideUnderline(
86.                         child: DropdownButtonFormField(
87.                           validator: (v) =>
88.                               v == null ? 'Please Select Company' : null,
89.                           isDense: true,
90.                           decoration: InputDecoration(
91.                             labelText: 'Company',
92.                             hintText: 'Select Company',
93.                           ),
94.                           autovalidate: _autoValidate,
95.                           items: companyMenuItems,
96.                           onChanged: (v) {
```

```
97.                         FocusManager.instance.primaryFocus.unfocus();
98.                         setState(() {
99.                           companyValue = v;
100.                        });
101.                      },
102.                      value: companyValue == -1 ? null : companyValue,
103.                    ),
104.                  ))
105.              : Padding(
106.                  padding: EdgeInsets.only(left: 10.0, right: 16.0),
107.                  child: TextFormField(
108.                    enabled: false,
109.                    decoration: InputDecoration(
110.                      icon: Icon(FontAwesomeIcons.building),
111.                      labelText: 'Company Name',
112.                    ),
113.                    controller: companyName,
114.                  ),
115.                ),
116.            Padding(
117.              padding: const EdgeInsets.only(left: 10.0, right: 16.0),
118.              child: TextFormField(
119.                controller: al1Controller,
120.                decoration: InputDecoration(
121.                  icon: Icon(Icons.place),
122.                  hintText: 'Enter Address Line 1',
123.                  labelText: 'Address Line 1',
124.                ),
125.                validator: (v) =>
126.                    v.isEmpty ? 'Address Line 1 is required .' : null,
127.              ),
128.            ),
129.            Padding(
130.              padding: const EdgeInsets.only(left: 10.0, right: 16.0),
131.              child: TextFormField(
132.                controller: al2Controller,
133.                decoration: InputDecoration(
134.                  icon: Icon(Icons.place),
135.                  hintText: 'Enter Address Line 2',
136.                  labelText: 'Address Line 2',
137.                ),
138.                validator: (v) =>
139.                    v.isEmpty ? 'Address Line 2 is required .' : null,
140.              ),
141.            ),
142.            Padding(
143.              padding: const EdgeInsets.only(left: 10.0, right: 16.0),
144.              child: TextFormField(
145.                controller: al3Controller,
146.                decoration: InputDecoration(
147.                  icon: Icon(Icons.place),
148.                  hintText: 'Enter Address Line 3',
149.                  labelText: 'Address Line 3',
150.                ),
151.                validator: (v) =>
152.                    v.isEmpty ? 'Address Line 3 is required .' : null,
```

```
153.               ),
154.             ),
155.             Padding(
156.               padding: const EdgeInsets.only(left: 10.0, right: 16.0),
157.               child: TextFormField(
158.                 controller: pincodeController,
159.                 decoration: InputDecoration(
160.                   icon: Icon(Icons.fiber_pin),
161.                   hintText: 'Enter Pincode',
162.                   labelText: 'Pincode',
163.                 ),
164.                 validator: (v) =>
165.                     v.isEmpty ? 'Pincode is required .' : null,
166.               ),
167.             ),
168.             Padding(
169.               padding: const EdgeInsets.only(left: 16.0, right: 16.0),
170.               child: StatefulBuilder(builder: (c, st) {
171.                 return Column(
172.                   crossAxisAlignment: CrossAxisAlignment.start,
173.                   children: <Widget>[
174.                     DropdownButtonHideUnderline(
175.                       child: DropdownButtonFormField<int>(
176.                         validator: (v) => v == null
177.                             ? 'Please Select Country'
178.                             : null,
179.                         isDense: true,
180.                         decoration: InputDecoration(
181.                           labelText: 'Country',
182.                           hintText: 'Select Country',
183.                         ),
184.                         autovalidate: _autoValidate,
185.                         items: countryMenuItems,
186.                         onChanged: (v) async {
187.                           FocusManager.instance.primaryFocus
188.                               .unfocus();
189.                           stateMenuItems.clear();
190.                           cityMenuItems.clear();
191.                           areaMenuItems.clear();
192.                           var r = await ApiCall.getDataFromApi(
193.                               Uri.GET_STATE_FROM_COUNTRY +
194.                                   '/${v.toString()}');
195.                           for (int i = 0; i < r.length; i++) {
196.                             stateMenuItems.add(
197.                               DropdownMenuItem<int>(
198.                                 value: r[i]['stateID'],
199.                                 child: Text(r[i]['stateName']),
200.                               ),
201.                             );
202.                           }
203.                           st(() {
204.                             stateAvail = true;
205.                             cityAvail = false;
206.                             areaAvail = false;
207.
208.                             stateValue = -1;
```

```
                        cityValue = -1;
                        areaValue = -1;
                        countryValue = v;
                      });
                    },
                    value: countryValue == -1
                        ? null
                        : countryValue,
                  ),
                ),
                DropdownButtonHideUnderline(
                  child: DropdownButtonFormField<int>(
                    validator: (v) {
                      if (stateAvail) {
                        if (v == null) {
                          return 'Please Select State';
                        } else {
                          return null;
                        }
                      } else {
                        return null;
                      }
                    },
                    decoration: InputDecoration(
                      labelText: 'State',
                      hintText: 'Please Select State',
                    ),
                    isDense: true,
                    autovalidate: _autoValidate,
                    items: stateMenuItems,
                    onChanged: stateAvail
                        ? (v) async {
                            FocusManager.instance.primaryFocus
                                .unfocus();
                            cityMenuItems.clear();
                            areaMenuItems.clear();
                            var r = await ApiCall.getDataFromApi(
                                Uri.GET_BUSINESS_CITY_FROM_STATE +
                                    '/${v.toString()}?ownerID=${CurrentUser.ownerId}');
                            for (int i = 0; i < r.length; i++) {
                              cityMenuItems.add(
                                DropdownMenuItem<int>(
                                  value: r[i][
                                      'businessCityForCompanyID'],
                                  child: Text(r[i][
                                      'businessCityForCompanyName']),
                                ),
                              );
                            }
                            st(() {
                              cityAvail = true;
                              areaAvail = false;

                              cityValue = -1;
                              areaValue = -1;
                              stateValue = v;
```

```
                                });
                              }
                            : null,
                        value: stateValue == -1 ? null : stateValue,
                      ),
                    ),
                    DropdownButtonHideUnderline(
                      child: DropdownButtonFormField<int>(
                        validator: (v) {
                          if (stateAvail && cityAvail) {
                            if (v == null) {
                              return 'Please Select City';
                            } else {
                              return null;
                            }
                          } else {
                            return null;
                          }
                        },
                        decoration: InputDecoration(
                          labelText: 'City',
                          hintText: 'Select City',
                        ),
                        isDense: true,
                        autovalidate: _autoValidate,
                        items: cityMenuItems,
                        onChanged: cityAvail
                            ? (v) async {
                                FocusManager.instance.primaryFocus
                                    .unfocus();
                                areaMenuItems.clear();
                                var r = await ApiCall.getDataFromApi(
                                    Uri.GET_BUSINESS_AREA_FROM_BUSINESS_CITY +
                                        '/${v.toString()}?ownerID=${CurrentUser.ownerId}');
                                for (int i = 0; i < r.length; i++) {
                                  areaMenuItems.add(
                                    DropdownMenuItem<int>(
                                        value: r[i][
                                            'businessAreaForCompanyID'],
                                        child: Text(r[i][
                                            'businessAreaForCompanyName'])),
                                  );
                                }
                                st(() {
                                  areaAvail = true;

                                  areaValue = -1;
                                  cityValue = v;
                                });
                              }
                            : null,
                        value: cityValue == -1 ? null : cityValue,
                      ),
                    ),
                    DropdownButtonHideUnderline(
                      child: DropdownButtonFormField<int>(
```

```
                    validator: (v) {
                      if (stateAvail &&
                          cityAvail &&
                          areaAvail) {
                        if (v == null) {
                          return 'Please Select Area.';
                        } else {
                          return null;
                        }
                      } else {
                        return null;
                      }
                    },
                    decoration: InputDecoration(
                      labelText: 'Area',
                      hintText: 'Please Select Area',
                    ),
                    isDense: true,
                    autovalidate: _autoValidate,
                    items: areaMenuItems,
                    onChanged: (v) {
                      FocusManager.instance.primaryFocus
                          .unfocus();
                      st(() {
                        areaValue = v;
                      });
                    },
                    value: areaValue == -1 ? null : areaValue,
                  ),
                ),
              ]);
            })),
          Container(
            padding:
                EdgeInsets.only(left: 60.0, top: 40.0, right: 60.0),
            height: 85.0,
            child: FlatButton(
              textColor: Theme.of(context).primaryColor,
              color: Colors.white,
              shape: RoundedRectangleBorder(
                borderRadius: new BorderRadius.circular(4.0),
                side: BorderSide(
                    color: Theme.of(context).accentColor),
              ),
              padding: const EdgeInsets.all(8.0),
              child: Text(
                "Submit",
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: 20.0,
                ),
              ),
              onPressed: () {
                if (_fbKey.currentState.validate()) {
                  addData();
                } else {
```

```
                        setState(() {
                          _autoValidate = true;
                        });
                      }
                    }),
                ),
                Container(
                  height: 20.0,
                ),
              ]),
            );
          }
        }),
      isLoading ? Center(child: CircularProgressIndicator()) : Container(),
    ]),
  );
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    key: _scaffoldKey,
    appBar: AppBar(
      title: Text('Add New Client'),
    ),
    body: makeForm(context),
  );
 }
}
```

2. API

```
1. @RestController
2. @RequestMapping("/client")
3. @CrossOrigin(origins = Constants.URL_TO_REQUEST)
4. public class ClientController
5. {
6. @ApiImplicitParams({
7. @ApiImplicitParam(name="Authorization",value="$
   {ivclient.request.authorization.description}",paramType="header")
8. })
9. @PostMapping(consumes = { MediaType.APPLICATION_JSON_VALUE,
   MediaType.APPLICATION_XML_VALUE },
10.  produces = { MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE })
11.  public ResponseEntity<Void> addClient(@Valid @RequestBody ClientInsert ci)
12.  {
13.  return cbl.addClient(ci);
14.  }}
15.
16.  @Service
17.  public class ClientDAOImpl implements ClientDAO
18.  {
19.  @Override
```

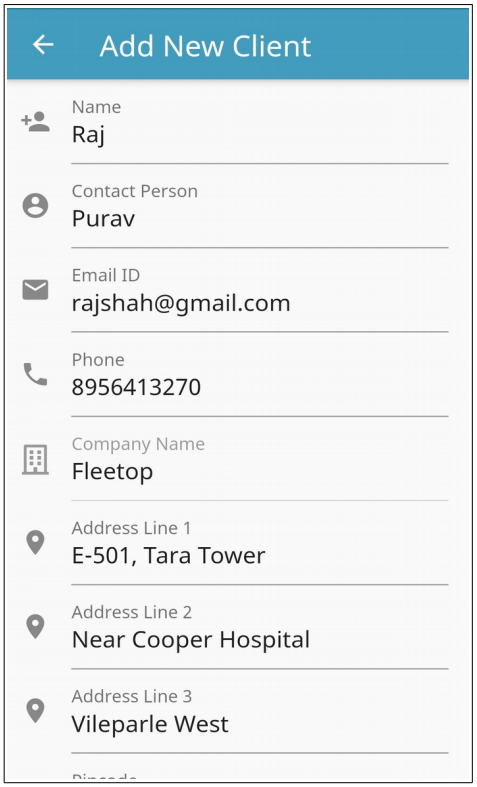
```
20.  public Boolean addClient(ClientInsert ci) throws SQLException, ClassNotFoundException
21.  {
22.  Connection c=ConnectionProvider.getConnection();
23.  CallableStatement                                                    stmt=c.prepareCall("SELECT
     client.\"fn_insertClient\"(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);");
24.  stmt.setString(1, ci.getContactName());
25.  stmt.setString(2, ci.getContactPerson());
26.  stmt.setString(3, ci.getEmailId());
27.  stmt.setString(4, ci.getContactNumber());
28.  stmt.setLong(5, ci.getCompanyId());
29.  stmt.setLong(6, ci.getCountryId());
30.  stmt.setLong(7, ci.getStateId());
31.  stmt.setLong(8, ci.getCityId());
32.  stmt.setLong(9, ci.getAreaId());
33.  stmt.setString(10, ci.getAddressLine1());
34.  stmt.setString(11, ci.getAddressLine2());
35.  stmt.setString(12, ci.getAddressLine3());
36.  stmt.setString(13, ci.getPincode());
37.  stmt.setString(14, ci.getLatitude());
38.  stmt.setString(15, ci.getLongitude());
39.  stmt.setTimestamp(16, ci.getCreatedOn());
40.  stmt.setLong(17, ci.getCreatedBy());
41.  ResultSet rs=stmt.executeQuery();
42.  c.commit();
43.  Boolean rsMain=false;
44.  if(rs.next())
45.  {
46.  rsMain=rs.getBoolean("fn_insertClient");
47.  }
48.  rs.close();
49.  stmt.close();
50.  c.close();
51.  return rsMain;
52.  }}
```

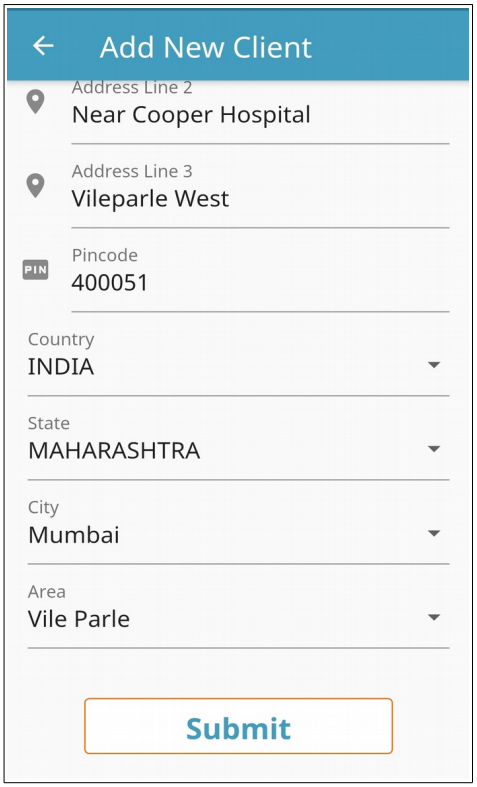
3. Database Scripts

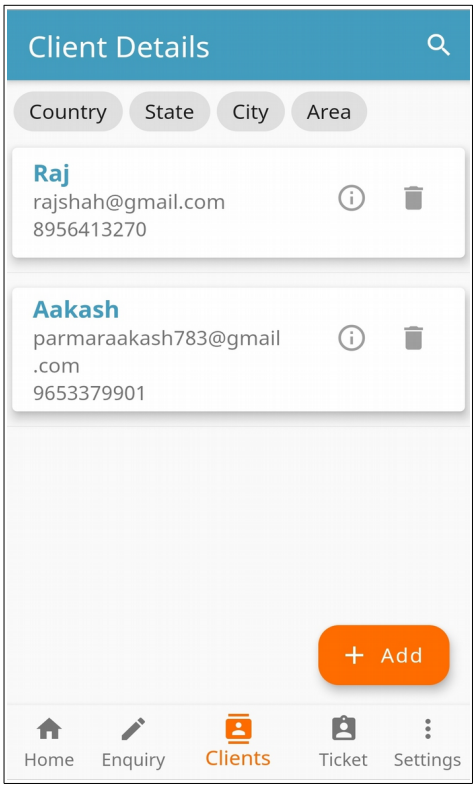
```
1.  CREATE FUNCTION client."fn_insertClient"(
2.  "ContactNameIn" text, "ContactPersonIn" text, "EmailIdIn" text, "ContactNumberIn" text, "CompanyIdIn" bigint,
    "CountryIn" bigint, "StateIn" bigint, "CityIn" bigint, "AreaIn" bigint, "AddressLine1In" text, "AddressLine2In"
    text, "AddressLine3In" text, "pincodeIn" text, "latitudeIn" text, "longitudeIn" text, "CreatedOnIn" timestamp
    without time zone, "CreatedByIn" bigint
3.  ) RETURNS boolean
4.     LANGUAGE plpgsql
5.     AS $$
6.  BEGIN
7.  WITH "ClientLocationGetId" AS
8.  (
9.     INSERT INTO "client"."ClientLocation"
10.                 (
11.                                 "Country",
12.                                 "State",
13.                                 "City",
```

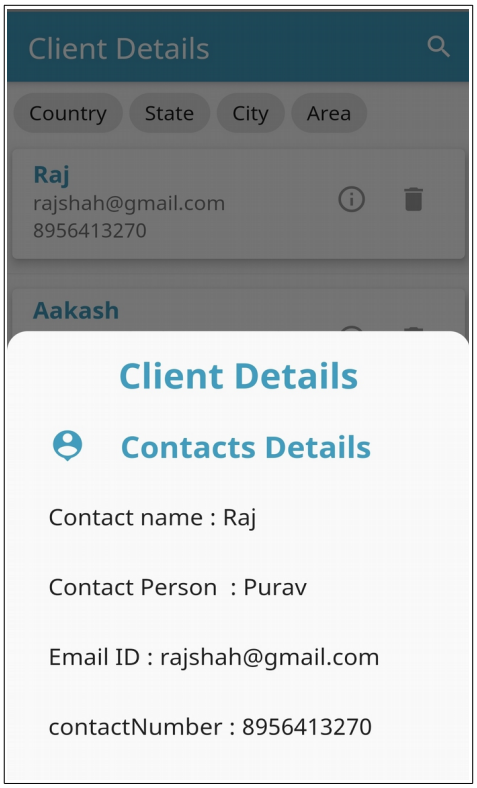
```
                    "Area",
                    "AddressLine1",
                    "AddressLine2",
                    "AddressLine3",
                    "pincode",
                    "latitude",
                    "longitude",
                    "CreatedOn",
                    "LastEditOn",
                    "CreatedBy",
                    "LastEditBy"
            )
VALUES
            (
                            "CountryIn",
                            "StateIn",
                            "CityIn",
                            "AreaIn",
                            "AddressLine1In",
                            "AddressLine2In",
                            "AddressLine3In",
                            "pincodeIn",
                            "latitudeIn",
                            "longitudeIn",
                            "CreatedOnIn",
                            "CreatedOnIn",
                            "CreatedByIn",
                            "CreatedByIn"
            )RETURNING "ClientLocationId"
)           INSERT INTO "client"."Client"
            (
                    "ContactName",
                    "ContactPerson",
                    "EmailId",
                    "ContactNumber",
                    "CompanyId",
                    "ClientLocationId","CreatedOn",
                    "LastEditOn","CreatedBy",
                    "LastEditBy"
            )
VALUES (
                            "ContactNameIn","ContactPersonIn",
                            "EmailIdIn","ContactNumberIn",
                            "CompanyIdIn",
                            (SELECT "ClientLocationId" FROM "ClientLocationGetId"),
                            "CreatedOnIn","CreatedOnIn",
                            "CreatedByIn", "CreatedByIn"
            );
RETURN true;

