

OUTDOOR TESTING (ODT) APP

Submitted By:

ADITYA PAL

B.E (4th Sem, CSE)

BIT Mesra, Ranchi

Certificate of Approval

*This is to certify that **ADITYA PAL**, a student of 4th semester B.E enrolled in C.S.E. Branch of **BIT Mesra, Ranchi** has worked under my supervision and guidance from **9th MAY 2017 – 9th JUNE 2017** and has completed the project for “**OUTDOOR TESTING (ODT) APP**” successfully. We appreciate his enthusiasm and commitment during the project tenure and wish his success in all his future endeavours.*

Mr. Pralay Pal

*(Deputy General
Manager)*

Mr. Kuntal Choudhary

*(Associate-Human
Resource)*

Mr. Krishna Singh

(Consultant)

Acknowledgment

It has been a great honour and privilege to undergo training at Tata Technologies Ltd., Jamshedpur. I have been able to complete this project only due to the support and guidance of many individuals. However, it would be wrong on my part to not express my sincere thanks to all of them.

I would like to take an opportunity to express my humble gratitude to Mr. Krishna Singh who has helped me execute this project. His constant guidance and willingness to share his vast knowledge made me understand this project and its manifestations in great depth. I am highly obliged to him without whose support this work would not have been accomplished. His invaluable guidance helped me understand the project better.

I am grateful to Mr. Kuntal Choudhury, the Human resources manager for providing all the facilities to meet my project requirements and giving me a chance to work with this organization.

ADITYA PAL

TABLE OF CONTENTS

<u>CHAPTER NO</u>	<u>PARTICULARS</u>	<u>PAGE NO</u>
1	Introduction	5
2	Objectives	6
3	Proposed features	6
4	Software Life cycle	7-8
5	Proposed method and Architecture	9-16
6	Description of the Activities	17-43
7	Conclusion	44
8	Bibliography	45

1. Introduction

OUTDOOR TESTING (ODT) App is an online app that allows data from vehicle testing sites to be directly uploaded to the webservice without the need for unnecessary paperwork. The current procedure for sending data from testing sites involves the test supervisor gathering the results and sending them manually through paper which gets uploaded by yet another person. This is pretty tedious and mistake-prone.

This ODT App resolves the above problems. It is an online app where the vehicle testing supervisor can search for the test and input the daily data in a hassle-free way. The test data entered in the app gets uploaded to a webservice on the click of a button. The app is quite user-friendly making it easier than writing data on sheets of paper. The app also increases productivity by reducing the cycle time for testing and uploading the data collected.

2. Objectives

- To provide instant, digitized and user-friendly way of uploading vehicle test results instead of manual process.
- Prevention of errors in data entry caused by the long manual process being used currently.
- Increase productivity by reducing the cycle time of the testing and data collection process.

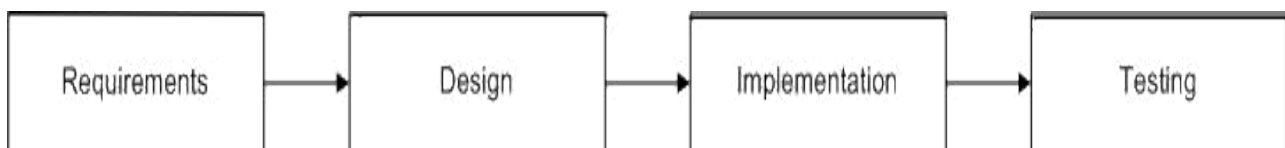
3. Proposed Features

- Role based view/updating of Daily Data
- Dynamic and Configurable Items.
- Easy Integration.
- PC Compatible.
- Uniformity throughout the application
- User-Friendly
- Custom Validation and Maintenance of data entry.

4. Software development lifecycle model

The General Model

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. The general, basic model is shown below:

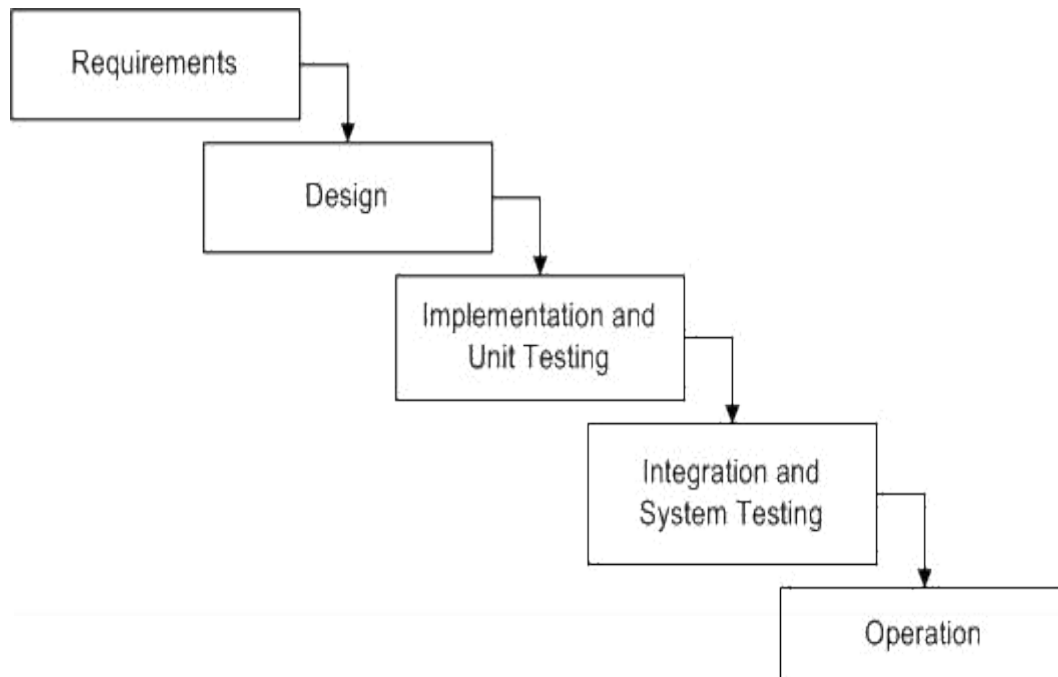


Each phase produces deliverable required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

Waterfall Model

This is the most common and classic of life cycle models, also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed in its entirety before the next phase can begin. At the end of each phase, a review

takes place to determine if the project is on the right path and whether or not to continue or discard the project. Unlike what I mentioned in the general model, phases do not overlap in a waterfall model.



Advantages

- Simple and easy to use.
- Easy to manage due to the rigidity of the model - each phase has specific deliverable and a review process.
- Phases are processed and completed one at a time.
- Easy to keep a record of Vehicle specifications and manage User details digitally.

5. Proposed Methods and Architecture

Android Elements Used :

Gradle Version : 3.3

Android Plugin Version : 2.3.2

Compile SDK version : API 25: Android 7.1.1 (Nougat)

Build Tools Version : 25.0.3

Operating System Requirements :

Minimum SDK Version : API 15 : Ice Cream Sandwich

Softwares Used:

- **Languages** : Java 8.0
- **Integrated Development Environment** : Android Studio
Bundle Version 2.3.3
- **Operating System Used** : Linux Mint 18.1 Serena

Java 8.0

Java 8 is a revolutionary release of the world's #1 development platform. It includes a huge upgrade to the Java programming model and a coordinated evolution of the JVM, Java language, and libraries. Java 8 includes features for productivity, ease of use, improved polyglot programming, security and improved performance.

Java 8 is the latest release for Java that contains new features, enhancements and bug fixes to improve efficiency to develop and run Java programs. The new release of Java is first made available to developers to give adequate time for testing and certification before being made available on the java.com website for end users to download.

Features of Java 8

Here is a brief summary of the enhancements included with the Java 8 release:

- **Lambda Expression and Virtual Extension Methods**
Highlighting feature of Java SE 8 is the implementation of Lambda expressions and supporting features to the Java programming language and platform.
- **Date and Time API**
This new API will allow developers to handle date and time in a more natural, cleaner and easier to understand way.
- **Nashorn JavaScript Engine**
A new lightweight, high performance implementation of JavaScript engine is integrated to JDK and is available to Java applications via existing APIs.
- **Improved Security** replacing the existing hand-maintained list of caller sensitive methods with a mechanism that accurately identifies such methods and allows their callers to be discovered reliably.

JVM, JRE and JDK

A **Java virtual machine (JVM)** is an abstract computing machine that enables a computer to run a Java program. There are three notions of the JVM: specification, implementation, and instance. The specification is a document that formally describes what is required of a JVM implementation. Having a single specification ensures all implementations are interoperable. A JVM implementation is a computer program that meets the requirements of the JVM specification. An instance of a JVM is an implementation running in a process that executes a computer program compiled into Java bytecode.

Java Runtime Environment (JRE) is a software package that contains what is required to run a Java program. It includes a Java Virtual Machine implementation together with an implementation of the Java Class Library. The Oracle Corporation, which owns the Java trademark, distributes a Java Runtime environment with their Java Virtual Machine called HotSpot.

Java Development Kit (JDK) is a superset of a JRE and contains tools for Java programmers, e.g. a javac compiler. The Java Development Kit is provided free of charge either by Oracle Corporation directly, or by the OpenJDK open source project, which is governed by Oracle.

