MOVIEWORLD

A PROJECT REPORT
Submitted By

MEENU SINGH
University Roll No. 190029014960

Submitted in partial fulfillment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
MS. NEELAM RAWAT

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

DECLARATION

I hereby declare that the work presented in this report entitled “Movie World", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences andgiven credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarismand the manipulation of the experiments and results, I shall be fully responsible andanswerable.

Name: Meenu Singh

RollNo:1900290149060

Branch: Master of Computer Application

(Candidate Signature)

CERTIFICATE

Certified that **Meenu Singh (University Roll No 1900290149060),** have carried out the project work having "**MovieWorld**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU),Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Name :- Meenu Singh
University Roll No 1900290149060

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Ms. Neelam Rawat
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Ajay Shrivastava
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

Mobile applications stood top among usability and user convenience. Many applications are available in the market to manage personal and group expenses. Not many applications provides a comprehensive view of both use cases. In this project, we develop a mobile application that keeps track of user personal expenses, his/her personal contribution towards group expenses; maintain monthly incomes, recurring and adhoc payments. It provides information of "who owes who and by how much". The proposed application would eliminate sticky note, spreadsheet and ledger that cause confusions, data inconsistency problems while recording and splitting of expenses. With our application user can manage his expenses more effectively. This application will not only helps users to manage their expenses but also help marketing executives to plan marketing according to the needs of users.

Movie World As the name itself suggests, this project is an attempt to manage our daily expenses in a more efficient and manageable way. Sometime we can't remember where our money goes. And we can't handle our cash flow.

For this problem, we need a solution that everyone can manage their expenses. So we decided to find an easier way to get rid of this problem. So, our application attempts to free the user with as much as possible the burden of manual calculation and to keep the track of the expenditure.

Instead of keeping a diary or a log of the expenses, this application enables the user to not just keep the control on the expenses but also to generate and save reports.

With the help of this application, the user can manage their expenses on a daily, weekly and monthly basis. Users can insert and delete transactions as well as can generate and save their reports.

The graphical representation of the application is the main part of the system as it appeals to the user more and is easy to understand.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Neelam Rawat** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications**, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

-Meenu Singh

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPION

Mobile applications stood top among usability and user convenience. Many applications are available in the market to manage personal and group expenses. Not many applications provides a comprehensive view of both use cases. In this project, we develop a mobile application that keeps track of user personal expenses, his/her personal contribution towards group expenses; maintain monthly incomes, recurring and adhoc payments. It provides information of "who owes who and by how much". The proposed application would eliminate sticky note, spreadsheet and ledger that cause confusions, data inconsistency problems while recording and splitting of expenses. With our application user can manage his expenses more effectively. This application will not only helps users to manage their expenses but also help marketing executives to plan marketing according to the needs ofusers.

Mobile applications stood top among usability and user convenience. Many applications are available in the market to manage personal and group expenses. Not many applications provides a comprehensive view of both use cases. In this project, we develop a mobile application that keeps track of user personal expenses, his/her personal contribution towards group expenses; maintain monthly incomes, recurring and adhoc payments. It provides information of "who owes who and by how much". The proposed application would eliminate sticky note, spreadsheet and ledger that cause confusions, data inconsistency problems while recording and splitting of expenses. With our application user can manage his expenses more effectively. This application will not only helps users to manage their expenses but also help marketing executives to plan marketing according to the needs ofusers.

ONLINE MOVIE TICKET BOOKING SYSTEM Future scope and further enhancement of the Project Future Scope The project E-ticket System for download Online ticket booking system project source I have downloaded online movie ticket booking system. pls send me online ticket reservation system Closed Online movie ticket reservation system This project received 33 bids from talented freelancers with an Download Cinema Reservation System in PHP More Downloads on

Online Movie Ticket Booking System >> List of List of other Online Reservation System ProjectsCinema hall ticket booking system or online movie ticket system project main objective is to develop a software for cinema halls for selling ticket through online.

Project Descriptio project made in PHP to implement Project MovieTicket Booking System

Project . YOU CAN BOOK, CANCEL, VIEW TICKETS HISTORY multiplex movie ticket booking system practical online Tickets reservation system for Cinema halls. This project is aimed at developing an online ticket Online.

The MovieWorld is a mobile application intended to run on android device namely smart phone. Expense Manager is designed to efficiently cater the needs of users by eliminating imparting costs and settling vows to friends. The application encourages corresponding users help in who owes who, and for what. Aim is use better approaches to help users and their companions to share expenses easily. This new application will let bunch users and their companions to have detailed view inside this application around individual costs. The app allows its users to add a remark to an expense, click on the expense name in any expense list. Bill posting will have space for comments and notes container with a "Post" catch underneath. The Expense Manager has notification option to notify each time somebody adds a remark to an expense user is on, or user can withdraw to posted bill. The additional feature that we are going to add in this application that enable us to collect the sample data of users expenses and use this to study patterns of expenses in certain area or by specific kinds of spending for market analysis. These patterns can be derived using some data mining techniques such as clustering, classification and association.

2. BACKGROUND STUDY

The idea of developing this project in platform arises with the frequent problems being experienced by people in sharing among them. RESEARCH ARTICLE OPEN ACCESS International Journal of Computer Techniques -– Volume 3 Issue 2, Mar- Apr 2016 ISSN :2394-2231

http://www.ijctjournal.org Page 61 Some of the concerns related dividing expenses are like maintaining a personal expense is a BIG problem, splitting the expenses among group is confusing. Some of the conventional methods used to tackle this problem in normal circumstances are like making use of a sticky note by normal users, Proficient people deal with this kind problems by using spreadsheet to record expenses and using a ledger to maintain large amounts data by especially by experts. As this shows that it is variable methods used by different people. This makes using this data inconsistent. There are still problems in areas like there is no assurance for data consistency, there are chances of critical inputs can be missed and the manual errors may creep in. The Data recorders are not always handy and it could be hectic process to have overall view of those expenses. We believe a handy design a handy mobile application which handles these problems. Such that app is capable of recording the expenses and giving comprehensive view with easy to use user interface and this app is intelligent enough to answer: 'Who owes who ? And by how much ??'

3. RELATED WORK

The mobile applications that are available in the market are very useful to the smartphone users and make their life easy. The expenses manager is also one of those applications, which much scope in daily life. As there are many similar applications available today we added some innovative features to make our application unique, easy to use and efficient. Apart from adding unique features like combining group expenses and personal expenses in to one application, we also added features like trends, estimations. Here, we have an idea of making use of application for the purpose of survey in the field of expenditures of user. This idea serves as main objective of research project. The research also includes syncing of the applications with some social networks and emails as well[5][6].

4. METHODOLOGY

This section of paper is very important and this will guide our team to successfully accomplish the goals set for research. Here, the research project methodology describes the steps and approaches to be fallowed to attain final product. As explained above our project is of splitting the expenses between the groups and also to efficiently manage the personal expenses as well. However, our projects will have additional features included as part of our research so that it makes our project unique in the market. These features would make the project more efficient and very useful for our users. Apart from the benefits user gets and there is an important use of the system that enables us to

use the data of the user with his prior permissions for the purpose of data mining for several other functionalities to be applied in market by analyzing user expenses[7][8][9].

4.1 User registration/creation This application like most of the applications will have user login screen and option for registration. The user must register in this application when he/she is using for first time. However, the user who is already registered can login to the application using his/her login credentials that are created by the user at the time of registration.

4.2 Creating, alter of user groups

1.1 Tools and Technologies

- **Hardware Processor**
- Intel ® Core™ i3-2370 CPU @2.40GHz
- **Installed Memory (RAM)**
  - GB or above
- **System Type**
  - 32/64 bit Operating System
  - **Software Interface**
- **Client-Side** Android Mobile **Software** Android Studio **Database Server** SQLite Database
- **Flaws in the current system**

- No offline data storage
- Overcrowded interface and inappropriate color schemes
- Unable to create multiple accounts
- Users get interrupted by annoying advertisements
- No privacy function
- Unable to generate PDF reports
- Unable to set budget mode (Weekly/Monthly)

1.2 Features of MovieWorld App Project

- Create multiple accounts/budget
- Delete account
- Background color
- Modify Transactions
- Offline datastore
- Passcode security
- Selecting budget mode(Weekly/Monthly)
- Generate reports as PDF files
- Fully customizable categories

- Cash flow (Pie/Bar/Graph)
- Expenses percentage
- Carryover
- Show transaction note
- Currency Symbol.

1.3 Modules of MovieWorld Android App Project

The modules which are currently covered are:

Add income/add expense

This module deals with adding income and expenses. The user has both options available for income and expense. But there is a condition if the user hasn't entered the amount yet then the user can't enter expenses. When the user enters any transaction then that transaction will be added in both Spending and Transaction tabs. If the user wants to delete that transaction then the user has to long click the transaction available in the spending tab then that transaction will be deleted from both tabs.

Modify Transactions

If the user wants to delete that transaction then the user has to click the transaction available in the spending tab then that transaction will be deleted from both tabs.

Filter Transaction view

In the transaction tab, the user can filter the transactions. In the Spinner, users can select the day, month and year and then click the filter button and according to the day, month and year transactions will appear. If the user wants to filter the transactions only on the basis of day, for example, user-selected Monday then all transactions will appear that were made on Monday.

PDF Report

In the transaction, the tab user has an option available for creating a report in PDF. Users click on the PDF button then PDF report will be generated and the user can view that report and that report will be automatically saved in the device.

Multiple Accounts

Users can create multiple accounts. In the account tab. User has the option available for creating a new account. Users will click the "+" sign button then a dialog will appear on the screen and the user can enter the name of the account then that name will be saved in the account tab. If a user wants to delete the particular account then the user has to l click the account name user want to delete. Then that account will be deleted.

Transactions overview as Pie/Bar/Graph

The user has three options available for graphical representation. When the user rotates the device then the pie chart will appear on the screen and also switch is available on the screen when the user will click on the bar chart will appear on the screen and when the user clicks on graph then Graph will appear on the screen.

Themes

At the top bar, the user has a setting option when the user clicks that then background option will appear user can select different background colors. After selecting the particular color background color will be changed.

Passcode

The passcode is available in setting option at the top bar. When the user clicks on the passcode switch when the user switches on then the passcode screen will appear and the user can choose the password and that password will be saved in the database. After that when the user will open the application user have to enter the passcode and that passcode will be matched with passcode saved in the database. If the user entered the wrong passcode then the error message will appear.

Currency Symbol

The currency symbol option is available at the top bar setting button. Users can select different currency symbols. If the user selects the dollar symbol then that symbol will appear on the spending tab.

