



Qualification		Module Number and Title
HD in Computing and Software Engineering		CSE5011- Mobile Application Development
Student Name & No.		Assessor
M.A.Chamath Shyamal		Mr. Roy Ian
Hand out date		Submission Date
2021-11-09		2022-01-09
Assessment type	Duration/Length of Assessment Type	Weighting of Assessment
Coursework	3000 words	100%
Learner declaration		
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.		

Marks Awarded		
First assessor		
IV marks		
Agreed grade		
Signature of the assessor		Date

FEEDBACK FORM
INTERNATIONAL COLLEGE OF BUSINESS & TECHNOLOGY

Module:

Student:

Assessor:

Assignment: Mobile Application for “Saloon SD”

Strong features of your work:

Areas for improvement:

Marks Awarded:

Coursework – Mobile Application for “Saloon SD”– 100 Marks

Learning outcomes covered

1. Explain mobile operating systems, development tools and technologies for the mobile application development
2. Design mobile application solutions
3. Develop mobile application solutions
4. Test mobile application solutions

Scenario

Saloon SD is a one of the famous saloons in Sri Lanka and they have expanded their services within a short period of time. Currently, all the appoints are managed manually and the management of the saloon is expecting to develop a user-friendly mobile application with multiple features supporting both customers and administrator.

There are two types of users in the system

Administrator's functions:

- Register & login to the system
- Add/update/delete appointments
- View appointments
- Finish appointment with the bill amount
- View history

Customer's functions:

- Register & login to the system
- Make an appointment
- Cancel an appointment
- View appointments

Provide a well-designed, user-friendly system addressing the following features:

- System should have different authentication levels. (Administrator, customer)
- Interactive user-friendly interfaces.
- Add suitable sets of reports, which you think will add more value to the entire business.

Tasks

- a) Provide the UML for the given problem with clear explanations on the design decisions. (Use case, Class, ER diagrams, Sequence diagrams etc.). (LO 1) (20 Marks)
- b) Design user interfaces to fulfill all functionalities. (LO 2) (50 Marks)
- c) Include test data and proper test plan for tasks 2 & 4 at the end of this document. (LO 4) (20 Marks)
- d) Create well formatted documentation. (LO 4) (10 Marks)

Acknowledgement

Primarily I thank God for being able to complete this assignment in a successful manner. Thus, I take this opportunity to express my deep sense of gratitude and my profound respect to the lecturer who guided and inspired me in doing this assignment. I had to get the guidance of a respected and responsible person at the preparation time and while continuing the assignment. So, I would like to thank our lecturer Mr. Roy Lan whose valuable guidelines and consultations been the ones that helped me patch this assignment and make this full proof success and finalize this successfully.

And also, his instructions which were given underlying the structure has served as the major contributor towards in completing this assignment and his instructions about the programming and coding was more helpful in implementing this Mobile Application for SalonSD. Through those and by this golden opportunity, I got the full knowledge on basics and some advance in Mobile Application Development as well as in Java Programming. I hope this knowledge you gave me will help me in learning more advance Mobile Application Development as well as my career. I really thankful for you because of the massive courage you had to teach us.

Then I would like to thank Mr. Chathura, Miss Manoda and Miss Ishani who is with us and help us in many sides since the beginning of the Bridging Program. However finally, I am really grateful because I managed to complete this assignment within the time period given by our lecturer Mr. Roy Lan. Thank you all!

Contents

Task 1 - UML Diagrams	10
1.1 - ER Diagram	10
1.2 - Use Case Diagram	14
1.3 - Class Diagram.....	17
1.4 - Sequence Diagrams	20
1.4.1 - Very First Home (Starting) Interface.....	20
1.4.2 - Admin Login.....	21
1.4.3 - Customer Login	22
1.4.4 - Admin Register.....	23
1.4.5 - Customer Register	24
1.4.6 - Admin/Customer View Appointments	25
1.4.7 - Admin/Customer Add/Update/Delete Appointments	26
1.4.8 - Admin Finish Billing Appointments	28
1.4.9 - View Finished Appointments History	29
Task 2 - User Interfaces	30
2.1 - Home (Starting) Interface.....	30
2.2 - Admin Login Interface	32
2.3 - Admin Menu Interface.....	34
2.4 - Admin Registration Interface	35
2.5 - Admin/Customer View Appointments Interface	37
2.6 - Admin Appointment Interface (Add/Update/Delete)	39
2.7 - Admin Finish Billing Appointments Interface	41
2.8 - Admin View Finished Appointments History Interface	43
2.9 - Customer Login Interface.....	45
2.10 - Customer Registration Interface	47
2.11 - Customer Add Appointments Interface	49
2.12 - Database Structure	51
2.12.1 - Admins Database Table	52
2.12.2 - Customers Database Table	53

2.12.3 - Appointments Database Table	54
2.12.4 - Finish Appointments Database Table	55
Task 3 - Testing	56
3.1 - Test Plan	57
3.2 - Test Cases	64

Introduction

This Assignment is regarding the Mobile Application Development Module and this is the seventeenth assignment we got in our HD Program. Basically, this coursework explains mobile operating systems, development tools and technologies for the mobile application development, design mobile application solutions, develop mobile application solutions and test mobile application solutions. Studying about Mobile Application Development is more important for a student who hopes to become a Software Engineer and a Mobile Application Developer as Java Programming directly explains the basics and advance things in building mobile applications and how complex codes are made using the basic stuffs in order to implement a mobile application with a low effort. In my assignment, I have ideally shown some theories used for UML diagrams designing as well as for coding along with the evidences.

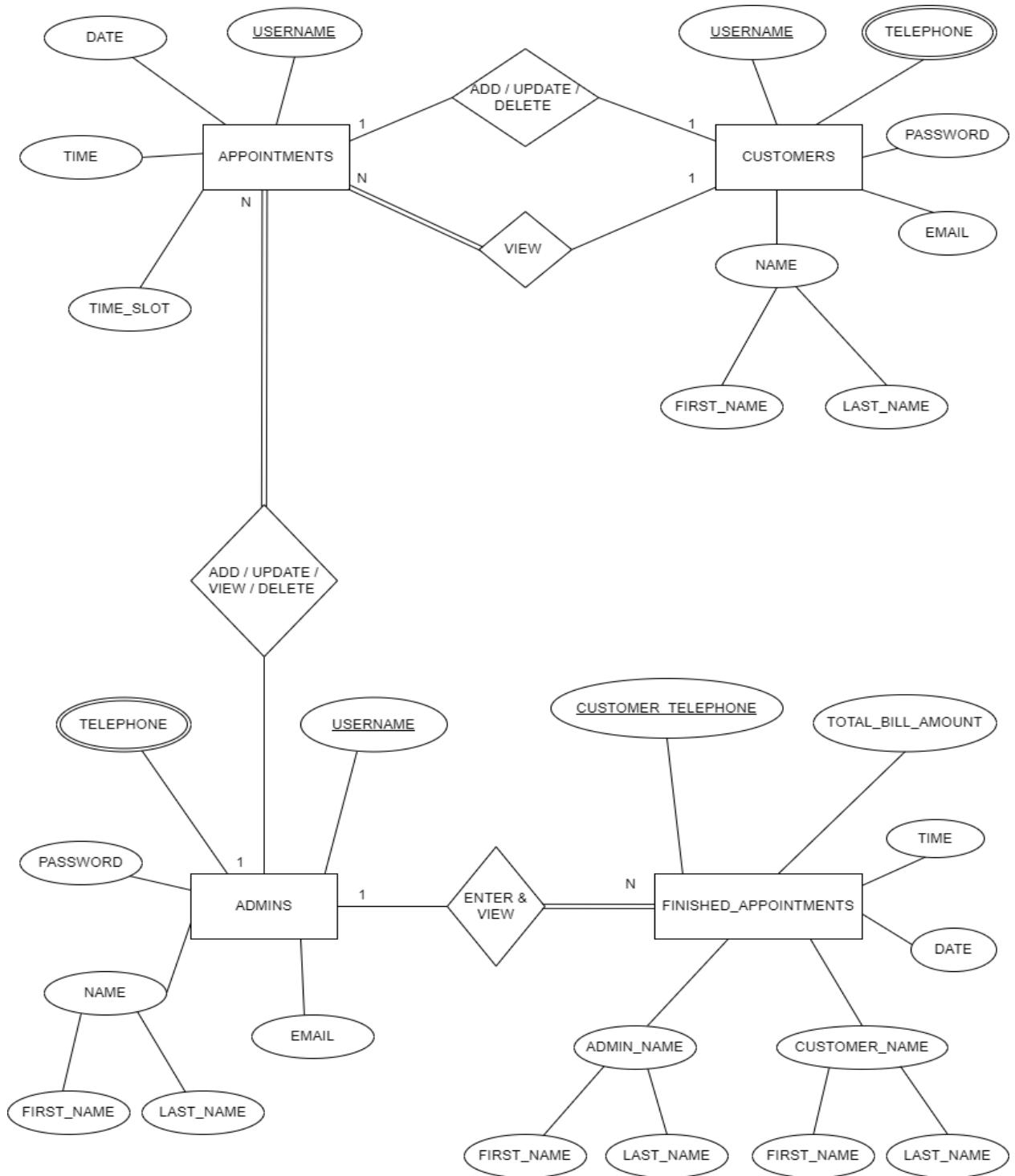
The primary and main objective of implementing this mobile application for Saloon SD is to support both customers and administrators to manage appointments which are currently handled manually. Saloon SD is a one of the famous saloons in Sri Lanka and they have expanded their services within a short period of time. They provide a quality service in hair cutting and give their admins control over their appointments is paramount at all SD Saloons. As this Mobile Application has two different authentication levels as Administrator and Customer, Admin will have the functionalities to register & login to the system, add/update/delete appointments, view booked appointments, finish appointment with the bill amount and view history of the finished appointments. Further, Customer will be able to register & login to the system, make an appointment, update an appointment, cancel an appointment and view booked appointments. In order to implement those functions, I had to include Java programming language which is more considerable when using to develop a Mobile Application in Android Studio. Thus, this SalonSD Mobile Application will help them to expand their business by increasing

customer base as this Mobile Application is more user friendly and easier to understand when using.

This document includes UML Diagrams such as an Entity Relationship diagram, an Use Case diagram, a Class diagram and Sequence diagrams with assumptions and explanations for each. In addition to them, this document consists with all the User Interfaces, analyzations along with description. Further, I have included a Test Plan and all relevant Test Cases to check validations and screenshots of the coding along with Android Virtual Device and assumptions that I have made. Thus, I hope this will be an ideal and clear assignment.

Task 1 - UML Diagrams

1.1 - ER Diagram



Within an information technology system like a Mobile Application, an ER diagram is an illustration which indicates relations with items, people, places, principles or activities. The major advantage of implementing an ERD is the database structure can be simply and perfectly understood after reviewing the diagram. Therefore, the above attached Entity Relationship diagram has been implemented to identify the entities, attributes and relationships in order to develop the database of the SalonSD Mobile Application (Arnicans, 1998).