Android Studio Bundle

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance the productivity when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to the running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

Project Structure

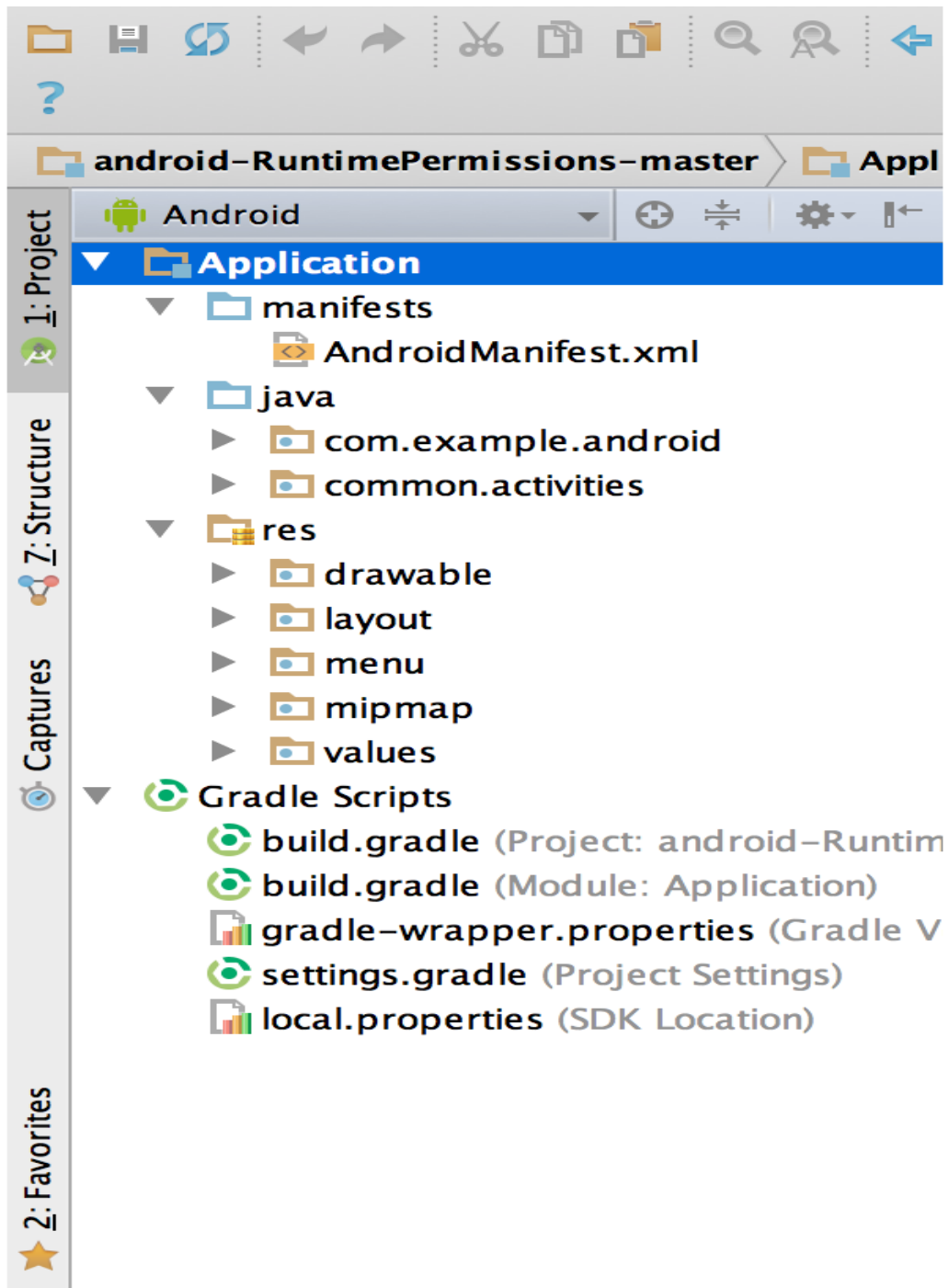


Figure 1. The project files in Android view.

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays the project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to the project's key source files.

All the build files are visible at the top level under **Gradle Scripts** and each app module contains the following folders:

- **manifests**: Contains the AndroidManifest.xml file.
- **java**: Contains the Java source code files, including JUnit test code.
- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

You can also customize the view of the project files to focus on specific aspects of the app development. For example, selecting the **Problems** view of the project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

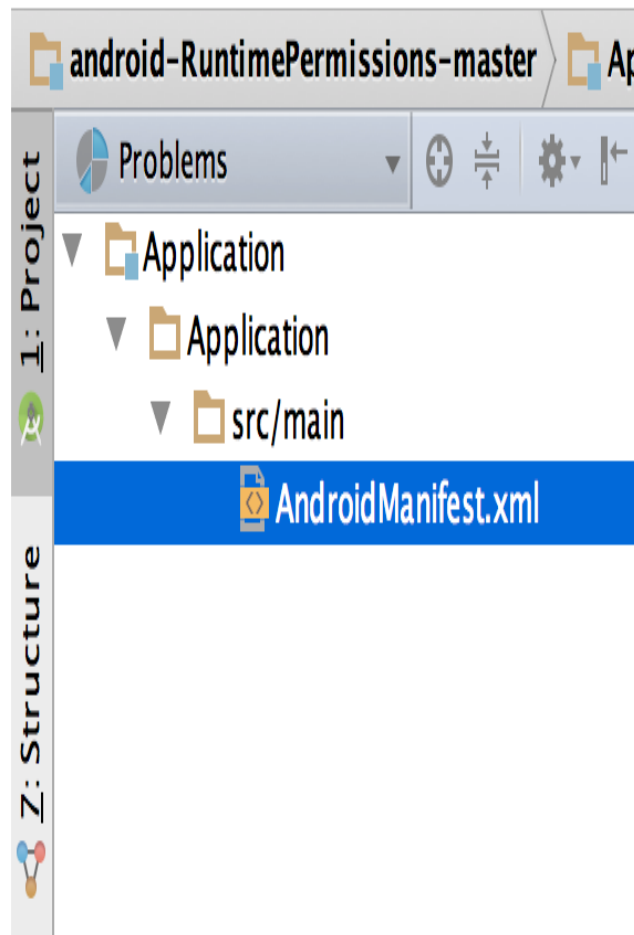


Figure 2. The project files in Problems view, showing a layout file with a problem.

Gradle Build System

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plugin for Gradle. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for the app, with different features using the same project and modules.

- Reuse code and resources across sourcesets.

By employing the flexibility of Gradle, you can achieve all of this without modifying the app's core source files. Android Studio build files are named `build.gradle`. They are plain text files that use Groovy syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

To learn more about the build system and how to configure, see [Configure the Build](#).

Build Variants

The build system can help you create different versions of the same application from a single project. This is useful when you have both a free version and a paid version of the app, or if you want to distribute multiple APKs for different device configurations on Google Play.

For more information about configuring build variants, see [Configuring Gradle Builds](#).

Multiple APK Support

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the `hdpi` and `mdpi` screen densities, while still considering them a single variant and allowing them to share test APK, `javac`, `dx`, and ProGuard settings.

For more information about multiple APK support, read [Build Multiple APKs](#).