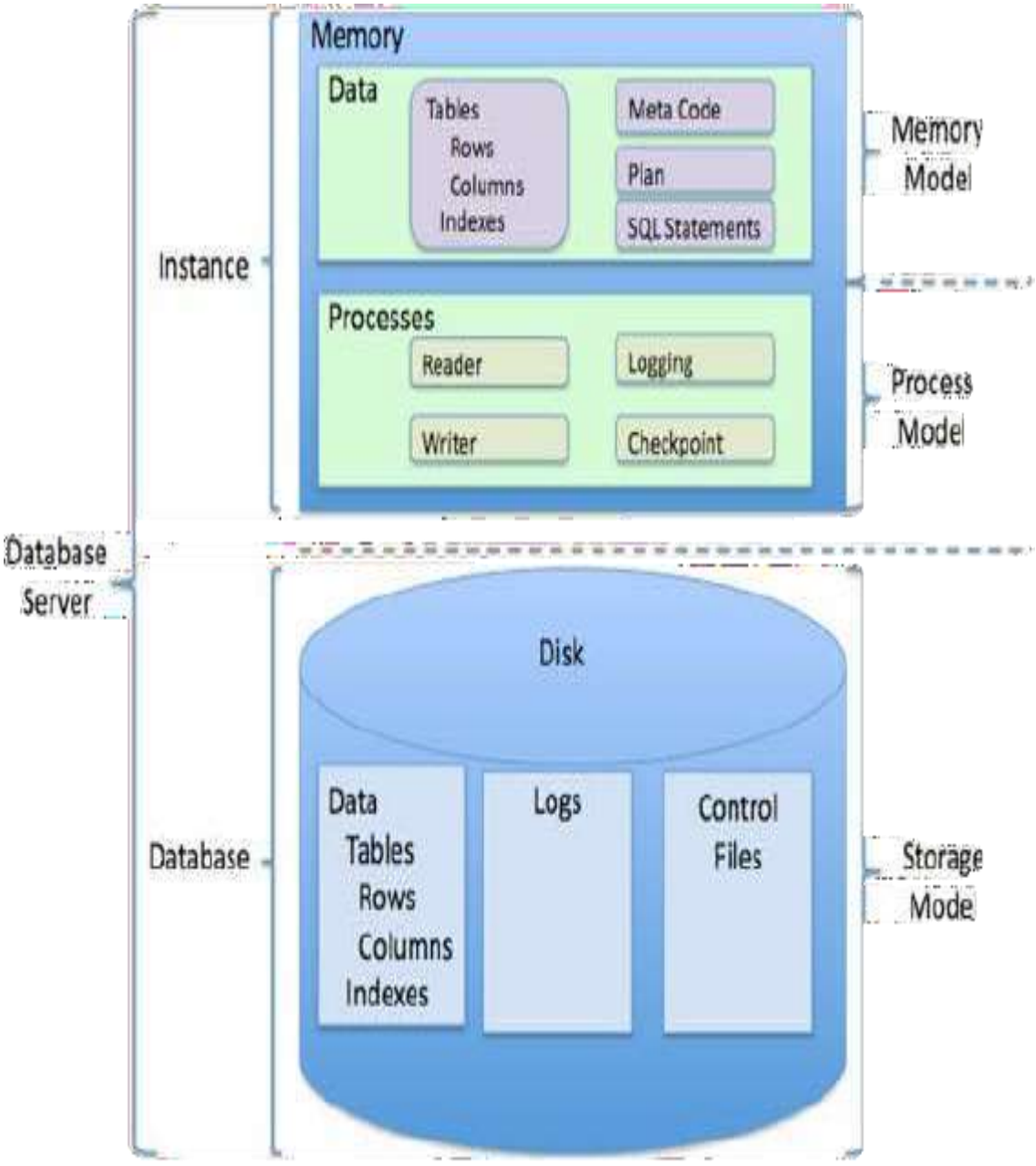
Functional Requirements

Identifier	Requirement
Req:1 Add transaction	This application will allow adding transaction.
Req:2 Delete transactio n	This application will allow the deleting transactions.
Req:3 Amount spent in categories	This application will allow adding the amount spent in a particular category.
Req:4 View all transactio ns	This application will allow viewing all previous transactions
Req:5 Total amount	This application will allow seeing the total amount, the amount spent in different categories and balance left.
Req:6 Overview	This application will allow viewing overall transactions.
Req:7 Graph representat ion	This application will show the graph which will help the users to visualize the budget.
Req:8 Pie representat ion	This application will show the pie.
Req:9 Bar representat ion	This application will show the bar.
Req:10 Change backgroun d	This application has the option to change the background.
Req:11 Passcode	This application has the option to set a passcode for security.
Req:12 Add multiple accounts	This application

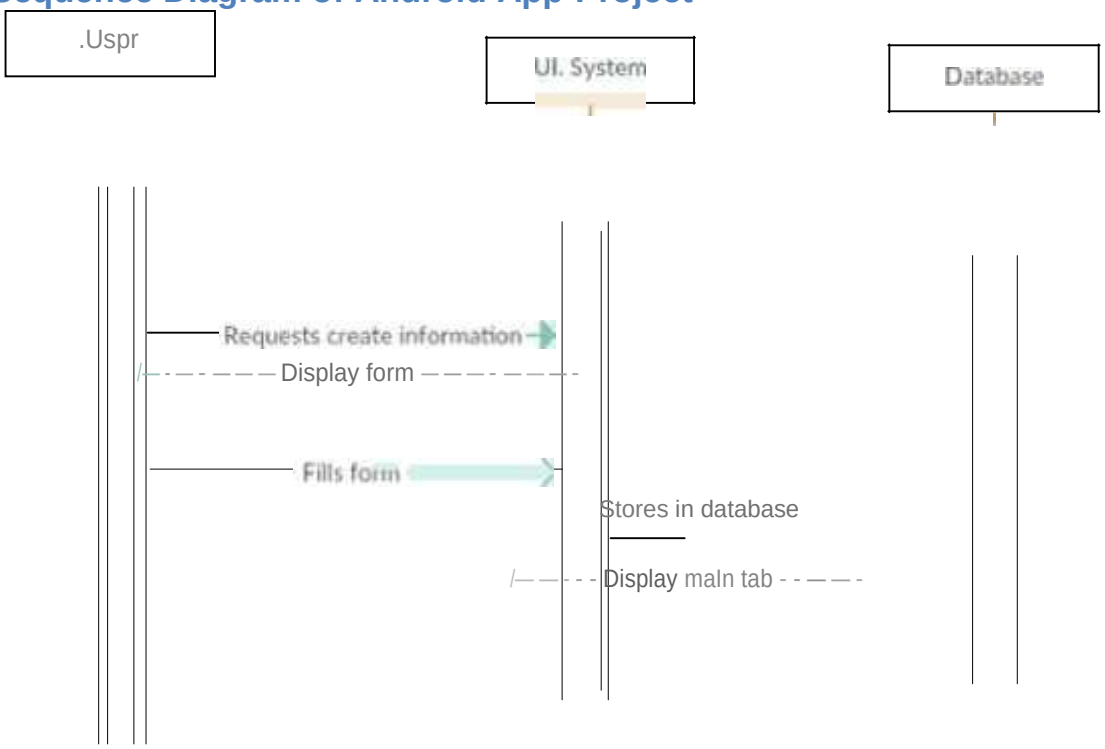
Req:13 Transactio n time/date	This application has the ability to show the transaction time along with the date on which it was created.
Req:14 Currency symbol	This application has many currency symbols as per user requirements.
Req:15 Reminder	This application has the option to set a reminder to make the transaction.
Req:16 Delete	This application will generate PDF reports of the transactions.
account	
Req:17 PDF report	This application has the option to view and filter transactions by day, month and year.
Req:18 Note	This application has the option to add a note about income and expenses.

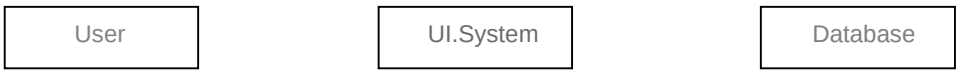
CHAPTER 3

System Design **Input/Output Form(Screens)**



1.4 Sequence Diagram of Android App Project





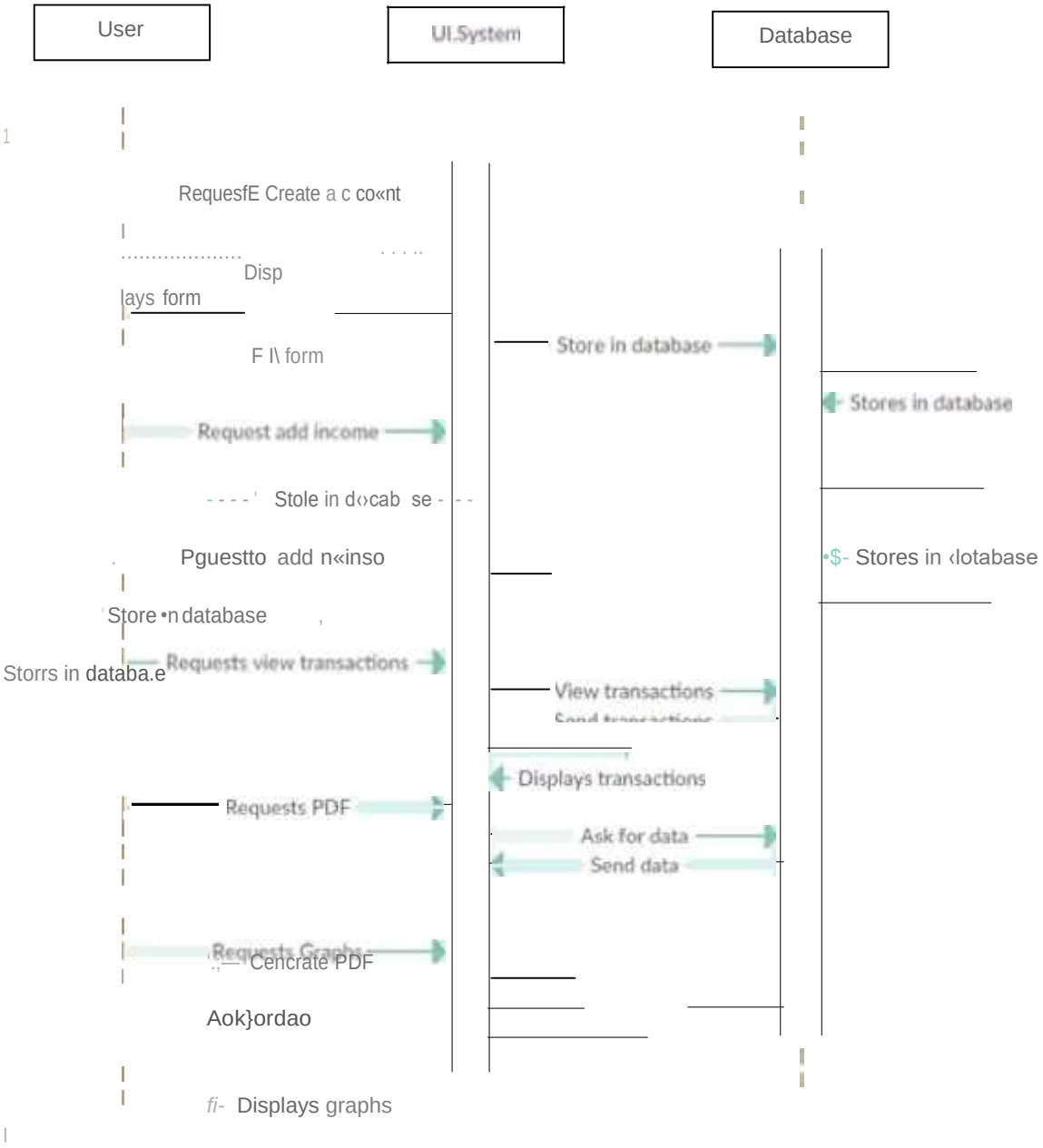
I Reque.sts to create accot›nt

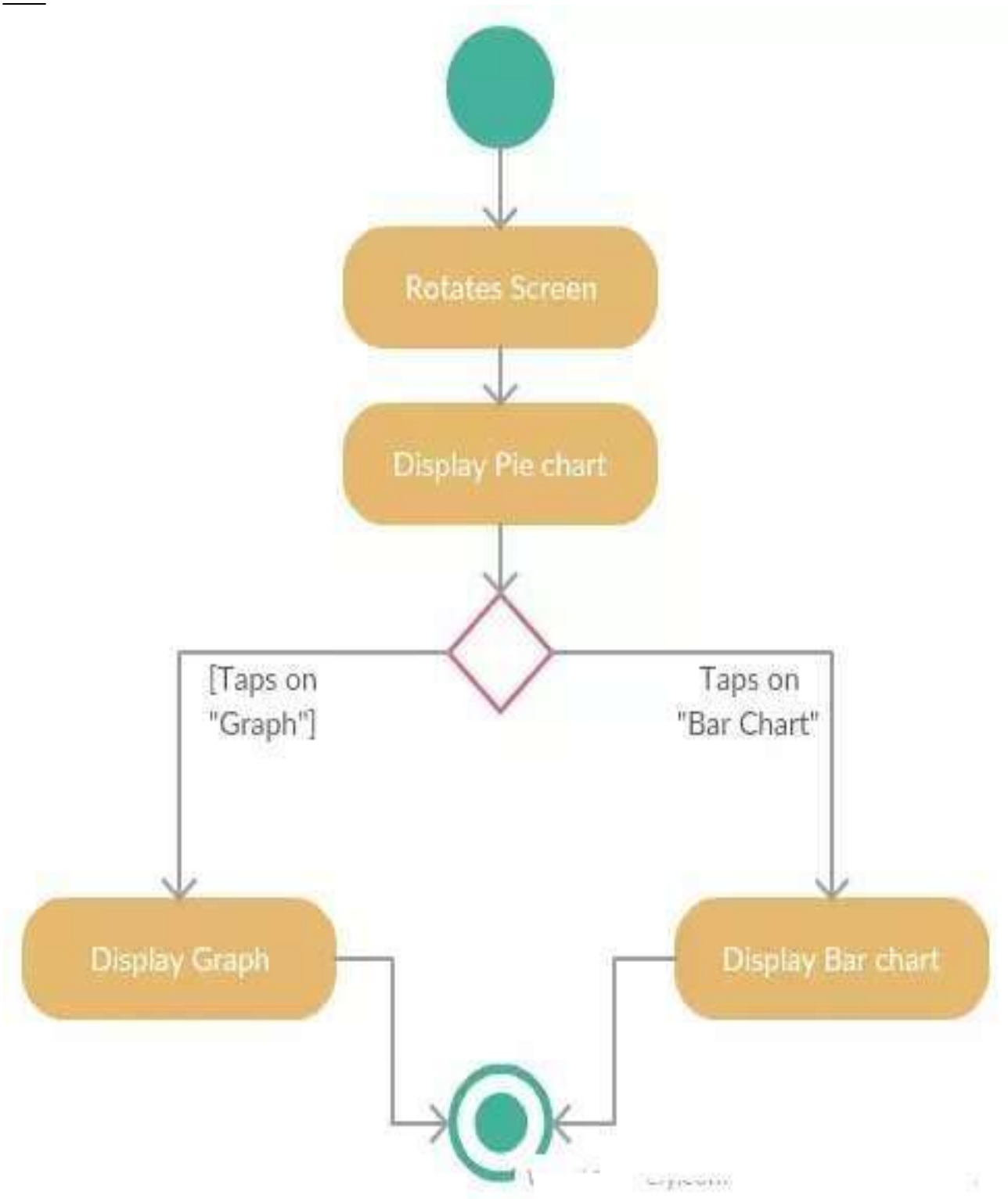
- " " Open account
tab • • • • EntwS
account nan›P