When it comes to the ERD attached above, there are four entities called Appointments, Customers, Admins and Finished Appointments. Those entities are the database tables used. However, each of those entities consist with some attributes.

In Appointment entity, there are four attributes named as username, date, time and time slot. In that, username will be the primary key as it's the unique attribute in that entity and that's why it is underlined in the diagram. In Customers entity, there are five attributes named as username, telephone, password, email and name. In that, username will be the primary key as it's the unique attribute in that entity and that's why it is underlined in the diagram. Telephone will be a multi valued attribute as the existence of more than one phone number is possible. As there are values that are to be stored in an attribute can be further divided into sub-values and that's why name has drawn as a composite attribute. When it comes to the SalonSD mobile application, only a customer can add, update or delete an appointment which is only for his/her username. One customer shouldn't be able to delete other customers appointments and that's why the relationship between customers and appointments have named as one to one relationship. Only one customer can add, update or delete only one appointment which is for his/her username. But in order to make an appointment, a customer should be able to view all the booked appointments. Then he/she can see the lastly put time slot and reserve the next time slot for him/her. So, one customer can view many appointments which are booked and that's why it has named as one to many relationship. From customers entity, a single line is drawn to the relationship called View as it's a partial participation which means every customer won't make appointments. But from appointment entity, double lines are

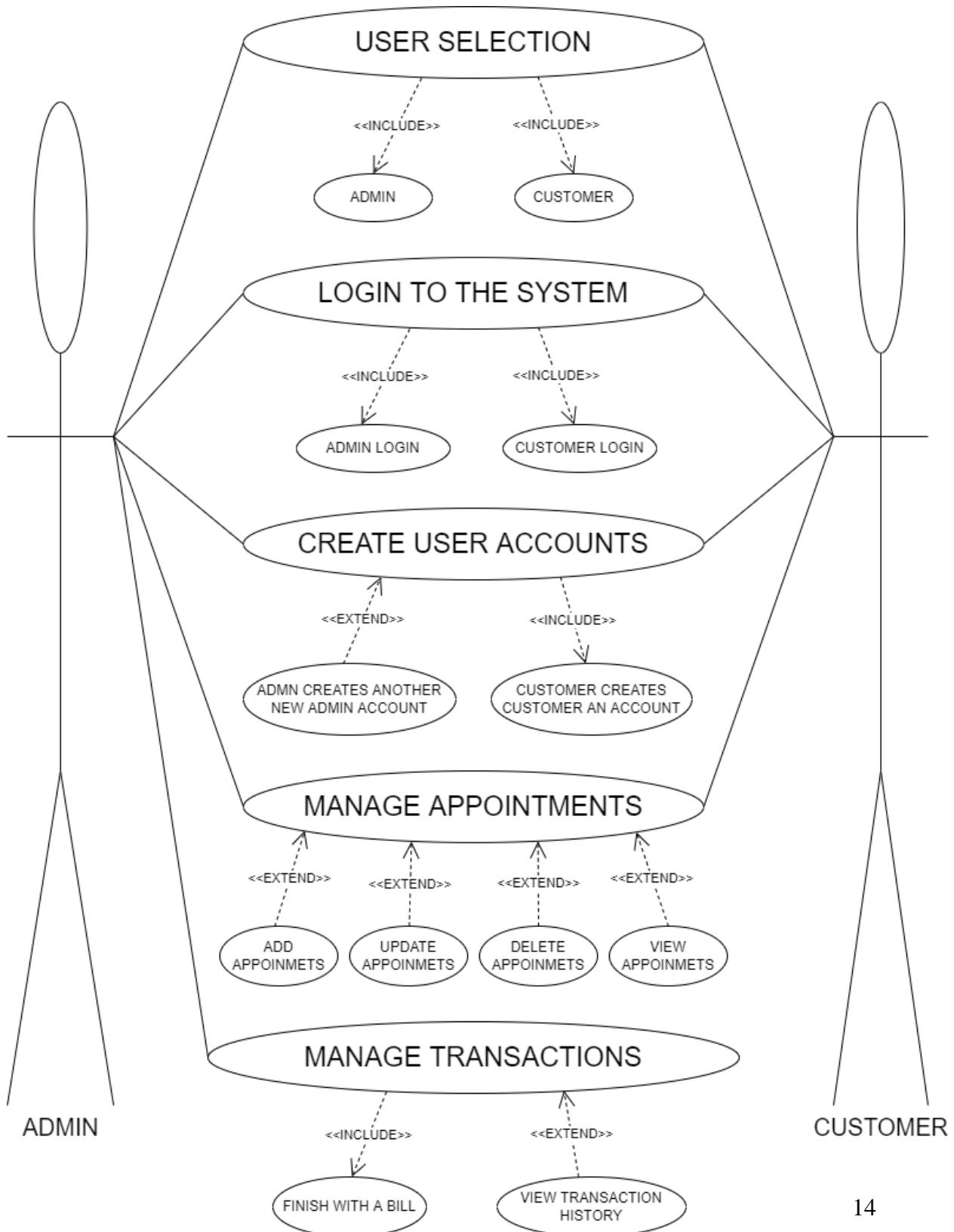
drawn to the relationship as it's a total participation which means one or many appointments can be made by one or many customers.

In Admins entity, there are five attributes named as username, telephone, password, email and name. In that, username will be the primary key as it's the unique attribute in that entity and that's why it is underlined in the diagram. Telephone will be a multi valued attribute as the existence of more than one phone number is possible. As there are values that are to be stored in an attribute can be further divided into sub-values and that's why name has drawn as a composite attribute. When it comes to the SalonSD mobile application, an admin has the functions to add, update, view or delete an appointment and appointment details. No matter that record is for whose username, an admin can update or delete any record from the appointment table and that's why the relationship between admins and appointments have named as one to many relationship. Because one admin can add, update, delete or view any and many appointments. From admin entity, a single line is drawn to the relationship as it's a partial participation which means every admin of the SalonSD won't make appointment. But from appointment entity, double lines are drawn to the relationship as it's a total participation which means one or many appointments can be made by one or many admins.

In Admins entity, there are six attributes named as customer telephone. Total bill amount, time, date, customer name and admin name. In that, customer telephone will be the primary key as it's the unique attribute in that entity and that's why it is underlined in the diagram. As there are values that are to be stored in an attribute can be further divided into sub-values and that's why admin name attribute and customer name attribute has drawn as composite attributes. When it comes to the SalonSD mobile application, an admin has the functions to enter finishing appointment details or view finished appointments and that's why the relationship between admins and finished appointments have named as one to many relationship. Because one admin can enter any and many finishing appointment details or view any or many finished appointments. From admin entity, a single line is drawn to the relationship called Enter & View as it's a partial

participation which means every admin of the SalonSD won't finish billing appointments. But from finished appointments entity, double lines are drawn to the relationship as it's a total participation which means one or many appointments can be finished and view by one or many admins.

1.2 - Use Case Diagram



The Use Case diagrams are being used to identify functions and how roles and actors interact with each of them. A Use Case diagram can be considered as a high-level picture of the system which emphasizes the roles that interact and the functionalities without focusing into the system's inner workings. (smartdraw, 1994-2022)

In the above attached Use Case diagram, there are five major functions. First main use case is User Selection where it's mandatory for a user of Salon SD application to select his/her user type whether user is a customer or an admin. That's why sub use cases are matched to the main use case by using two include arrows. As shown in the diagram, both the users are accessible for that function in the first main use case.

Second main use case is Login To The System where it's mandatory for both the users of Salon SD application to log in to the system in order to make appointments and move to the menu. Both the users have two different Login interfaces for each. As it's mandatory for both users to login in order to make appointments and view menu, sub use cases are matched to the main use case by using two include arrows. As shown in the diagram, both the users are accessible for that function in the second main use case.

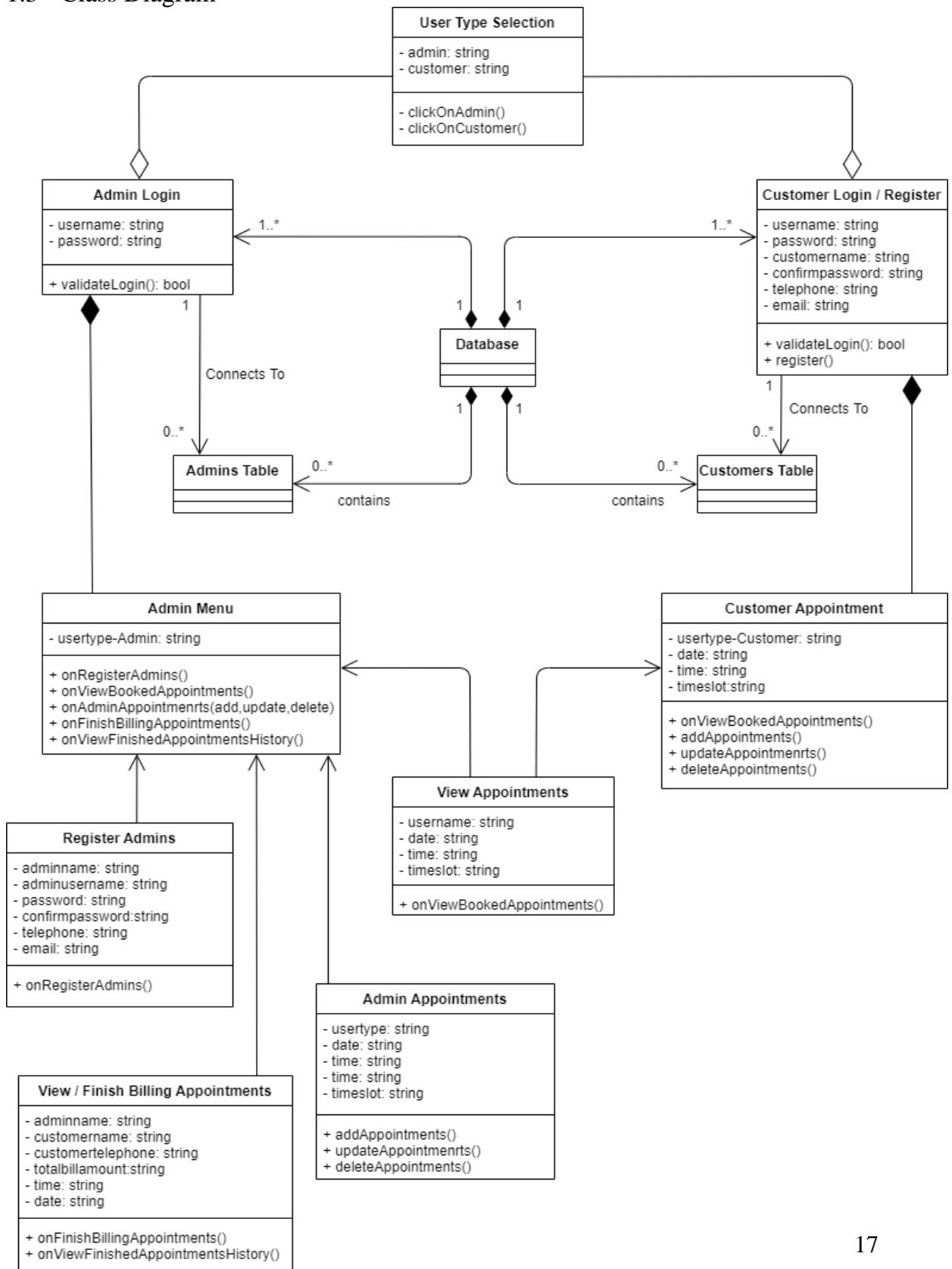
Third main use case is Create User Accounts where it's not mandatory for an admin to create another admin account while it's a mandatory for a customer to register and then log in to make an appointment. Reason why it's not mandatory for an admin to create another admin account is there is a default username and a default password which are hardcoded for the main admin of the SalonSD to log in and manage works. That's why Creates Another Admin Account sub use case is matched to the main use case by using one extend arrow whereas Creates Customer Account sub use case is matched to the main use case by using an include arrow. As shown in the diagram, both the users are accessible for that function in the third main use case.