EXCEPTION WHEN OTHERS THEN
RETURN false;
ROLLBACK;
END;
$$;
```

Screenshots:





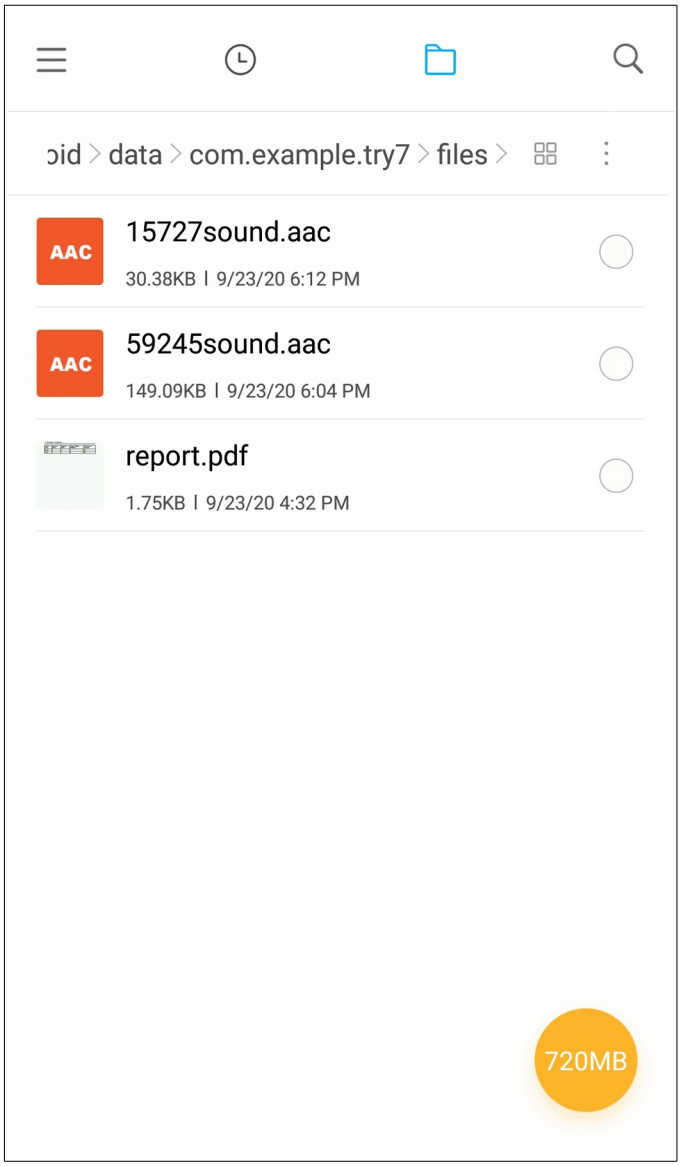




10.1.4 Recording calls:

Our application records the call conversation of the executive and client to understand the standard of marketing. When the executive receives the call our app which is running in the background compares the contact number against the database of the client's contact number if the match is found then the call is recorded otherwise it gets discarded. This process is done in the background and generates an .aac file.

Screenshots:



10.1.5 Notifications:

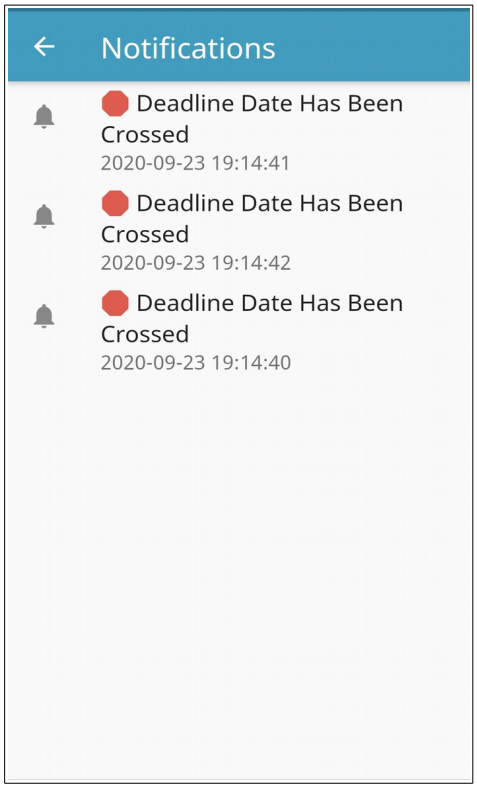
Executives receive a notification on various occasions.

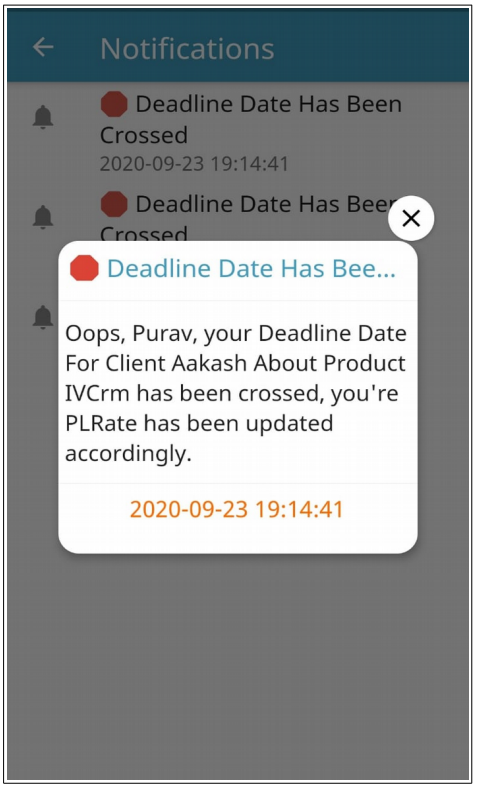
1. When a ticket is assigned
2. When ticket crosses its deadline
3. When PL rate changes

The notification is provided in two forms:

1. Floating notifications
2. In app notifications

Screenshots:





10.1.6 Extensive search and filter support:

Quick search and multiple filter support provide executive ease of access to information. The application has a real-time searching feature that provides suggestions as to the user type. Filter support provides quick access to complex information and fulfils the needs.

Code Snippet:

```
class EnquiryState extends State<Enquiry> with TickerProviderStateMixin{
  Widget build(BuildContext context) {
    filterList = Consumer<ThemeNotifier>(builder: (context, data, child) {
      return Container(
        height: 50.0,
        width: double.infinity,
        child: ListView(scrollDirection: Axis.horizontal, children: <Widget>[
          Padding(
            padding: const EdgeInsets.only(right: 5.0, left: 5.0),
              child: FilterChip(
                label: Text(filterChipCountry),
                onSelected: (b) {
                  return showDialog(
                      context: context,
                      builder: (c) {
                        return AlertDialog(
                          title: Text('Select Country',
                              style: TextStyle(
                                  color: Theme.of(context).primaryColor)),
                          content: Container(
                            constraints: BoxConstraints(
                              maxHeight: 100.0,
                            ),
                            child: SingleChildScrollView(
                              child:
                                  CustomChip(countries, 'country', 'enquiryPage',
                                      onSelectionChanged:
                                          (selectedValue, selectIndex) {
                                        if (selectIndex) {
                                          filterChipCountry = selectedValue;
                                        } else {
                                          filterChipCountry = 'Country';
                                        }
                                      }),
                            ),
                          ),
                          actions: <Widget>[
                            FlatButton(
                              child: Text('OK',
                                  style: TextStyle(
                                      color: Theme.of(context).primaryColor)),
```

```
                    onPressed: () {
                      Navigator.of(c).pop();
                      setState(() {
                        if (filterChipCountry == 'Country') {
                          getNewData = true;
                          filter = '/${CurrentUser.id}';
                          filterChipCountrySelected = false;
                        } else {
                          getNewData = true;
                          filter =
                                  '/country/${countriesMapping[filterChipCountry]}?companyExecutiveId=${CurrentUser.id}';
                          filterChipCountrySelected = true;
                        }
                      });
                    },
                  ),
                ],
              );
            });
        },
        selected: filterChipCountrySelected,
        selectedColor: data.getTheme().brightness == Brightness.dark
            ? Color(0xff4FAACA)
            : Color(0xff79cef1),
        backgroundColor: data.getTheme().brightness == Brightness.dark
            ? Colors.grey
            : Colors.grey[300],
      ),
    ),
    Padding(
      padding: const EdgeInsets.only(right: 5.0),
      child: FilterChip(
        label: Text(filterChipState),
        onSelected: (b) {
          return showDialog(
              context: context,
              builder: (c) {
                return AlertDialog(
                  title: Text('Select State',
                      style: TextStyle(
                          color: Theme.of(context).primaryColor)),
                  content: Container(
                    constraints: BoxConstraints(
                      maxHeight: 100.0,
                    ),
                    child: SingleChildScrollView(
                      child: CustomChip(states, 'state', 'enquiryPage',
                          onSelectionChanged:
                              (selectedValue, selectIndex) {
                        if (selectIndex) {
                          filterChipState = selectedValue;
                        } else {
                          filterChipState = 'State';
                        }
                      }),
```

```
97.                 ),
98.               ),
99.               actions: <Widget>[
100.                FlatButton(
101.                  child: Text('OK',
102.                      style: TextStyle(
103.                          color: Theme.of(context).primaryColor)),
104.                  onPressed: () {
105.                    Navigator.of(c).pop();
106.                    setState(() {
107.                      if (filterChipState == 'State') {
108.                        getNewData = true;
109.                        filter = '/${CurrentUser.id}';
110.                        filterChipStateSelected = false;
111.                      } else {
112.                        getNewData = true;
113.                        filter =
114.                            '/state/${statesMapping[filterChipState]}?companyExecutiveId=${CurrentUser.id}';
115.                        filterChipStateSelected = true;
116.                      }
117.                    });
118.                  },
119.                ),
120.              ],
121.            );
122.          });
123.      },
124.      selected: filterChipStateSelected,
125.      selectedColor: data.getTheme().brightness == Brightness.dark
126.          ? Color(0xff4FAACA)
127.          : Color(0xff79cef1),
128.      backgroundColor: data.getTheme().brightness == Brightness.dark
129.          ? Colors.grey
130.          : Colors.grey[300],
131.    ),
132.  ),
133.  Padding(
134.    padding: const EdgeInsets.only(right: 5.0),
135.    child: FilterChip(
136.      label: Text(filterChipCity),
137.      onSelected: (b) {
138.        return showDialog(
139.            context: context,
140.            builder: (c) {
141.              return AlertDialog(
142.                title: Text('Select City',
143.                    style: TextStyle(
144.                        color: Theme.of(context).primaryColor)),
145.                content: Container(
146.                  constraints: BoxConstraints(
147.                    maxHeight: 100.0,
148.                  ),
149.                  child: SingleChildScrollView(
150.                    child: CustomChip(cities, 'city', 'enquiryPage',
151.                        onSelectionChanged:
152.                            (selectedValue, selectIndex) {
```

```
153.                     if (selectIndex) {
154.                       filterChipCity = selectedValue;
155.                     } else {
156.                       filterChipCity = 'City';
157.                     }
158.                   }),
159.                 ),
160.               ),
161.               actions: <Widget>[
162.                 FlatButton(
163.                   child: Text('OK',
164.                     style: TextStyle(
165.                       color: Theme.of(context).primaryColor)),
166.                   onPressed: () {
167.                     Navigator.of(c).pop();
168.                     setState(() {
169.                       if (filterChipCity == 'City') {
170.                         getNewData = true;
171.                         filter = '/${CurrentUser.id}';
172.                         filterChipCitySelected = false;
173.                       } else {
174.                         getNewData = true;
175.                         filter =
176.                           '/city/${citiesMapping[filterChipCity]}?companyExecutiveId=${CurrentUser.id}';
177.                         filterChipCitySelected = true;
178.                       }
179.                     });
180.                   },
181.                 ),
182.               ],
183.             );
184.           });
185.       },
186.       selected: filterChipCitySelected,
187.       selectedColor: data.getTheme().brightness == Brightness.dark
188.           ? Color(0xff4FAACA)
189.           : Color(0xff79cef1),
190.       backgroundColor: data.getTheme().brightness == Brightness.dark
191.           ? Colors.grey
192.           : Colors.grey[300],
193.     ),
194.   ),
195.   Padding(
196.     padding: const EdgeInsets.only(right: 5.0),
197.     child: FilterChip(
198.       label: Text(filterChipArea),
199.       onSelected: (b) {
200.         return showDialog(
201.           context: context,
202.           builder: (c) {
203.             return AlertDialog(
204.               title: Text('Select Area',
205.                 style: TextStyle(
206.                   color: Theme.of(context).primaryColor)),
207.               content: Container(
208.                 constraints: BoxConstraints(
```

```
                  maxHeight: 100.0,
                ),
                child: SingleChildScrollView(
                  child: CustomChip(areas, 'area', 'enquiryPage',
                      onSelectionChanged:
                          (selectedValue, selectIndex) {
                        if (selectIndex) {
                          filterChipArea = selectedValue;
                        } else {
                          filterChipArea = 'Area';
                        }
                      }),
                ),
              ),
              actions: <Widget>[
                FlatButton(
                  child: Text('OK',
                      style: TextStyle(
                          color: Theme.of(context).primaryColor)),
                  onPressed: () {
                    Navigator.of(c).pop();
                    setState(() {
                      if (filterChipArea == 'Area') {
                        getNewData = true;
                        filter = '/${CurrentUser.id}';
                        filterChipAreaSelected = false;
                      } else {
                        getNewData = true;
                        filter =
                            '/area/${areasMapping[filterChipArea]}?companyExecutiveId=${CurrentUser.id}';
                        filterChipAreaSelected = true;
                      }
                    });
                  },
                ),
              ],
            );
          });
    },
    selected: filterChipAreaSelected,
    selectedColor: data.getTheme().brightness == Brightness.dark
        ? Color(0xff4FAACA)
        : Color(0xff79cef1),
    backgroundColor: data.getTheme().brightness == Brightness.dark
        ? Colors.grey
        : Colors.grey[300],
  ),
),
Padding(
  padding: const EdgeInsets.only(right: 5.0),
  child: FilterChip(
    label: Text(filterChipClient),
    onSelected: (b) {
      return showDialog(
          context: context,
          builder: (c) {
```