5tc•re in dâtahase

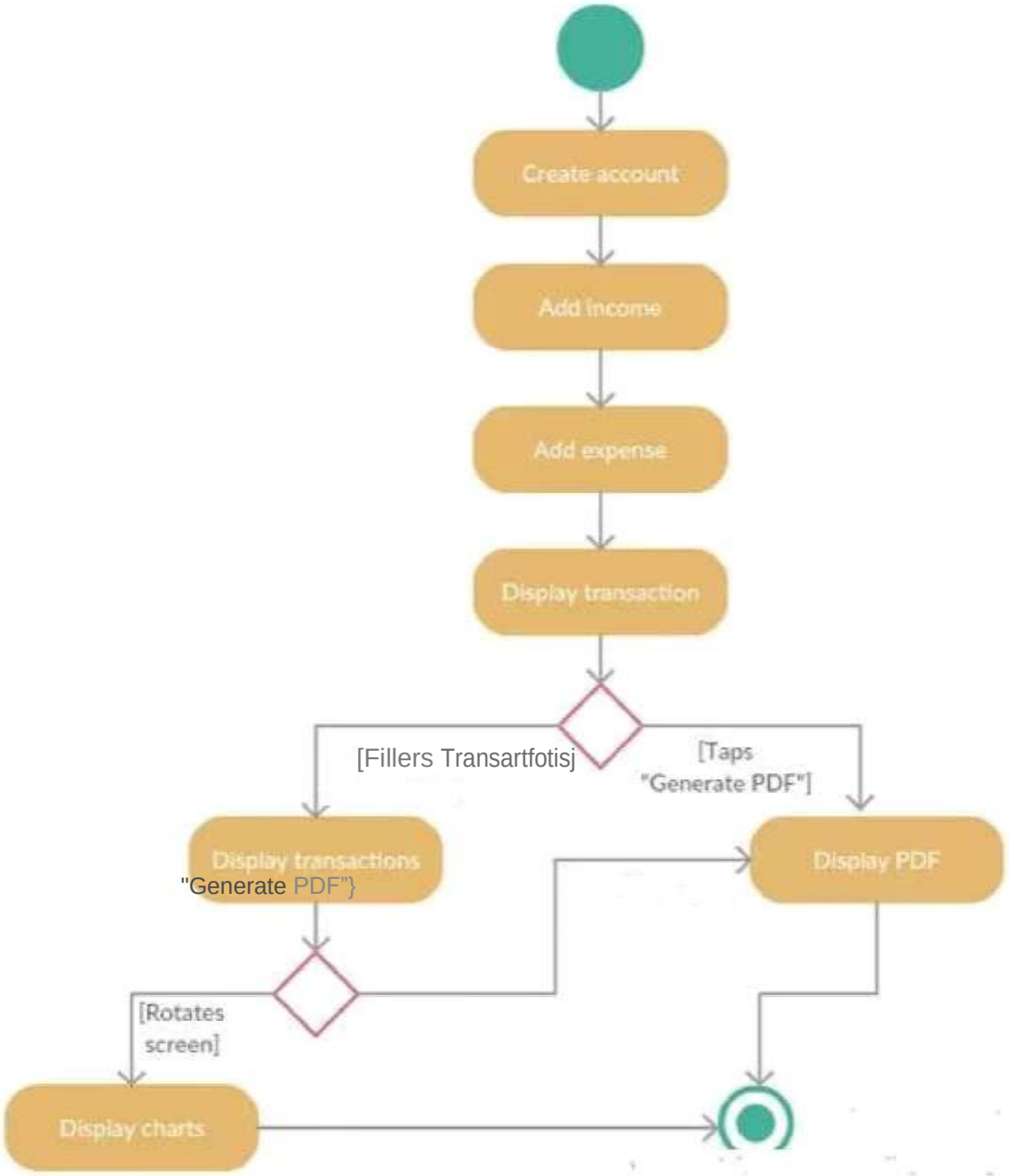
Stores in database

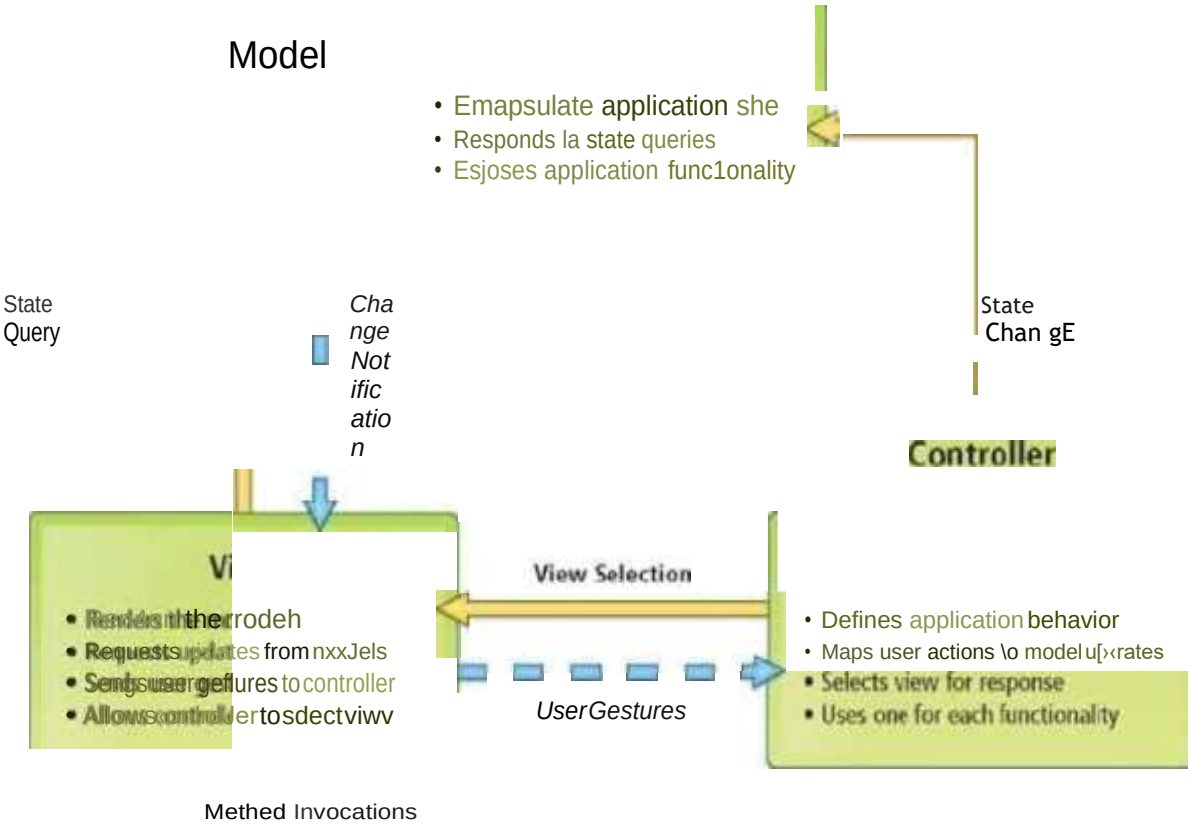
' Disp1Jyttuccessulmtssag'e.--


User
UI.System
Database
RequesfE Create a c co«nt
Disp
lays form
F I\ form
Store in database
Stores in database
Request add income
Stole in d‹›cab se
Pguestto add n«inso
Stores in ‹lotabase
Store •n database
Requests view transactions
Storrs in databa.e
View transactions
Displays transactions
Requests PDF
Ask for data
Send data
Requests Graphs
Cencrate PDF
Aok}ordao
Displays graphs