Fourth main use case is Manage Appointments where it's not mandatory for any user (admin/customer) to add, update, delete or view appointments of Salon SD application. That's why all four sub use cases are matched to the main use case by using four extend arrows. Because if a user wants to add, update, delete or view appointments only, he/she

can do that. As shown in the diagram, both the users are accessible for that function in the fourth main use case.

Last main use case is Manage Transactions where it's mandatory for an admin to Finish With A Bill while it's not mandatory for an admin to View Transaction History. Reason why its mandatory for an admin to finish with a bill is an admin should finish the appointments which are finished for a day. But its not mandatory to view transaction history by an admin. That's why Finish With A Bill sub use case is matched to the main use case by using an include arrow whereas View Transaction History sub use case is matched to the main use case by using an extend arrow. As shown in the diagram, only an admin is accessible for that function in the last main use case.

1.3 - Class Diagram



Class diagrams are essential to the object modeling process because they represent a system's blueprints. Further, Class Diagrams are being used to represent the items that build a particular system like SalonSD Mobile Application, depict their connections and define what those objects perform. Basically, they explain how a system should be implemented. So, they are useful for visualizing, and documenting structural characteristics of SalonSD Mobile Application

System. In Class Diagrams, an attribute inside a class is a significant piece of data containing values that describe each instance of that class. Attributes are also known as fields, variables or properties. After attributes, methods are put which are also known as functions or operations of a system. Methods allow to specify any behavioral feature of a particular class. (JavaTpoint., 2011-2021)

The visibility or "- sign" of an attribute or a method sets the accessibility for the attribute or method. In the above attached Class Diagram, I've used the visibility only for attributes to set the accessibility for those attributes. Further, this "- sign" indicates that all of the attributes are private. Methods inside all classes are public (indicated with + sign) as they can be accessed by any other class.

I have used Aggregation type of association which specify a whole and its parts where a part can exist outside the whole. The aggregation associations have been noted with an open diamond in the above attached Class Diagram.

The very first interface loaded is User Type Selection interface. That's why two methods have been used as one for Admins and the other for Customers. If an admin wants to login, there is a function that validates user login credentials. In both admin and customer login, there is a public method created for validating user returning a Boolean value.

Also, I have used relationships where the part can't exist outside the whole. Those type of relationships have been depicted with composition where I have noted with a colored and closed diamond. In composition type of relationships, a child object wouldn't be able to exist without its parent object.

Moreover, there won't load the Customer Menu interface if there is no any registered and logged in admins. Same as that, customers won't be able to make appointments and do things if there is no any registered and logged in customers. That's why there are two composition relationships (with close diamond) from Admin Login class to Admin Menu class and Customer Login/Register class to Customer Appointment Class. If so, the database records won't exist.

Furthermore, I have used multiplicity in the above class diagram designed for SalonSD Mobile Application. Because Multiplicity allows to set numerical constraints on the relationships. In the class diagram, I have used specific number (1) multiplicity and zero to many (0..*) multiplicity. In attached diagram, it depicts that there can be only one database for the whole Mobile Application and that's why specific number (1) multiplicity has been used. Also there can be only one Admin Login for all the admins to log into the system and that's why specific number (1) multiplicity has been used. Same as for that, there can be only one Customer Login for all the customers of the SalonSD to log into the system and that's why specific number (1) multiplicity has been used there.

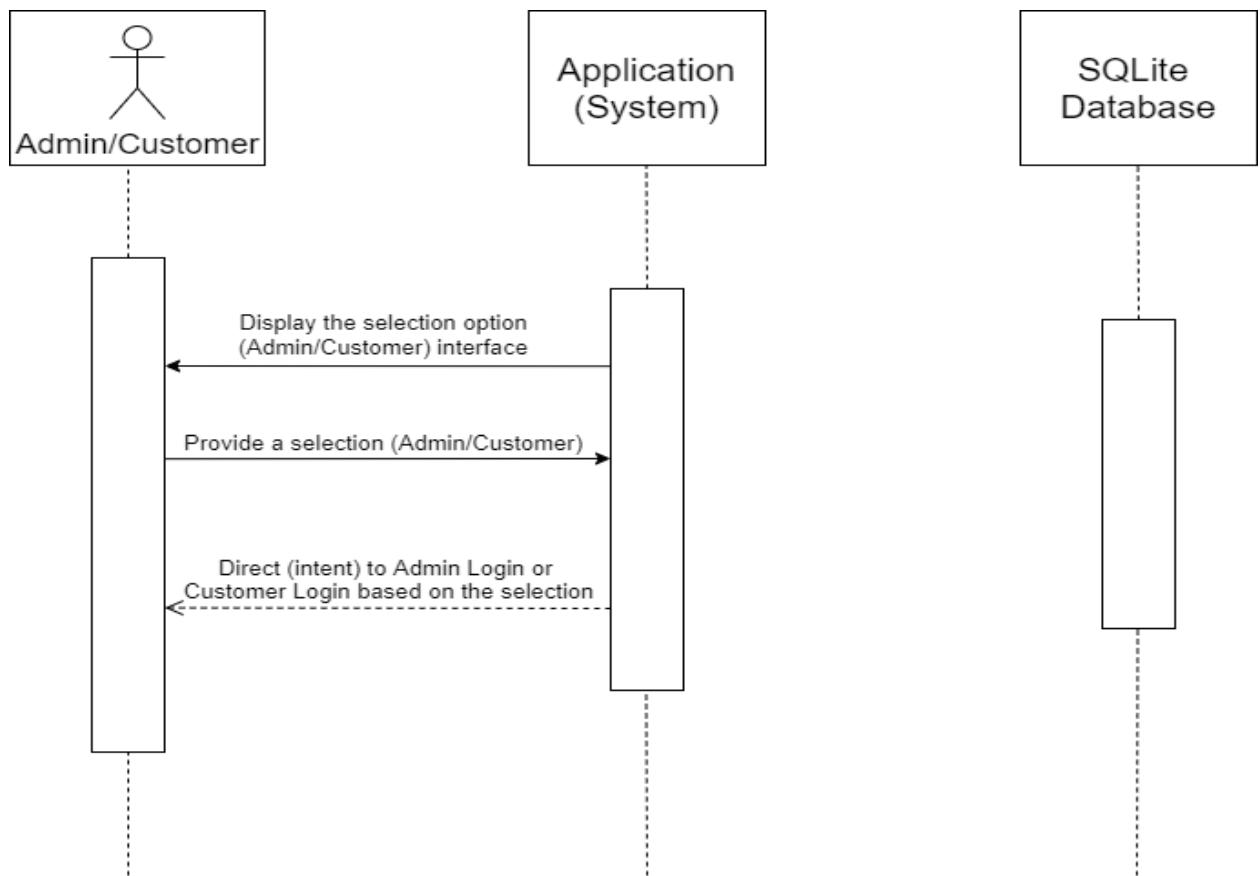
But there can be many numbers of Admin logins as well as Customer Logins. They will be connected with the Admins database table and Customers database table. Inside both Admins and Customers database tables, they may contain multiple user accounts for admins and customers. That's what meant by the areas where zero to many (0..*) notation has been used.

All the classes which are matched to the Admin Menu class are child classes which are inherited from the parent class called Admin Menu. So, Methods inside the Admin Menu class have inherited to the child classes. Same as that, Customer Appointment parent class inherit the method of View Appointments. It's shown that both the Admin Menu and Customer Appointment parent classes inherit a same method called View Appointments which is a common child of both of those parent classes. But Customer Appointment has its own methods too.

1.4 - Sequence Diagrams

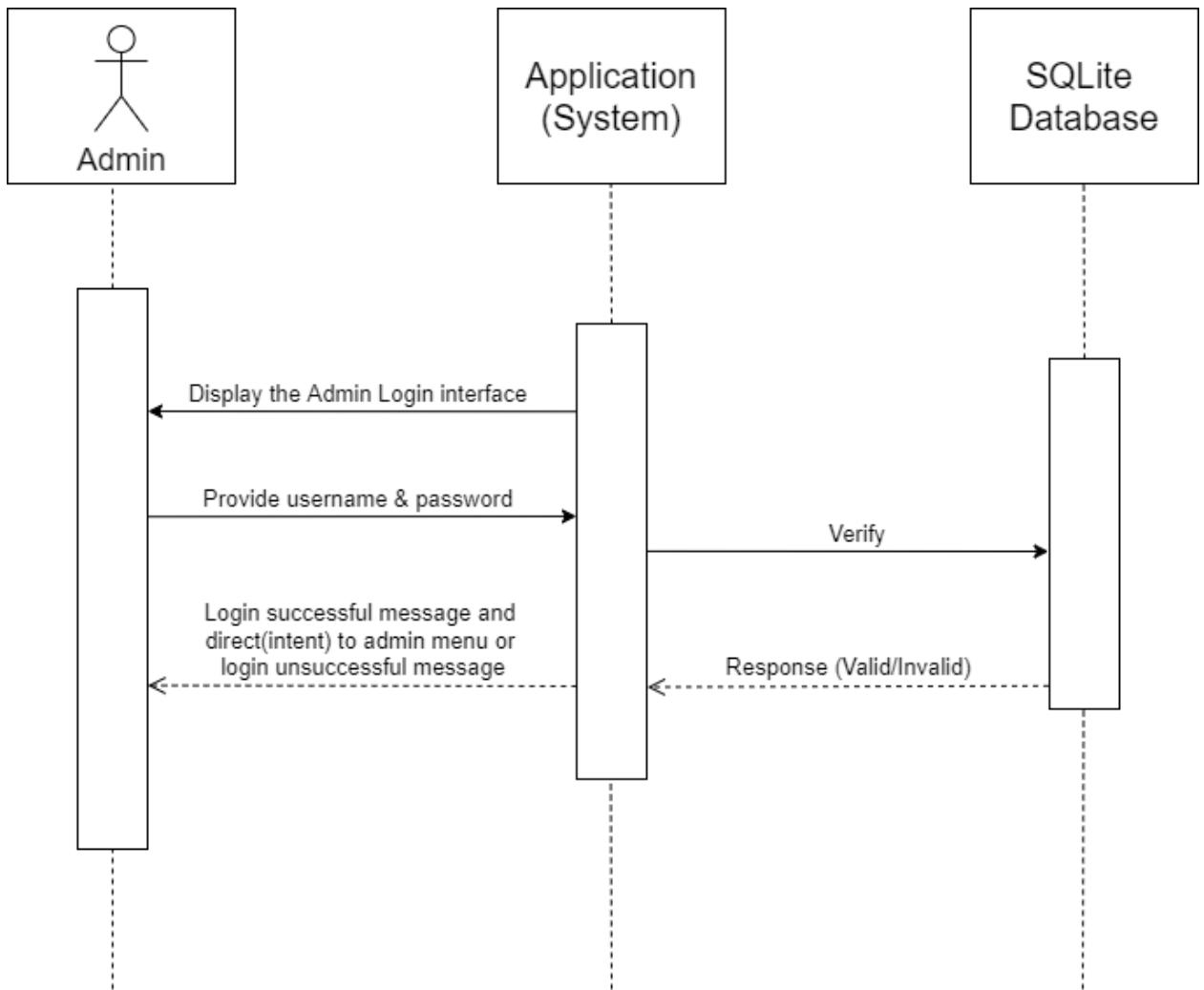
Sequence Diagram is also another kind of a Unified Modeling Language diagrams which describe the flow of messages, events and actions among objects. (Bell, 2004) When it comes to the SalonSD Mobile Application, there can be identified three objects as Admin/Customer, Application (System) and the SQLite Database.