```
265.                 return AlertDialog(
266.                  title: Text('Select Client',
267.                    style: TextStyle(
268.                      color: Theme.of(context).primaryColor)),
269.                  content: Container(
270.                   constraints: BoxConstraints(
271.                    maxHeight: 100.0,
272.                   ),
273.                   child: SingleChildScrollView(
274.                    child: CustomChip(clients, 'client', 'enquiryPage',
275.                      onSelectionChanged:
276.                        (selectedValue, selectIndex) {
277.                      if (selectIndex) {
278.                       filterChipClient = selectedValue;
279.                      } else {
280.                       filterChipClient = 'Client';
281.                      }
282.                     }),
283.                   ),
284.                  ),
285.                  actions: <Widget>[
286.                   FlatButton(
287.                    child: Text('OK',
288.                      style: TextStyle(
289.                        color: Theme.of(context).primaryColor)),
290.                    onPressed: () {
291.                     Navigator.of(c).pop();
292.                     setState(() {
293.                      if (filterChipClient == 'Client') {
294.                       getNewData = true;
295.                       filter = '/${CurrentUser.id}';
296.                       filterChipClientSelected = false;
297.                      } else {
298.                       getNewData = true;
299.                       filter =
300.                         '/client/${clientsMapping[filterChipClient]}?companyExecutiveId=${CurrentUser.id}';
301.                       filterChipClientSelected = true;
302.                      }
303.                     });
304.                    },
305.                   ),
306.                  ],
307.                 );
308.                });
309.           },
310.           selected: filterChipClientSelected,
311.           selectedColor: data.getTheme().brightness == Brightness.dark
312.               ? Color(0xff4FAACA)
313.               : Color(0xff79cef1),
314.           backgroundColor: data.getTheme().brightness == Brightness.dark
315.               ? Colors.grey
316.               : Colors.grey[300],
317.         ),
318.        ),
319.        Padding(
320.         padding: const EdgeInsets.only(right: 5.0),
```

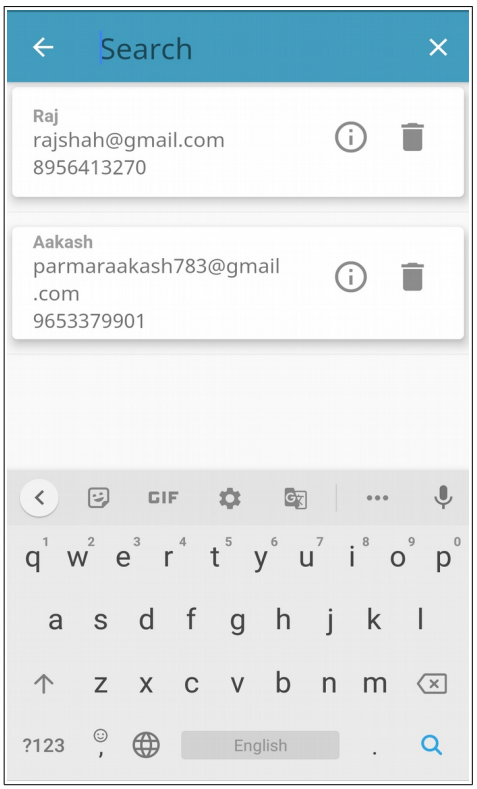
```
321.          child: FilterChip(
322.            label: Text(filterChipEnquiryType),
323.            onSelected: (b) {
324.              return showDialog(
325.                context: context,
326.                builder: (c) {
327.                  return AlertDialog(
328.                    title: Text('Select Enquiry Type',
329.                      style: TextStyle(
330.                        color: Theme.of(context).primaryColor)),
331.                    content: Container(
332.                      constraints: BoxConstraints(
333.                        maxHeight: 100.0,
334.                      ),
335.                      child: SingleChildScrollView(
336.                        child: CustomChip(
337.                          enquiryTypes, 'enquiryType', 'enquiryPage',
338.                          onSelectionChanged:
339.                              (selectedValue, selectIndex) {
340.                            if (selectIndex) {
341.                              filterChipEnquiryType = selectedValue;
342.                            } else {
343.                              filterChipEnquiryType = 'Enquiry Type';
344.                            }
345.                          }),
346.                      ),
347.                    ),
348.                    actions: <Widget>[
349.                      FlatButton(
350.                        child: Text('OK',
351.                          style: TextStyle(
352.                            color: Theme.of(context).primaryColor)),
353.                        onPressed: () {
354.                          Navigator.of(c).pop();
355.                          setState(() {
356.                            if (filterChipEnquiryType == 'Enquiry Type') {
357.                              getNewData = true;
358.                              filter = '/${CurrentUser.id}';
359.                              filterChipEnquiryTypeSelected = false;
360.                            } else {
361.                              getNewData = true;
362.                              filter =
363.                                                  '/enquiryType/${enquiryTypesMapping[filterChipEnquiryType]}?
    companyExecutiveId=${CurrentUser.id}';
364.                              filterChipEnquiryTypeSelected = true;
365.                            }
366.                          });
367.                        },
368.                      ),
369.                    ],
370.                  );
371.                });
372.          },
373.          selected: filterChipEnquiryTypeSelected,
374.          selectedColor: data.getTheme().brightness == Brightness.dark
375.              ? Color(0xff4FAACA)
```

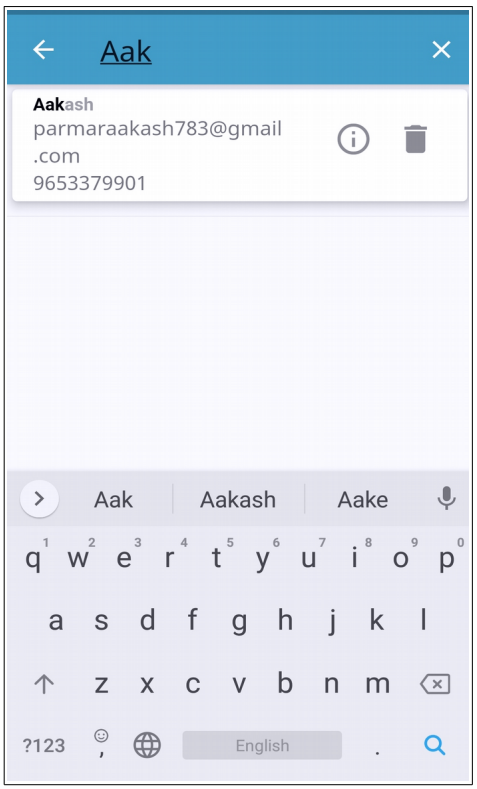
```
376.              : Color(0xff79cef1),
377.          backgroundColor: data.getTheme().brightness == Brightness.dark
378.              ? Colors.grey
379.              : Colors.grey[300],
380.        ),
381.      ),
382.      Padding(
383.        padding: const EdgeInsets.only(right: 5.0),
384.        child: FilterChip(
385.          label: Text(filterChipProduct),
386.          onSelected: (b) {
387.            return showDialog(
388.                context: context,
389.                builder: (c) {
390.                  return AlertDialog(
391.                    title: Text('Select Product',
392.                        style: TextStyle(
393.                            color: Theme.of(context).primaryColor)),
394.                    content: Container(
395.                      constraints: BoxConstraints(
396.                        maxHeight: 100.0,
397.                      ),
398.                      child: SingleChildScrollView(
399.                        child:
400.                            CustomChip(products, 'product', 'enquiryPage',
401.                                onSelectionChanged:
402.                                    (selectedValue, selectIndex) {
403.                              if (selectIndex) {
404.                                filterChipProduct = selectedValue;
405.                              } else {
406.                                filterChipProduct = 'Product';
407.                              }
408.                            }),
409.                      ),
410.                    ),
411.                    actions: <Widget>[
412.                      FlatButton(
413.                        child: Text('OK',
414.                            style: TextStyle(
415.                                color: Theme.of(context).primaryColor)),
416.                        onPressed: () {
417.                          Navigator.of(c).pop();
418.                          setState(() {
419.                            if (filterChipProduct == 'Product') {
420.                              getNewData = true;
421.                              filter = '/${CurrentUser.id}';
422.                              filterChipProductSelected = false;
423.                            } else {
424.                              getNewData = true;
425.                              filter =
426.                                          '/product/${productsMapping[filterChipProduct]}?companyExecutiveId=$
    {CurrentUser.id}';
427.                              filterChipProductSelected = true;
428.                            }
429.                          });
430.                        },
```

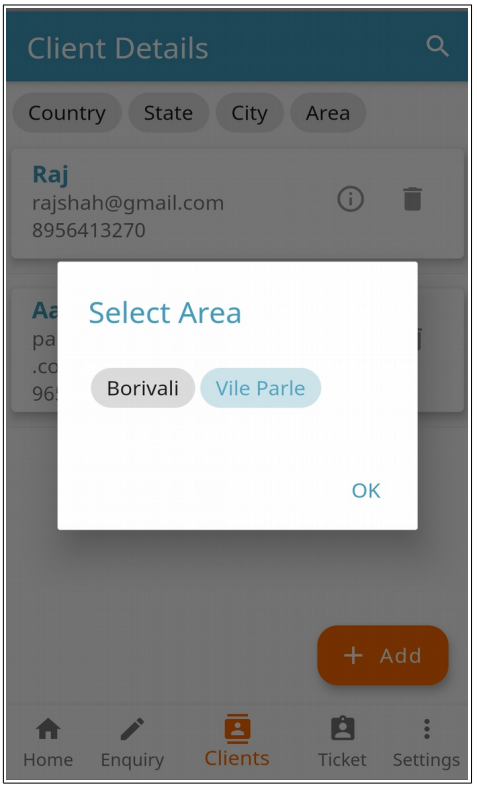
```
                    ),
                  ],
                );
              });
          },
          selected: filterChipProductSelected,
          selectedColor: data.getTheme().brightness == Brightness.dark
              ? Color(0xff4FAACA)
              : Color(0xff79cef1),
          backgroundColor: data.getTheme().brightness == Brightness.dark
              ? Colors.grey
              : Colors.grey[300],
        ),
      ),
    ]),
  );
});

return Scaffold(
  key: _scaffoldKey,
  //backgroundColor : data.getTheme().brightness == Brightness.dark ? Colors.grey : Colors.grey[300],
  appBar: AppBar(
    automaticallyImplyLeading: false,
    elevation: 1.0,
    title: Text(
      'Enquiry Details',
      style: TextStyle(fontFamily: 'NotoSans'),
    ),
    actions: <Widget>[
      IconButton(
        icon: Icon(Icons.search),
        onPressed: () {
          showSearch(context: context, delegate: DataSearch())
              .then((value) {
            if (value == 'delete') {
              showMessage('Deleted Successfully.!');
            }
            setState(() {
              getNewData = true;
            });
          });
        },
      ),
    ]),
  body: refresh(context),
  floatingActionButton: FloatingActionButton.extended(
    label: Text('Add'),
    icon: Icon(Icons.add),
    shape: RoundedRectangleBorder(
      borderRadius: BorderRadius.all(Radius.circular(16.0)),
    ),
    onPressed: () async {
      setState(() {
        isLoading = true;
      });
      var r5 = await ApiCall.getDataFromApi(
```

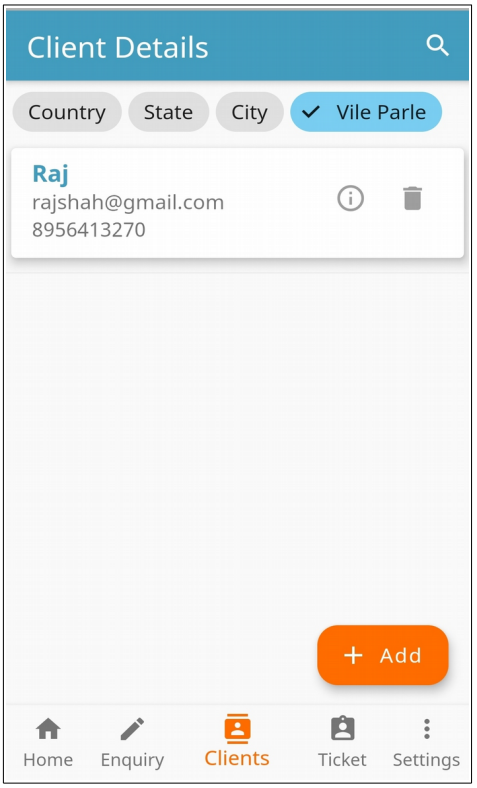
```
487.           Uri.GET_ENQUIRY_TYPE + "/${CurrentUser.companyId}");
488.
489.       if (r5 == 'nothing') {
490.         setState(() {
491.           isLoading = false;
492.         });
493.         missingDialog('ENQUIRY TYPE');
494.         return;
495.       }
496.       var r6 = await ApiCall.getDataFromApi(
497.           Uri.GET_CLIENT + "/company/${CurrentUser.companyId}");
498.       if (r6 == 'nothing') {
499.         setState(() {
500.           isLoading = false;
501.         });
502.         missingDialog('CLIENT');
503.         return;
504.       }
505.       var r7 = await ApiCall.getDataFromApi(
506.           Uri.GET_STATUS + "/${CurrentUser.companyId}");
507.       if (r7 == 'nothing') {
508.         setState(() {
509.           isLoading = false;
510.         });
511.         missingDialog('STATUS');
512.         return;
513.       }
514.       var r8 = await ApiCall.getDataFromApi(
515.           Uri.GET_PRODUCT + "/company/${CurrentUser.companyId}");
516.       if (r8 == 'nothing') {
517.         setState(() {
518.           isLoading = false;
519.         });
520.         missingDialog('PRODUCT');
521.         return;
522.       }
523.
524.       if (r5 != 'nothing' &&
525.           r6 != 'nothing' &&
526.           r7 != 'nothing' &&
527.           r8 != 'nothing') {
528.         setState(() {
529.           isLoading = false;
530.         });
531.         Navigator.pushNamed(context, '/add_enquiry').then((value) {
532.           if (value != null) {
533.             showMessage('Record Successfully Added.!');
534.             setState(() {
535.               getNewData = true;
536.             });
537.           }
538.         });
539.       }
540.     },
541.   ),);
542. ))
```

Screenshots:









10.2 Owner

10.2.1 Login and Signup process:

The signup process consists of entering payment details that is debit or credit card detail, owner information, and email id. An email is sent that provides a secret key at the end of the signup process.

The owner provides the username, password, and secret key for the login process. An additional layer of security is added with the use of the secret key.

Code Snippet:

1. Frontend

```
class _LoginScreenState extends State<LoginScreen> {
@override
  Widget build(BuildContext context) {
    return Scaffold(
       body: IgnorePointer(
          ignoring: isLoading,
          child: Stack(
            children: <Widget>[
              Container(
                  decoration: BoxDecoration(
                    image: DecorationImage(
                       image: AssetImage('Assets/Images/CRM.jpg'),
                       fit: BoxFit.fitWidth,
                       alignment: Alignment.topCenter),
                  ),
                ),
                Container(
                  width: MediaQuery.of(context).size.width,
                  margin: EdgeInsets.only(top: 200),
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.circular(20),
                    color: Colors.white,
                  ),
                  child: Padding(
                    padding: EdgeInsets.all(23),
                    child: Form(
                       key: fbKey,
                       autovalidate: _autoValidate,
                       child: ListView(
```

```
30.                 children: <Widget>[
31.                   Padding(
32.                     padding: EdgeInsets.only(top: 0),
33.                     child: Container(
34.                       color: Color(0xfff5f5f5),
35.                       child: TextFormField(
36.                         controller: userController,
37.                         style: TextStyle(
38.                           color: Colors.black,
39.                         ),
40.                         decoration: InputDecoration(
41.                             border: OutlineInputBorder(),
42.                             labelText: 'Username',
43.                             hintText: "Enter Username",
44.                             prefixIcon: Icon(Icons.person_outline),
45.                             labelStyle: TextStyle(
46.                               fontSize: 20,
47.                             )),
48.                         validator: (v) => v.isEmpty
49.                             ? 'Username is required .'
50.                             : null,
51.                       ),
52.                     ),
53.                   ),
54.                   Container(
55.                     padding: EdgeInsets.only(top: 20),
56.                     color: Color(0xfff5f5f5),
57.                     child: TextFormField(
58.                       controller: passwordController,
59.                       obscureText: password,
60.                       style: TextStyle(
61.                         color: Colors.black,
62.                       ),
63.                       decoration: InputDecoration(
64.                         border: OutlineInputBorder(),
65.                         labelText: 'Password',
66.                         hintText: "Enter password",
67.                         prefixIcon: Icon(Icons.lock_outline),
68.                         suffixIcon: GestureDetector(
69.                           onTap: () {
70.                             setState(() {
71.                               password = !password;
72.                               if (password) {
73.                                 icon = Icon(Icons.visibility_off,
74.                                     color: Theme.of(context)
75.                                         .primaryColor);
76.                               } else {
77.                                 icon = Icon(Icons.visibility,
78.                                     color: Theme.of(context)
79.                                         .primaryColor);
80.                               }
81.                             });
82.                           },
83.                           child: icon,
84.                         ),
85.                         labelStyle: TextStyle(
```

```
                      fontSize: 20,
                    ),
                  ),
                  validator: (v) =>
                    v.isEmpty ? 'Password is required .' : null,
                ),
              ),
              AnimatedSwitcher(
                duration: Duration(milliseconds: 1000),
                child: myWidget,
              ),
              Padding(
                padding: EdgeInsets.only(top: 20),
                child: MaterialButton(
                  onPressed: () {
                    if (fbKey.currentState.validate()) {
                      checkCredentials();
                    } else {
                      setState(() {
                        _autoValidate = true;
                      });
                    }
                  },
                  child: Text(
                    'SIGN IN',
                    style: TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                  color: Theme.of(context).primaryColor,
                  elevation: 0,
                  minWidth: 400,
                  height: 50,
                  textColor: Colors.white,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),),),
              Container(
                padding: EdgeInsets.only(top: 20),
                height: 70,
                width: 400,
                child: OutlineButton(
                  borderSide: BorderSide(
                    color: Theme.of(context)
                        .primaryColor, //Color of the border
                    style:
                        BorderStyle.solid, //Style of the border
                    width: 0.8, //width of the border
                  ),
                  onPressed: () {
                    setState(() {
                      skvisible = !skvisible;
                      if (skvisible) {
                        myWidget = Padding(
                          padding: EdgeInsets.only(top: 20),
```