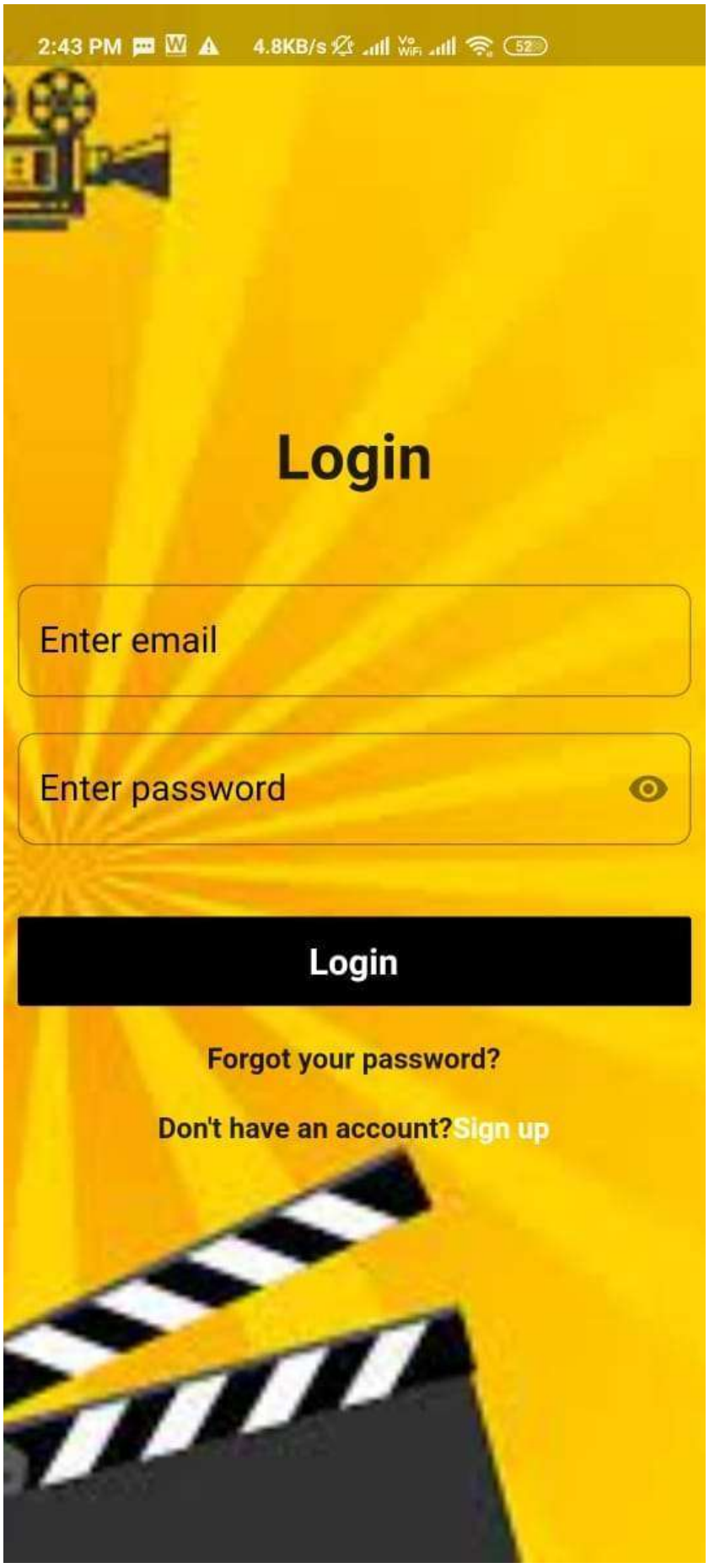


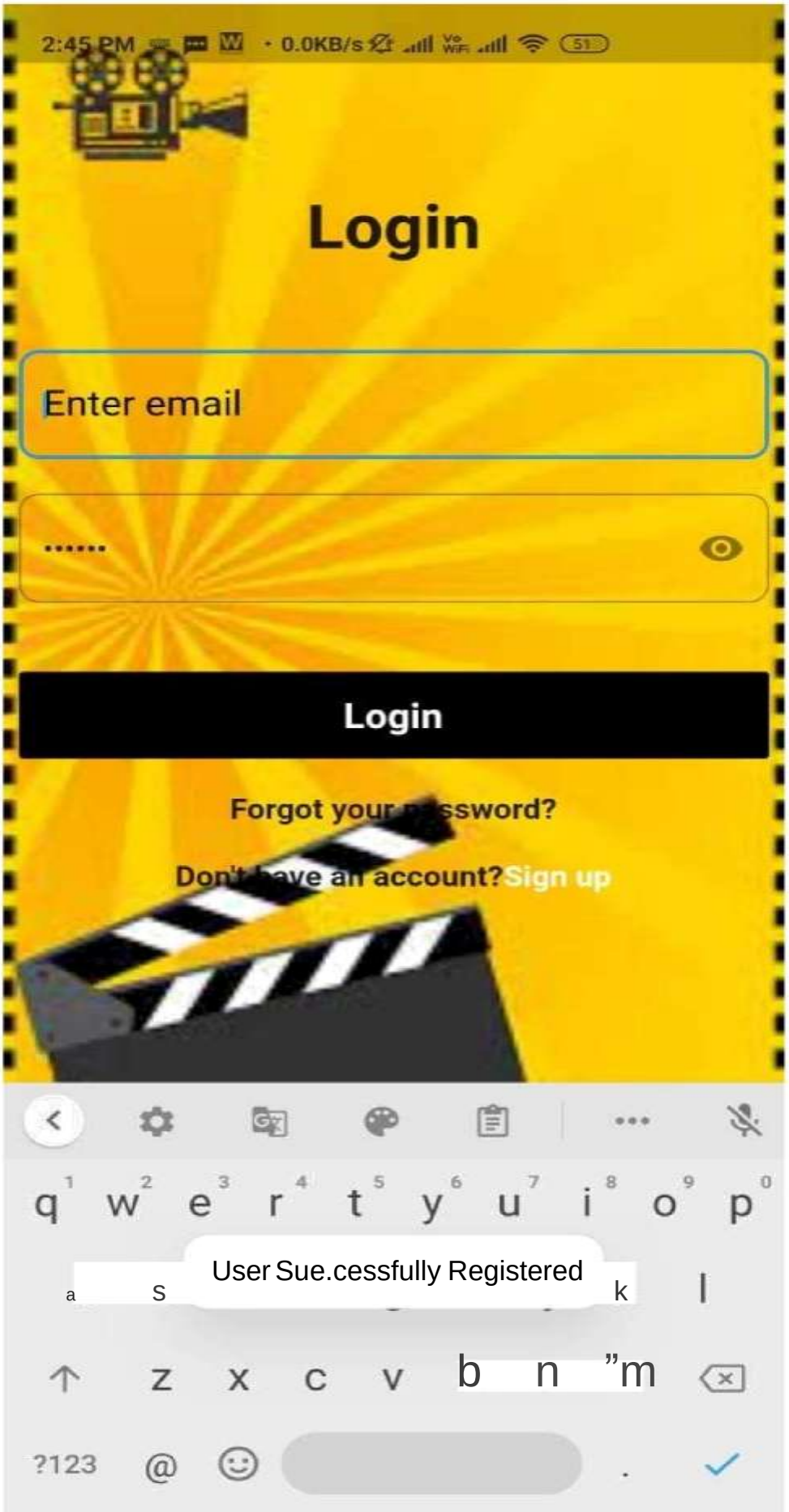
1.5 Activity Diagram


Create account
Add income
Add expense
Display transaction
[Fillers Transartfotisj
[Taps
"Generate PDF"]
Display transactions
"Generate PDF"}
Display PDF
[Rotates
screen]
Display charts


Model
• Emapsulate application she
• Responds la state queries
• Esjoses application func1onality
State
Query
Cha
nge
Not
ific
atio
n
State
Chan gE
Controller
View Selection
• Defines application behavior
• Maps user actions \o model u[›‹rates
• Selects view for response
• Uses one for each functionality
UserGestures
Methed Invocations

CHAPTER 3
Form Design Input/Output Form(Screens)


2:43 PM
4.8KB/s
Login
Enter email
Enter password
Login
Forgot your password?
Don't have an account?Sign up


2:45 PM
0.0KB/s
Login
Enter email
Login
Forgot your password?
Don't have an account?Sign up
User Sue.cessfully Registered

CHAPTER 4

Coding

Main.dart

```
impor'package:delivery_boy_oc/Utils/AppsharedPref.dart';
import'package:delivery_boy_oc/View/HomePage.dart';
import'package:delivery_boy_oc/loginpage/LoginPage.dart';
import 'package:flutter/material.dart';
import 'package:flutter_localizations/flutter_localizations.dart';

void main() {
WidgetsFlutterBinding.ensureInitialized();
  AppSharedPref.isLogin().then((v) {
    if (v) {
      runApp(HomePage("", "", "", "", "", "", false));
    } else { runApp(LoginScreen());
    }
  });
}

class LoginScreen extends StatelessWidget {

  static const String appName = "OpenCart Delivery Boy";
  @override
  Widget build(BuildContext context) { return new MaterialApp(
```

```
      title: "DeliveryBoy",
      theme: new ThemeData(primarySwatch: Colors.blue),
      routes: <String, WidgetBuilder>{'/': (BuildContext
              context) => LoginPage(appName)
      },
      debugShowCheckedModeBanner: false, localizationsDelegates: [
        // ... app-specific localization delegate[s] here
        GlobalMaterialLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
      ],
      supportedLocales: [
        const Locale('en', ''), // English
        // ... other locales the app supports
      ],
    );
  }
}
```

Login.dart

```
import 'dart:convert';

import 'package:delivery_boy_oc/AppConstant/AppConstant.dart'; import
'package:delivery_boy_oc/Callback/GetResponse.dart'; import
```

```
'package:delivery_boy_oc/Helper/Constant.dart';
import 'package:delivery_boy_oc/Helper/Method.dart';
import 'package:delivery_boy_oc/Helper/MobikulTheme.dart';
import 'package:delivery_boy_oc/NetworkManger/NetworkCall.dart';
import'package:delivery_boy_oc/Utils/AppsharedPref.dart';
import 'package:delivery_boy_oc/View/AddDeliveryBoy.dart';
import 'package:delivery_boy_oc/View/HomePage.dart';
import 'package:delivery_boy_oc/loginpage/LoginResponseModel.dart';
import'package:firebase_messaging/firebase_messaging.dart';
import 'package:flutter/material.dart';
import'package:http/http.dart'ashttp;
import 'package:provider/provider.dart';

const TAG = "LoginPage";

class LoginPage extends StatelessWidget implements GetResponse{

  LoginPage(this.title);

  final String title;
  static const String userNameHint = "Email"; static const String
  passwordHint = "Password"; static const String signInHint = "Login";
  static const String enterYourEmail = "Enter Your Email"; static const
  String forgetPasswordTitle = "Forget Password";
  final userNameController = TextEditingController(text: Constant.isDemo
  ?
Constant.demoAdminEmail : "");
  final passWordController = TextEditingController(text: Constant.isDemo
```

```
  ?
Constant.demoAdminPassword : "");
  final forgetPassWordController = TextEditingController(); final
  _loginFormKey = GlobalKey<FormState>(); BuildContext _context;

  LoginResponseModel loginResponseModel = LoginResponseModel();

  @override
  Widget build(BuildContext context) {

    _context = context;
    final userNameField = Container(
      margin: EdgeInsets.fromLTRB(Constant.spacing_normal,
Constant.spacing_small, Constant.spacing_normal, Constant.spacing_zero),
      child: TextFormField(
        keyboardType: TextInputType.emailAddress, autofocus: false,
        controller: userNameController, validator: (value) {
          if (value.isEmpty) {
            return "$userNameHint can't be empty";
          }
          return null;
        },
        autovalidateMode: AutovalidateMode.disabled, decoration:
        InputDecoration(
          labelText: userNameHint, border: OutlineInputBorder(),
          prefixIcon: Icon(Icons.email),
          contentPadding: EdgeInsets.symmetric(vertical:
```

```
Constant.spacing_normal, horizontal: Constant.spacing_tiny),
        ),
      ),
    );

    final signInButton = Container(
      margin:                    EdgeInsets.fromLTRB(Constant.spacing_normal,
Constant.spacing_small, Constant.spacing_normal, Constant.spacing_tiny),
     width: double.infinity, child: ElevatedButton( onPressed: () {
           if          (_loginFormKey.currentState.validate())          {
             onClickLogin(context);
           }
         },
         style:                          ElevatedButton.styleFrom(primary:
         MobikulTheme.accentColor), child: Text(
           signInHint.toUpperCase(),
         )),
    );

    final signUpButton = Container(
      margin:                    EdgeInsets.fromLTRB(Constant.spacing_normal,
Constant.spacing_tiny, Constant.spacing_normal, Constant.spacing_tiny),
     width: double.infinity, child: OutlinedButton( onPressed: () {
     onSignUpButtonClick(context);
         },
         style:                 OutlinedButton.styleFrom(          primary:
            MobikulTheme.accentColor, side: BorderSide(
```

```
            color: MobikulTheme.accentColor,
          )),
        child: Text( AppConstant.createAccount.toUpperCase(),
        )),
    );

    final forgetPassword = new TextButton( onPressed: () {
        showDialog(
            context: context,
            builder: (BuildContext context) { return AlertDialog(
              title: new Container(
                margin: EdgeInsets.fromLTRB(8.0, 8.0, 8.0, 8.0),
                padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0),
                child: new Center(
                  child: new Column(
                    mainAxisAlignment: MainAxisAlignment.spaceEvenly,
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: <Widget>[
                      new Text( AppConstant.FORGET_PASSWORD,
                        style: TextStyle(fontWeight: FontWeight.bold,
fontSize: 16),
                      ),
                      new Container(
                        padding: EdgeInsets.fromLTRB(0, 20, 0, 0),
                        child: new TextFormField(
                          controller: forgetPassWordController,
                        ),
```

```
                    ),
                  ],
                ),
              ),
            ),
          ),
          actions: <Widget>[ new TextButton(
                onPressed: () { Navigator.of(context).pop();
                },
                child: new Text("Close")), new TextButton(
                onPressed: () { Navigator.pop(context);
                  makeForgetPassCall(forgetPassWordController,
context);
                },
                child: new Text("Ok"))
          ],
        );
      });
    },
    child: new Row(
      mainAxisAlignment: MainAxisAlignment.end, children: <Widget>[
        new Text( forgetPasswordTitle, textAlign: TextAlign.end,
          style: new TextStyle(fontSize: 14.0, color:
Colors.blue[600], decoration: TextDecoration.underline),
        )
      ],
    ));
```

```
return ChangeNotifierProvider<LoginResponseModel>( create: (context)
  => loginResponseModel,
  child: Consumer<LoginResponseModel>(
    builder: (context, model, child) => Scaffold( body: Scaffold(
      body: SafeArea( child: Container(
        height: double.infinity, child:
        SingleChildScrollView(
          physics: BouncingScrollPhysics(parent:
AlwaysScrollableScrollPhysics()),
          child: model.loading
            ? Container(child: Center(child:
CircularProgressIndicator()), height: MediaQuery.of(context).size.height,
width: MediaQuery.of(context).size.width,)
            : Column(
          mainAxisAlignment: MainAxisAlignment.center, children:
          <Widget>[
            Container(
              margin: EdgeInsets.fromLTRB(8,
Constant.spacing_large, 8, 20),
              padding: EdgeInsets.all(4), child: Text(
                title,
                style: TextStyle(fontSize: 26, fontWeight:
FontWeight.bold, color: Colors.blueAccent),
              ),
            ),
            Form(
              key: _loginFormKey, child: Column(
```

```
children: [ userNameField, Container(
margin:
EdgeInsets.fromLTRB(Constant.spacing_normal, Constant.spacing_small,
Constant.spacing_normal, Constant.spacing_zero),
child: TextFormField(
obscureText: !model.showPassword, autofocus:
false,
controller: passWordController, validator:
(value) {
if (value.isEmpty) {
return "$passwordHint can't be empty";
}
return null;
},
autovalidateMode:
AutovalidateMode.disabled,
decoration: new InputDecoration(
labelText: passwordHint, border:
OutlineInputBorder(), prefixIcon: Icon(Icons.lock), contentPadding:
EdgeInsets.symmetric(vertical: Constant.spacing_normal, horizontal:
Constant.spacing_tiny),
suffixIcon: IconButton( onPressed: () {
model.showPassword =
!model.showPassword;
},
icon: Icon(model.showPassword ?
Icons.visibility : Icons.visibility_off),
), // hintText: passwordHint,
```

```
    ),
),
                    ),
                    forgetPassword, signInButton, signUpButton,
                ],
            ),
        ),
        Constant.isDemo
            ? Container( child: Column(
            children: <Widget>[ Container(
                padding: EdgeInsets.all(8), margin:
                EdgeInsets.all(12), child: Text(
                  "To Login as Delivery boy try the below
Credentials",
                  style: TextStyle(fontWeight:
FontWeight.bold, fontSize: 18),
                ),
              ),
              Container(
                padding: EdgeInsets.fromLTRB(12, 4, 12, 12),
                child: Column(
                  children: <Widget>[ Text('Email:
${Constant.demoDeliveryBoyEmail}\nPassword:
${Constant.demoDeliveryBoyPassword}', style: TextStyle(fontWeight:
FontWeight.bold, fontSize: 18, color: Colors.blue[600])),
                ],
              ),
```

```
                        )
                      ],
                    ),
                  )
                      : Container(),
                ],
              ),
            )),
          ),
        ),
      ),
    );

  }

  onClickLogin(BuildContext context) async { loginResponseModel.loading
    = true; FirebaseMessaging().getToken().then((firebaseToken) {
      debugPrint(TAG + " : Firebase token - " + firebaseToken);
      AppSharedPref.getToken().then((v) { debugPrint(TAG + " : WK token -
        " + v); Map body = {'wk_token': v, 'width':
MediaQuery.of(context).size.width.toString(), 'username':
userNameController.text, 'password': passWordController.text,
'device_token': firebaseToken};
        debugPrint(TAG + "Body---->" + body.toString());
        ApiCall.makeCall(Method.POST, Constant.USER_LOGIN, body, this);
```

```
      }).catchError((err) {
        debugPrint(TAG + "error:-------->" + err.toString());
      });
    }).whenComplete(() {
      debugPrint(TAG + " : Firebase token - Null");
    });
  }

  onSignUpButtonClick(BuildContext context) { Navigator.push(context,
    MaterialPageRoute(builder: (context) =>
AddDeliveryBoy(true)),).then((value) {
      if(value!=null && value is String) { print("NEW TEST LOG --> " +
        value); Map res = json.decode(value);
        int approveStatus = res['approve_status'] ?? 0;
        if(approveStatus==1) {
          getResponse(value);
        }
      }
    });
  }

  @override
  void getResponse(String response) { Map res = json.decode(response);
    if (res['error'] != 1 && res['fault'] != 1) {
      Navigator.of(_context).pushReplacement(new
      MaterialPageRoute(builder:
(context) => HomePage(res['email'], res['id'], res['image'], res['name'],
```

```
res['status'], res['user_type'], true)));
    } else if (res['error'] == 1) { loginResponseModel.loading = false;

      showDialog(
          context: _context,
          builder: (BuildContext context) { return AlertDialog(
              title: new Container(
                child: new Text(res['message']),
              ),
              actions: <Widget>[ new TextButton(
                    onPressed: () {
                      Navigator.of(context, rootNavigator: true).pop();
                    },
                    child: new Text("OK"))
              ],
            );
          });
    }
  }

  void makeForgetPassCall(TextEditingController
forgetPassWordController, BuildContext context) {
    AppSharedPref.getToken().then((token) {
      Map body = {"wk_token": token, "username":
forgetPassWordController.text};
      loginResponseModel.loading = true;
```

```
      debugPrint("ForgetResponse" + "=====>" + body.toString());
      http.post(Constant.BASE_URL + Constant.FORGET_PASSWORD, body:
json.encode(body)).then((v) {
        debugPrint("ForgetResponse" + "=====>" + v.body);
        loginResponseModel.loading = false;

        Map res = json.decode(v.body); if (res['fault'] == 1) {
          makeApiLoginCall(forgetPassWordController.text,
            token).then((res) {
            makeForgetPassCall(forgetPassWordController, context);
          });
        } else {
//          if (res['error'] == 1) { forgetPassWordController.clear();
          showDialog(
              context: context,
              builder: (BuildContext context) {
            return AlertDialog(title: Text(res['message']), actions:
<Widget>[
                  new TextButton( onPressed: () {
                      Navigator.of(context).pop();
                    },
                    child: new Text( "Ok",
              style: TextStyle(fontSize: 14, fontWeight:
FontWeight.bold),
                    ))
                ]);
```

```
                });
 //           }
}
});
    });
  }

 }
```