1.4.1 - Very First Home (Starting) Interface



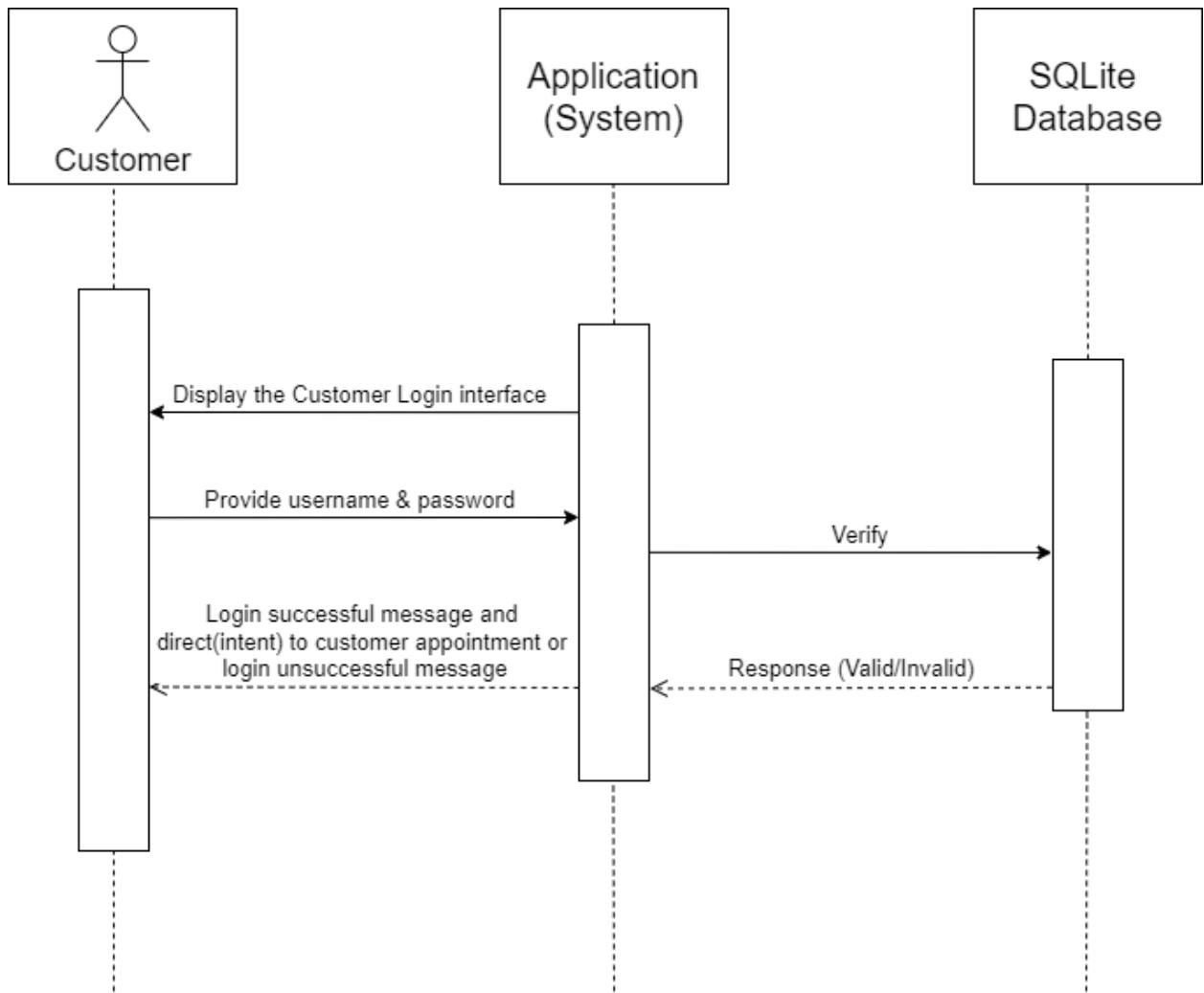
In the above sequence diagram, that is the very first interface which will be visible for both the users admin and customer. So, a user can select the button assigned for his/her user type and click on Admin or Customer. After user entering the relevant button, the system will direct user either to the Admin Login interface or the Customer Login Interface based on the user's input.

1.4.2 - Admin Login



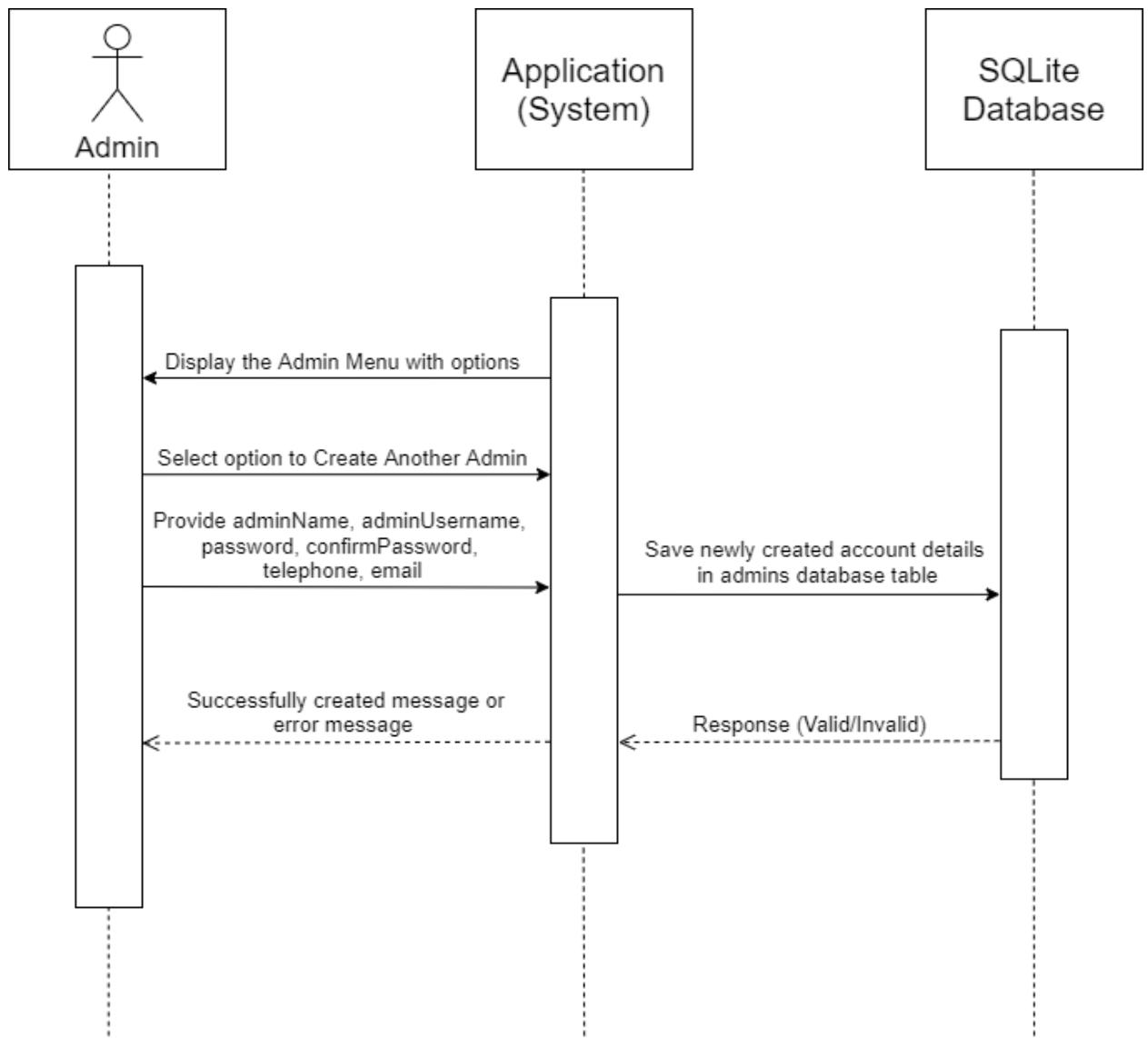
According to the above sequence diagram for Admin Login, the login interface will be displayed by the system for the user. Thus, admin should provide necessary details (username and password) and login. After admin entering the relevant data, the system verifies them with the use of SQLite Database and then return the response as Successful or not to the admin. If the admin entered login credentials are exists in the admins SQLite Database table, he/she is will be eligible to move to the Admin Menu Interface.