```
                        child: Container(
                          color: Color(0xfff5f5f5),
                          child: TextFormField(
                            controller: secretKeyController,
                            style: TextStyle(
                              color: Colors.black,
                            ),
                            decoration: InputDecoration(
                                border: OutlineInputBorder(),
                                labelText: 'Secret Key',
                                hintText:
                                    "Enter Secret key sent you to mail",
                                prefixIcon: Icon(Icons.vpn_key),
                                labelStyle: TextStyle(
                                  fontSize: 20,
                                )),
                            validator: (v) => v.isEmpty
                                ? 'Secret key is required .'
                                : null,
                          ),),);
                      } else {
                        myWidget = Container();
                      }
                    });},
                  child: Text(
                    skvisible
                        ? 'LOG IN AS USER'
                        : 'LOG IN AS OWNER',
                    style: TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ),),
                  color: Colors.white,
                  textColor: Theme.of(context).primaryColor,
                  shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(10),
                  ),),),
              Material(
                  child: Padding(
                    padding: const EdgeInsets.only(
                        top: 20.0, left: 70.0),
                    child: InkWell(
                      splashColor: Colors.grey,
                      onTap: () {
                        Navigator.push(
                          context,
                          MaterialPageRoute(
                              builder: (context) =>
                                  CreditCardPage()),
                        ).then((value) {
                          if (value) {
                            return showDialog(
                                barrierDismissible: false,
                                context: context,
                                builder: (c) {
                                  return AlertDialog(
```

```
                                    title: Text('Owner Added',
                                      style: TextStyle(
                                        color: Theme.of(
                                            context)
                                          .primaryColor)),
                                    content: Text(
                                        'Congratulations.You have successfully registered as a new Owner.\nNow Sign In as a owner(Secret key is been sent to you by Email).'),
                                    actions: [
                                     FlatButton(
                                      child: Text('OK'),
                                      onPressed: () {
                                       Navigator.of(c)
                                          .pop();
                                      },
                                     ),]);});
                                });
                              },
                              child: Text(
                               "BECOME A NEW OWNER",
                               textDirection: TextDirection.rtl,
                               style: TextStyle(
                                color: Theme.of(context).primaryColor,
                                fontWeight: FontWeight.bold,
                                fontSize: 17.0,
                               ),
                              ),
                             ))),
                        Material(
                          child: Padding(
                            padding: const EdgeInsets.only(
                              top: 20.0, left: 130.0),
                            child: InkWell(
                             splashColor: Colors.grey,
                             onTap: () {
                              print("hi");
                             },
                             child: Text(
                              "Forgot Password ?",
                              textDirection: TextDirection.rtl,
                              style: TextStyle(
                                color: Theme.of(context).primaryColor,
                                fontWeight: FontWeight.bold,
                               ),
                              ),
                             ))),
                      ],
                    ))),
              isLoading
                  ? Center(
                      child: CircularProgressIndicator(),
                    )
                  : Center(child: Text('')),
            ],
          )));
  }}
```

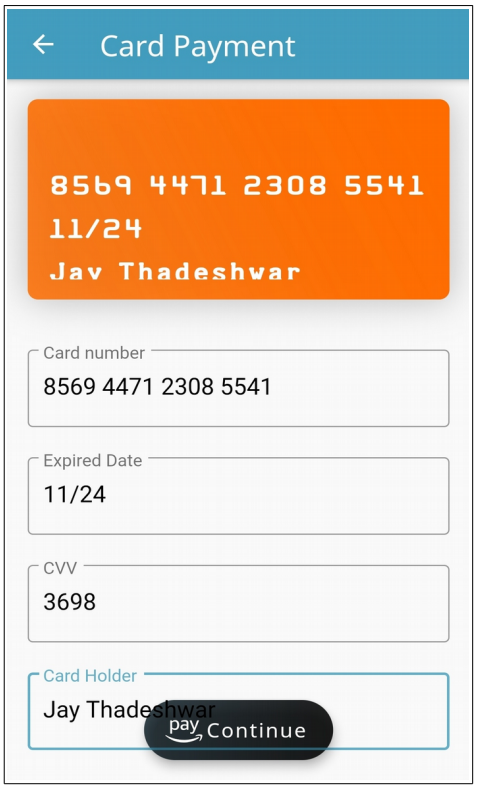
2. API

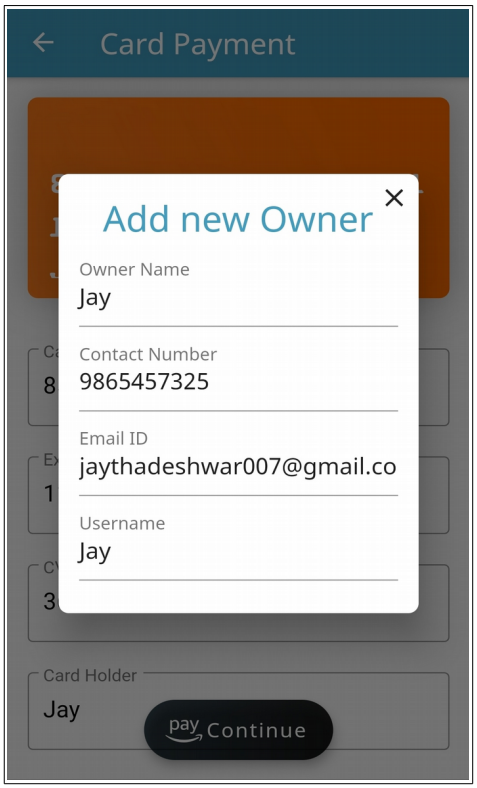
```
1. @RestController
2. @RequestMapping("/owner")
3. @CrossOrigin(origins = Constants.URL_TO_REQUEST)
4. public class OwnerController {
5.
6. @Autowired
7. OwnerBusinessLogic obl;
8.
9. @PostMapping(path="/login",produces = { MediaType.APPLICATION_JSON_VALUE,
   MediaType.APPLICATION_XML_VALUE })
10.  public ResponseEntity<OwnerLoginResponseModel> loginOwner(@RequestBody @Valid
   OwnerLoginCredentials olc)
11.  {
12.  return obl.loginOwner(olc);
13.  }}
14.
15.  @Service
16.  public class OwnerDAOImpl implements OwnerDAO{
17.
18.  @Override
19.  public Long loginOwner(String userName, String secretKey)throws SQLException, ClassNotFoundException
20.  {
21.  Long ownerId=null;
22.  Connection c=ConnectionProvider.getConnection();
23.  CallableStatement stmt=c.prepareCall("SELECT * FROM \"owner\".\"fn_loginOwnerContact\"(?,?);");
24.  stmt.setString(1, userName);
25.  stmt.setString(2, secretKey);
26.  ResultSet rs=stmt.executeQuery();
27.  if(rs.next()) {
28.  ownerId=rs.getLong("OwnerId");
29.  }
30.  rs.close();
31.  stmt.close();
32.  c.close();
33.  return ownerId;
34.  }}
```

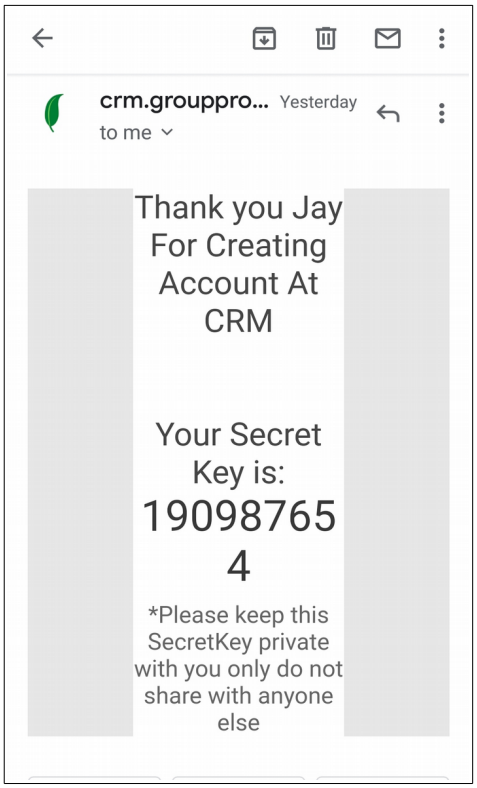
3. Database Scripts

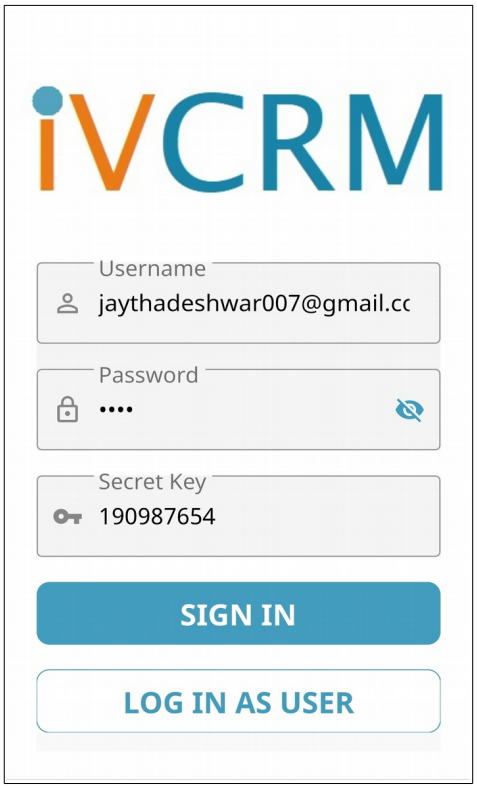
```
1. CREATE FUNCTION owner."fn_loginOwnerContact"("UserNameIn" text, "SecretKeyIn" text) RETURNS
   TABLE("OwnerId" bigint)
2.    LANGUAGE sql
3.    AS $$
4. SELECT id FROM "owner"."OwnerContact"
5. WHERE "owner"."OwnerContact"."ownerEmail"="UserNameIn"
6. AND "owner"."OwnerContact"."SecretKey"="SecretKeyIn"
7. AND "owner"."OwnerContact"."MarkForDelete"=false
8. $$;
```

Screenshots:









10.2.2 Reports for performance tracking:

The application provides the feature of management reporting over several criteria. Reports enable the management to evaluate the performance of the organization and support them in a quick decision-making process.

The application provides different types of reports which are as follows:

1. Status Report
2. Priority Report
3. Time-based report
4. Executive report
5. Product report

Reports are generated in pdf format and can be shared through the app over various communication mediums. The owner can generate specific information by applying a location-based filter to the reports.

Code Snippet:

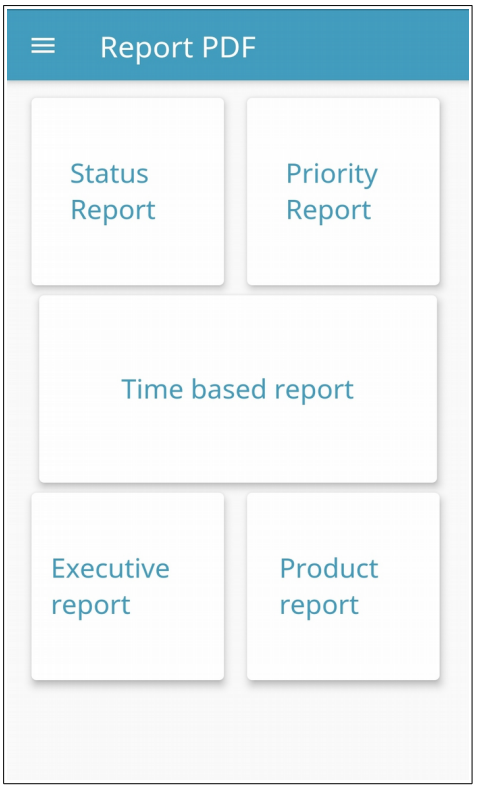
```
class ReportPdfState extends State<ReportPdf> {
void callerMeth(BuildContext context, int reportID, String location) {
   if (reportID == 1) {
    getStatusReport(1, location).then((res) {
      reportGen(res, "Status Report", context);
    });
   } else if (reportID == 2) {
    getPriorityReport(1, location).then((res) {
      reportGen(res, "Priority Report", context);
      });
    } else if (reportID == 3) {
      getTimeReport(1, location).then((res) {
        reportGen(res, "Time Report", context);
      });
    }
   }