deliveryBoyList.dart

```
import 'dart:async';
 import 'dart:convert';
import 'dart:developer';
import 'package:delivery_boy_oc/AppConstant/AppConstant.dart';
import'package:delivery_boy_oc/Callback/GetResponse.dart';
import 'package:delivery_boy_oc/Helper/Constant.dart';
import'package:delivery_boy_oc/Helper/MobikulTheme.dart';
import 'package:delivery_boy_oc/Model/BaseModel.dart'; import
'package:delivery_boy_oc/Model/DeliveryBoyModel/DeliveryboyModel.dart'
;
import'package:delivery_boy_oc/NetworkManger/NetworkCall.dart';
import'package:delivery_boy_oc/Utils/AppsharedPref.dart';
import'package:delivery_boy_oc/View/AddDeliveryBoy.dart';
import'package:delivery_boy_oc/View/DeliveryBoyDetails.dart';
import 'package:flutter/foundation.dart';
import'package:flutter/material.dart';
import 'package:http/http.dart' as http;
```

```
import 'package:liquid_pull_to_refresh/liquid_pull_to_refresh.dart';

const TAG = "DeliveryBoyFragment";

class DeliveryBoyListFragment extends StatefulWidget { bool fromOrder;
  String orderId; bool userType;
  bool automaticOrderAssign;
  GlobalKey<ScaffoldState> _homeScaffoldKey;

  DeliveryBoyListFragment(this.fromOrder, this.orderId, this.userType,
this.automaticOrderAssign, this._homeScaffoldKey);

  @override
  DeliveryBoy createState() => new DeliveryBoy(fromOrder, orderId,
userType, automaticOrderAssign, _homeScaffoldKey);
}

int page = 1; int limit = 5;
BuildContext mContext; GetResponse responseData;

class DeliveryBoy extends State<DeliveryBoyListFragment> implements
GetResponse {
  String token; bool fromOrder; bool deliveryBoy; String orderId;
  String showErrorText;
  bool automaticOrderAssign; GlobalKey<ScaffoldState> _homeScaffoldKey;

  bool isApiCallComplete = false;
  bool isDeliveryBoyApprovedApiLoading = false; bool isLongPressEnabled
```

```
= false;
List<DeliveryboyModel> deliveryBoyApiResponseList = List.empty(growable: true);

// List<DeliveryboyModel> model =new List<DeliveryboyModel>();

ScrollController _scrollController = ScrollController();
final GlobalKey<LiquidPullToRefreshState> _refreshIndicatorKey = GlobalKey<LiquidPullToRefreshState>();

DeliveryBoy(this.fromOrder, this.orderId, this.deliveryBoy, this.automaticOrderAssign, this._homeScaffoldKey);

@override
void dispose() { fromOrder = false; deliveryBoy = false;
  super.dispose();
}

@override
void initState() {

  AppSharedPref.getToken().then((v) { setState(() {
      token = v; fetchDeliveryBoy(token).then((value) {
        setState(() { deliveryBoyApiResponseList = value;
          isApiCallComplete = true;
        });
      });
      debugPrint(TAG + "-----getToken------>" + v);
```

```
      });
    });
    page;
    _scrollController.addListener(() {
      if(_scrollController.position.pixels ==
 _scrollController.position.maxScrollExtent)//_fetchFive();
debugPrint('get more data');
}
});
    super.initState();
  }

  List<DeliveryboyModel> selectionList = List.empty(growable: true);

  @override
  Widget build(BuildContext context) { responseData = this;
    mContext = context; return new Scaffold(
        appBar: getAppBar(),  body: getViewByApiState(),
        floatingActionButton: (deliveryBoy || fromOrder)
            ? Container()
            : FloatingActionButton( onPressed: () {
            Navigator.of(context).push(MaterialPageRoute(
              builder:                    (context)                    =>
            AddDeliveryBoy(false))).then((value) { setState(() {
              isApiCallComplete = false;
            });
            _handleRefresh();
```

```
        });
      },
      child: Icon(Icons.add),
    ));
}

Widget getAppBar() {
  if (fromOrder) { return AppBar(
    title: new Text(AppConstant.DELIVERY_BOY), leading: new
    IconButton(
      icon: new Icon(Icons.arrow_back), onPressed: () {
        Navigator.of(context).pop();
      }),
  );
  } else {
    if (isLongPressEnabled) { return AppBar(
      title: new Text('${selectionList.length} Selected'), leading:
      new IconButton(
        icon: Icon( Icons.close,
          color: Colors.white,
        ),
        onPressed: () => removeSelection()), actions: <Widget>[
      IconButton(
        icon: Icon(Icons.select_all), tooltip: 'Select All',
        onPressed: () => selectAll(),
      ),
      IconButton(
```

```
icon: Icon( Icons.pending_actions, color: Colors.white,
),
onPressed: () {
  setState(() { isDeliveryBoyApprovedApiLoading = true;
  });
  approveDeliveryBoy(token,
  filterDeliveryBoyId().toString())
      .then((value)                                    {
    ScaffoldMessenger.of(context).showSnackBar(SnackBar(
      content: Text( '${value.message}',
        style: TextStyle(color: Colors.white),
      ),
      backgroundColor: Colors.black87,
    ));
    setState(()  {  isDeliveryBoyApprovedApiLoading = false;
      if (value.error == 0) {
        selectionList.clear(); page = 1;
        fetchDeliveryBoy(token).then((value)  {  setState(()
          {
            isLongPressEnabled              =              false;
            deliveryBoyApiResponseList        =        value;
            isApiCallComplete = true;
          });
        });
      }
    });
```

```
            });
          })
      ],
    );
  } else {
    return AppBar(
      title: new Text(AppConstant.DELIVERY_BOY), leading: new
      IconButton(
          icon: new Icon(Icons.menu,
            color: Colors.white,
          ),
          onPressed: () {
            _homeScaffoldKey.currentState.openDrawer();
          }),
    );
  }
 }
}

Widget getViewByApiState() { if (isApiCallComplete) {
  if (showErrorText == null) { return Stack(
    children: [ createListView(deliveryBoyApiResponseList),
      Visibility(
          visible: isDeliveryBoyApprovedApiLoading, child: Center(
            child: CircularProgressIndicator(),
```

```
            ))
        ],
      );
    } else {
      return Center( child: new Text(
          showErrorText,
          textAlign: TextAlign.center, style: TextStyle(
              fontSize: 16,
              fontWeight:              FontWeight.bold,       color:
              Colors.grey[600]),));

    }
  } else {
    return Center(
      child: CircularProgressIndicator(),
    );
  }
}

Widget createListView(data) {
  //contoller:_scrollController; List<DeliveryboyModel> model = data;
  return LiquidPullToRefresh(
    key:   _refreshIndicatorKey,  showChildOpacityTransition:   false,
    onRefresh: _handleRefresh,
    child: AnimatedList(
      key:        GlobalKey<AnimatedListState>(),    initialItemCount:
```

```
      model.length,
      itemBuilder: (BuildContext context, int index, Animation
        animation) { return _buildItem(model[index], animation);
        // if (index == model.length) {
        //  return Center(child: CircularProgressIndicator());
        // }else {
        //
        // }
      },
    ), // scroll view
  );
}

void manageLongClick(DeliveryboyModel model) { setState(() {
    if (selectionList.contains(model)) { model.isSelected = false;
      selectionList.remove(model);
    } else {
      model.isSelected = true; selectionList.add(model);
    }
    if (selectionList.length == 0) { isLongPressEnabled = false;
    }
  });
}

Widget _buildItem(DeliveryboyModel model, Animation _animation) { var
  isSelected = <bool>[
```

```
    model.isAccountApproved(),
    !model.isAccountApproved()
  ];
  if (model.isAccountApproved()) { isSelected = <bool>[true];
  }
  var screenSize = MediaQuery.of(context).size; return SizeTransition(
    sizeFactor: _animation, child: Card(
      color: model.isSelected ? Colors.grey[300] : Colors.white, shape:
      RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(8),
      ),
      margin: EdgeInsets.all(8), child: InkWell(
        onTap: () {
          if (fromOrder) {
            AppSharedPref.getToken().then((v) {
              assignOrder(v, model.id, orderId, automaticOrderAssign);
            });
          } else {
            if (isLongPressEnabled) manageLongClick(model);
            else
              Navigator.of(context).push(MaterialPageRoute(      builder:
                  (context) => new
DeliveryBoyDetails(model.id))).then((value)  {
                print("THIS ONE IS FROM POP-POP. $value");
                setState(() { isApiCallComplete = false;
                });
                _handleRefresh();
```

```
        });
      }
    },
    onLongPress: () { if (!fromOrder) {
      isLongPressEnabled = true; manageLongClick(model);
      }
    },
    child: Padding(
      padding: const EdgeInsets.all(8.0), child: Row(
        children: [ Container(
          width: screenSize.width / 5, height: screenSize.width /
          5, decoration: BoxDecoration(
            shape: BoxShape.circle,
            image: DecorationImage(
      image: NetworkImage(model.image), fit:
BoxFit.fill),
            color: MobikulTheme.accentColor),
        ),
        Padding(
          padding: const EdgeInsets.only(left: 10), child:
          Container(
            width: screenSize.width / 1.5, child: Column(
              crossAxisAlignment: CrossAxisAlignment.start, children:
              [
                Row(
                  mainAxisAlignment:
```

```
MainAxisAlignment.spaceBetween, children: [
  Flexible(
    child: Text(model.name, maxLines: 1,
        overflow:  TextOverflow.ellipsis, style:
        TextStyle(
            fontWeight: FontWeight.bold, fontSize:
            16)),
  ),
  Container(
    margin:       EdgeInsets.all(4),    alignment:
    Alignment.bottomRight,                  width:
    screenSize.width / 30, height: screenSize.width
    / 30, decoration: new BoxDecoration(
      color: model.delivery_status == "1"
          ? Colors.green
          : Colors.grey, shape: BoxShape.circle,
    ),
  ),
 ],
),
getText(model.email,          isBold:          true),
getText(model.getMobileString(context)),
getText(model.getVehicleTypeString(context)),
Padding(
  padding:  const  EdgeInsets.only(top:  4), child:
  Row(
    children: [ Padding(
```

```
                padding: const EdgeInsets.only(right: 10),
                child: getText(AppConstant.accountStatus),
              ),
              ToggleButtons(
                selectedColor: model.isAccountApproved()
                    ? Colors.green
                    : Colors.redAccent,
            selectedBorderColor:
model.isAccountApproved()
                    ? Colors.green
                    : Colors.redAccent,
                fillColor: model.isAccountApproved()
                    ? Colors.green.withOpacity(0.08)
                    :     Colors.redAccent.withOpacity(0.08),
                splashColor: model.isAccountApproved()
                    ? Colors.green.withOpacity(0.12)
                    :     Colors.redAccent.withOpacity(0.12),
                hoverColor: model.isAccountApproved()
                    ? Colors.green.withOpacity(0.04)
                    :     Colors.redAccent.withOpacity(0.04),
                borderRadius: BorderRadius.circular(4.0),
                constraints:        BoxConstraints(minHeight:
                20.0), isSelected: isSelected,
                onPressed: (index) { if (!fromOrder &&
!model.isAccountApproved() && index==0 && !isLongPressEnabled) {
                    print("Toggle Button index : $index");
```

```
                                setState(() {
                                  isDeliveryBoyApprovedApiLoading =
                                  true;          selectionList.clear();
                                  selectionList.add(model);
                                  approveDeliveryBoy(token,
filterDeliveryBoyId().toString()).then((value) {

ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                                    content: Text( '${value.message}',
                                      style: TextStyle(
                                          color: Colors.white),
                                    ),
                                    backgroundColor: Colors.black87,
                                  ));
                                  setState(() {
                                    isDeliveryBoyApprovedApiLoading =
false;
                                    if (value.error == 0) {
                                      model.approve_status = '1';
                                    }
                                    selectionList.clear();
              });
            });
          });
}
                              },
                              children:
```

```
  getToggleButtons(isSelected.length),
)
                          ],
                        ),
                      )
                    ],
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
      clipBehavior: Clip.antiAlias,
    ),
  );
}

List<Widget> getToggleButtons(int count) { if (count == 2) {
    return [ Padding(
      padding: EdgeInsets.symmetric(horizontal: 10.0, vertical: 4.0),
      child: Text(
        AppConstant.approved,
      ),
    ),
```

```
      Padding(
        padding: EdgeInsets.symmetric(horizontal: 10.0, vertical: 4.0),
        child: Text(AppConstant.unapproved,),
      ),
    ];
  } else {
    return [ Padding(
        padding: EdgeInsets.symmetric(horizontal: 10.0, vertical: 4.0),
        child: Text(
          AppConstant.approved,
        ),
      ),
    ];
  }
}

Widget getText(String text, {bool isBold = false}) => Text( text,
  overflow: TextOverflow.ellipsis, style: TextStyle(
      fontSize: 13,
      fontWeight: isBold ? FontWeight.bold : FontWeight.normal),
);

@override
void getResponse(String response) { setState(() {
    showErrorText = response;
  });
}
```

```
Future<void> _handleRefresh() async { page = 1;
  final Completer<void> completer = Completer<void>(); var value =
  await fetchDeliveryBoy(token); isLongPressEnabled = false;
  deliveryBoyApiResponseList = value; isApiCallComplete = true;
  setState(() {});
  completer.complete();
  return completer.future.then<void>((_) {});
}

List<String> filterDeliveryBoyId() { List<String> id =
  List.empty(growable: true); for (DeliveryboyModel i in
  selectionList) {
    id.add(i.id);
  }
  return id;
}

void removeSelection() {
  for (DeliveryboyModel i in deliveryBoyApiResponseList) {
    i.isSelected = false;
  }
  setState(() { selectionList.clear(); isLongPressEnabled = false;
  });
}

void selectAll() {
```

```
    for (DeliveryboyModel i in deliveryBoyApiResponseList) {
      i.isSelected = true;
      if (!selectionList.contains(i)) selectionList.add(i);
    }
    setState(() { isLongPressEnabled = true;
    });
  }
assignOrder(String token, String id, String orderId, bool
automaticOrderAssign) async {
  Map body = { 'wk_token': token, 'order_id': orderId, 'id': id,
    'assign_status': automaticOrderAssign ? '1' : '0'
  };

  debugPrint(
      TAG + "-------URL-------->" + Constant.BASE_URL +
Constant.ASSIGN_ORDER);  http.Response response = await http
      .post(Constant.BASE_URL + Constant.ASSIGN_ORDER, body:
json.encode(body));
  debugPrint(TAG + "-------Body-------->" + body.toString()); Map res =
  json.decode(response.body);
  debugPrint(TAG + "------Response------>" + response.body); if
  (res['fault'] != 1) {
    Navigator.pop(mContext, res['message']);
  } else { AppSharedPref.getEmailId().then((email) {
      makeApiLoginCall(email, token).then((v) {
        AppSharedPref.getToken().then((res) {
          assignOrder(token, id, orderId, automaticOrderAssign);
```

```
        }
        );

      }
      );

    }
    );

  }

}
Future<List<DeliveryboyModel>> fetchDeliveryBoy(String token) async { if
  (token != null) {
    Map body = { 'wk_token': token,
      'width': MediaQuery.of(mContext).size.width.toString(), 'page': page,
      'limit': limit
    };
    debugPrint(TAG + "requestBody:------->" + body.toString());
    http.Response response = await http.post(Constant.BASE_URL +
Constant.GET_DELIVERY_BOYS, body: json.encode(body));
    debugPrint(TAG + "DeliveryBoyResponse:------->" + response.body);
    log(response.body);
    Map res = json.decode(response.body);
    if (res['fault'] != 1 && res['error'] != 1) {
      if ((res['delivery_boys'] as List).length != 0) { return
        compute(parseDeliveryBoys, response.body);
      } else {
        responseData.getResponse("No Delivery Boy Available");
      }
    } else if (res['error'] == 1) {
      responseData.getResponse(res['message']);
```

```
    } else { AppSharedPref.getEmaillId().then((email) {
        makeApiLoginCall(email, token).then((v) {
          AppSharedPref.getToken().then((res) {});
        });
      });
    }
  }
  return null;
}
Future<BaseModel> approveDeliveryBoy(String token, String idList) async {
  if (token != null) {
    Map body = {'wk_token': token, 'selected': idList};
    debugPrint(TAG + "DeliveryBoyResponse:-------->" + jsonEncode(body));
    http.Response response = await http.post(
        Constant.BASE_URL + Constant.APPROVE_DELIVERY_BOY, body:
        json.encode(body));
    debugPrint(TAG + "DeliveryBoyResponse:-------->" + response.body);
    log(response.body);
    Map res = json.decode(response.body); if (res['fault'] == 1) {
      AppSharedPref.getEmaillId().then((email) { makeApiLoginCall(email,
        token).then((v) {
          AppSharedPref.getToken().then((res) {});
        });
      });
    } else {
      return BaseModel(message: res['message'], error: res['error']);
```

```
    }
  }
  return null;
}

// fetchFive(){
//   debugPrint('get more data');
//   // for(int i=0; i<5; i++){
//   //   fetch();
  //}
  // List<DeliveryboyModel> model;
  // page = page+1;
  //if(deliveryBoyApiResponseList.)
//}

// Future<List<DeliveryboyModel>> fetch() async{
//   http.Response response = await http.post(Constant.BASE_URL +
Constant.GET_DELIVERY_BOYS,
//       // body: json.encode(body)
//   );
//   debugPrint(TAG + "DeliveryBoyResponse:------->" + response.body);
//   log(response.body);
//   Map res = json.decode(response.body);
//   if (res['fault'] != 1 && res['error'] != 1) {
//     if ((res['delivery_boys'] as List).length != 0) {
//       return compute(parseDeliveryBoys, response.body);
```

```
//     } else {
//       responseData.getResponse("No Delivery Boy Available");
//     }
//   }
// }
List<DeliveryboyModel> parseDeliveryBoys(String response) { Map res =
  json.decode(response);
  List<DeliveryboyModel> model;
  if ((res['delivery_boys'] as List).length != 0) { model = [];
    final items = (res['delivery_boys'] as List); for (final i in items)
    {
      model.add(new DeliveryboyModel( i['id'],
          i['name'],
          i['email'],
          i['status'],
          i['mobile'],      i['vehicle_type'],      i['vehicle_number'],
          i['address'],
          i['image'],
          i['delivery_status'], i['approve_status']));
    }
  }
  return model;
}
```