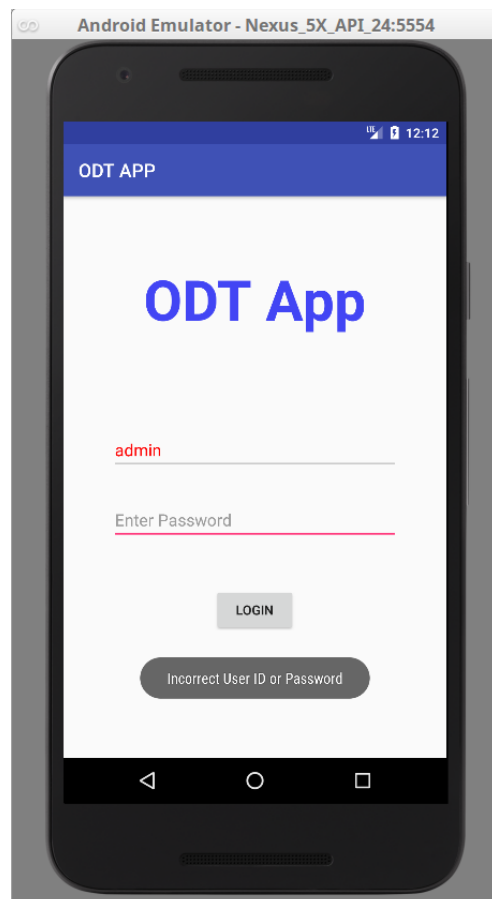
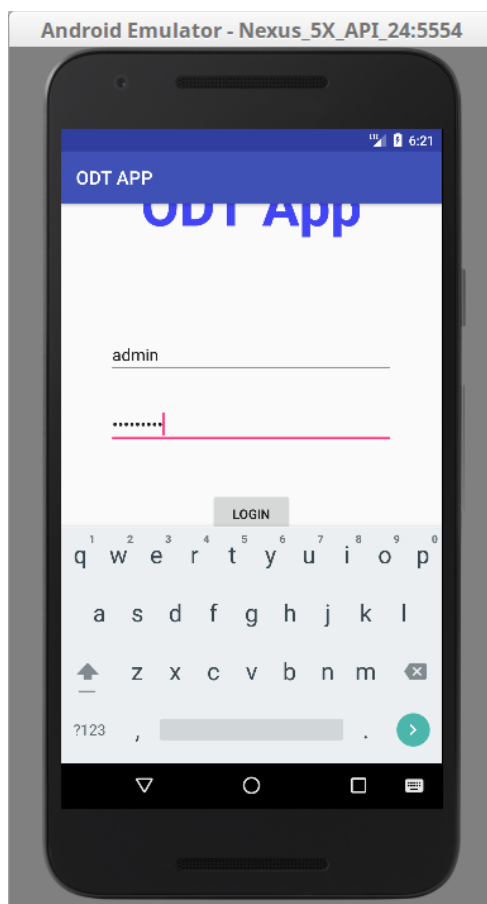
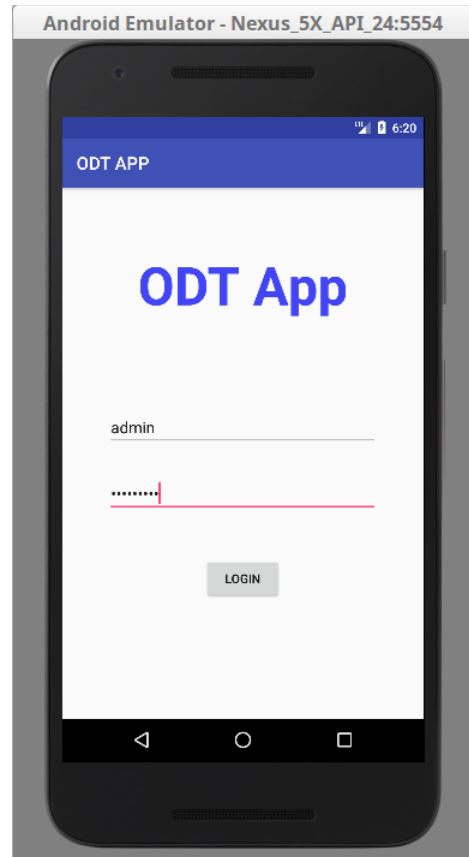
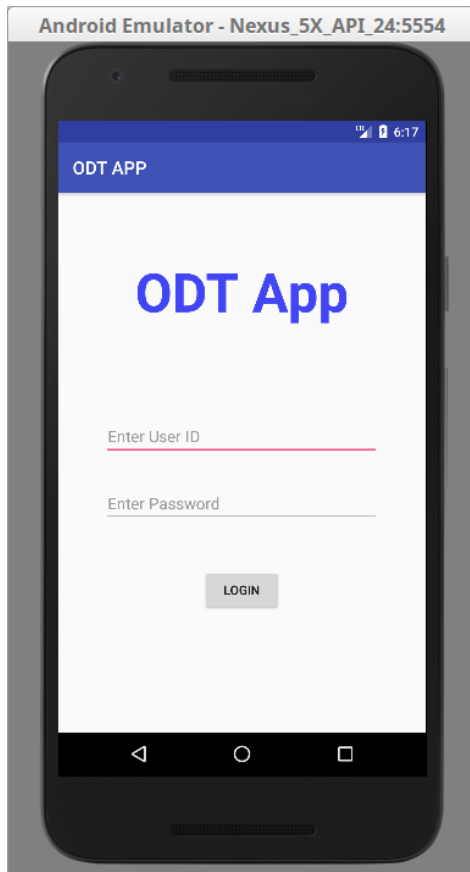
6. Description of the Activities

1. Login Activity

In this activity, there are two text fields – User ID and Password which allows a user to login to the app using his/her login credentials. There is a login button which the user clicks after he/she enters his/her login details. Once the login button is clicked, the app connects to a web-service which returns the validity of the entered user ID and password. If the entered details are valid, the login activity goes to the Home Page Activity. If the entered details are incorrect, the User ID field is highlighted in red and a toast is made prompting the user to re-enter details. Also the password field is reset.

The Login Activity has the following features :

- Scrollable Screen
- User ID and Password Fields with hints
- Password field is of “password type – asterisks shown”
- Login Button
- Highlighted User ID if entered details are invalid
- Toast to enter correct details if entered details are invalid



2. Data Home Activity

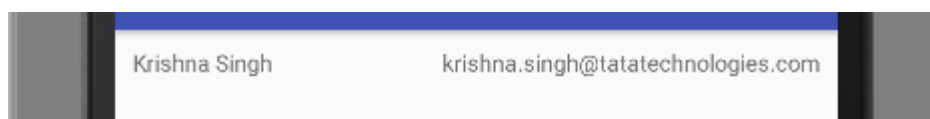
Once the login details of the user are verified in the Login Activity, the user is directed to the Data Home Activity. There are the following elements in the Data Home Activity:

- Header
- Input Fields and Buttons
- Data Display
- Footer

Description of the Data Home Activity Elements :

Header

The Header of the Data Home Activity contains the user information received from the web-service when user authentication takes place. The header includes the name of the user and the email ID of the user obtained from the web-service. It is a proof that the user is logged into his account properly.

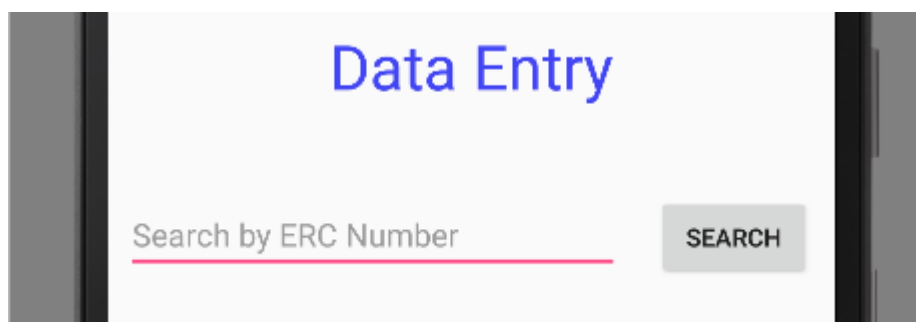


Input Fields and Buttons

Below the header of the Data Home Activity, there are two fields – A text field to search a test by ERC Number of the Test and a text field to search a test by Model Number of Vehicle. The following are the features of each field :

Search By ERC Number

- This Text Field can be used to enter the ERC number of the test directly if the user has prior knowledge of the test. The Text Field is followed by a Search Button, which when clicked verifies the correctness of the ERC Number entered. If the entered ERC Number is found to be correct according to the web-service, the user is redirected to the screen containing the inputs to be given to the test. If the user enters an invalid ERC Number, the ERC Number text box gets highlighted in red and a toast is made to re-enter a proper ERC Number. If no ERC Number is entered but the Search button is clicked, the user is prompted to enter the ERC Number before Submitting.



Search By Model Number

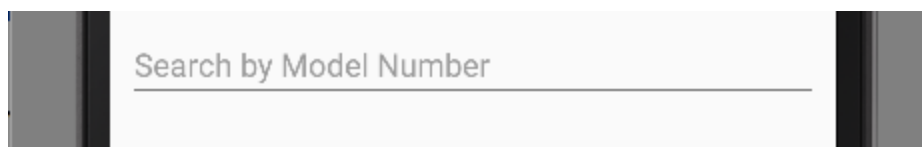
- This Text Field can be used to search for the Model Number of the test. When a user clicks on the Search By Model Number Text Box, a scrollable List View opens containing the tests available to the user. The tests have the following nature :