   @override
   Widget build(BuildContext context) {
    return Scaffold(
      drawer: CustomDrawer(reportSelected: true),
      appBar: AppBar(
        title: Text('Report PDF'),
      ),
```

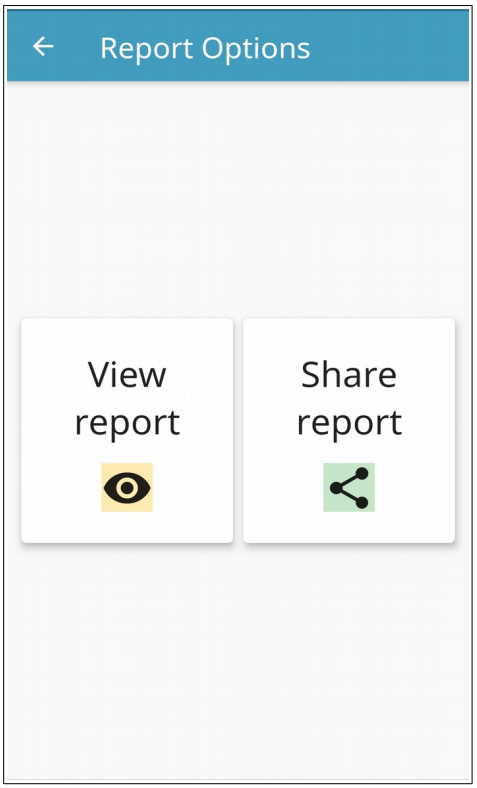
```
25.       body: Container(
26.         child: Column(
27.           children: <Widget>[
28.             Container(
29.               child: Row(
30.                 children: <Widget>[
31.                   Card(
32.                     child: InkWell(
33.                       splashColor: Colors.blue.withAlpha(30),
34.                       onTap: () => showFancyCustomDialog(context, 1),
35.                       child: Container(
36.                         width: 150,
37.                         height: 150,
38.                         child: Center(
39.                           child: Text(
40.                             'Status Report',
41.                             textScaleFactor: 1.5,
42.                             style: TextStyle(
43.                               color: Theme.of(context).primaryColor,
44.                               fontWeight: FontWeight.w500,
45.                             ),
46.                           ),
47.                         ),
48.                         padding: EdgeInsets.all(30.0),
49.                       ),
50.                     ),
51.                     elevation: 5.0,
52.                   ),
53.                   SizedBox(width: 10),
54.                   Card(
55.                     child: InkWell(
56.                       splashColor: Colors.blue.withAlpha(30),
57.                       onTap: () => showFancyCustomDialog(context, 2),
58.                       child: Container(
59.                         width: 150,
60.                         height: 150,
61.                         child: Center(
62.                           child: Text(
63.                             'Priority Report',
64.                             textScaleFactor: 1.5,
65.                             style: TextStyle(
66.                               color: Theme.of(context).primaryColor,
67.                               fontWeight: FontWeight.w500,
68.                             ),
69.                           ),
70.                         ),
71.                         padding: EdgeInsets.all(30.0),
72.                       ),
73.                     ),
74.                     elevation: 5.0,
75.                   )
76.                 ],
77.               ),
78.               padding: EdgeInsets.only(top: 10, left: 15)),
79.             Container(
80.               child: Card(
```

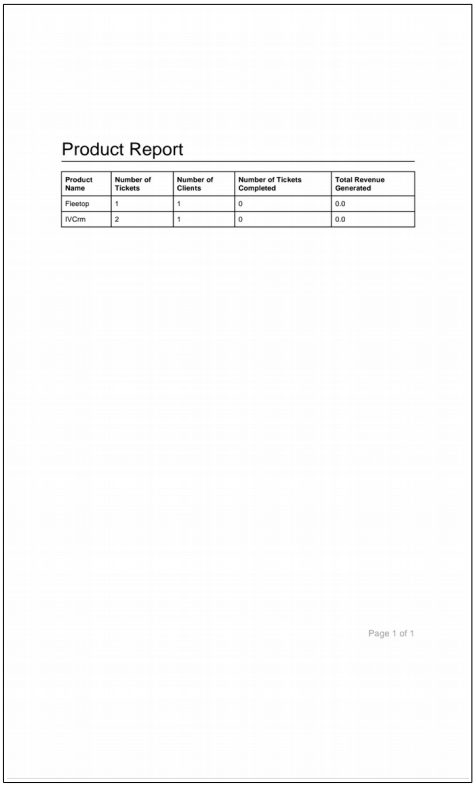
```
81.             child: InkWell(
82.               splashColor: Colors.blue.withAlpha(30),
83.               onTap: () => showFancyCustomDialog(context, 3),
84.               child: Container(
85.                 width: 310,
86.                 height: 150,
87.                 child: Center(
88.                   child: Text(
89.                     'Time based report',
90.                     textScaleFactor: 1.5,
91.                     style: TextStyle(
92.                       color: Theme.of(context).primaryColor,
93.                       fontWeight: FontWeight.w500,
94.                     ),
95.                   ),
96.                 ),
97.                 padding: EdgeInsets.all(15.0),
98.               ),
99.             ),
100.            elevation: 5.0,
101.          ),
102.        ),
103.        Container(
104.          child: Row(
105.            children: <Widget>[
106.              Card(
107.                child: InkWell(
108.                  splashColor: Colors.blue.withAlpha(30),
109.                  onTap: () {
110.                    getExecutiveReport(1).then((res) {
111.                      reportGen(res, "Executive Report", context);
112.                    });
113.                  },
114.                  child: Container(
115.                    width: 150,
116.                    height: 150,
117.                    child: Center(
118.                      child: Text(
119.                        'Executive report',
120.                        textScaleFactor: 1.5,
121.                        style: TextStyle(
122.                          color: Theme.of(context).primaryColor,
123.                          fontWeight: FontWeight.w500,
124.                        ),
125.                      ),
126.                    ),
127.                    padding: EdgeInsets.all(15.0),
128.                  ),
129.                ),
130.                elevation: 5.0,
131.              ),
132.              SizedBox(width: 10),
133.              Card(
134.                child: InkWell(
135.                  splashColor: Colors.blue.withAlpha(30),
136.                  onTap: () {
```

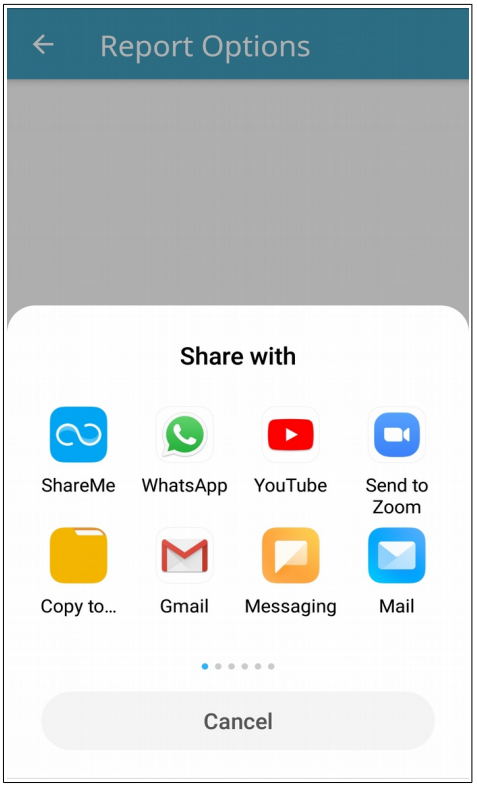
```
                    getProductReport().then((res) {
                      reportGen(res, "Product Report", context);
                    });
                  },
                  child: Container(
                    width: 150,
                    height: 150,
                    child: Center(
                      child: Text(
                        'Product report',
                        textScaleFactor: 1.5,
                        style: TextStyle(
                          color: Theme.of(context).primaryColor,
                          fontWeight: FontWeight.w500,
                        ),
                      ),
                    ),
                    padding: EdgeInsets.all(25.0),
                  ),
                ),
                elevation: 5.0,
              )
            ],
          ),
          padding: EdgeInsets.only(left: 15)),
        ],
      ),
    ),
  );
 }
}
```

Screenshots:









10.2.3 Geolocating on map:

The owner has instantaneous access to all the firm's resources. The application displays all the resources on the map with the help of markers. It provides the owner's current location, directions to any of the markers, and all the possible routes to reach the marker location through google map.

The application provides different types of geolocation, which are as follow:

1. Company executive
2. Company
3. Client
4. Enquiry
5. Company branch

Code Snippet:

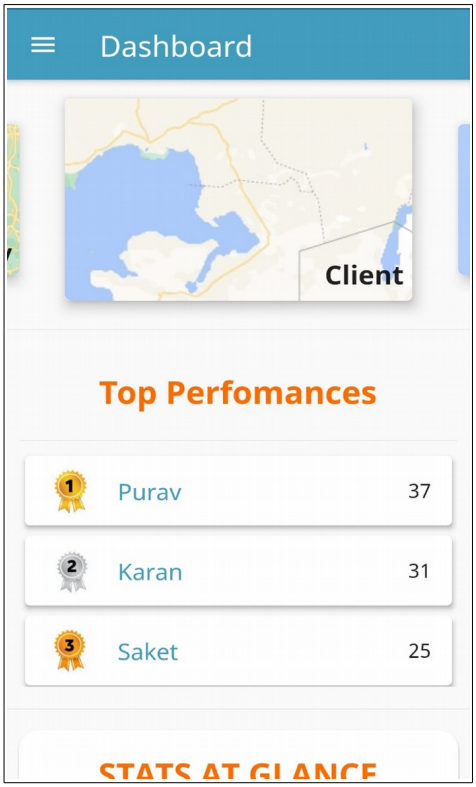
```
class _AdminHomeState extends State<AdminHome> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      drawer: CustomDrawer(dashboardSelected: true),
      appBar: AppBar(title: Text('Dashboard')),
      body: SingleChildScrollView(
        child: Container(
          height: MediaQuery.of(context).size.height * 1.4,
            child: FutureBuilder(
              future: getDashboardData(),
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  dashboardData = snapshot.data;
                  return Column(
                    children: <Widget>[
                      Expanded(
                        flex: 2,
                        child: CarouselSlider(
                          options: CarouselOptions(
                            autoPlay: true,
                            aspectRatio: 2.0,
                            enlargeCenterPage: true,
                          ),
                          items: imageAutoSliders(context),
                        ),
                      ),
                      Divider(),
                      Expanded(
```

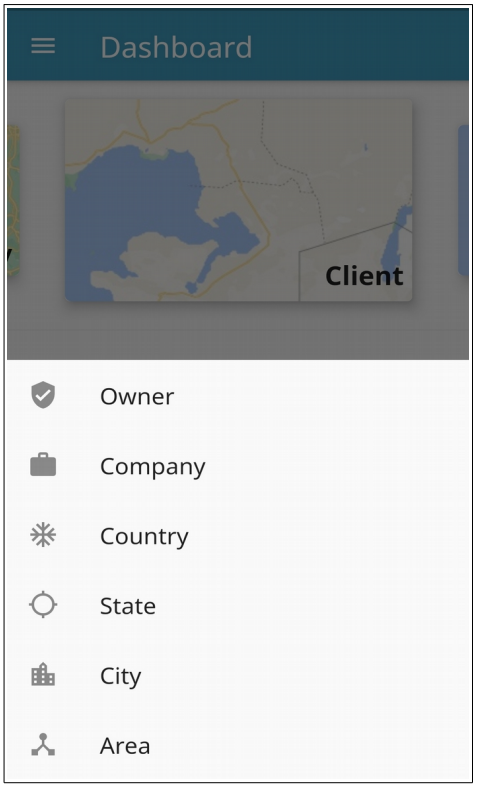
```
                flex: 3,
                child: Padding(
                  padding: const EdgeInsets.all(10.0),
                  child: Column(
                    children: [
                      Expanded(
                        flex: 1,
                        child: Center(
                          child: Text(
                            'Top Perfomances',
                            textScaleFactor: 1.8,
                            style: TextStyle(
                              color: Theme.of(context).accentColor,
                              fontWeight: FontWeight.bold,
                            ),
                          ),
                        ),
                      ),
                      Divider(),
                      Expanded(
                        flex: 3,
                        child: ListView.builder(
                          itemCount:
                              dashboardData.topExecutives.length,
                          itemBuilder: (con, index) {
                            var item =
                                dashboardData.topExecutives[index];
                            return Card(
                              elevation: 3,
                              child: ListTile(
                                leading: Container(
                                  width: 40.0,
                                  height: 40.0,
                                  decoration: new BoxDecoration(
                                    color: Colors.transparent,
                                    shape: BoxShape.circle,
                                    image: new DecorationImage(
                                      fit: BoxFit.fitHeight,
                                      image: index == 0
                                          ? AssetImage(
                                              'Assets/Images/first.jpg')
                                          : index == 1
                                              ? AssetImage(
                                                  'Assets/Images/second.jpg')
                                              : AssetImage(
                                                  'Assets/Images/third.jpg'))))),
                                title: Text(
                                  '${item.executiveName}',
                                  style: TextStyle(
                                    color: Theme.of(context)
                                        .primaryColor,
                                  ),
                                ),
                                trailing: Text('${item.plRate}'),
                              ),
                            );
```

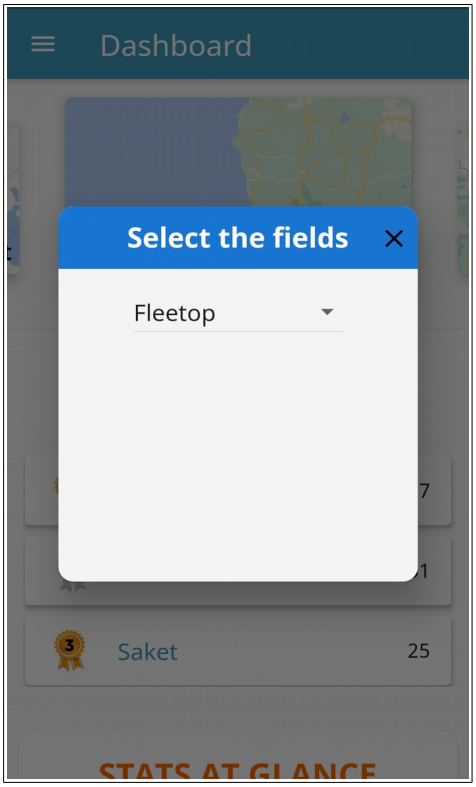
```
                        }),
                    ),
                  ],
                ),
              ),
            ),
            Divider(),
            Expanded(
              flex: 4,
              child: Padding(
                padding: const EdgeInsets.all(5.0),
                child: Card(
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(15.0)),
                  elevation: 1,
                  child: Column(
                    children: <Widget>[
                      Expanded(
                          flex: 2,
                          child: Center(
                              child: Text(
                            'STATS AT GLANCE',
                            textScaleFactor: 1.8,
                            style: TextStyle(
                              fontWeight: FontWeight.bold,
                              color: Theme.of(context).accentColor,
                            ),
                          ))),
                      Divider(),
                      Expanded(
                        flex: 4,
                        child: Row(
                          children: [
                            Expanded(
                              child: displayCard(
                                  Icons.shopping_cart,
                                  'Products',
                                  '${dashboardData.numberOfProducts}',
                                  0xffffc400),
                            ),
                            Expanded(
                              child: displayCard(
                                  Icons.people,
                                  'Clients',
                                  '${dashboardData.numberOfClients}',
                                  0xff2c75b0),
                            )
                          ],
                        ),
                      ),
                      Expanded(
                        flex: 4,
                        child: Row(
                          children: [
                            Expanded(
                                child: displayCard(
```

```
                                    Icons.pages,
                                    'Tickets',
                                    '${dashboardData.numberOftickets}',
                                    0xffeb6963)),
                                Expanded(
                                  child: displayCard(
                                    Icons.business_center,
                                    'Executives',
                                    '${dashboardData.numberOfExecutives}',
                                    0xff28b061),
                                )
                              ],
                            ),
                          ),
                          SizedBox(height: 5)
                        ],
                      ),
                    ),
                  ),
                ),
              ],
            );
          }
          if (snapshot.hasError) {
            return Center(
              child: SimpleDialog(
              title: Center(
                child: Text('Please check your internet connection',
                  textAlign: TextAlign.center)),
              children: [
                IconButton(
                  icon: Icon(Icons.refresh),
                  onPressed: () {
                    Navigator.of(context)
                      .pushReplacementNamed('/admin_home');
                  },
                )
              ],
            ));
          }
          return Align(
            alignment: Alignment(0.0, -0.45),
            child: CircularProgressIndicator(),
          );
        })),
    ),
  );
  }
}
```

Screenshots:



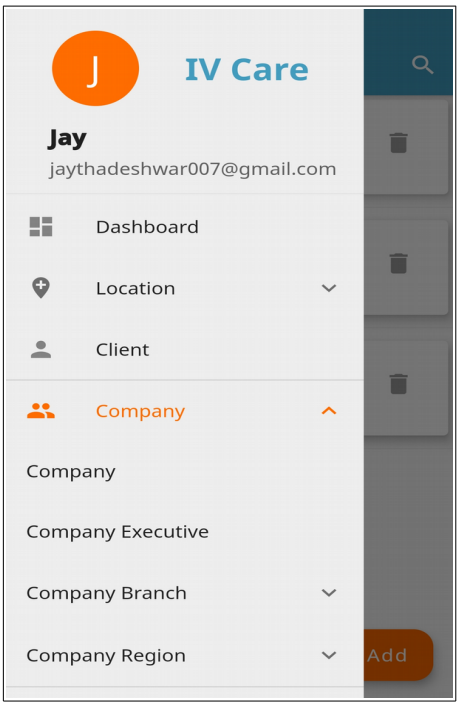


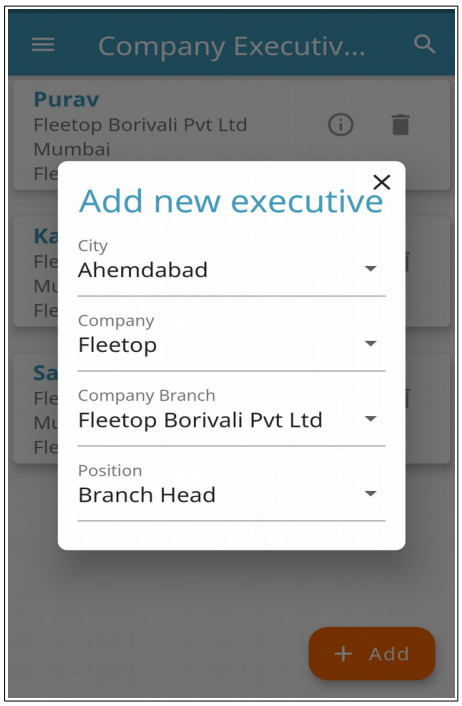


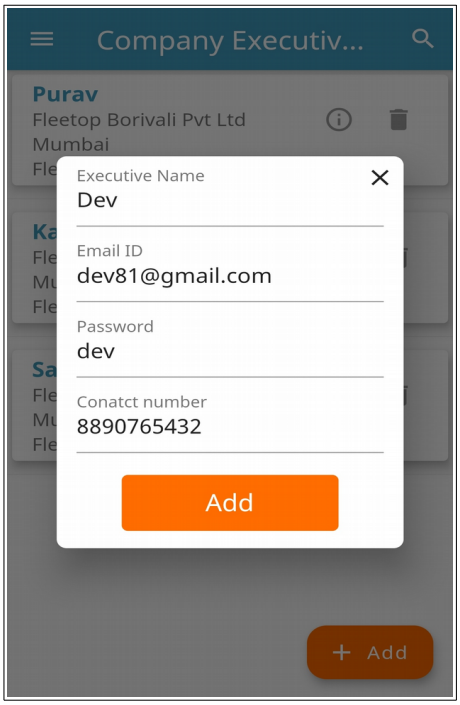
10.2.4 Adding new executives:

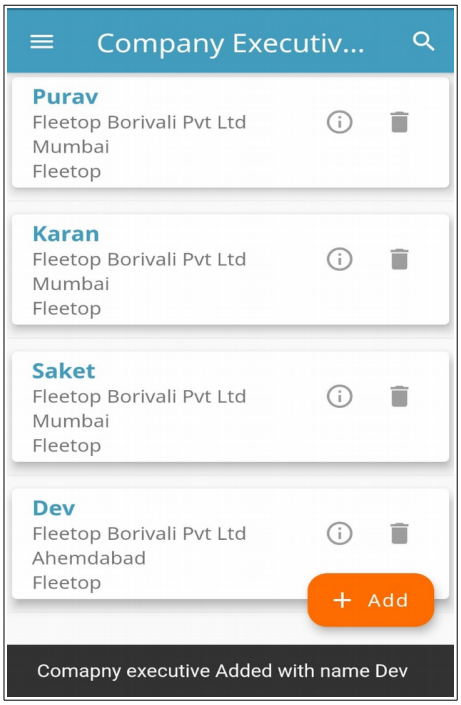
The owner of the company can only create a new executive. The credentials generated by the owner becomes the username and password for executive login.

Screenshots:









10.2.5 Adding new areas and cities:

The owner of the company can add new areas and cities where the company currently operates. This feature of application helps in understanding the marketspace covered and the scope of the organization. This information provides a base for the foundation of the CRM application.

Code Snippet:

1. Frontend

```
1.    class _AreaState extends State<Area>{
2.    @override
3.    Widget build(BuildContext context){
4.                return Scaffold(
5.                        key: _scaffoldKey,
6.                        appBar : AppBar(
7.                                title : Text('Area Details'),
8.                                actions : <Widget>[
9.                                        IconButton(
10.                                                icon : Icon(Icons.search),
11.                                                onPressed:(){
12.                                                  showSearch(context : context , delegate :
                          DataSearch()).then((value){
13.                                                    if(value=='delete'){
14.                                                            showMessage('Deleted Successfully.!!');
15.                                                            setState((){
16.                                                                 getNewData = true;
17.                                                               });
18.                                                    }
19.                                                    else if(value == 'update'){
20.                                                            showMessage('Record            Successfully
    updated.!!');
21.                                                            setState((){
22.                                                                 getNewData = true;
23.                                                               });
24.                                                    }
25.                                                  });
26.                                                  },
27.                                                )
28.                                        ],
29.                                ),
30.                                drawer : CustomDrawer(areaSelected : true),
31.                                body : refresh(context),
32.                                floatingActionButton : FloatingActionButton.extended(
33.                                        icon : Icon(Icons.add),
34.                                        label : Text('Add'),
35.                                        onPressed : (){
36.                                                addArea();
```

```
                                        },
                                        shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.all(Radius.circular(16.0))
            ),
                                )
                        );
            }
            }
```

2. API