DeliveryModel.dart

```
import 'package:delivery_boy_oc/AppConstant/AppConstant.dart'; import 'package:delivery_boy_oc/Model/BaseModel.dart';

class DeliveryboyModel extends BaseModel{ String id;
  String name; String email; String status; String mobile;
  String vehicle_type; String vehicle_number; String address; String _image;
  String delivery_status; String approve_status;
  bool isSelected = false;  //used to manage multi-selection UI.

  String get image { if(_image == null) {
      return '';
    }
    return _image;
  }
  set image(String image) {
    _image = image;
  }

  DeliveryboyModel( this.id, this.name, this.email, this.status,
      this.mobile, this.vehicle_type,
      this.vehicle_number, this.address,

      this._image, this.delivery_status, this.approve_status);
```

```
  bool isAccountApproved() => approve_status == '1' ? true : false;

  String getMobileString(context) => '${AppConstant.mobile}: $mobile';

  String getVehicleTypeString(context) => '${AppConstant.vehicleType}:
$vehicle_type';

}
```

AppSharedPreferene.dart

```
import 'dart:async';

import 'package:shared_preferences/shared_preferences.dart';

class AppSharedPref {
  static const String WK_TOKEN = "wktoken";

  static const String USER_ID = "userid"; static const String EMAIL_ID =
  "emailid"; static const String USERNAME = "username"; static const
  String IMAGE = "image"; static const String STATUS = "status"; static
  const String USERTYPE = "userType"; static const String LOGIN =
  "login"; static const String TAG = "AppsharedPref"; static const String
  LAT = "lat";
  static const String LNG = "lng";

  static setLogin(bool login) async {
    SharedPreferences preferences = await SharedPreferences.getInstance();
    preferences.setBool(LOGIN, login);
  }
```

```
static clearSharedPref() async {
  SharedPreferences preferences = await SharedPreferences.getInstance();
  preferences.clear().catchError((err) {
    print(TAG + err);
  });
}

static Future<bool> isLogin() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); bool
  token = pref.getBool(LOGIN) ?? false;
  return token;
}

static storeToken(String wkToken) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(WK_TOKEN, wkToken);
  print(TAG + "---UpdateToken---->" + wkToken);
}

static Future<String> getToken() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(WK_TOKEN) ?? "none";
  return token;
}

static setUserId(String id) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
```

```
  pref.setString(USER_ID, id);
}

static Future<String> getUserId() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(USER_ID) ?? "none";
  return token;
}

static setUserEmaillId(String email) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(EMAIL_ID, email);
}

static Future<String> getEmaillId() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token

  pref.getString(EMAIL_ID) ?? "none";
  return token;
}

static setUserName(String userName) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(USERNAME, userName);
}
```

```
static Future<String> getUserName() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(USERNAME) ?? "none";
  return token;
}

static setUserImage(String image) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(IMAGE, image);
}

static Future<String> getUserImage() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(IMAGE) ?? "none";
  return token;
}

static Future<String> getEmail() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(EMAIL_ID) ?? "none";
  return token;
}

static setStatus(String image) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(STATUS, image);
}
```

```
static Future<String> getStatus() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(STATUS) ?? "none";
  return token;

}

static setUserType(String type) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(USERTYPE, type);
}

static Future<String> getUserType() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); String
  token = pref.getString(USERTYPE) ?? "none";
  return token;

}

static setUserLat(String lat) async {
  SharedPreferences pref = await SharedPreferences.getInstance();
  pref.setString(LAT, lat);
}

static Future<String> getUserLat() async {
  SharedPreferences pref = await SharedPreferences.getInstance(); return
  pref.getString(LAT);
```

```
  }

  static setUserLng(String lng) async {
    SharedPreferences pref = await SharedPreferences.getInstance();
    pref.setString(LNG, lng);
  }

  static Future<String> getUserLng() async {
    SharedPreferences pref = await SharedPreferences.getInstance(); return
    pref.getString(LNG);
  }
}
```