1.4.3 - Customer Login



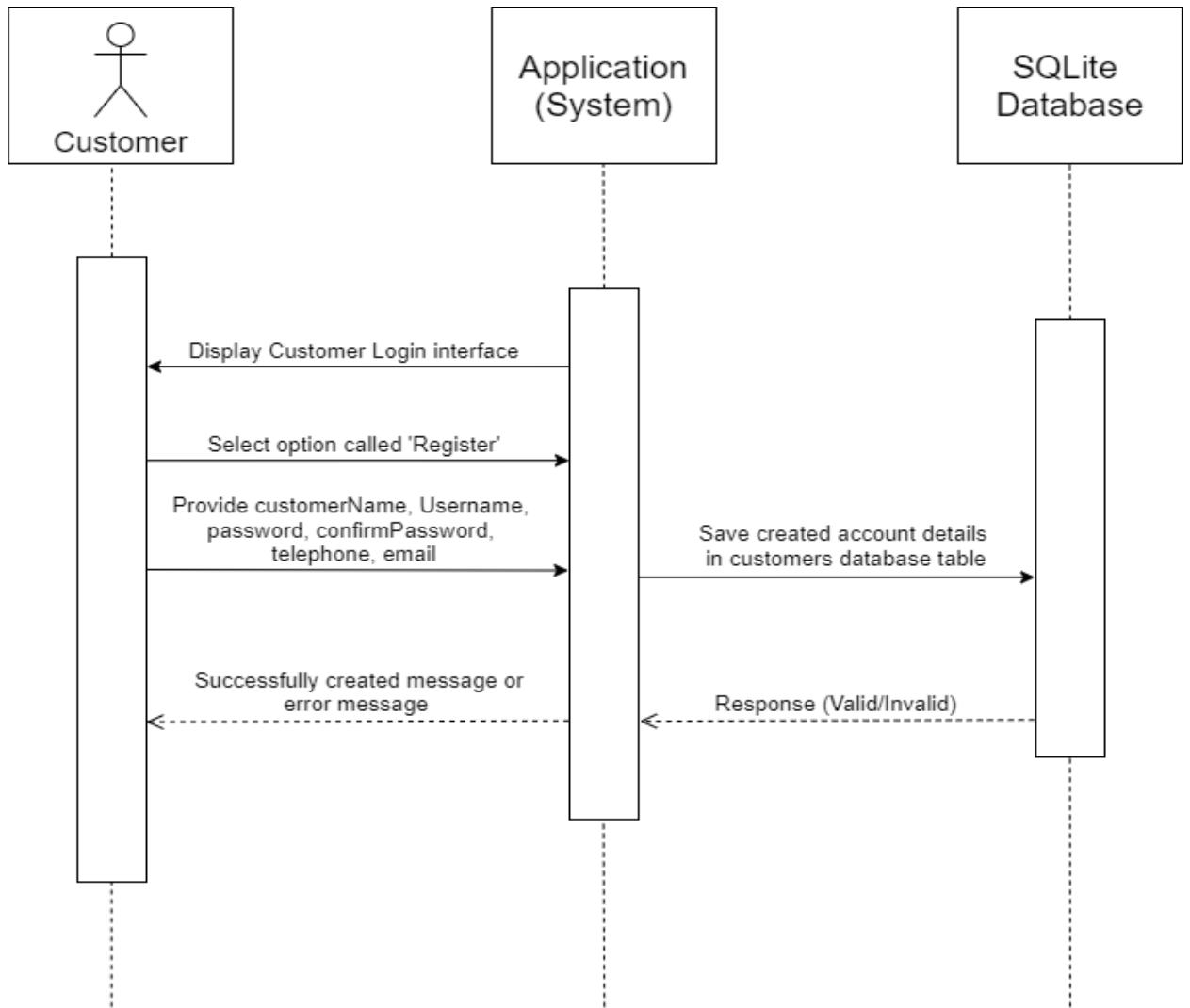
According to the above sequence diagram for Customer Login, the login interface will be displayed by the system for the user. But, a customer should register first in order to login. However, customers should provide necessary details (username and password) and login. After a customer entering the relevant data, the system verifies them with the use of SQLite Database and then return the response as Successful or not to the customer. If the customer entered login credentials are exists in the customers SQLite Database table, he/she is will be eligible to move to the Customer Appointment Interface and go ahead with making an appointment after viewing the booked appointments.

1.4.4 - Admin Register



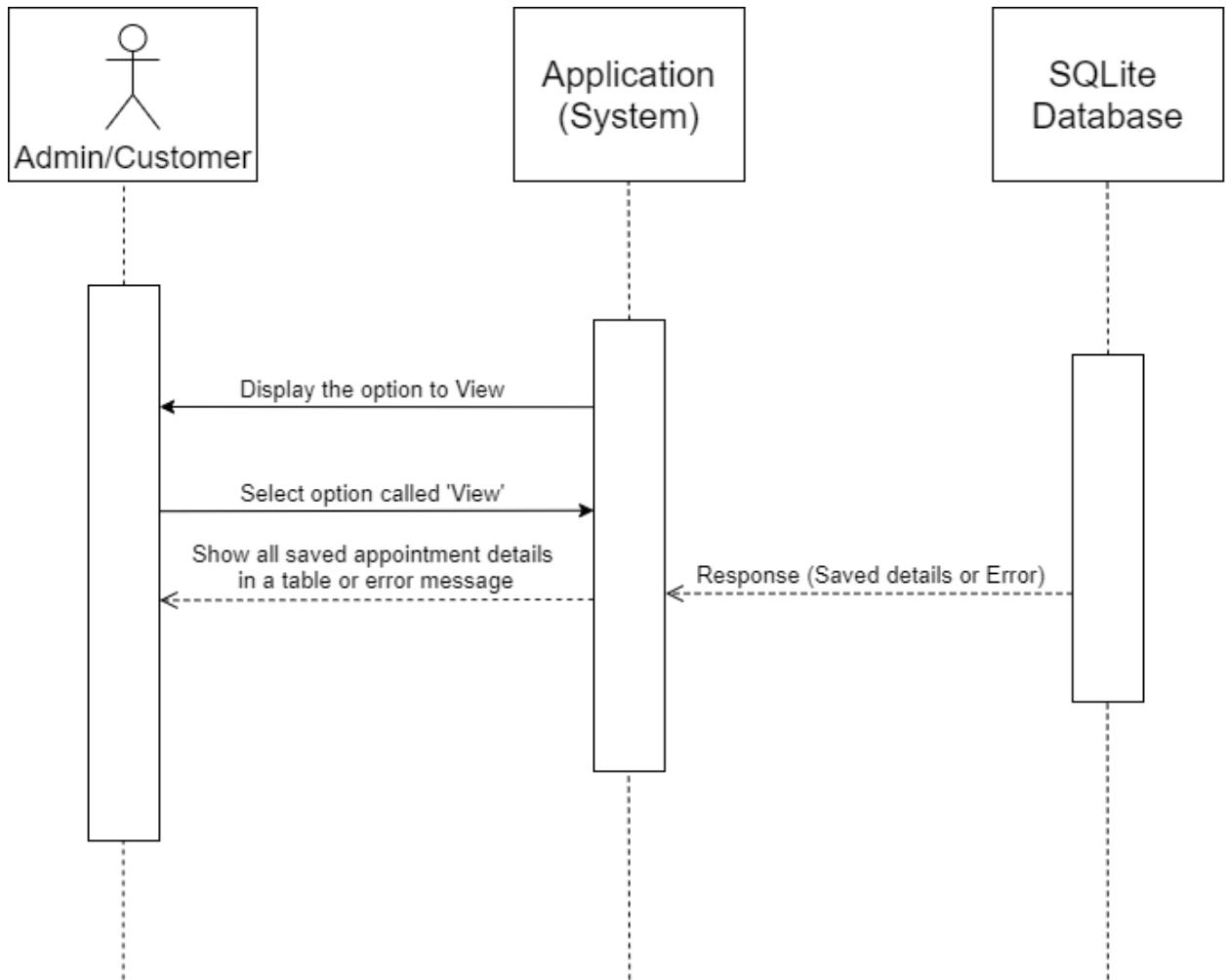
At first it is convenient to mention that this sequence diagram is relatable with admin as an admin can create more new accounts for admins. Thus, in the above sequence diagram for create Admin Account/s (Admin Registration), the system will display the manager menu primarily after logging. Then manager can select option to Create Another Admin Account and provide necessary information. That information will be saved in the relevant admin database table. Finally, a response will be sent to the user (Admin) mentioning whether account is successfully created or not.

1.4.5 - Customer Register



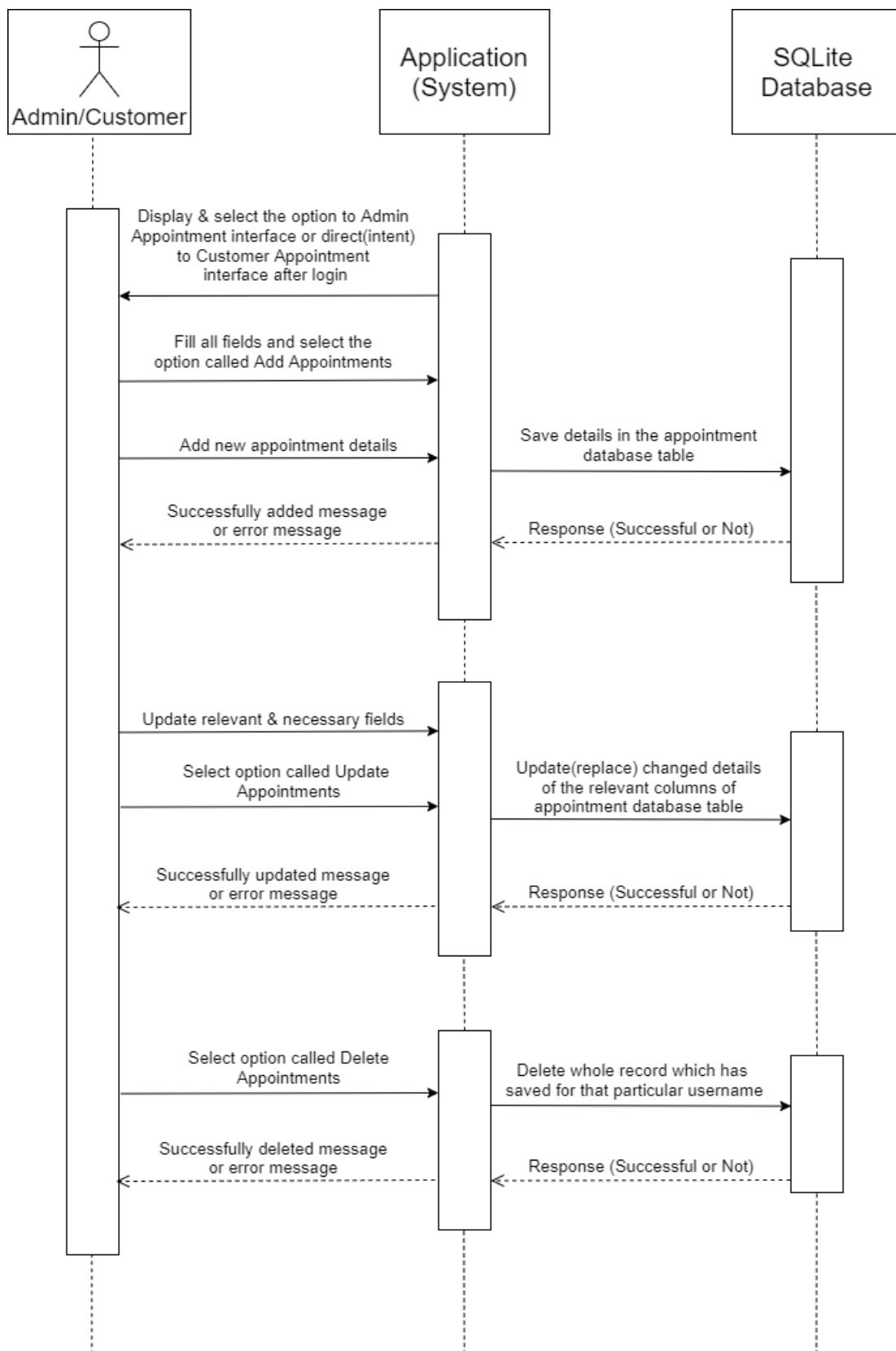
Here also, it is convenient to mention that this sequence diagram is relatable with customers as an customers should be registered first in order to login and make appointments after viewing booked appointments. Here the registration button will be in the login interface and customer should select on it. Then he/she will be directed into the customer registration interface. Thus, in the above sequence diagram for create Customer Registration. Then a customer can provide necessary information register. That information will be saved in the relevant customer database table. Finally, a response will be sent to the user (Customer) mentioning whether account is successfully created or not.