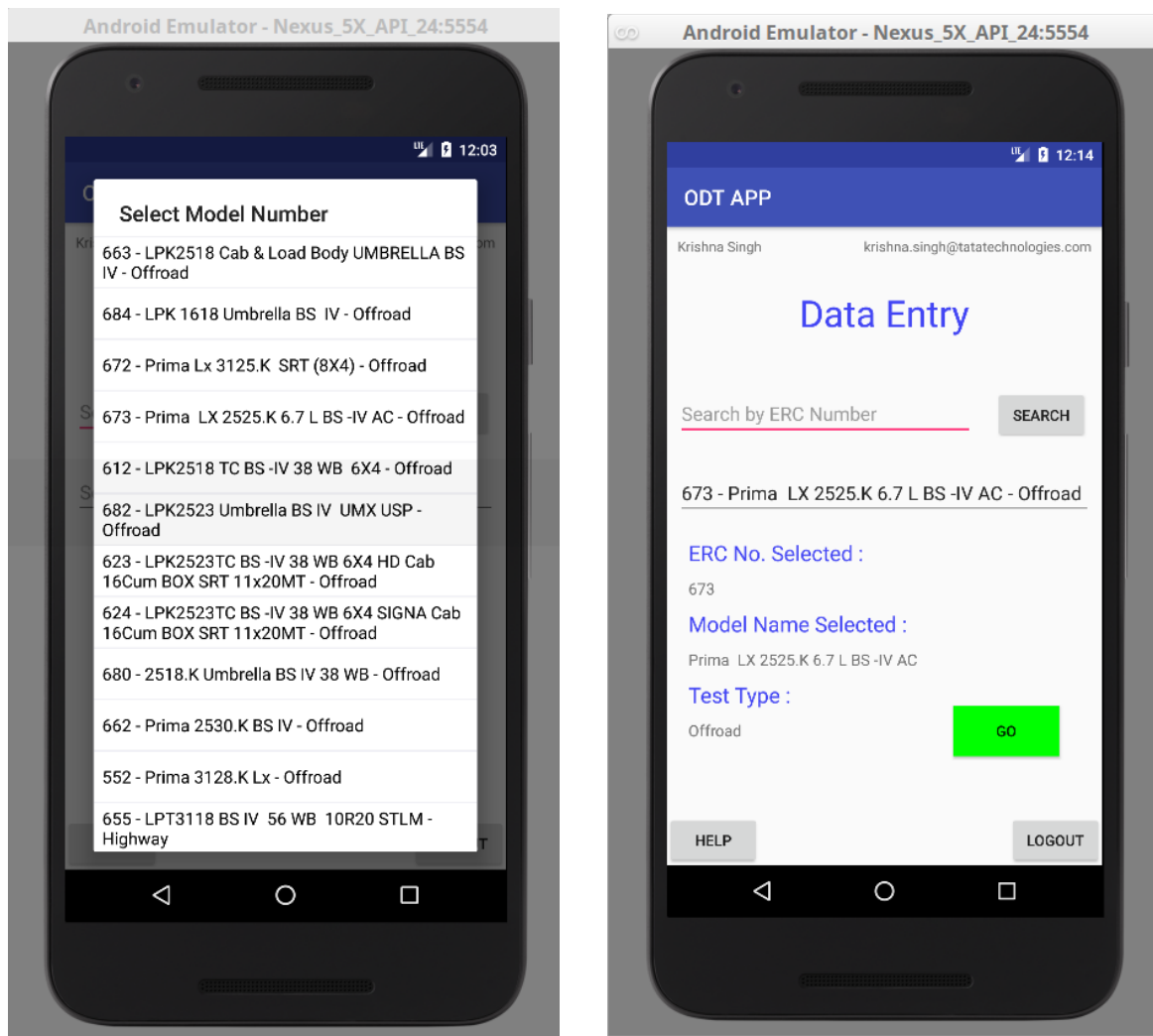
ERC Number – Model Name – Test Type

The ERC Number is the same as that can be entered in the search by ERC Number. The model name has the name of the model number of the vehicle tested. The test type is of three types :

- Highway Testing
- Offroad Testing
- Torture Testing

On selecting an option from the List View, the Search By Model Number Text Field changes to the value selected.





Data Display

The Data Display Area is used to display the Test details selected by the user in the List View of the Search by Model Number. The Display contains the following data :

- ERC Number Selected
- Model name Selected
- Test Type Selected

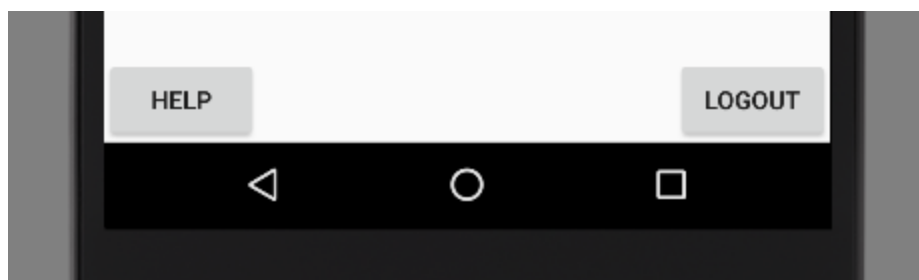
The Display also contains a green coloured button which enables the user to go to the selected test type activity

which can be highway, offroad or torture. Once the Go Button is clicked, the selected information is sent to the respective test activity.

Footer

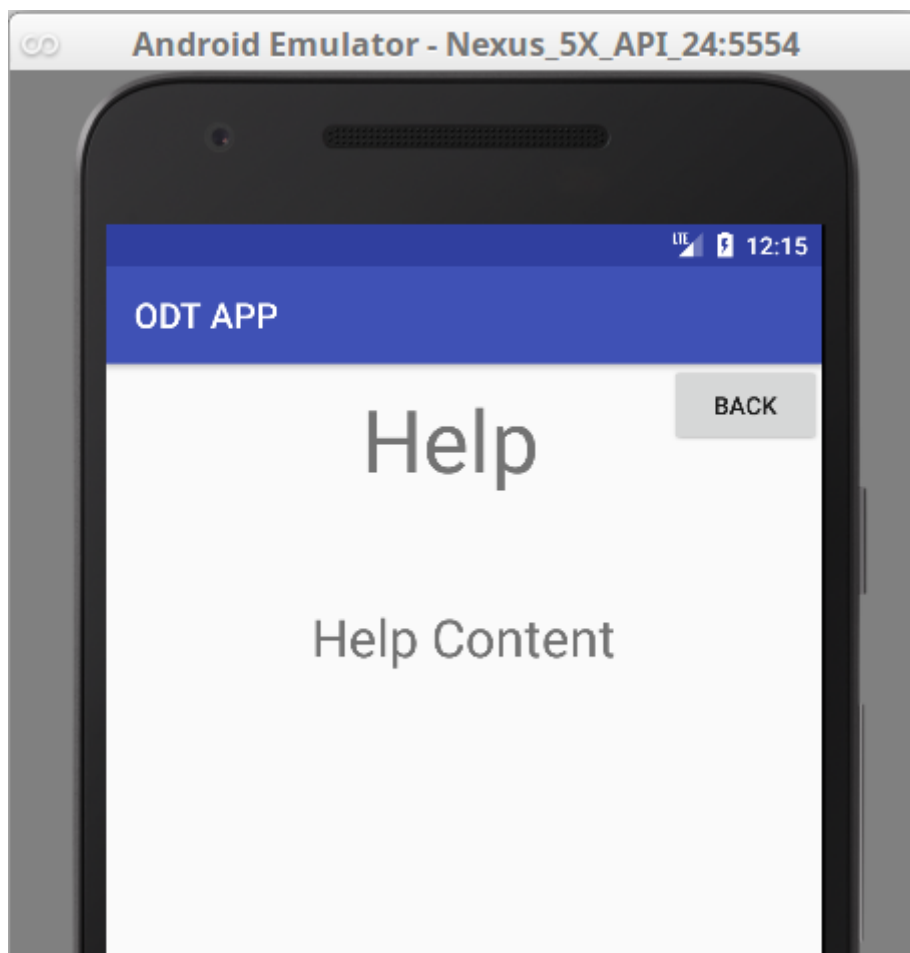
The Header of the Data Home Activity contains two buttons – Help and Logout.

- The Help button is used to reach the help page which directs the user how to use the app. It is a manual which the user can refer to in case he is unable to understand a specific functioning of the app.
- The logout button is used to Log the user out of the current session. When the logout button is clicked, the user is redirected to the login page and his session is terminated.



3. Help / Manual Activity

This Activity can be accessed from the Data Home Activity by the click of the Help Button present on the footer of the Data Home Activity. This page is basically a map of the app which is useful to provide a manual to explain the working of the app in case a user is unable to navigate through the app. The Help Activity contains the help content which has not been updated. It also contains a Back Button at the top of the page which when clicked, allows the user to return to the Data Home Activity page.



4. DATA ENTRY ACTIVITIES

The Test type selected can be of three types :

- Highway Testing (Test Type 1 or TT1)
- Torture Testing (Test Type 2 or TT2)
- Offroad Testing (Test Type 3 or TT3)

Different Activities have been created for each of the above mentioned tests. The following pages describe each Test Activity in detail :

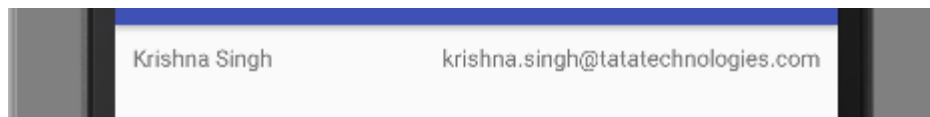
Highway Testing Activity

The Highway Testing Activity is the page for Daily data entry of test data after the completion of a highway test. It is scrollable in nature. The Highway Testing Activity contains the following elements :

- Header
- Title
- Data Input Fields
- Footer

Header

The Header of the HighWay Testing Activity contains the user information received from the web-service when user authentication takes place. The header includes the name of the user and the email ID of the user obtained from the web-service.



Title

This area is used to display the ERC Number and the model name of the Test Vehicle that the user had selected in the Data Home Activity. The title of the page also specifies the type of test whose data is being entered currently.

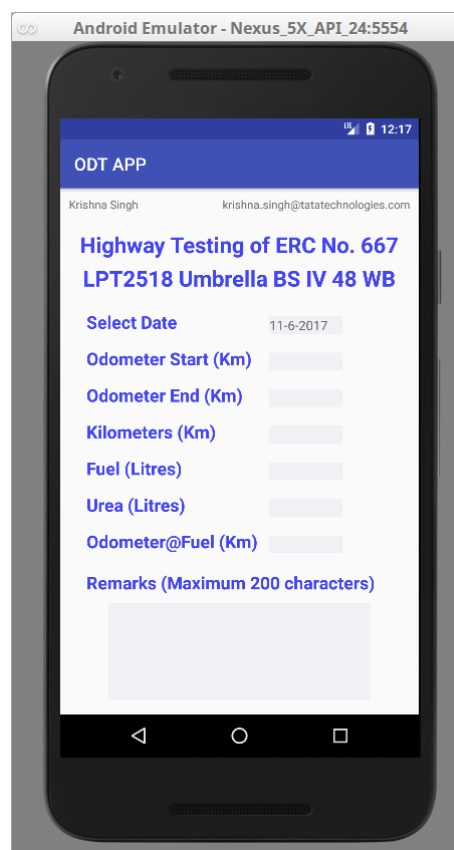
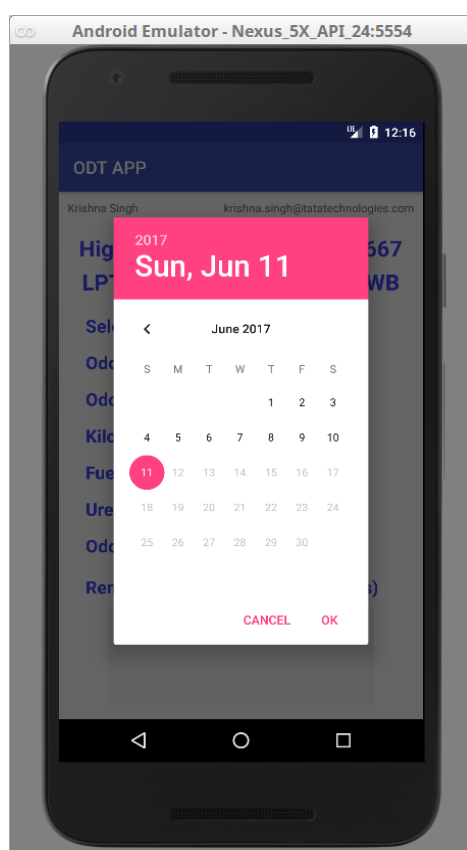