Constant.dart

```
class Constant {

//  static const String BASE_URL =
"https://octest.webkul.com/mobikul/dboy_3.x/";      //Test Server
  static const String BASE_URL =
  "https://octest.webkul.com/mobikul/dboy/";
//Test Server New Features
// static const String BASE_URL = "http://192.168.15.160/opencart-
2.3.0.2/";
//Local Test Server New Features
// static const String BASE_URL =
"https://oc.webkul.com/mobikul/Network/";
//Live Demo Server
  static const String API_KEY = "admin"; static const String API_PASSWORD
  = "admin"; static const int timeout = 200;
```

```
static const bool isDemo = true;
static const String demoAdminEmail = "admin@webkul.com"; static const String demoAdminPassword = "webkul12#";
static const String demoDeliveryBoyEmail = "johndoe@webkul.com"; static const String demoDeliveryBoyPassword = "webkul12#";

static const String FCM_TOPIC = "opencart_delivery_boy";

static const double spacing_large = 32; static const double spacing_normal = 16; static const double spacing_small = 8; static const double spacing_tiny = 4; static const double spacing_zero = 0;

static const String USER_LOGIN = "?route=api/wkrestapi/deliveryboy/userLogin";
static const String API_LOGIN = "?route=api/wkrestapi/deliveryboy/apiLogin";
static const String FORGET_PASSWORD = "?route=api/wkrestapi/deliveryboy/forgotPassword";
static const String ACCOUNT_INFO = "?route=api/wkrestapi/deliveryboy/myAccount";
static const String USER_LOGOUT = "?route=api/wkrestapi/deliveryboy/userLogout";
static const String GET_DELIVERY_BOYS = "?route=api/wkrestapi/deliveryboy/getAllDeliveryBoy";
static const String ADD_DELIVERY_BOYS = "?route=api/wkrestapi/deliveryboy/addDeliveryBoy";
static const String GET_DELIVERY_BOY_INFO = "?route=api/wkrestapi/deliveryboy/getDeliveryBoy";
```

```
  static const String EDIT_DELIVERY_BOY_INFO =
"?route=api/wkrestapi/deliveryboy/editDeliveryBoy";
  static const String VALIDATE_EMAIL_ADDRESS =
"?route=api/wkrestapi/deliveryboy/checkEmail";
  static const String DELETE_DELIVERY_BOY =
"?route=api/wkrestapi/deliveryboy/deleteDeliveryBoy";
  static const String UPLOAD_IMAGE =
"?route=api/wkrestapi/deliveryboy/uploadImage";
  static const String GET_ORDERS =
"?route=api/wkrestapi/deliveryboy/getOrders";
  static const String GET_ORDER_DETAILS =
"?route=api/wkrestapi/deliveryboy/getOrder";
  static const String ASSIGN_ORDER =
"?route=api/wkrestapi/deliveryboy/assignOrder";
  static const String UPDATE_ONLINE_STATUS =
"?route=api/wkrestapi/deliveryboy/changeBoyStatus";
  static const String CHART_API =
"?route=api/wkrestapi/deliveryboy/getGraphData";
  static const String CONFIRM_ORDER =
"?route=api/wkrestapi/deliveryboy/confirmOrder";
  static const String SET_TRACK_DATA =
"?route=api/wkrestapi/deliveryboy/setTrackData";

  static const String ADMIN_PERMISSION =
"?route=api/wkrestapi/deliveryboy/getPermissions";

  static const String CHANGE_PASSWORD =
"?route=api/wkrestapi/deliveryboy/changePassword";
  static const String ACCEPT_ORDER =
"?route=api/wkrestapi/deliveryboy/acceptOrder";
```

static const String DECLINE_ORDER = "?route=api/wkrestapi/deliveryboy/declineOrder";

static const String APPROVE_DELIVERY_BOY = "?route=api/wkrestapi/deliveryboy/approveDeliveryBoy";

static const String PENDING = "pending"; static const String COMPLETED = "completed"; static const String PROCESSED = "processed";

static List vehicleList = ["Bike", "Cycle", "Car"];

}

AddDeliveryBoy.drt import 'dart:convert';

import 'package:delivery_boy_oc/Callback/GetResponse.dart'; import 'package:delivery_boy_oc/Helper/Constant.dart'; import 'package:delivery_boy_oc/Helper/Method.dart';

import 'package:delivery_boy_oc/NetworkManger/NetworkCall.dart'; import 'package:delivery_boy_oc/Utils/AppsharedPref.dart'; import 'package:flutter/material.dart';

import 'package:flutter/services.dart'; import 'package:http/http.dart' as http;

import 'package:image_picker/image_picker.dart';

String addDeliveryBoyTitle = "Add Delivery Boy";

String createDeliveryBoyTitle = "Create Delivery Boy Account";
BuildContext mContext;

```
const String TAG = "AddDeliveryBoy";

class AddDeliveryBoy extends StatefulWidget { final bool isFromLoginPage;

  AddDeliveryBoy(this.isFromLoginPage);

  @override
  AddDeliveryBoyState createState() => AddDeliveryBoyState(isFromLoginPage);
}

class AddDeliveryBoyState extends State<AddDeliveryBoy> implements GetResponse {
  static final firstNameController = new TextEditingController(); static final lastNameController = new TextEditingController(); static final emailController = new TextEditingController(); static final telephoneController = new TextEditingController(); static final passWordController = new TextEditingController();
  static final confirmPasswordController = new TextEditingController();
  static final vehicleController = new TextEditingController();
  static final addressController = new TextEditingController(); static var firstNameHint = "Name";
  static var addressHint = "Address"; static var emailHint = "Email";
  static var vehicleNumberHint = "Vehicle Number"; static var passwordHint = "Password";
  static var confirmPasswordHint = "Confirm Password"; static var telephoneHint = "Telephone";
```

```
static List statusList = ["Disable", "Enable"]; String signInHint =
"Submit";
String currentVehicle; String currentStatus;

bool loading = false;
String uploadImage, thumbImage; bool showPassword = false;
bool showConfirmPassword = false;

final bool isFromLoginPage;

AddDeliveryBoyState(this.isFromLoginPage);

@override
void dispose() { firstNameController.clear();
  lastNameController.clear(); emailController.clear();
  telephoneController.clear(); passWordController.clear();
  confirmPasswordController.clear(); vehicleController.clear();
  addressController.clear(); super.dispose();
}

@override initState() {
  currentStatus = statusList[0]; currentVehicle =
  Constant.vehicleList[0]; super.initState();
}

@override
Widget build(BuildContext context) { mContext = context;
  final firstName = new Container(
```

```
        margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
        padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new
        BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),
        child: new TextFormField(
          keyboardType: TextInputType.emailAddress, autofocus: false,
          controller: firstNameController,
          decoration: InputDecoration(border: InputBorder.none, hintText:
firstNameHint, contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0)),
        ));

    final telephone = new Container(
        margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
        padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new
        BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),
        child: new TextFormField( keyboardType: TextInputType.number,
          autofocus: false,
          controller: telephoneController,
          decoration: InputDecoration(border: InputBorder.none, hintText:
telephoneHint, contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0)),
        ));

    final email = new Container(
        margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
        padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new
```

```
        BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),

        child: new TextFormField(
          keyboardType: TextInputType.emailAddress, autofocus: false,

          controller: emailController,

          decoration: InputDecoration(border: InputBorder.none, hintText:
emailHint, contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0)),

        ));


    final passWordField = new Container(

      margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),

      padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new

      BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),

      child: new TextFormField(

        obscureText: showPassword ? false : true, autofocus: false,

       controller: passWordController, decoration: new InputDecoration(

        border: InputBorder.none, hintText: passwordHint,

            contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),

            suffixIcon: IconButton(

              onPressed: () {

                setState(() {

showPassword = !showPassword;

                });

              },

              icon: Icon(showPassword ? Icons.visibility :
```

```
Icons.visibility_off),
            )),
      ),
    );

final confirmPasswordField = new Container(
      margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
      padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new
      BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),
      child: new TextFormField(
        obscureText: showConfirmPassword ? false : true, autofocus: false,
        controller: confirmPasswordController, decoration: new
        InputDecoration(
            border: InputBorder.none, hintText: confirmPasswordHint,
            contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
            suffixIcon: IconButton(
              onPressed: () {
                setState(() {
                  showConfirmPassword = !showConfirmPassword;
                });
              },
              icon: Icon(showConfirmPassword ? Icons.visibility :
Icons.visibility_off),
            )),
      ),
```

```
);

final vehicleNumber = new Container(
    margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
    padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new
    BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),
    child: new TextFormField( keyboardType:

     TextInputType.text, autofocus: false,
      controller: vehicleController,
      decoration: InputDecoration(border: InputBorder.none, hintText:
vehicleNumberHint, contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0,
10.0)),
    ));

final address = new Container(
    margin: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),
    padding: EdgeInsets.fromLTRB(0.0, 4.0, 4.0, 4.0), decoration: new
    BoxDecoration(border: new Border.all(color:
Colors.blueGrey), borderRadius: new BorderRadius.all(const
Radius.circular(8.0))),
    child: new TextFormField( keyboardType: TextInputType.text,
      maxLines: 5,
      autofocus: false,
      controller: addressController,
      decoration: InputDecoration(border: InputBorder.none, hintText:
```

```
addressHint, contentPadding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0)),
      ));

  List<DropdownMenuItem<String>> getVehicleList() {
    List<DropdownMenuItem<String>> items = new List(); for (String
    vehicle in
  Constant.vehicleList) {
      items.add(new DropdownMenuItem( child: new Container(
          child: new Row(
            mainAxisSize: MainAxisSize.max, children: <Widget>[new
            Text(vehicle)],
          ),
        ),
        value: vehicle));
    }
    return items;
  }

  List<DropdownMenuItem<String>> getStatusList() {
    List<DropdownMenuItem<String>> items = new List(); for (String
    status in statusList) {
      items.add(new DropdownMenuItem( child: new Container(
          child: new Row(
            mainAxisSize: MainAxisSize.max, children: <Widget>[new
            Text(status)],
          ),
        ),
```

```
            value: status));
      }
      return items;
    }

    void updateStatusList(String value) { setState(() {
        currentStatus = value;
      });
    }

    void updateVehicleType(String value) { setState(() {
        currentVehicle = value;
      });
    }

    final statusDropDown = new DropdownButton(items: getStatusList(),
onChanged: updateStatusList, value: currentStatus);

    final vehicleType = new DropdownButton(items: getVehicleList(),
onChanged: updateVehicleType, value: currentVehicle);

    final statusDropDownContainer = new Align( alignment:
      Alignment.topLeft,
      child: new Container(
        padding: EdgeInsets.fromLTRB(20.0, 10.0, 10.0, 10.0), child: new
        Column(
          crossAxisAlignment: CrossAxisAlignment.start, mainAxisAlignment:
          MainAxisAlignment.start, children: <Widget>[
```

```
            new Text("Delivery Boy Status"), statusDropDown,
          ],
        ),
      ),
    );

    final vehicleDropDownContainer = new Align( alignment:
      Alignment.topLeft,
      child: new Container(
        padding: EdgeInsets.fromLTRB(20.0, 10.0, 10.0, 10.0), child: new
        Column(
          crossAxisAlignment: CrossAxisAlignment.start, mainAxisAlignment:
          MainAxisAlignment.start,
          children: <Widget>[new Text("Vehicle Type"), vehicleType],
        ),
      ),
    );

    final submitButton = new GestureDetector( onTap: () {
          bool isEmail(String em) { String p =
r'^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[
[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-
9]+\.)+[a-zA- Z]{2,}))$';
            RegExp regExp = new


            RegExp(p); return regExp.hasMatch(em);
          }
```

```
if (firstNameController.text.trim() == "") { showDialog(
    context: context,
    builder: (BuildContext context) { return AlertDialog(
      title: new Container(
        child: new Text("First Name is required"),
      ),
      actions: <Widget>[ new FlatButton(
          onPressed: () { Navigator.of(context).pop();
          },
          child: new Text("OK"))
      ],
    );
  });
} else if (telephoneController.text.trim() == "") { showDialog(
    context: context,
    builder: (BuildContext context) { return AlertDialog(
      title: new Container(
        child: new Text("Telephone number is required"),
      ),
      actions: <Widget>[
        new FlatButton( onPressed: () {
            Navigator.of(context).pop();
          },
          child: new Text("OK"))
      ],
```

```
        );
      });
    } else if (!isEmail(emailController.text.trim())) {
      showDialog(
        context: context,
        builder: (BuildContext context) { return AlertDialog(
          title: new Container(
            child: new Text("Enter valid Email"),
          ),
          actions: <Widget>[ new FlatButton(
                onPressed: () { Navigator.of(context).pop();
                },

                child: new Text("OK"))
          ],
        );
      });
    } else if (passWordController.text.trim() == "") { showDialog(
        context: context,
        builder: (BuildContext context) { return AlertDialog(
          title: new Container(
            child: new Text("Password is required"),
          ),
          actions: <Widget>[ new FlatButton(
                onPressed: () { Navigator.of(context).pop();
                },
                child: new Text("OK"))
          ],
```

```
            );
          });
        } else if (passWordController.text != confirmPasswordController.text) {
          showDialog(
            context: context,
            builder: (BuildContext context) { return

            AlertDialog(
              title: new Container(
                child: new Text("Confirm password doesn\'t match."),
              ),
              actions: <Widget>[ new FlatButton(
                  onPressed: () { Navigator.of(context).pop();
                  },
                  child: new Text("OK"))
              ],

            );
          });
        } else if (vehicleController.text.trim() == "") { showDialog(
            context: context,
            builder: (BuildContext context) { return AlertDialog(
              title: new Container(
          child: new Text("Vehicle number is required"),
              ),
              actions: <Widget>[ new FlatButton(
```

```
                    onPressed: () { Navigator.of(context).pop();
                    },
                    child: new Text("OK"))
                ],
              );
            });
        } else { setState(() {
          loading = true;
        });
        AppSharedPref.getToken().then((v) { Map body = {
          'wk_token': v, 'name': firstNameController.text.trim(),
          'email': emailController.text.trim(),
          'password': passWordController.text.trim(), 'mobile':
          telephoneController.text.trim(), 'vehicle_type':
          currentVehicle.toLowerCase(), 'vehicle_number':
          vehicleController.text.trim(), 'address':

          addressController.text.trim(), 'image': thumbImage};

          if(!isFromLoginPage) {
            body.putIfAbsent('status', () => currentStatus == "Disable"
            ?
'O' : '1');
          }

          debugPrint(TAG + "bodyAddDeliveryBoy:-------->" +
body.toString());
```

```
            ApiCall.makeCall(Method.POST,      Constant.ADD_DELIVERY_BOYS,
            body,
this);

          }).catchError((err) {

            debugPrint(TAG + "error:-------->" + err.toString());

          });

        }

      },

      child: loading

          ? Center(

            child: CircularProgressIndicator(),

          )

          : new Container( width: 310.0,

            height: 50.0,

            alignment: FractionalOffset.center,

            padding: EdgeInsets.fromLTRB(10.0, 10.0, 10.0, 10.0),

            margin:   EdgeInsets.fromLTRB(10.0,   10.0,   10.0,   10.0),

            decoration: new BoxDecoration(

              color: Colors.blueAccent,

              borderRadius:          new          BorderRadius.all(const
Radius.circular(30.0)),

            ),

            child: new Text( signInHint,

              style: new TextStyle( color: Colors.white, fontSize: 16.0,

                fontWeight: FontWeight.bold, letterSpacing: 0.3,

              ),

            ),
```

```
    ));
    Future getImage() async {
  var image = await ImagePicker.pickImage(source: ImageSource.gallery);
  debugPrint("MultipartImageRes=====>" + image.toString());
  if (image != null) { setState(() {
    loading = true;
  });
  String base64Image = base64Encode(image.readAsBytesSync()); String
  fileName = image.path.split("/").last;
  debugPrint("MultipartImageRes=====>" + base64Image.toString() +
"======>" + fileName);
  http.post(Constant.BASE_URL + Constant.UPLOAD_IMAGE, body: {
    "image": base64Image,
    "name": fileName,
  }).then((res) { setState(() {
    loading = false;
  });
  debugPrint("MultipartImageRes=====>" + res.body.toString());
  print("StatusCode===>" + res.statusCode.toString());
  Map response = json.decode(res.body); if (response['error'] != 1)
  {
    setState(() {
      uploadImage = response['image']; thumbImage =
      response['thumb'];
    });
  }
  }).catchError((err) { setState(() {
```

```
          loading = false;
        });
        print(err);
      });
    }
  }

  final profileImage = new Container( margin: EdgeInsets.all(8),
    child: Stack(
      alignment: AlignmentDirectional.bottomEnd, children: <Widget>[
        GestureDetecto

        r( onTap: () {
              getImage();
            },
            child: uploadImage == null ? Container(width: 100.0, height: 100.0, decoration: new BoxDecoration(shape: BoxShape.circle, image: new DecorationImage(fit: BoxFit.fill, image: new AssetImage('assets/profile.png')))) : Container(width: 100.0, height: 100.0, decoration: new BoxDecoration(shape: BoxShape.circle, image: new DecorationImage(fit: BoxFit.fill, image: NetworkImage(uploadImage))))),
        GestureDetector( onTap: () {
            getImage();
          },
          child: Icon(Icons.edit),
        )
      ],
    ),
```

```
    );


    return new Scaffold( appBar: new AppBar(
        leading: new IconButton(
            icon: new Icon(Icons.arrow_back), onPressed: () {
//              var home = HomePage("", "", "", "", "", "", false);// home.currentPos = 2;
//              Navigator.of(context).pushReplacement(new MaterialPageRoute(builder: (context) => home));
                Navigator.of(context).pop();
            }),
        title: Text(isFromLoginPage ? createDeliveryBoyTitle : addDeliveryBoyTitle),
      ),
      body: new Container( color: Colors.white,
        child: new SingleChildScrollView( child: new Column(
            mainAxisSize: MainAxisSize.max,
            children: <Widget>[profileImage, firstName, telephone, email, passWordField, confirmPasswordField, vehicleNumber, address, isFromLoginPage ? Container() : statusDropDownContainer, vehicleDropDownContainer, submitButton],
          ),
        ),
      ),
    );
  }
@override
```

```
void getResponse(String response) { Map res = json.decode(response); if
(res['error'] != 1) {
    setState(() { loading = false;
    });
    showDialog(
        context:    mContext,   barrierDismissible:    false,   builder:
        (BuildContext context) {
          return AlertDialog(
            title: new Text(res['message']), actions: <Widget>[
              new TextButton( onPressed: () {
                    Navigator.of(context).pop();
                    Navigator.of(context).pop(response);
                  },
                  child: new Text("OK"))
            ],
          );
        });
  } else { showDialog(
        context: mContext,
        builder: (BuildContext

        context) { return AlertDialog(
            title: new Text(res['message']), actions: <Widget>[
              new FlatButton( onPressed: () {
                    setState(() { loading = false;
                    });
                    Navigator.of(context).pop();
```

```
                },
                child: new Text("OK"))
            ],
          );
        });
    }
  }
}
```