```
1.  @RestController
2.  @RequestMapping("businessAreaForCompany")
3.  @CrossOrigin(origins = Constants.URL_TO_REQUEST)
4.  public class BusinessAreaForCompanyController {
5.  @Autowired
6.  BusinessAreaForCompanyBusinessLogic businessAreaForCompany_business_logic;
7.
8.  @ApiImplicitParams({
9.  @ApiImplicitParam(name="Authorization",value="$
    {ivclient.request.authorization.description}",paramType="header")
10.   })
11.   @PostMapping(consumes = { MediaType.APPLICATION_JSON_VALUE,
    MediaType.APPLICATION_XML_VALUE }, produces = {
12.   MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE })
13.   public ResponseEntity<Void> addBusinessAreaForCompanyName(
14.   @Valid @RequestBody BusinessAreaForCompanyInsert businessAreaForCompany_info) {
15.   return
    businessAreaForCompany_business_logic.addBusinessAreaForCompany(businessAreaForCompany_info);
16.   }
17.   }
18.
19.   @Service
20.   public class BusinessAreaForCompanyDaoImpl implements BusinessAreaForCompanyDao {
21.   @Override
22.   public Boolean addBusinessAreaForCompany(BusinessAreaForCompanyInsert
    businessAreaForCompany_info) {
23.   if (businessAreaForCompany_info == null)
24.   return null;
25.
26.   String sql = "SELECT location.\"fn_insertIntoBusinessAreaForCompany\"(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
27.   Connection c = null;
28.   CallableStatement st = null;
29.   try {
30.   c = ConnectionProvider.getConnection();
31.   st = c.prepareCall(sql);
32.   st.setLong(1, businessAreaForCompany_info.getCompanyID());
33.   st.setLong(2, businessAreaForCompany_info.getCountryID());
34.   st.setLong(3, businessAreaForCompany_info.getStateID());
35.   st.setLong(4, businessAreaForCompany_info.getCityID());
36.   st.setString(5, businessAreaForCompany_info.getAreaName());
37.   st.setString(6, businessAreaForCompany_info.getAreaCode());
38.   st.setString(7, businessAreaForCompany_info.getAreaDescription());
39.   st.setTimestamp(8, businessAreaForCompany_info.getCreatedOn());
```

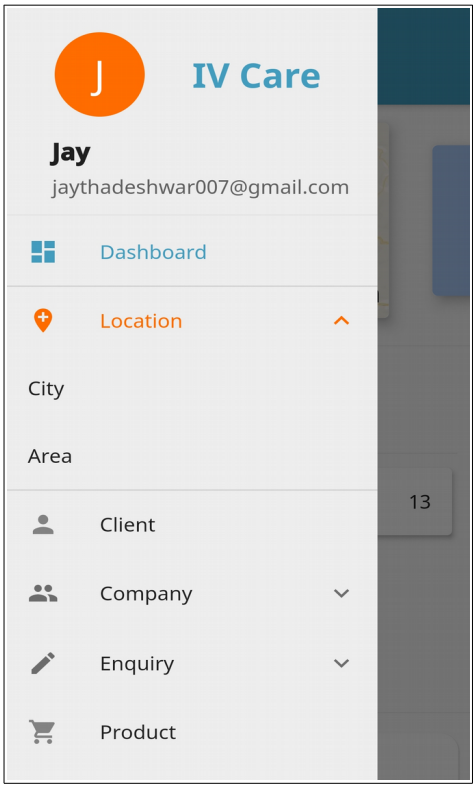
```
40.  st.setLong(9, businessAreaForCompany_info.getCreatedBy());
41.  st.setInt(10, businessAreaForCompany_info.getDeviceType());
42.  st.setTimestamp(11, businessAreaForCompany_info.getCreatedOn());
43.  st.setLong(12, businessAreaForCompany_info.getCreatedBy());
44.  st.setInt(13, businessAreaForCompany_info.getDeviceType());
45.
46.  boolean rs = st.execute();
47.  c.commit();
48.  if (rs) {
49.  return rs;
50.  } else {
51.  return rs;
52.  }
53.  } catch (SQLException s) {
54.  s.printStackTrace();
55.  return null;
56.  } catch (ClassNotFoundException e) { logger.error("Exception: "+e.getMessage());
57.  ;
58.  return null;
59.  } finally {
60.  try {
61.  c.close();
62.  st.close();
63.  } catch (Exception e) {
64.  }
65.  }}
```

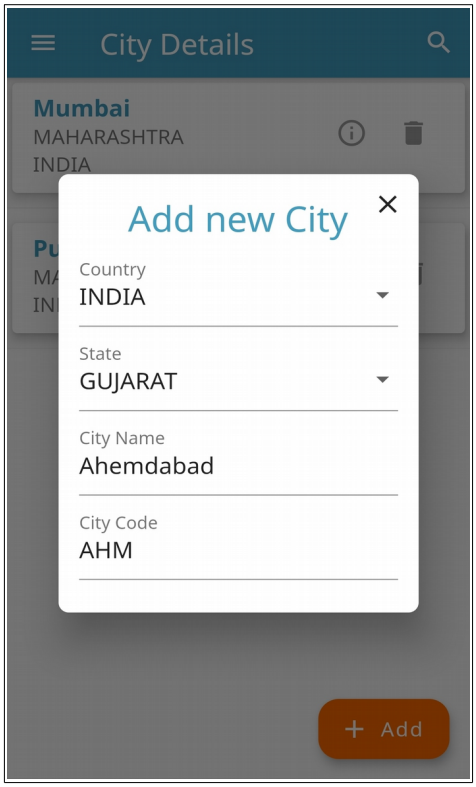
3. Database Scripts

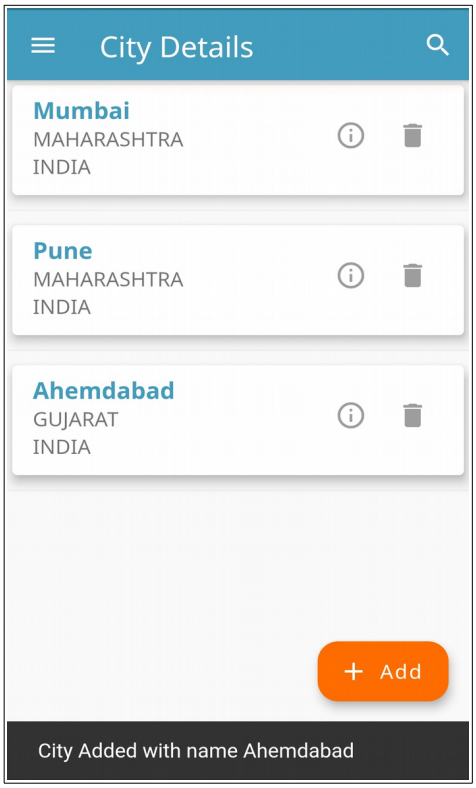
```
1.  CREATE OR REPLACE FUNCTION location."fn_insertIntoBusinessAreaForCompany"(
2.     "p_CompanyID" bigint,
3.     "p_CountryID" bigint,
4.     "p_StateID" bigint,
5.     "p_CityID" bigint,
6.     "p_AreaName" text,
7.     "p_AreaCode" text,
8.     "p_AreaDescription" text,
9.     "p_CreatedOn" timestamp without time zone,
10.                 "p_CreatedBy" bigint,
11.                 "p_DeviceType" integer,
12.                 "p_LastEditOn" timestamp without time zone,
13.                 "p_LastEditBy" bigint,
14.                 "p_LastEditDeviceType" integer)
15.       RETURNS boolean
16.       LANGUAGE 'sql'
17.
18.       COST 100
19.       VOLATILE STRICT
20.   AS $BODY$
21.   WITH "SelectFromAreaInfo" AS (
22.                 INSERT INTO location.areainfo("AreaName", "AreaCode", "AreaDescription",
23.                 "CreatedOn", "CreatedBy", "DeviceType", "LastEditOn", "LastEditBy",
```

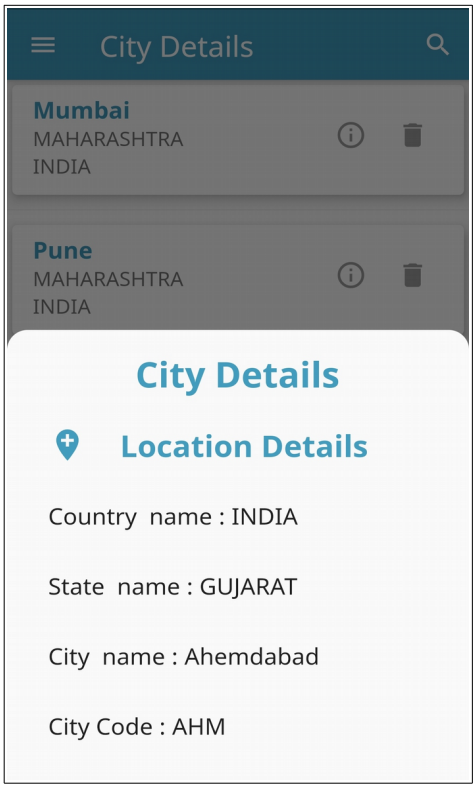
```
            "LastEditDeviceType", "CityID", "StateID", "CountryID")
            VALUES ("p_AreaName", "p_AreaCode", "p_AreaDescription", "p_CreatedOn",
                                    "p_CreatedBy","p_DeviceType", "p_LastEditOn", "p_LastEditBy",
                                    "p_LastEditDeviceType", "p_CityID", "p_StateID", "p_CountryID")
            RETURNING "AreaID" AS "AreaID"
),
"SelectFromBusinessAreaInfo" AS (
            INSERT INTO location."businessAreaForCompany"("CompanyID", "AreaID",
                        "CityID", "CreatedOn", "CreatedBy", "DeviceType",
                        "LastEditOn", "LastEditBy", "LastEditDeviceType")
            VALUES ("p_CompanyID", (SELECT "AreaID" FROM "SelectFromAreaInfo"),
                                    "p_CityID", "p_CreatedOn", "p_CreatedBy", "p_DeviceType",
                                    "p_LastEditOn", "p_LastEditBy", "p_LastEditDeviceType")
            RETURNING true
)

SELECT * FROM "SelectFromBusinessAreaInfo";
$BODY$;
```

Screenshots:





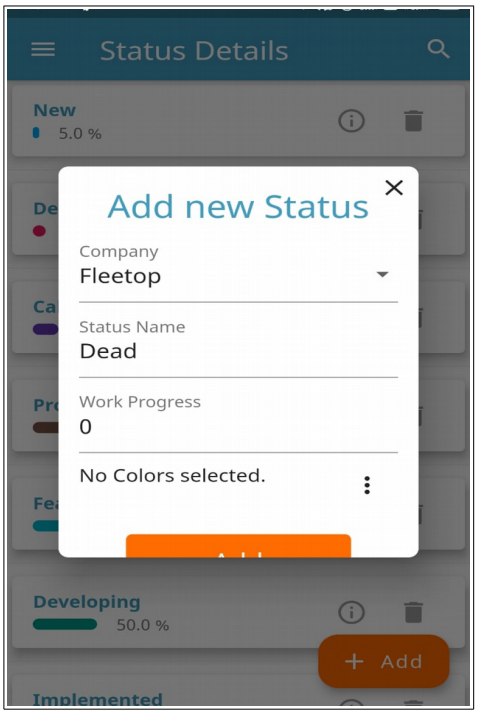


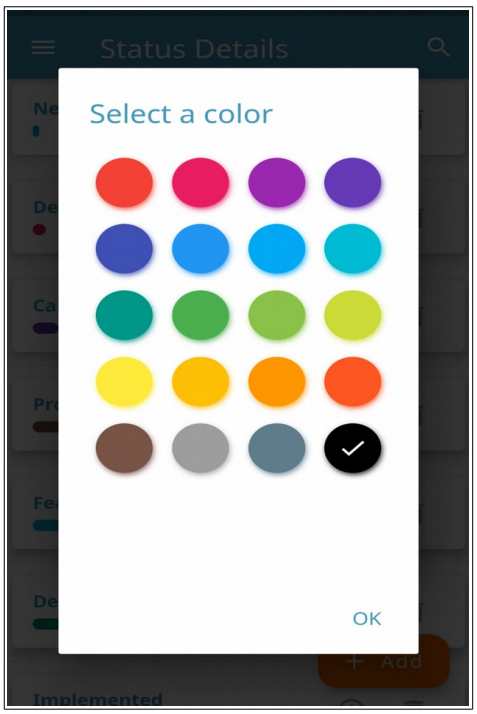


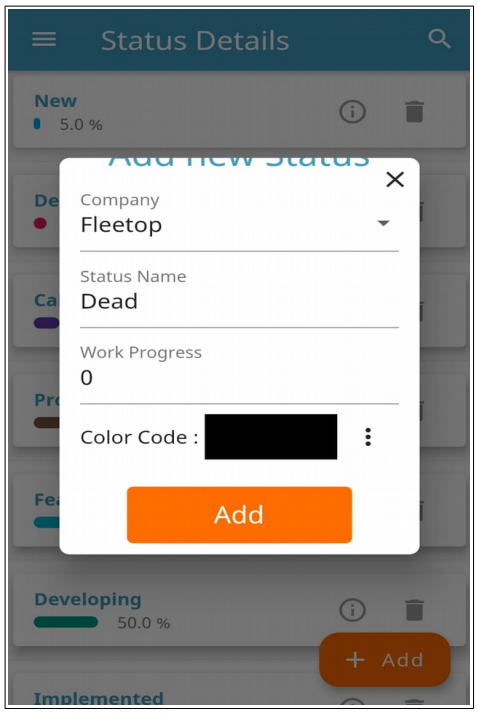
10.2.6 Adding new status for enquiry:

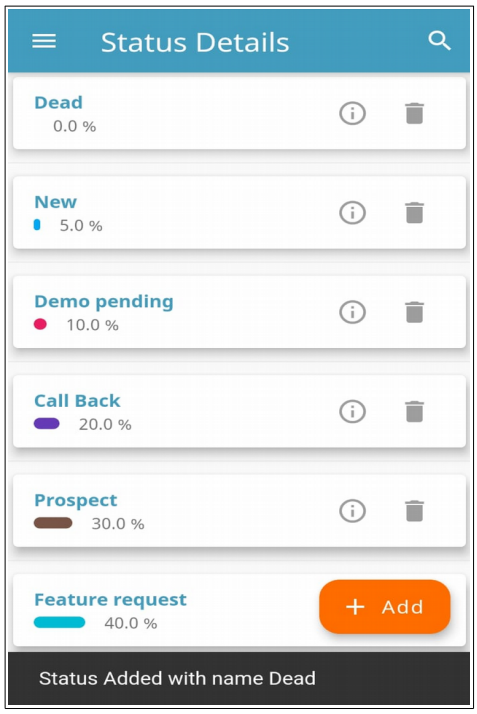
The owner of the company can add status for each stage of the enquiry. The status helps in understanding the current stage of enquiry. The status added by the owner generates a lifecycle for each enquiry.

Screenshots:





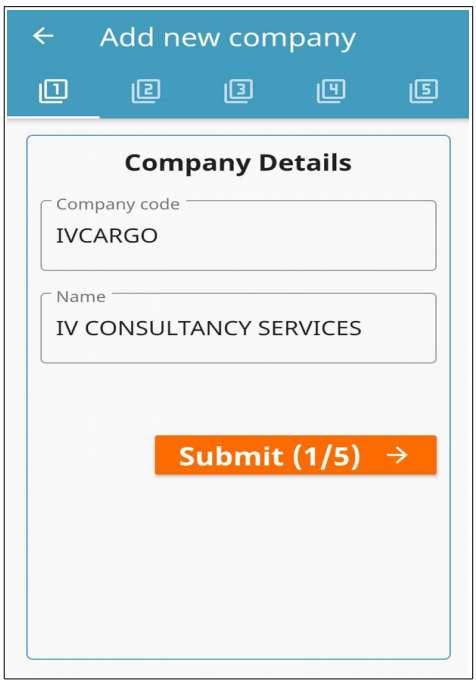


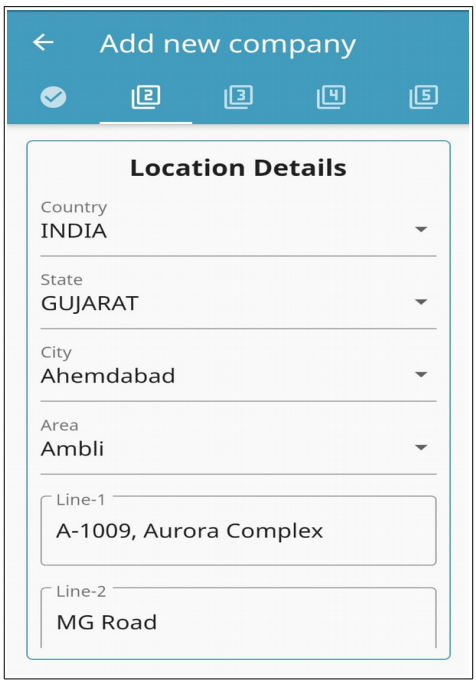


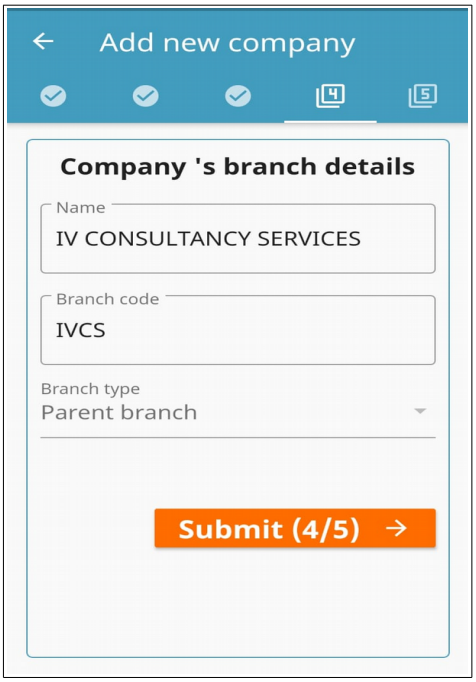
10.2.7 Creating new company and company branch:

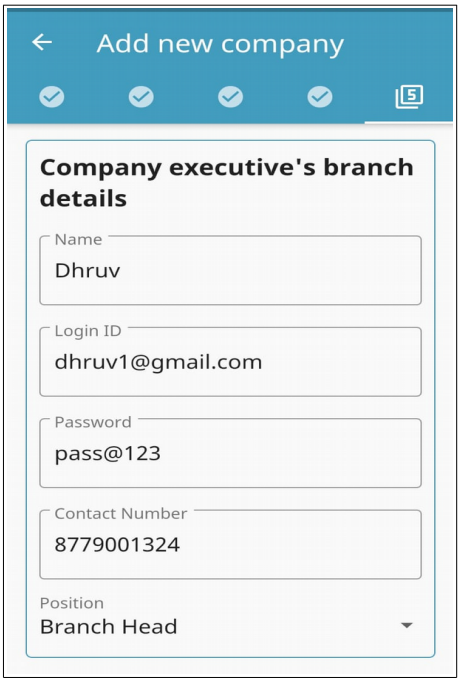
The owner can have multiple companies and each company can have multiple branches. The application provides information and manages the operations of all the companies of the owner. This feature of the application enables the owner to create new companies and their branches.

Screenshots:





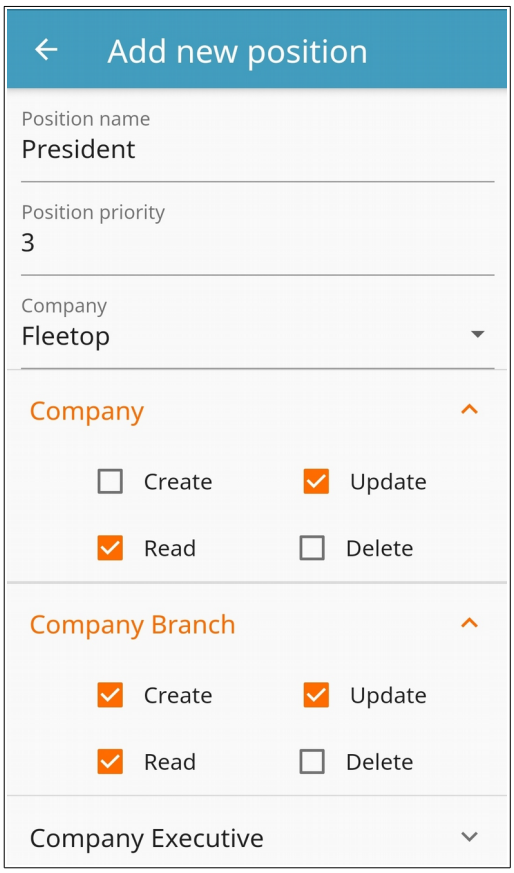


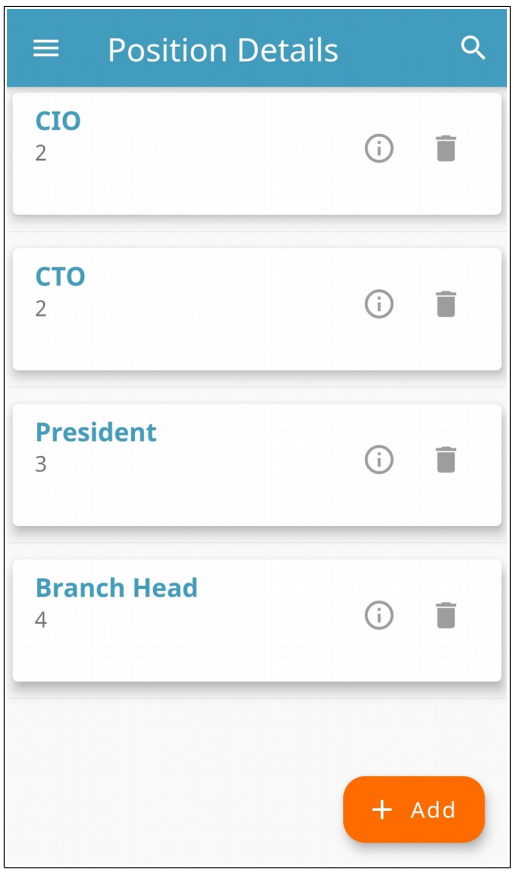


10.2.8 Creating new positions for executive:

The employees perform the task at several levels in the organization. Each employee has a different role to perform in the organization. The application provides the feature to generate new positions to serve this hierarchical structure of the organization. The owner can create new positions and provide access rights to perform specific tasks.

Screenshots:





10.2.9 Listening recorded call for quality control:

The owner can listen to the recorded conversation between executives and clients that helps in understanding the quality of marketing. The application provides the recorded call on the following basis:

1. Employee
2. Date

The owner or management can monitor and evaluate the performance of the employee and place them in the proper market segment.

Code Snippet:

1. Frontend