Data Input Fields

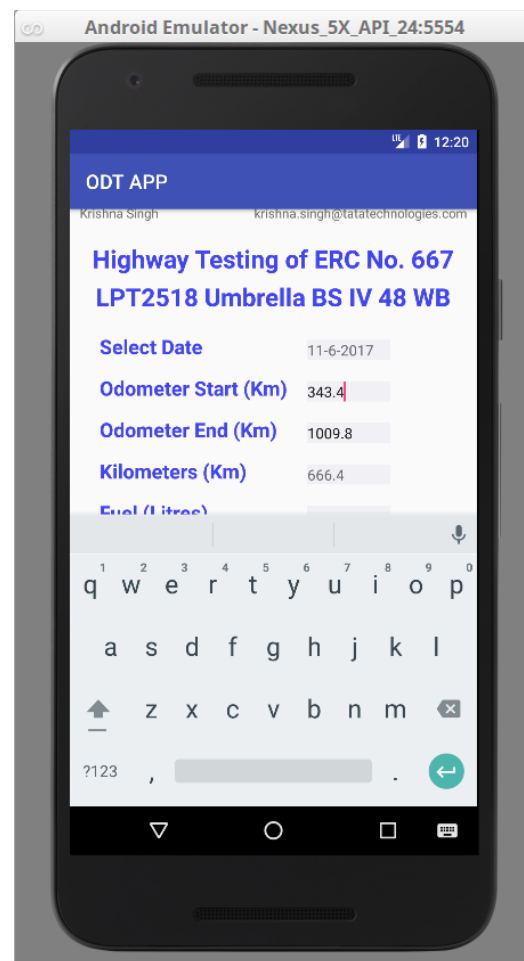
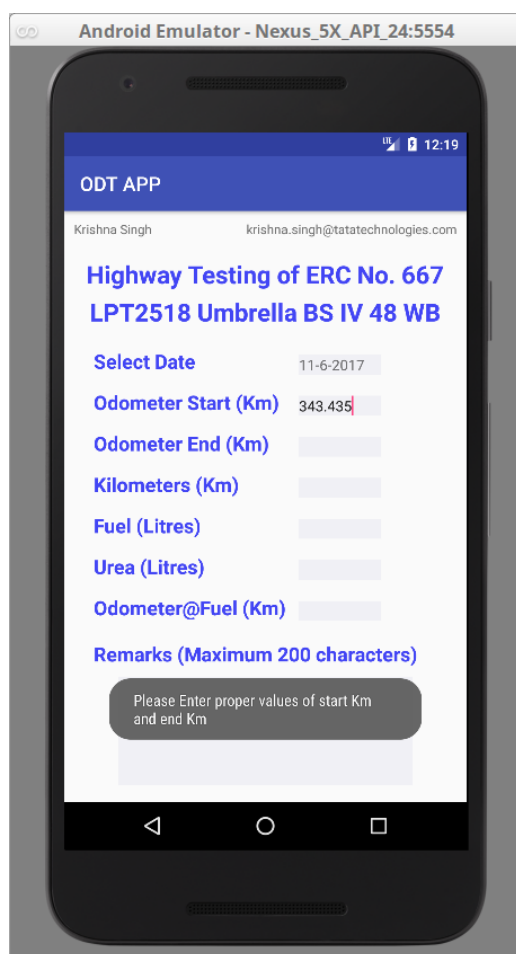
This Data Entry Fields are located below the Title of the page. The data fields are the following :

Required Data Fields and their use :

- **Select date** - The user is expected to enter the date of testing in this field. When the text field is clicked, a Date Picker is opened in a dialog box. This provides a user-friendly way of date selection without the need to worry about the date format. The maximum date allowed in the selection is the current date. Once the user selects the date from the date dialog box, the dialog box closes and the date gets displayed in the date field in DD-MM-YYYY format.



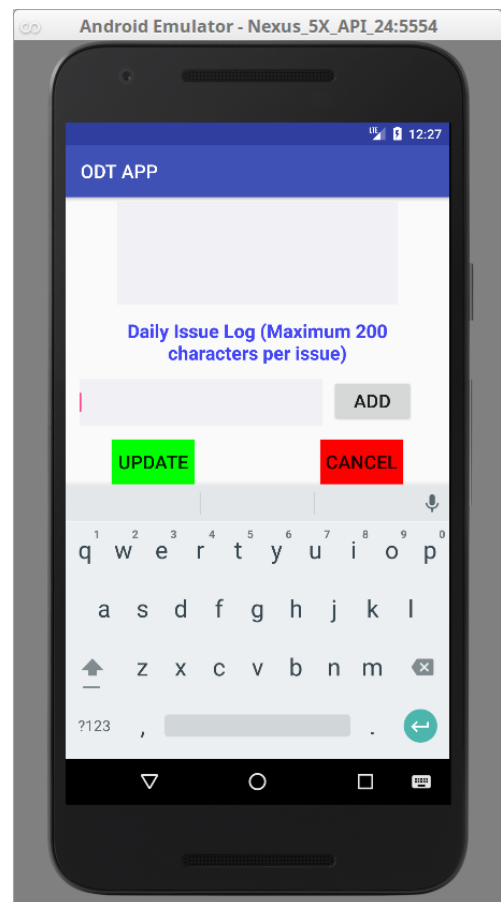
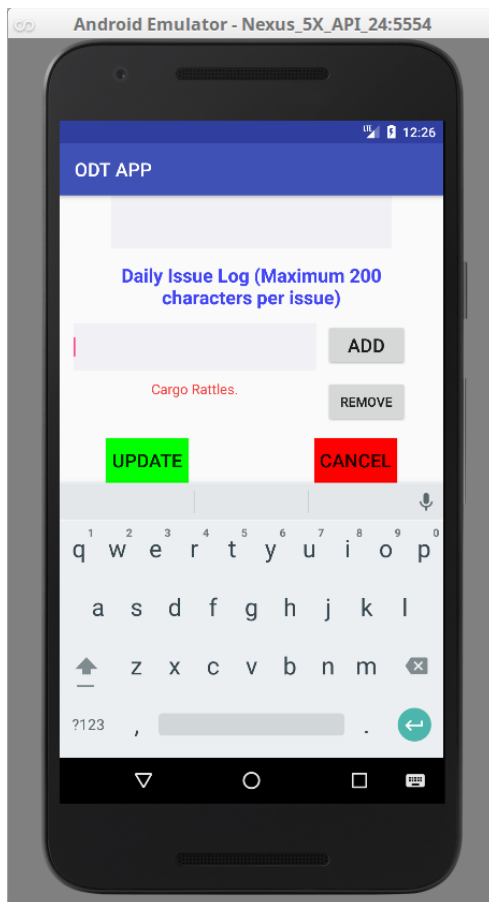
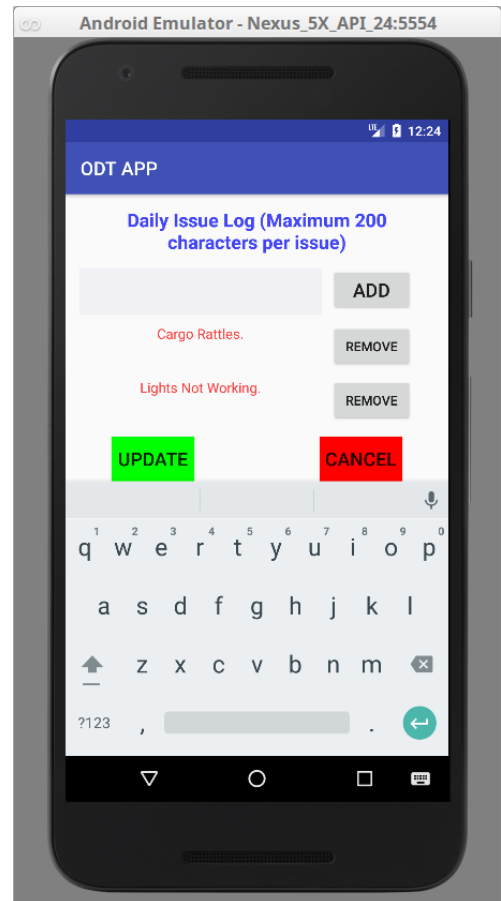
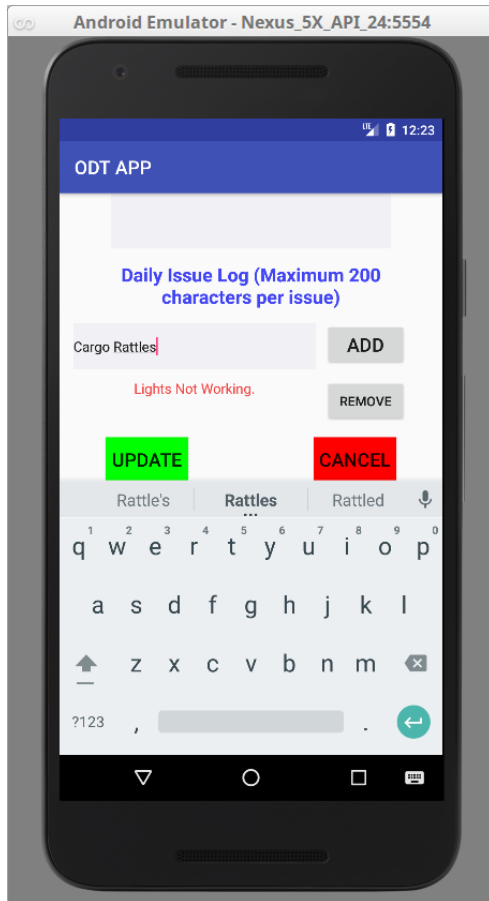
- **Odometer Start** - This text field takes a double value specifying the starting value of the odometer in kilometers before the test is conducted.
- **Odometer End** - This text field takes a double value specifying the ending value of the odometer in kilometers after the test is conducted.
- **Kilometers** - This text field does not take any input from the user. When this text field is clicked, it automatically calculates the total kilometers travelled during the testing by subtracting the odometer start value from the odometer end value. If it does not find the proper values of Odometer Start and Odometer End text fields, it displays an error stating the same.