CHAPTER 6

TESTING

5.1 INTRODUCTION

Testing is the integral part of any System Development Life Cycle insufficient and interested application tends to crash and result in loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation.

"Software Testing can be looked upon as one among much process, an organization performs, and that provides the last opportunity to correct any flaws in the developed system. Software Testing includes selecting test data that have more probability of giving errors." The first step in System testing is to develop the plan that all aspect of system .Complements, Correctness, Reliability and Maintainability.

Software is to be tested for the best quality assurance, an assurance that system meets the specification and requirement for its intended use and performance.

System Testing is the most useful practical process of executing the program with the implicit intention of finding errors that makes the program fail.

5.2 Types of Testing:-

- **Black Box (Functional) Testing:**

Testing against specification of system or components. Study it by examining its inputs and related outputs. Key is to devise inputs that have a higher likelihood of causing outputs that reveal the presence of defects. Use experience and knowledge of domain to identify such test cases. Failing this a systematic approach may be necessary. Equivalence partitioning is where the input to a program falls into a

number of classes, e.g. positive numbers vs. negative numbers. Programs normally behave the same way for each member of a class. Partitions exist for both input and output. Partitions may be discrete or overlap. Invalid data (i.e. outside the normal partitions) is one or more partitions that should be tested.

Internal System design is not considered in this type of testing. Tests are based on requirements and functionality.

This type of test case design method focuses on the functional requirements of the software, ignoring the control structure of the program. Black box testing attempts to find errors in the following categories:

- Incorrect or missing functions.

5.2.1 Interface errors.

5.2.2 Errors in data structures or external database access.

5.2.3 Performance errors.

5.2.4 Initialization and termination errors.

➢ White Box (Structural) Testing:

Testing based on knowledge of structure of component (e.g. by looking at source code). Advantage is that structure of code can be used to find out how many test case need to be performed. Knowledge of the algorithm (examination of the code) can be used to identify the equivalence partitions. Path testing is where the tester aims to exercise every independent execution path through the component. All conditional statements tested for both true and false cases. If a unit has no control statements, there will be up to 2n possible paths through it. This demonstrates that it is much easier to test small program units than large ones. Flow graphs are a

pictorial representation of the paths of control through a program (ignoring assignments, procedure calls and I/O statements). Use flow graph to design test cases that execute each path. Static tools may be used to make this easier in programs that have a complex branching structure. Tools support. Dynamic program analyzers instrument a program with additional code. Typically this will count how many times each statement is executed. At end print out report showing which statements have and have not been executed. Problems with flow graph derived testing:

5.2.5 Data complexity could not take into account.

5.2.6 We cannot test all paths in combination.

5.2.7 In really only possible at unit and module testing stages because beyond that complexity is too high.

This testing is based on knowledge of the internal logic of an application's code. Also known as a Glass Box Testing .Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.

- **Unit Testing:**

Unit testing concentrates on each unit of the software as implemented in the code. This is done to check syntax and logical errors in programs. At this stage, the test focuses on each module individually, assuring that it functions properly as a unit. In our case, we used extensive white-box testing at the unit testing stage.

A developer and his team typically do the unit testing do the unit testing is done in parallel with coding; it includes testing each function and procedure.

- **Incremental Integration Testing:**

Bottom up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately done by programmers or by testers.

- **Integration Testing:**

Testing of integration modules to verify combined functionality after integration .Modules are typically code modules, individual applications, client and server and distributed systems.

- **Functional Testing:**

This type of testing ignores the internal parts and focus on the output is as per requirement or not .Black box type testing geared to functionality requirements of an application.

- **System Testing:**

Entire system is tested as per the requirements. Black box type test that is based on overall requirement specifications covers all combined parts of a system.

CHAPTER 6

REFERENCES

[1] Fundamentals of Database systems – book by Elmasri and Navathe.

[2] Scrum Methodology & Agile Scrum Methodologies.

[3] Developer.android.com.

[4] Akshi Kumar, Web Technology, CRC Press, 1st Edition, PP 135..

[5] Amos Q. Haviv, MEAN Web Development, Second Edition, PP 84

[6] Textbook-Data Mining: Concepts and Techniques (3rd Edition)by J. Han, M. Kamber, and J. Pei Morgan Kaufmann Publ. 2012 ISBN: 978-0-12-381479-1

[7] IEEE Transactions on software engineering, vol. 31, No. 3, March 2005 [9]R.Pressman, software engineering A practitioner's approach. Fifth edition McGrawHill, 2001.

[8] Amos Q. Haviv, MEAN Web Development, Second Edition, PP 84

[9] Gerard O'Regan, Concise Guide to Software Testing, Springer Nature Switzerland AG, 1st Edition, PP 49.

[11] Start Programming Using HTML, CSS, and JavaScript

[12] Recent Developments and the New Direction in Soft-Computing Foundations and Applications

[13] Olga Filipova, and Rui Vilao, Software Development from A to Z, PP 183.

[14] Steve Fenton, Pro TypeScript Application -Scale JavaScript Develoment, 1st Edition, PP

[15] Gerald D. Everett, and Raymond McLeod, Jr, Software Testing, Testing Across the Entire Software Development Life Cycle, PP 93.

[16] Martin P. Robillard, Introduction to Software Design with Java, Springer Nature Switzerland AG, 1st Edition, PP 91.

[17] Iztok Fajfar, Start Programming Using HTML, CSS, and Javascript, CRC Press, 1st

BIBLIOGRAPHY

- Beginning App Development with Flutter **by Rap Payne.**
- Beginning Flutter: A Hands On Guide to App Development **by Marco L. Napoli.**
- Watanabe, Y., Suzuki, S., Sugihara, M., & Sueoka, Y. (2002). An experimental study of paper flutter. *Journal of fluids and Structures*, *16*(4), 529-542.
- Theodorsen, T., & Garrick, I. E. (1940). *Mechanism of flutter a theoretical and experimental investigation of the flutter problem*. NATIONAL AERONAUTICS AND SPACE ADMINISTRATION WASHINGTON DC.
- Salerno, D. M., Dias, V. C., Kleiger, R. E., Tschida, V. H., Sung, R. J., Sami, M., ... & Groupabcdefg, F. S. (1989). Efficacy and safety of intravenous diltiazem for treatment of atrial fibrillation and atrial flutter. *The American journal of cardiology*, *63*(15), 1046-1051.