1.4.6 - Admin/Customer View Appointments



As in the above sequence diagram, it's visible that there can be seen the function in the system as View Appointments. This particular function of viewing booked appointments can be accessed by both the users. If any admin wants to view appointments, first the admin menu will be displayed. Then admin has to select the option called View Booked Appointments and a response will be sent to the user by showing all the saved appointment details in a table or an error message. For customer, when they logged in, the Customer Appointment interface will be loaded and there is a transparent button in that interface with the texts on it called “Booked Appointments”. Therefore, customer can click on that and view the appointments.

1.4.7 - Admin/Customer Add/Update/Delete Appointments



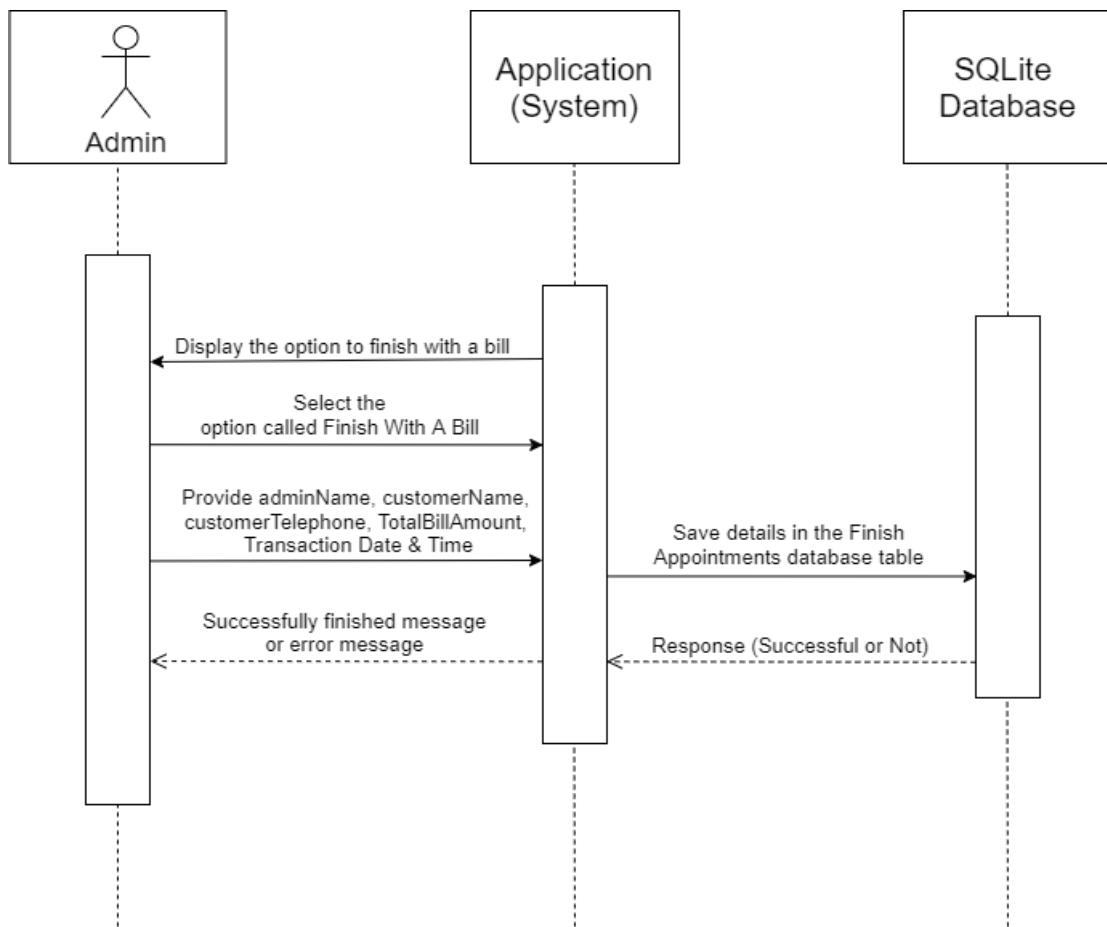
The above sequence diagram showing functions are accessible for both users.

Customers can move to the interface of making an appointment after logged in whereas admin can move to the interface of making an appointment from the admin menu. If any of the users want to add an appointment, it's essential to view the booked appointments before making an appointment. Then user has to select the option called add appointment after entering particular appointment details on relevant fields. After that, particular appointment details will be saved in the appointment database table and a response will be sent to the user as successfully added or not.

If any of the users want to update an appointment, it's essential mention that admin can update any user's appointment details based on the username whereas customer can update only his/her appointment. Then user has to select the option called update appointment after changing particular appointment details on relevant fields. After that, particular appointment details will be updated in the appointment database table and a response will be sent to the user as successfully updated or not.

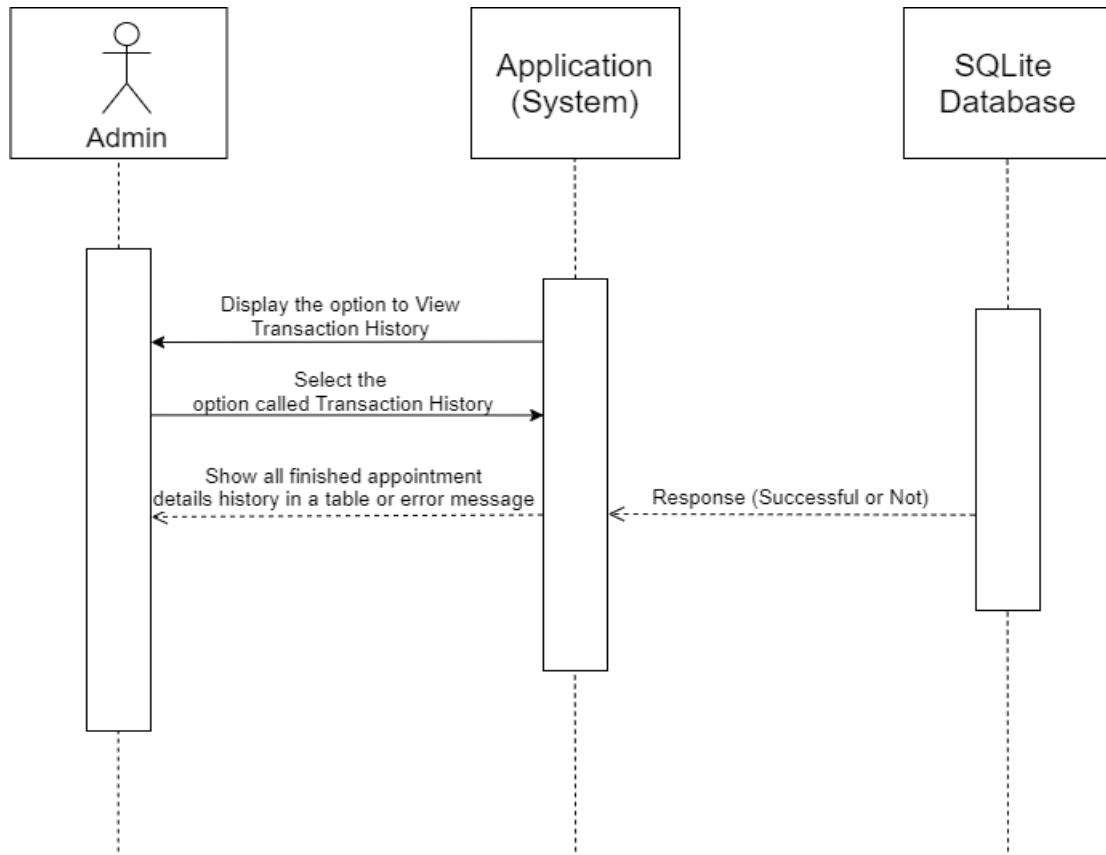
If any of the users want to delete an appointment, it's essential mention that admin can delete any user's appointment based on the username whereas customer can delete only his/her appointment. Admin should enter the deleting appointment's put user's username on the user type field and press on delete button while customer does not have anything to do expect just simply pressing on delete appointment button. After that, particular appointment record will be deleted from the appointment database table and a response will be sent to the user as successfully deleted or not.

1.4.8 - Admin Finish Billing Appointments



The above sequence diagram showing function of finish billing an appointment is accessible only for the user type admin. Admins can move to the interface of finish billing an appointment from the admin menu. Then user has to select the option called Finish With A Bill after entering particular appointment finishing details on relevant fields. After that, particular finished appointment details will be saved in the Finish Appointment database table and a response will be sent to the user as successfully finished or not.

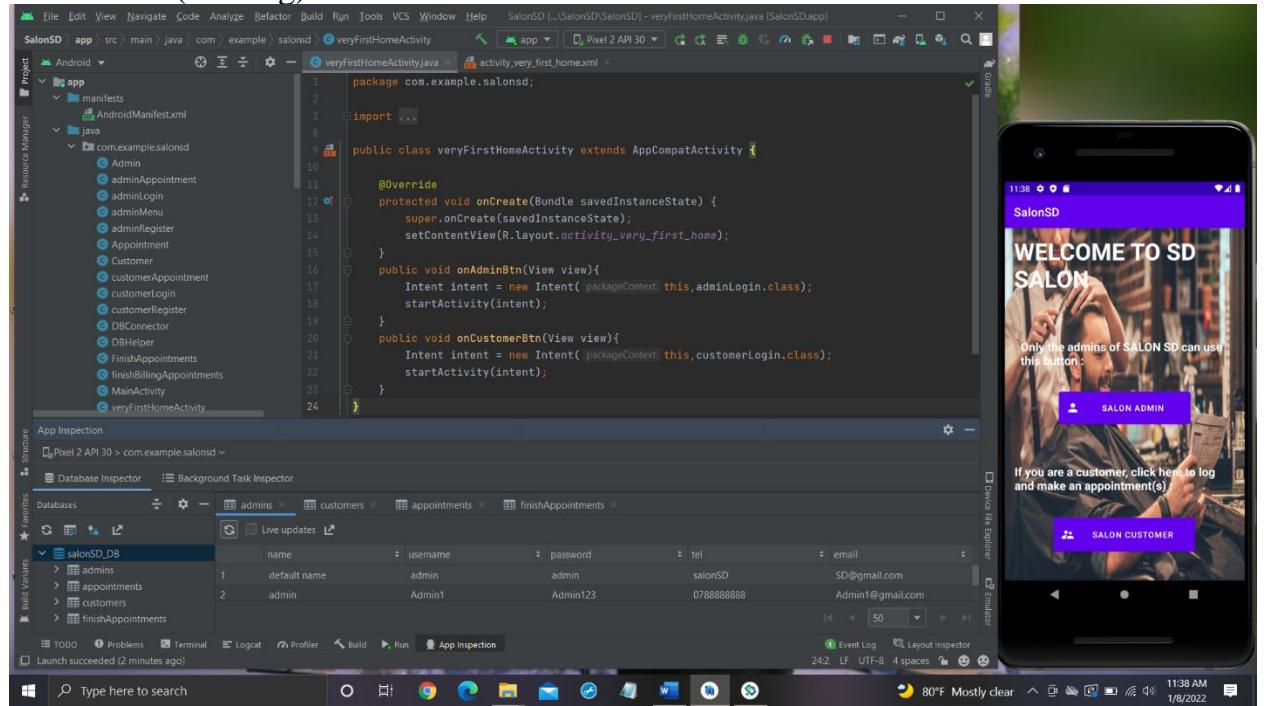
1.4.9 - View Finished Appointments History



As in the above sequence diagram, it's visible that there can be seen the function in the system as View Finished Appointments. This particular function of viewing finished appointments can be accessed only by an admin. If any admin wants to view finished appointments, first the admin menu will be displayed. Then admin has to select the option called View Transaction History and a response will be sent to the user by showing all the saved finished appointment details in a table or an error message.