- **Fuel (Litres)** - This text field takes a double value specifying the amount of fuel in Litres used during the testing procedure.
- **Urea (Litres)** - This text field takes a double value specifying the amount of urea in Litres used during the testing procedure.
- **Odometer@Fuel** - This text field takes a double value specifying the rate of odometer change in kilometers with respect to the amount of fuel in Litres used during the testing procedure.

Optional Data Fields and their use :

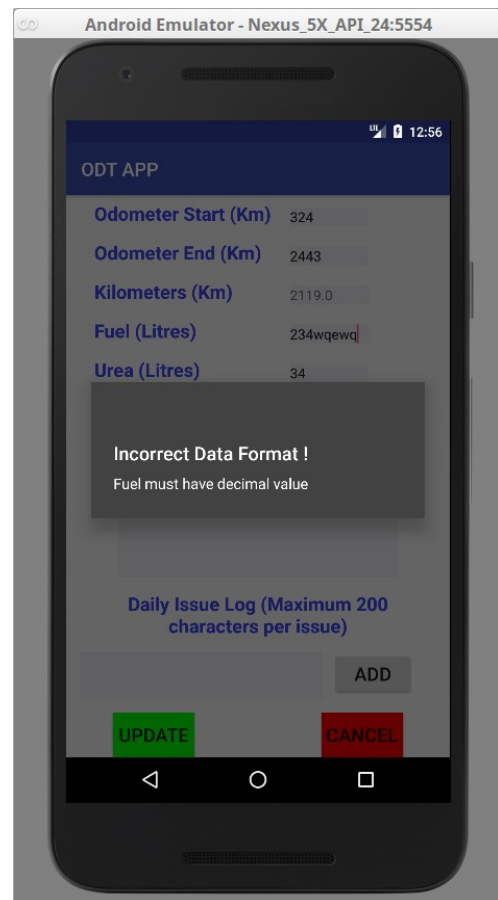
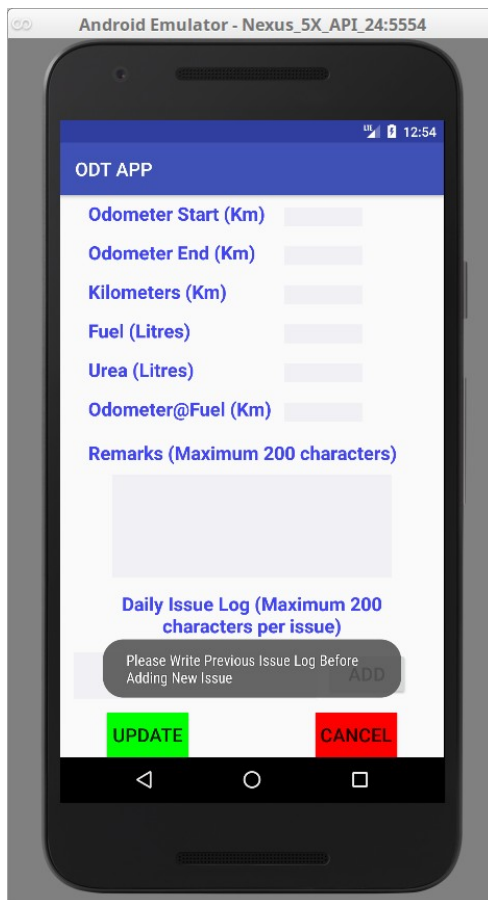
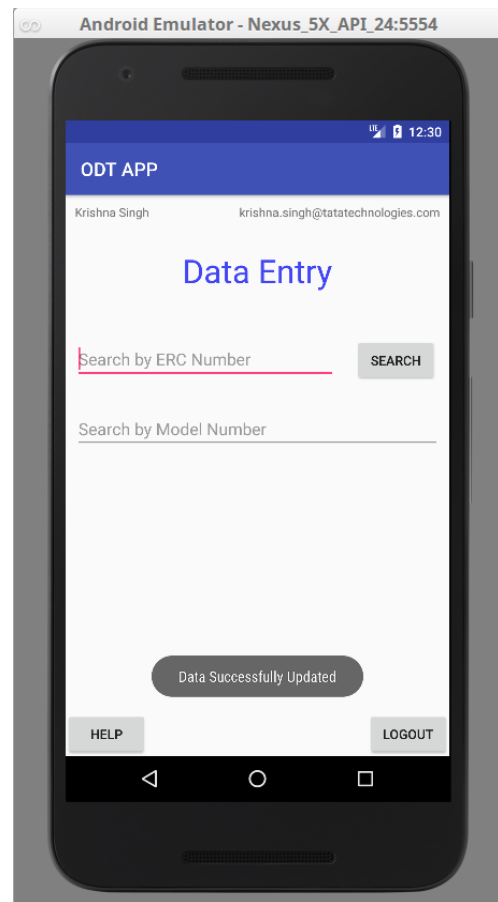
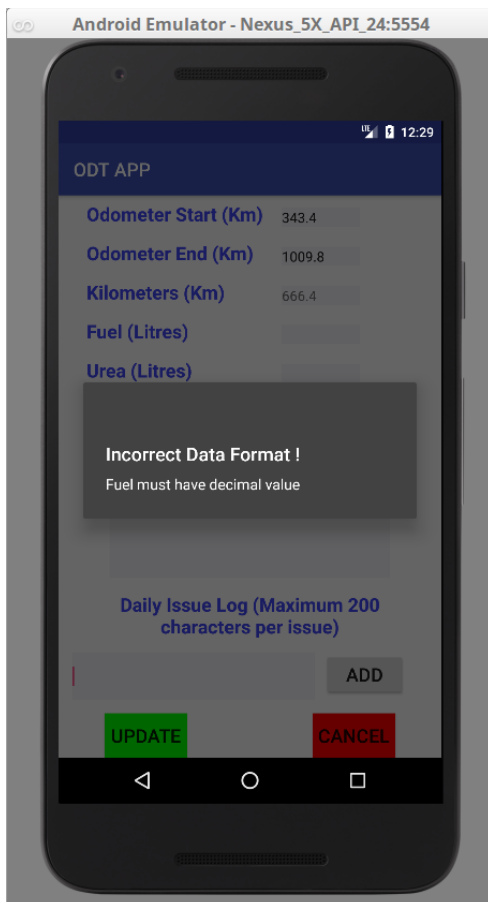
- **Remarks** - This text field is used to write the remarks (if any) of the test. The maximum number of characters allowed in this field is 200.
- **Daily Issue Log** - This text field is used to specify the issues/problems found in the vehicle after the test. The user can add issues one by one by clicking on the ADD button present on the side of the text field. Whenever the ADD button is pressed, the app checks the data entered in the text box. If no data is entered, the user is prompted to enter the issue. If the user has clicks the add button after typing the issue, then the issue gets added and it appears in a text view field below the daily issue log text field with a REMOVE option. The REMOVE button can be used to delete individual issues. Once an issue is added, the daily issue log entry text box is reset and a new issue can be added. The following figures show how daily issue can be added or removed.



Footer

The Footer contains two buttons – a green-coloured UPDATE button and a red-coloured CANCEL button. When the UPDATE button is clicked, the date entered is first verified. If some/all of the required fields is missing, an alert is generated in the app which prompts the user to enter the details before submitting. If all the data is entered in the correct manner, then the app connects to the web-service and passes the data to the web-service for updation to the database. Once the data is updated from the webservice, the user is redirected to the data home activity page and a toast is made stating the success/failure updation of the updation operation.

On the other hand, if the CANCEL button is clicked, no further connections are made to the webservice and the user is redirected to the data home activity page without any updates being made in the database of the web-service.