```
class _CallRecordingState extends State<CallRecording> {
  bool isLoading = false;

  refresh(context) {
    return FutureBuilder(
        future: getData(),
        builder: (c, s) {
          if (s.data == null) {
            return Center(child: CircularProgressIndicator());
          } else {
            if (s.data.length == 0) {
              return Center(child: Text('No Data Exists.!!'));
            } else {
              return IgnorePointer(
                ignoring: isLoading,
                child: Stack(children: <Widget>[
                  ListView.builder(
                      itemCount: s.data.length,
                      itemBuilder: (c, i) {
                        return makeCard(s.data[i]);
                      }),
                  isLoading
                      ? Center(child: CircularProgressIndicator())
                      : Container(),
                ]),
              );
            }
          }
        });
  }

  Future<List<CallRecordingClass>> getData() async {
    var json = await ApiCall.getDataFromApi(
        Uri.GET_CALL_TRANSACTION + '/${widget.id}');
    List<CallRecordingClass> callRecordings = [];
```

```
  if (json == "nothing") {
    return callRecordings;
  }
  for (int i = 0; i < json.length; i++) {
    int _callTransactionId = json[i]['callTransactionId'];
    int _clientId = json[i]['clientId'];
    int _companyExecutiveId = json[i]['companyExecutiveId'];
    String _clientNameOnCompanyExecutiveList =
        json[i]['clientNameOnCompanyExecutiveList'];
    String _clientContactNumber = json[i]['clientContactNumber'];
    int _callType = json[i]['callType'];
    dynamic _talkDuration = json[i]['talkDuration'];
    String _callTime = json[i]['callTime'];
    String _fileURL = json[i]['fileURL'];
    print(_fileURL);
    String _filePath = json[i]['filePath'];

    CallRecordingClass obj = new CallRecordingClass.retrieve(
        _callTransactionId,
        _clientId,
        _companyExecutiveId,
        _clientNameOnCompanyExecutiveList,
        _clientContactNumber,
        _callType,
        _talkDuration,
        _callTime,
        _fileURL,
        _filePath);

    callRecordings.add(obj);
  }
  return callRecordings;
}

Widget makeCard(var obj) {
  String callTimeFormatted = obj.callTime;
  List<String> data = callTimeFormatted.split(" ");
  print(data);
  List<String> data1 = data[0].split("-");
  callTimeFormatted =
      data1[2] + "/" + data1[1] + "/" + data1[0] + " " + data[1];
  print('defefwefwf  ${obj.talkDuration}');
  return Column(
    children: [
      Card(
          elevation: 7,
          child: ListTile(
            dense: false,
            isThreeLine: true,
            title: Text(obj.clientNameOnCompanyExecutiveList),
            subtitle: Column(
              mainAxisSize: MainAxisSize.min,
              children: <Widget>[

                Flexible(
                  fit: FlexFit.loose,
                  child:
                      AudioProvider(obj.fileURL, obj.talkDuration.toString()),
                )
              ],
```

```
            ),
            onTap: null,
          )),
      Divider(),
    ],
  );
 }

 @override
 void initState() {
  super.initState();
 }

 @override
 Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Call Recordings'),
    ),
    body: refresh(context),
  );
 }
}
```

2. API

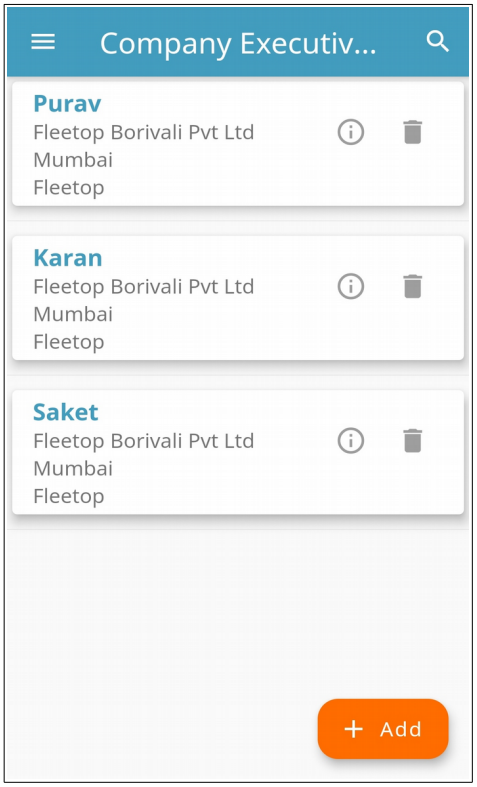
```
@RestController
@RequestMapping("/callTransaction")
@CrossOrigin(origins = Constants.URL_TO_REQUEST)
public class CallTransactionController {

@Autowired
CallTransactionBusinessLogic ctbl;

@GetMapping(path="{companyExecutiveId}")
public                                  ResponseEntity<List<@Valid                                  CallTransactionSelect>>
selectCallTransactionsByCompanyExecutiveId(@PathVariable @NotNull Long companyExecutiveId)
{
            return ctbl.selectCallTransactionsByCompanyExecutiveId(companyExecutiveId);
}}

@Override
public List<CallTransactionSelect> selectCallTransactionsByCompanyExecutiveId(
                        Long companyExecutiveId) throws SQLException, ClassNotFoundException {

            Connection c=ConnectionProvider.getConnection();
            CallableStatement                                  stmt=c.prepareCall("SELECT                                  *
FROM \"callTransactions\".\"fn_selectCallTransactionByCompanyExecutiveId\"(?);");
            stmt.setLong(1, companyExecutiveId);
            ResultSet rs=stmt.executeQuery();
            List<CallTransactionSelect> lr=new ArrayList<CallTransactionSelect>();
            while(rs.next())
            {
                                    lr.add(
                                                new CallTransactionSelect(
                                                            rs.getLong("CallTransactionId"),
                                                            rs.getLong("ClientId"),
                                                            rs.getLong("CompanyExecutiveId"),
```

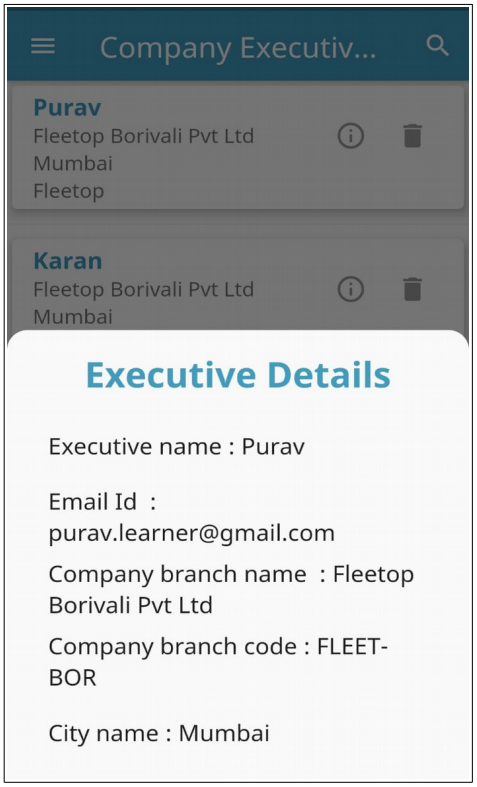
```
                                                rs.getString("ClientNameOnExecutiveContactList"),
                                                rs.getString("ClientContactNumber"),
                                                rs.getInt("CallType"),
                                                rs.getDouble("TalkDuration"),
                                                rs.getTimestamp("CallTime"),
                                                rs.getString("FileURL"),
                                                "PRIVATE FILE PATH"
                                                //rs.getString("FilePath").replace("\\","/")
                                        )
                                );