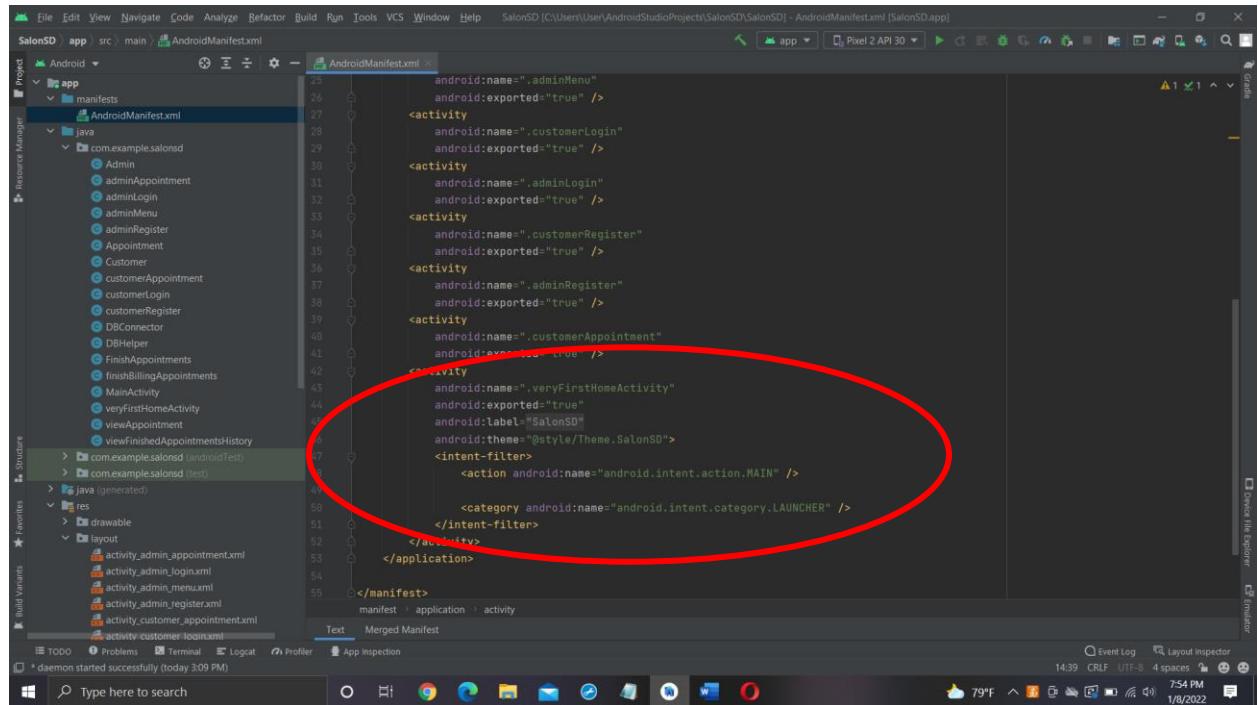
Task 2 - User Interfaces

2.1 - Home (Starting) Interface



The interface shown in the attached screenshot is the very first interface which can be seen by a user. This UI consists with two buttons and three text views along with a background image. This Salon SD Mobile Application has only two user authentications. One user level is Admin while the other one is Customer. From the back-end, inside the class called `veryFirstHomeActivity`, I have created two on clicks in buttons and intent them separately as Admin button to the Admin Login interface and Customer button to the Customer Login. Thus, its clear that the very first interface which loads after run the application simple direct users to their respective login interfaces.

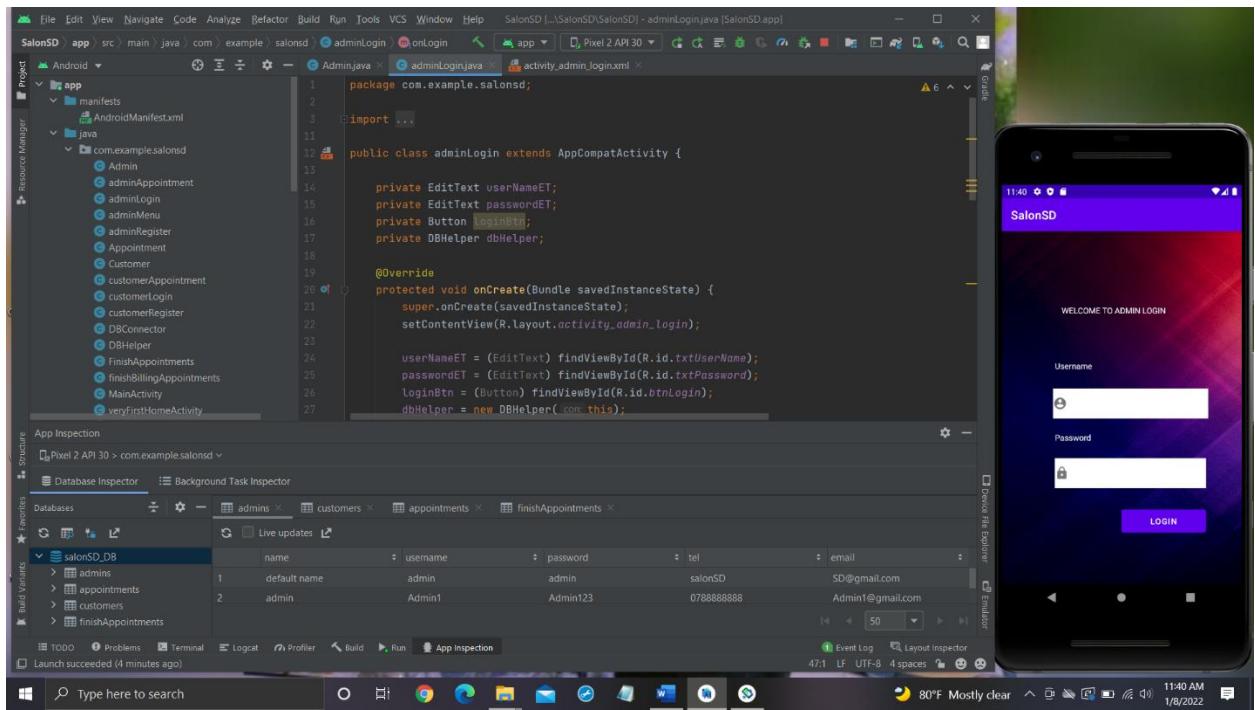
The below screenshot shows the codes used to load the veryFirstHomeActivity activity as the first activity to be loaded after running the application.



The screenshot shows the AndroidManifest.xml file in the Android Studio Project Structure. A red oval highlights the section of code that defines the veryFirstHomeActivity as the main activity:

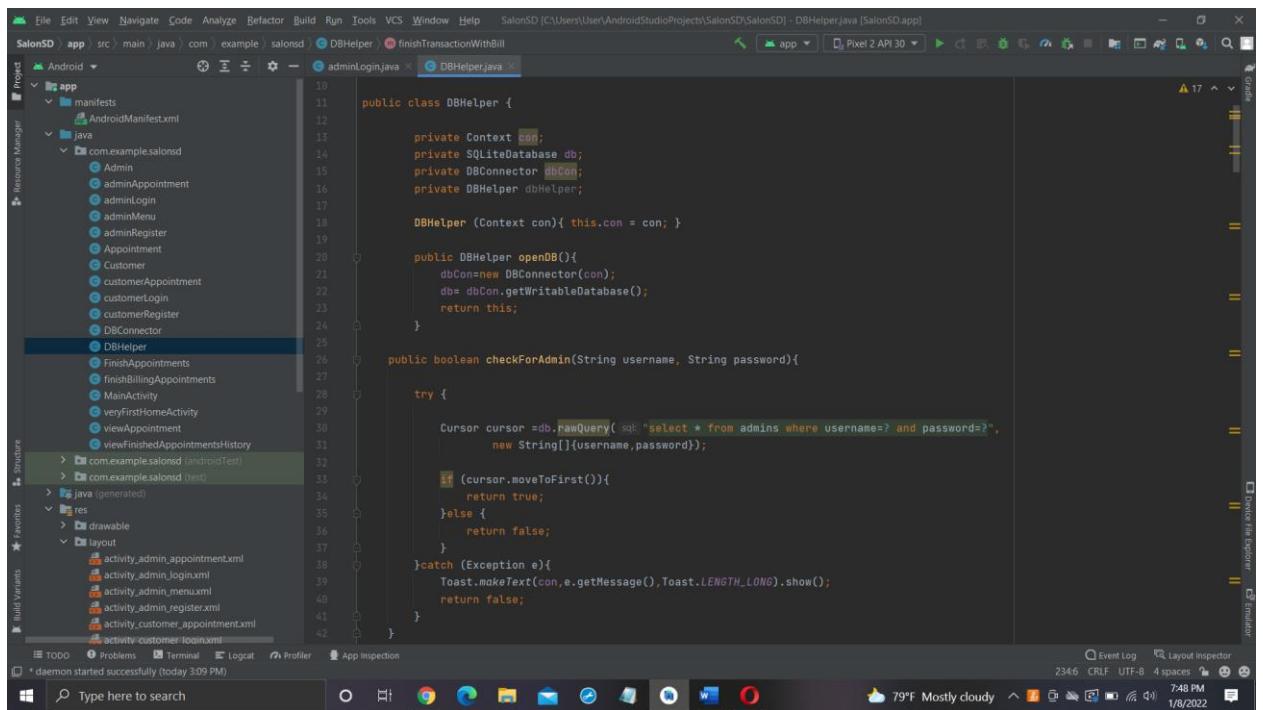
```
        android:name=".veryFirstHomeActivity"
        android:exported="true"
        android:label="SalonSD"
        android:theme="@style/Theme.SalonSD">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

2.2 - Admin Login Interface



The interface shown in the attached screenshot can be seen by only a user who click on the admin button from the very first loaded interface. This Login interface is only for the Admins of the SalonSD. This UI consists with one button, two plain texts and three text views along with a background image. From the back-end, inside the class called adminLogin, I have created one on click in login button and intent it to the admin menu where admin functionalities are there. Here, when an admin sees this interface, he/she will have a default username called “admin” and a default password called “admin”. Only the admin who is with those two-default username and password can see the admin menu and create another admin if he/she wants. The validations used for this admin login interfaces include are empty field validation and correct username/password validation. Further, toast messages have used to show the messages inside the validations. All those coding has done inside the on click of the login button.

The below screenshot shows the database connection that is used to check the correctness of an admin's username and password. Select query has been used for that.



The screenshot displays the Android Studio interface with the project 'SalonSD' open. The left sidebar shows the project structure with packages like 'com.example.salonsd' containing classes such as Admin, adminAppointment, adminLogin, adminMenu, adminRegister, Appointment, Customer, customerAppointment, customerLogin, customerRegister, DBConnector, DBHelper, FinishAppointments, finishBillingAppointments, MainActivity, veryFirstHomeActivity, viewAppointment, and viewFinishedAppointmentsHistory. The right pane shows the code editor with the 'DBHelper.java' file open. The code defines a DBHelper class that interacts with an SQLite database to check if an admin's credentials are correct. A cursor is used to execute a select query on the 'admins' table where the username and password match the provided parameters. If a row is found, it returns true; otherwise, it returns false. An exception is caught to handle any errors that might occur during the database operation.