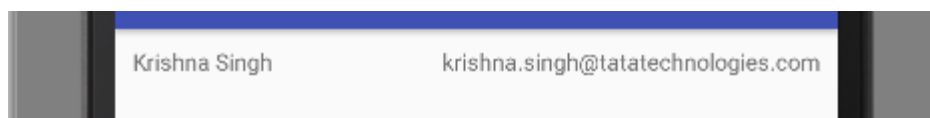
Torture Testing Activity

The Torture Testing Activity is the page for Daily data entry of test data after the completion of a highway test. It is scrollable in nature. The Torture Testing Activity contains the following elements :

- Header
- Title
- Data Input Fields
- Footer

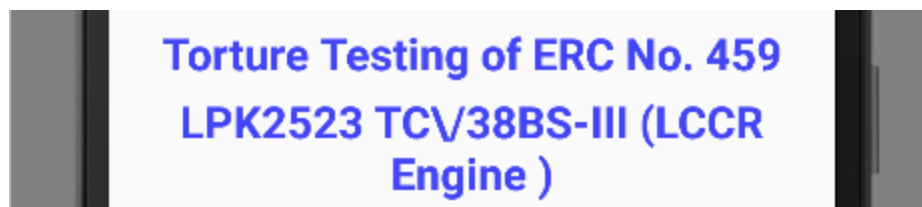
Header

The Header of the Torture Testing Activity contains the user information received from the web-service when user authentication takes place. The header includes the name of the user and the email ID of the user obtained from the web-service.



Title

This area is used to display the ERC Number and the model name of the Test Vehicle that the user had selected in the Data Home Activity. The title of the page also specifies the type of test whose data is being entered currently.



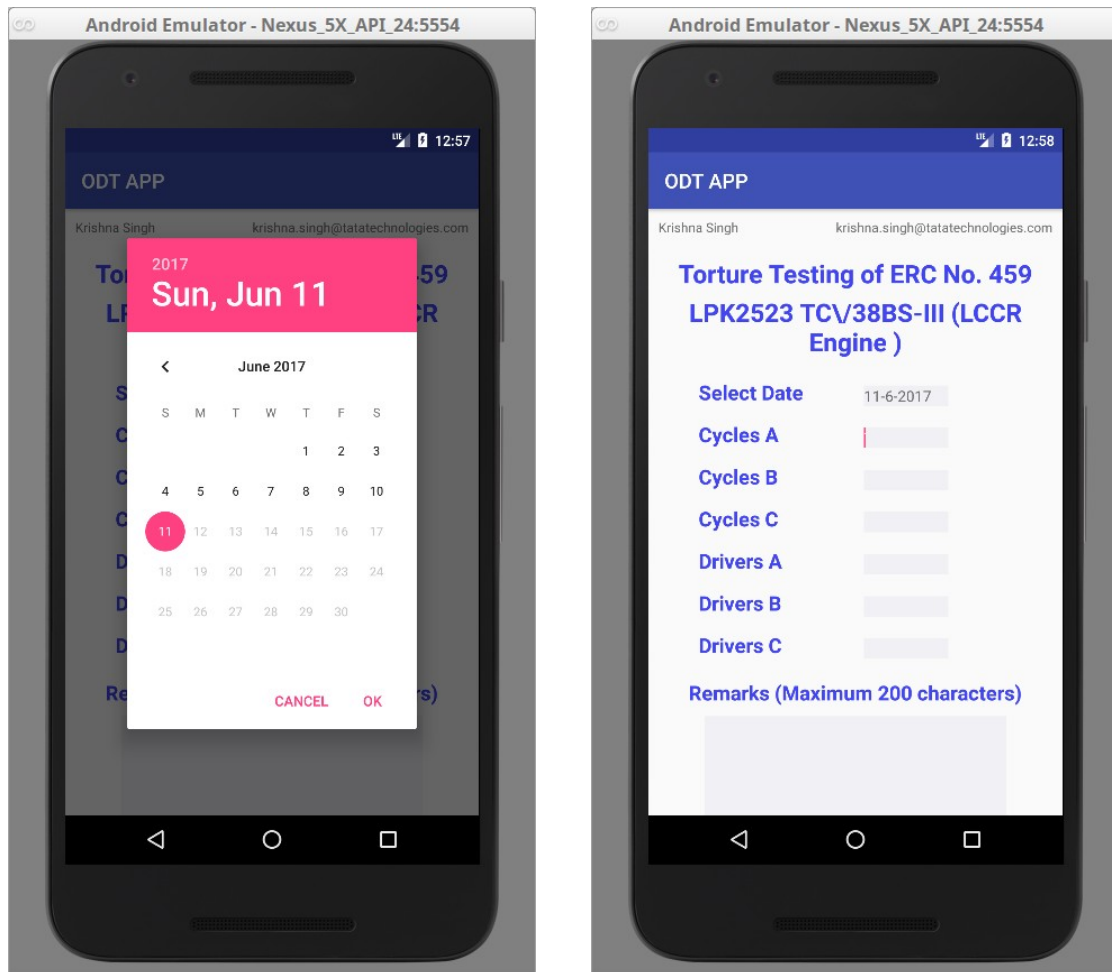
Data Input Fields

This Data Entry Fields are located below the Title of the page. The data fields are the following :

Required Data Fields and their use :

- **Select date** - The user is expected to enter the date of testing in this field. When the text field is clicked, a Date Picker is opened in a dialog box. This provides a user-friendly way of date selection without the need to worry about the date format. The maximum date allowed in the selection is the current date. Once the user selects the date from the date dialog box, the dialog box closes and

the date gets displayed in the date field in DD-MM-YYYY format.



- **Cycles A** - This integer text field is used to enter the number of cycles of torture testing completed during A shift.
- **Cycles B** - This integer text field is used to enter the number of cycles of torture testing completed during B shift.
- **Cycles C** - This integer text field is used to enter the number of cycles of torture testing completed during C shift.

- **Drivers A** - This integer text field is used to enter the number of drivers operating during the torture testing of the vehicle in A shift.
- **Drivers B** - This integer text field is used to enter the number of drivers operating during the torture testing of the vehicle in B shift.
- **Drivers C** - This integer text field is used to enter the number of drivers operating during the torture testing of the vehicle in C shift.

Optional Data Fields and their use :

- **Remarks** - This text field is used to write the remarks (if any) of the test. The maximum number of characters allowed in this field is 200.
- **Daily Issue Log** - This text field is used to specify the issues/problems found in the vehicle after the test. The user can add issues one by one by clicking on the ADD button present on the side of the text field. Whenever the ADD button is pressed, the app checks the data entered in the text box. If no data is entered, the user is prompted to enter the issue. If the user has clicks the add button after typing the issue, then the issue gets added and it appears in a text view field below the daily issue log text field with a REMOVE option. The REMOVE button can be used to delete individual issues. Once an issue is added, the daily issue log entry text box is reset and a new issue can be added. The following figures show how daily issue can be added or removed.

- **Footer**

The Footer contains two buttons - a green-coloured UPDATE button and a red-coloured CANCEL button. When the UPDATE button is clicked, the date entered is first verified. If some/all of the required fields is missing, an alert is generated in the app which prompts the user to enter the details before submitting. If all the data is entered in the correct manner, then the app connects to the web-service and passes the data to the web-service for updation to the database. Once the data is updated from the webservice, the user is redirected to the data home activity page and a toast is made stating the success/failure updation of the updation operation.

On the other hand, if the CANCEL button is clicked, no further connections are made to the webservice and the user is redirected to the data home activity page without any updates being made in the database of the web-service.

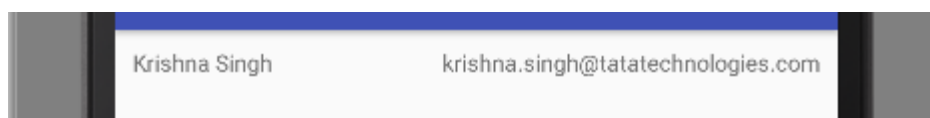
Offroad Testing Activity

The Offroad Testing Activity is the page for Daily data entry of test data after the completion of a highway test. It is scrollable in nature. The Offroad Testing Activity contains the following elements :

- Header
- Title
- Data Input Fields
- Footer

Header

The Header of the Offroad Testing Activity contains the user information received from the web-service when user authentication takes place. The header includes the name of the user and the email ID of the user obtained from the web-service.



Title

This area is used to display the ERC Number and the model name of the Test Vehicle that the user had selected in the Data Home Activity. The title of the page also specifies the type of test whose data is being entered currently.