                }
                rs.close();
                stmt.close();
                c.close();
                return lr;
}
```

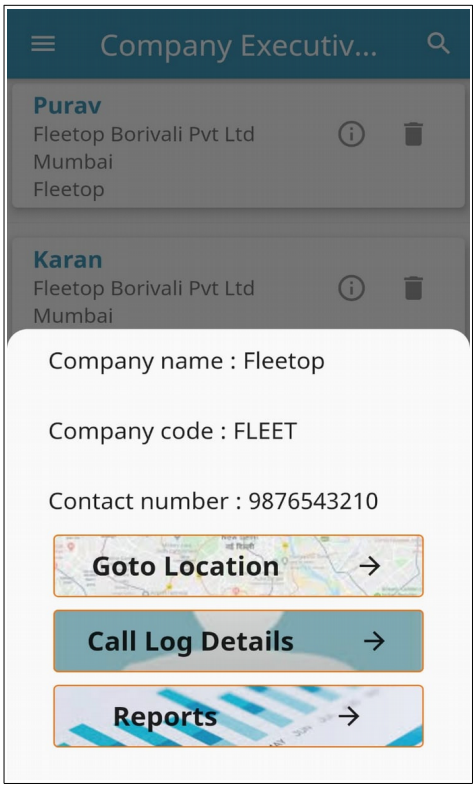
3. Database Scripts

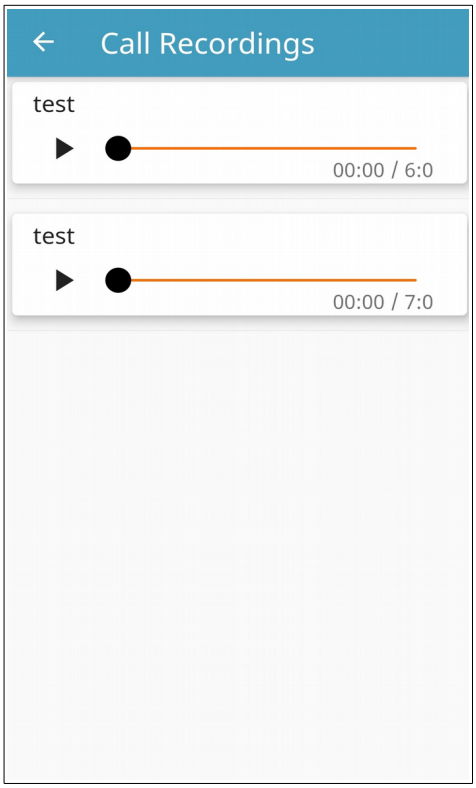
```
1.  CREATE OR REPLACE FUNCTION "callTransactions"."fn_selectCallTransactionByCompanyExecutiveId"(
2.      "CompanyExecutiveIdIn" bigint)
3.              RETURNS TABLE("CallTransactionId" bigint, "ClientId" bigint, "CompanyExecutiveId" bigint,
    "ClientNameOnExecutiveContactList" text, "ClientContactNumber" text, "CallType" integer, "TalkDuration"
    double precision, "CallTime" timestamp without time zone, "FileURL" text, "FilePath" text)
4.      LANGUAGE 'sql'
5.
6.      COST 100
7.      VOLATILE
8.      ROWS 1000
9.  AS $BODY$
10.     SELECT
11.       "CallTransactionId",
12.       "ClientId",
13.       "CompanyExecutiveId",
14.       "ClientNameOnExecutiveContactList",
15.       "ClientContactNumber",
16.       "CallType",
17.       "TalkDuration",
18.       "CallTime",
19.       "FileURL",
20.       "FilePath"
21.                 FROM "callTransactions"."CallTransaction"
22.       WHERE "MarkForDelete"=false
23.       AND "CompanyExecutiveId"="CompanyExecutiveIdIn";
24.   $BODY$;
```

Screenshots:





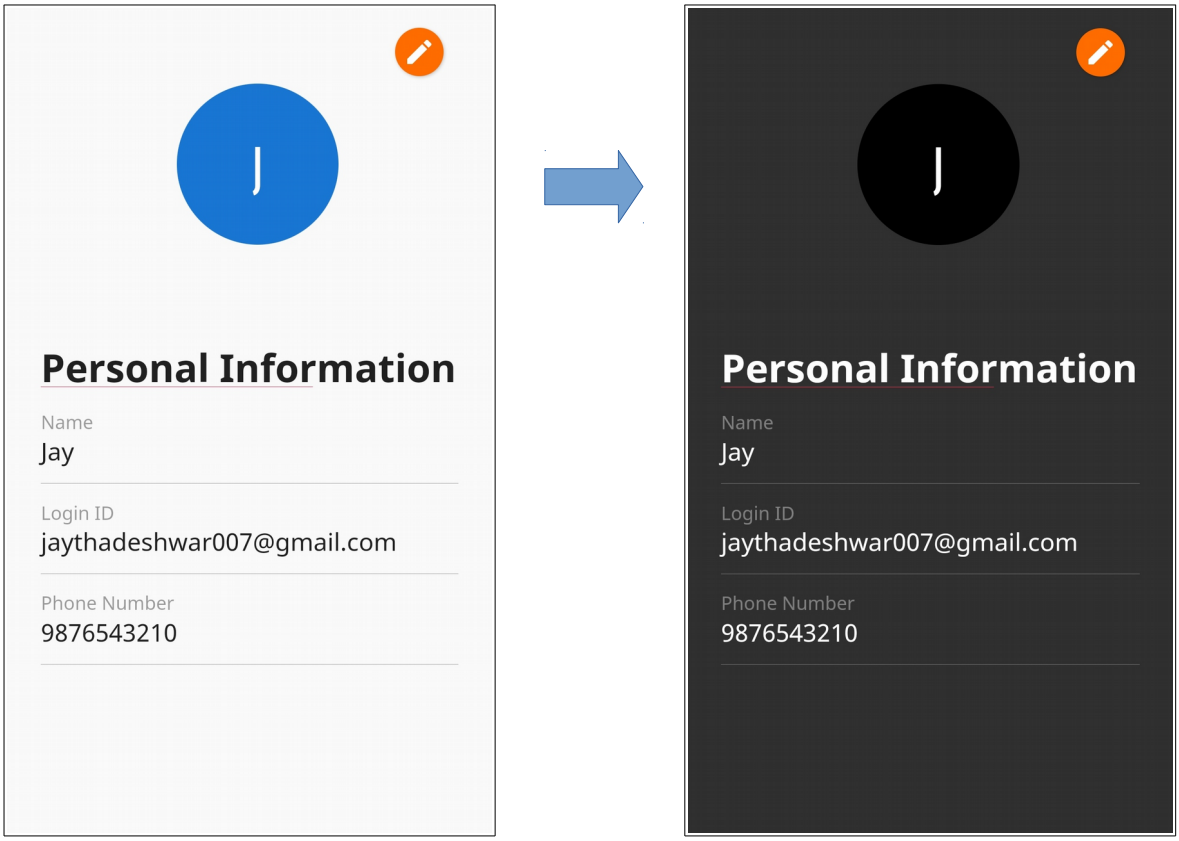




10.3 Dark mode:

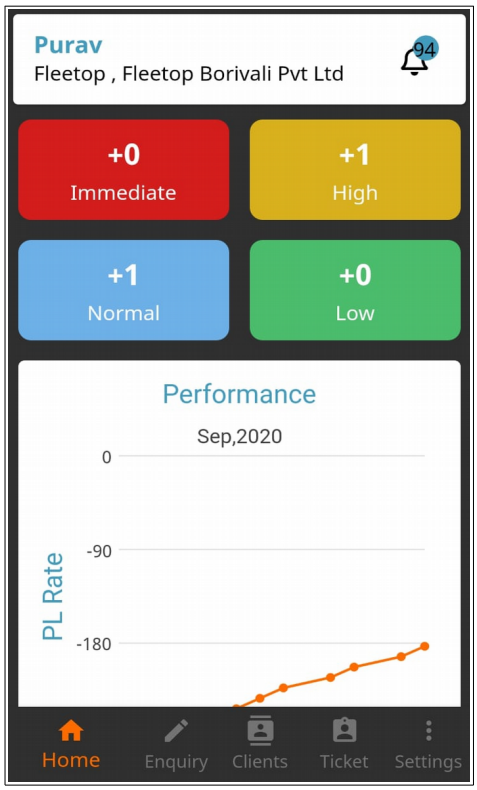
The application provides the feature of dark mode for both executive and owner interface. Dark mode apps can prolong the battery life of your smartphone. Google has confirmed that using dark mode on OLED screens has been a huge help for battery life. It not only provides great visual effect but also reduces eye strains.

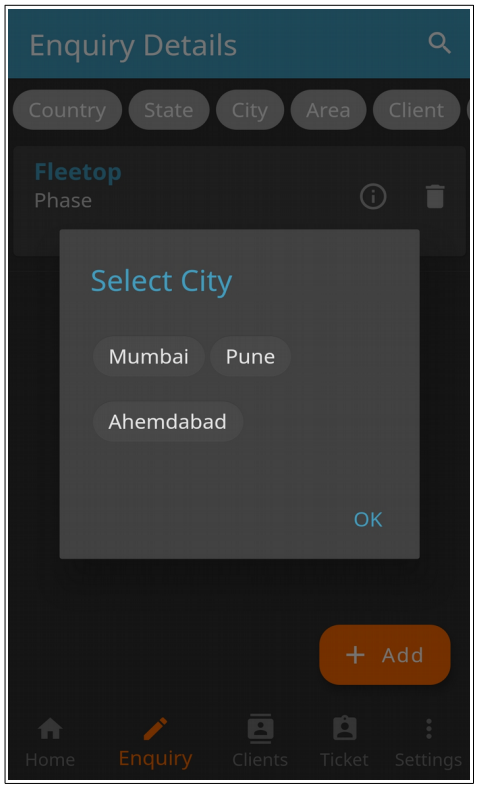
Screenshots:

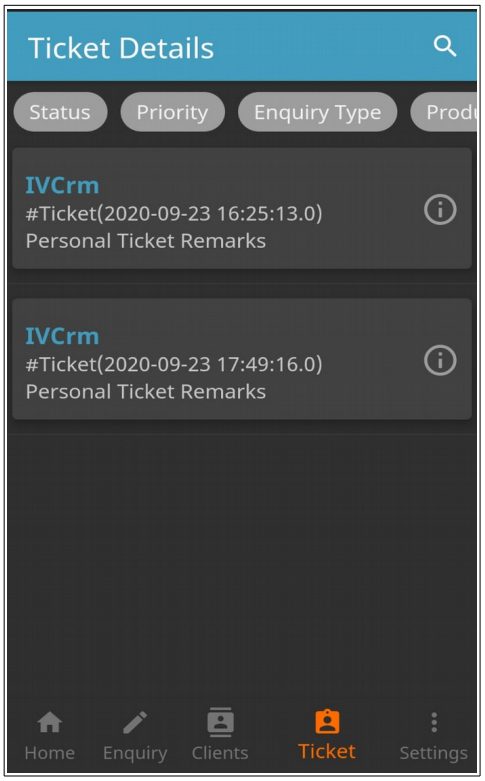


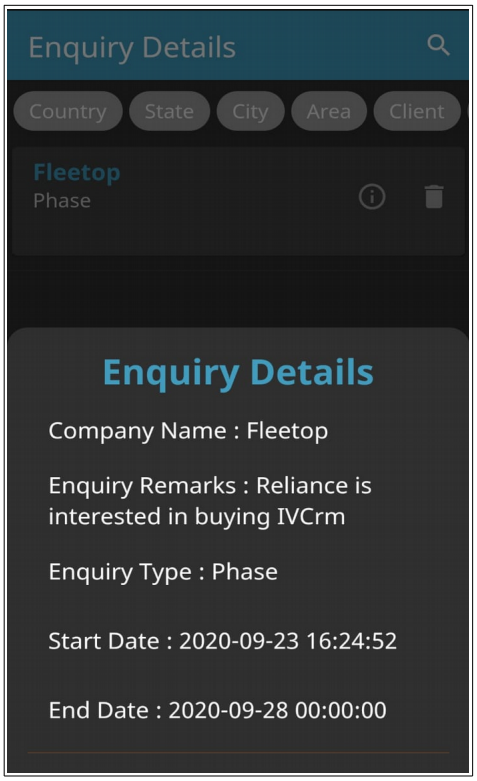
NOTE: *Clicking on blue button changes app from light to dark mode.*

Executive App

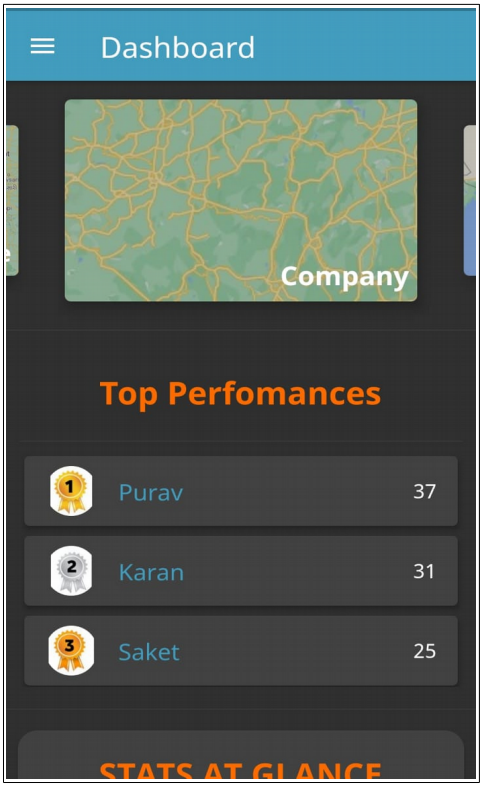


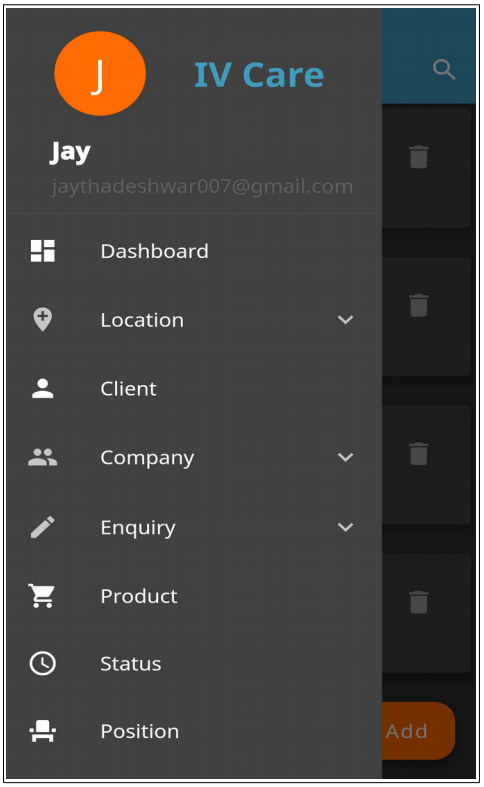


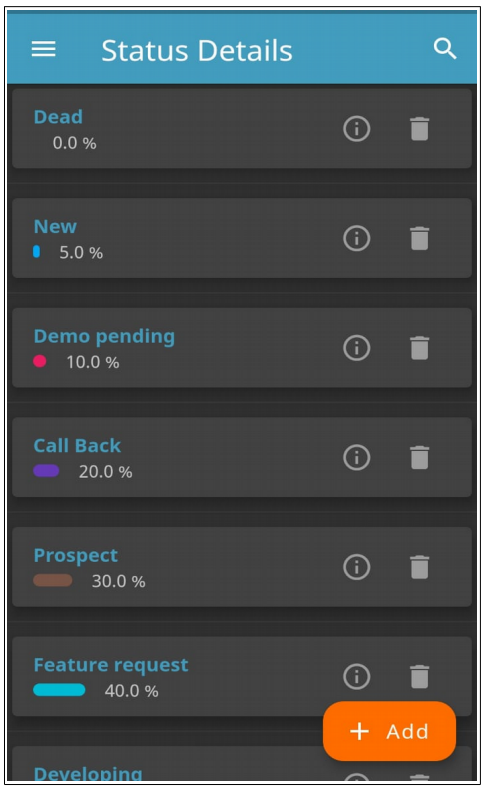


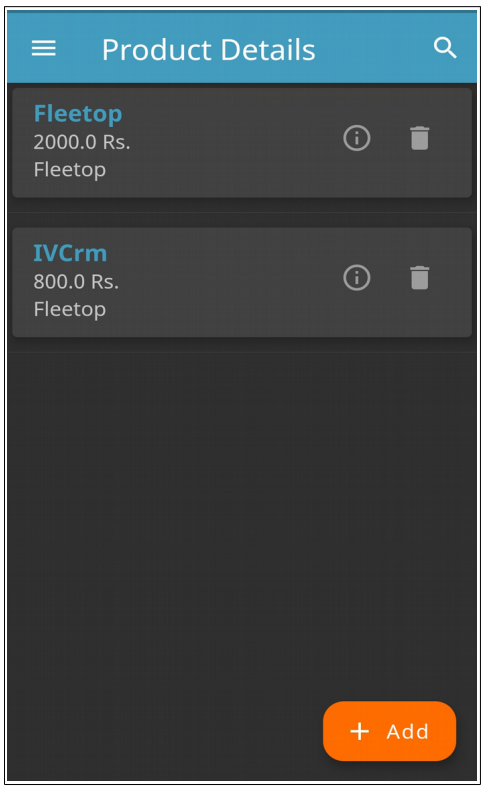


Owner App


Dashboard
Company
Top Perfomances
Purav 37
Karan 31
Saket 25


IV Care
Jay
jaythadeshwar007@gmail.com
Dashboard
Location
Client
Company
Enquiry
Product
Status
Position
Add


Status Details
Dead 0.0 %
New 5.0 %
Demo pending 10.0 %
Call Back 20.0 %
Prospect 30.0 %
Feature request 40.0 %
Add


Product Details
Fleetop
2000.0 Rs.
Fleetop
IVCrm
800.0 Rs.
Fleetop
Add

10.4 API Functionalities:

The API in framed with a secure mechanism and too has a rich documentation service with a Logging framework which enriches the use and functionalities of the API.

16 Limitations

- **Call recording:**

  The application provides a one-side call recording and if kept on the speaker then two-way call recording with low sound. Google currently supports only native libraries for the two-way call recording feature in Android. Implementation of native libraries in flutter would take quite an amount of time and can be considered a separately new project.

- **Lack of user knowledge:**

  The user of the application is required to understand the goals that can be achieved through the application. A wrong interpretation of functionalities might end up in wastage of time and money. The user must understand each aspect of an application and its working flow. That will not only help in the management of resources but also provides an efficient infrastructure for the company to follow.

17 Applications

- Helps management keep everything in one place, allowing sales, marketing, and service departments to work more efficiently.
- Once the customer data is available, management can begin to formulate and target marketing that is more apt to client conversions.
- Reports generated from the application help management evaluate their business's daily activities or problems that arise, make decisions and track progress.
- By tracking all communication with the customers, the application helps salespeople to know exactly when customers need to be contacted; for example, for product replacement, contract renewal, or for an upsell to a new product or service.
- Helps streamline the entire sales cycle, which results in closing deals in your sales pipeline and helping everyone in the team to reach targets faster.

18 Future Scope

- The signup process of the application requires payment gateway integration for revenue generation.
- A strong bond can be developed between the sales force and sales management by adding a shared calendar feature. This feature unites all the team members and keeping everyone up to date regarding future work.
- The one-way call recording feature of the application can be extended into the two-way recording.
- The tracking system that monitors the communication channels of the firm and captures all the feedback, messages, and information. That can be further used for data analysis and generating useful information.

19 Conclusion

Change is inevitable in all the prospects, to cope up with evolving technological trends in business and maintain competitive advantage, CRM is a complete solution. IVCRM plays a very critical role in identifying, acquiring, and retaining the customers, and thereby managing a healthy relationship with them.

It standardizes a sales cycle and eliminates the common sales issues such as getting quotations about the product, generating customized products and proposals, and lead management. The IVCRM provides comprehensive reports, geolocation, and call recording feature which supports management effectively in the decision-making process.

20 References

[1] https://www.udemy.com/course/restful-web-services-with-spring-framework-a-quick-start/

[2] https://flutter.dev/docs/development/ui/layout

[3] https://medium.com/@dltlabs/how-to-build-a-flutter-card-list-in-less-than-10-minutes-9839f79a6c08

[4] https://codelabs.developers.google.com/codelabs/google-maps-in-flutter

[5] https://pub.dev/packages/flutter_sound

[6] https://dzone.com/articles/java-springboot-rest-api-to-uploaddownload-file-on

[7] https://medium.com/flutterdevs/implement-dark-mode-in-flutter-using-provider-158925112bf9

[8] https://api.flutter.dev/flutter/material/Chip-class.html

[9] https://medium.com/@info_67212/how-to-perform-a-search-in-flutter-app-1365bd317a0c

[10] https://dzone.com/articles/spring-boot-security-json-web-tokenjwt-hello-world

[11] https://www.postgresql.org/docs/9.2/plpgsql-structure.html

[12] https://stackoverflow.com/questions/41128944/how-to-share-a-file-using-flutter

[13] https://google.github.io/charts/flutter/gallery.html

[14] https://www.youtube.com/playlist?list=PLOU2XLYxmsIL0pH0zWe_ZOHgGhZ7UasUE

[15] https://www.springboottutorial.com/spring-boot-swagger-documentation-for-rest-services

[16] https://www.baeldung.com/exception-handling-for-rest-with-spring

[17] https://medium.com/@mahmudahsan/how-to-create-validate-and-save-form-in-flutter-e80b4d2a70a4