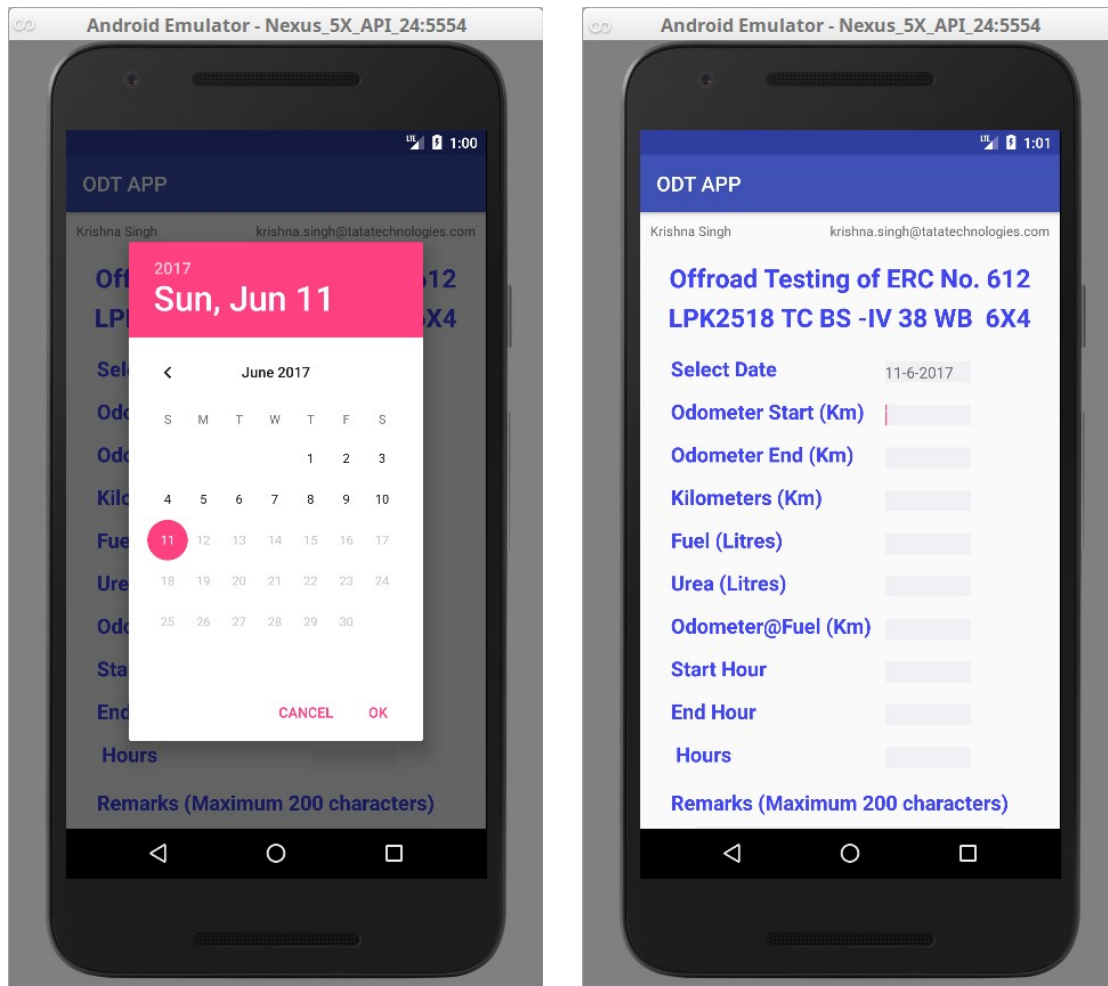
Data Input Fields

This Data Entry Fields are located below the Title of the page. The data fields are the following :

Required Data Fields and their use :

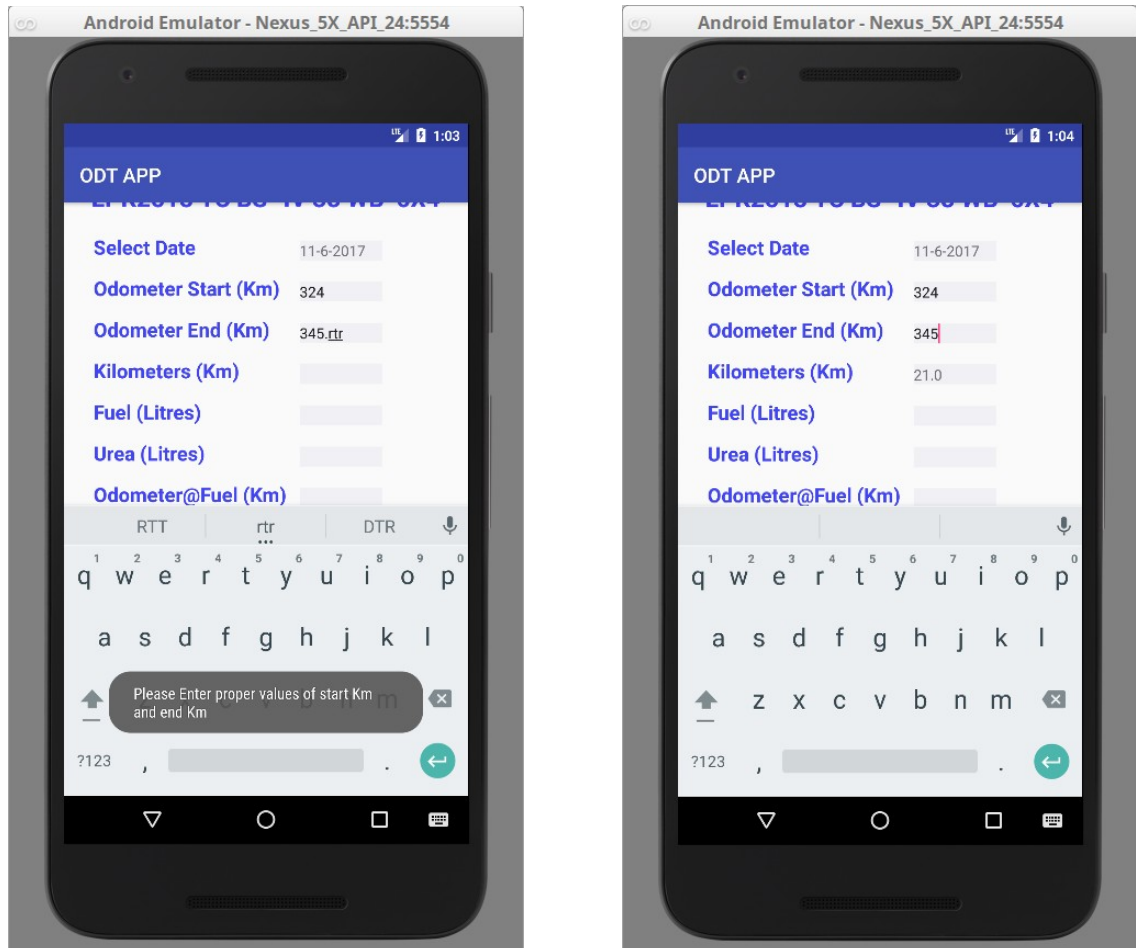
- **Select date** - The user is expected to enter the date of testing in this field. When the text field is clicked, a Date Picker is opened in a dialog box. This provides a user-friendly way of date selection without the need to worry about the date format. The maximum date allowed in the selection is the current date. Once the user selects the date from the date dialog box, the dialog box closes and

the date gets displayed in the date field in DD-MM-YYYY format.



- **Odometer Start** - This text field takes a double value specifying the starting value of the odometer in kilometers before the test is conducted.
- **Odometer End** - This text field takes a double value specifying the ending value of the odometer in kilometers after the test is conducted.
- **Kilometers** - This text field does not take any input from the user. When this text field is clicked, it automatically calculates the total kilometers travelled during the testing by subtracting the odometer start

value from the odometer end value. If it does not find the proper values of Odometer Start and Odometer End text fields, it displays an error stating the same.



- **Fuel (Litres)** - This text field takes a double value specifying the amount of fuel in Litres used during the testing procedure.
- **Urea (Litres)** - This text field takes a double value specifying the amount of urea in Litres used during the testing procedure.
- **Odometer@Fuel** - This text field takes a double value specifying the rate of odometer change in kilometers with respect to the amount of fuel in Litres used during the testing procedure.

- **Start Hour** - This text field is used to note the starting time of the torture testing. It takes a decimal value.
- **End Hour** - This text field is used to note the ending time of the torture testing. It takes a decimal value.
- **Hours** - This text field does not take any input from the user. When this text field is clicked, it automatically calculates the total hours in the testing by subtracting the start hour from the end hour. If it does not find the proper values of Start Hour and End Hour text fields, it displays an error stating the same.

Optional Data Fields and their use :

- **Remarks** - This text field is used to write the remarks (if any) of the test. The maximum number of characters allowed in this field is 200.
- **Daily Issue Log** - This text field is used to specify the issues/problems found in the vehicle after the test. The user can add issues one by one by clicking on the ADD button present on the side of the text field. Whenever the ADD button is pressed, the app checks the data entered in the text box. If no data is entered, the user is prompted to enter the issue. If the user has clicks the add button after typing the issue, then the issue gets added and it appears in a text view field below the daily issue log text field with a REMOVE option. The REMOVE button can be used to delete individual issues. Once an issue is added, the daily issue log entry text box is reset and a new issue can be added. The following figures show how daily issue can be added or removed.

- **Footer**

The Footer contains two buttons – a green-coloured UPDATE button and a red-coloured CANCEL button. When the UPDATE button is clicked, the date entered is first verified. If some/all of the required fields is missing, an alert is generated in the app which prompts the user to enter the details before submitting. If all the data is entered in the correct manner, then the app connects to the web-service and passes the data to the web-service for updation to the database. Once the data is updated from the webservice, the user is redirected to the data home activity page and a toast is made stating the success/failure updation of the updation operation.

On the other hand, if the CANCEL button is clicked, no further connections are made to the webservice and the user is redirected to the data home activity page without any updates being made in the database of the web-service.

7. CONCLUSION

This project, made by me, is a small effort made towards the development of larger programs, and it involves limited aspects which are required in training management. In my app, I hope I have been successful in speeding up the process of updating test data from the test site directly without having to go through the lengthy process of writing data on sheets of paper and then updating them individually. The role based updation and viewing of data were not covered by me due to time constraints. The software development although is a very difficult task but it can be carried out successfully. After the completion of this project, I have learned different things about software and its development and hope to employ the skills which I have acquired during this tenure for development of technologies.

8. BIBLIOGRAPHY

Website reference

1. stackoverflow.com
2. google.com
3. <https://developer.android.com>
4. <https://www.google.com/android/>
5. tools.android.com/download/studio/builds/android-studio-2-2-3