```
public class DBHelper {
    private Context con;
    private SQLiteDatabase db;
    private DBConnector dbCon;
    private DBHelper dbHelper;

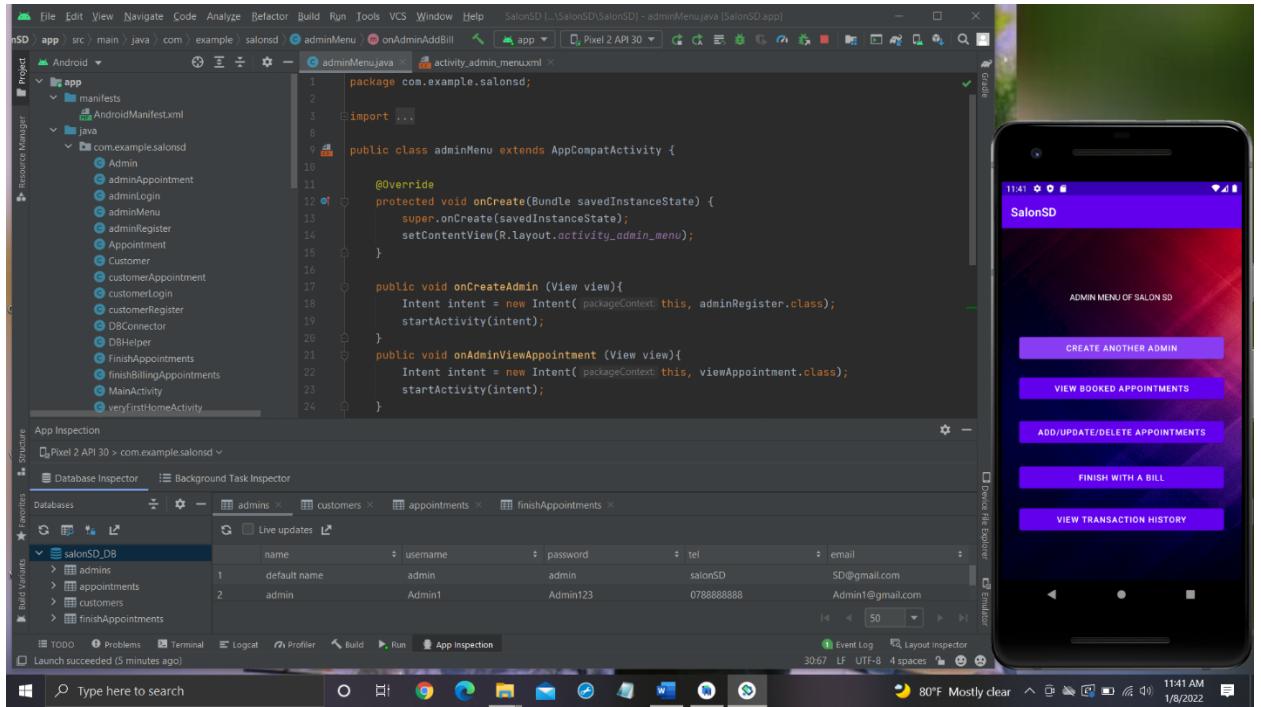
    DBHelper (Context con){ this.con = con; }

    public DBHelper openDB(){
        dbCon=new DBConnector(con);
        db= dbCon.getWritableDatabase();
        return this;
    }

    public boolean checkForAdmin(String username, String password){
        try {
            Cursor cursor=db.rawQuery("select * from admins where username=? and password=?", new String[]{username,password});

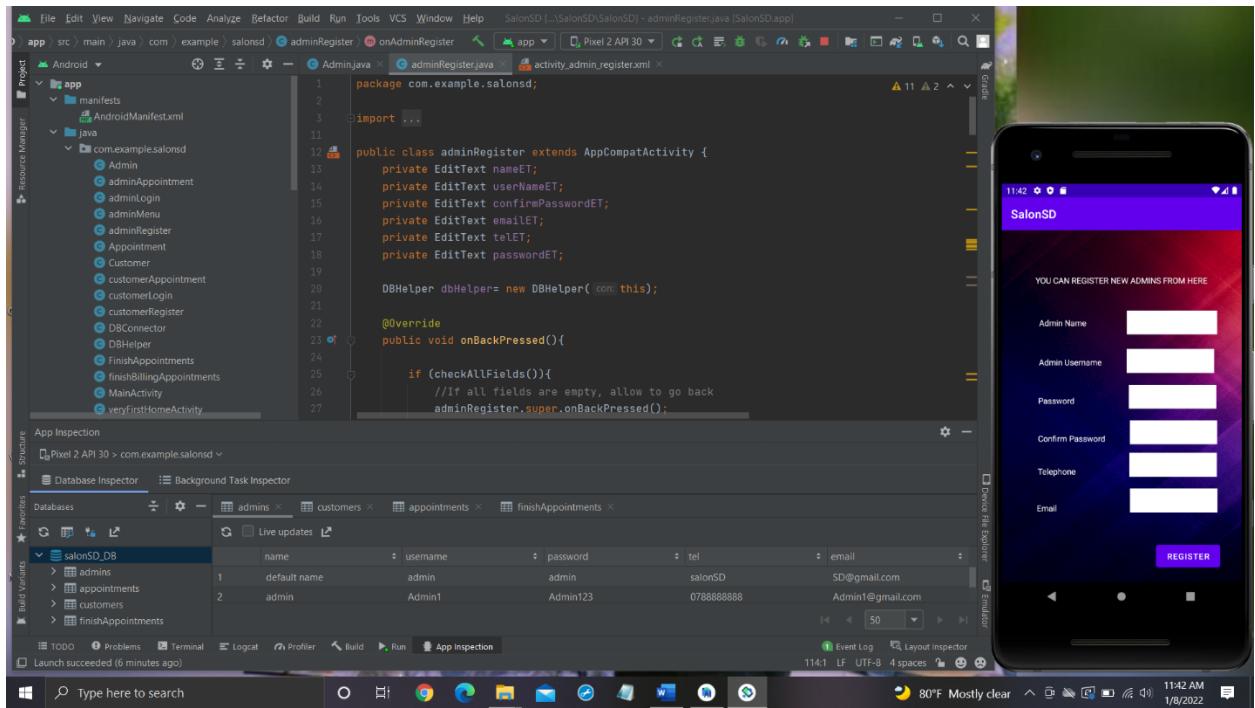
            if(cursor.moveToFirst()){
                return true;
            }else {
                return false;
            }
        }catch (Exception e){
            Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
            return false;
        }
    }
}
```

2.3 - Admin Menu Interface



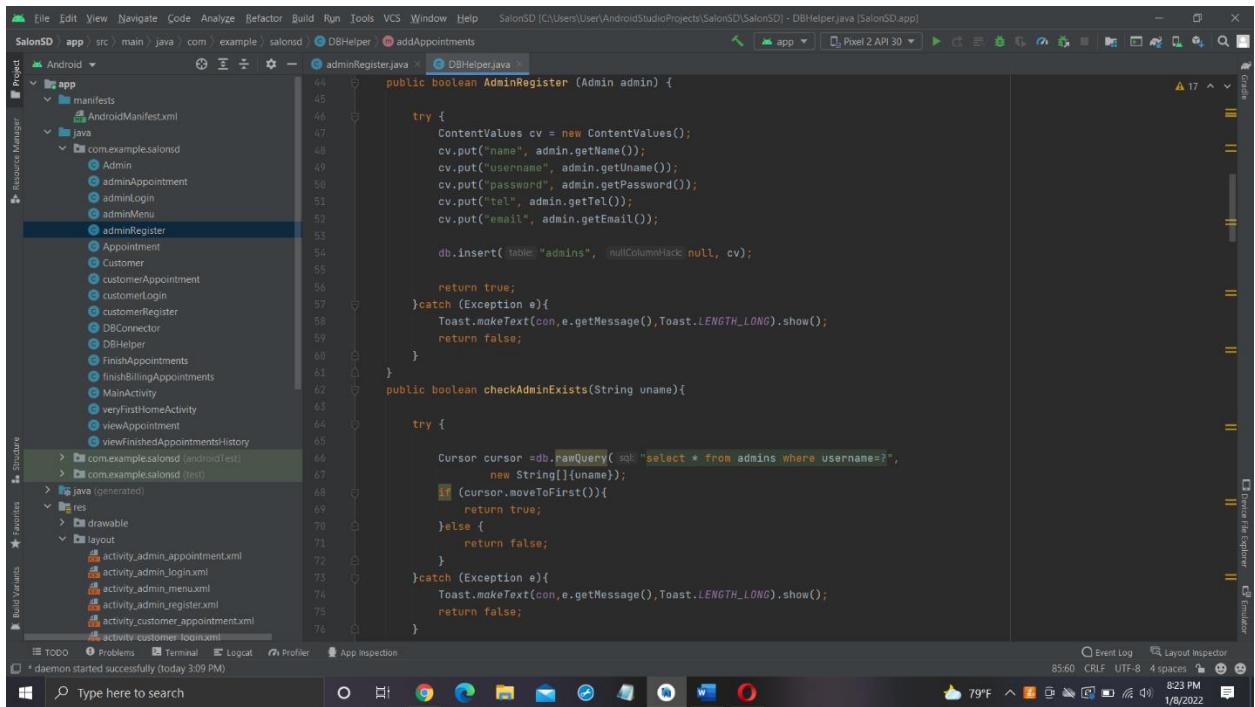
The interface shown in the above screenshot can be only seen by an admin. Because admin menu is the interface which loads when an admin entered correct username/password and logged in. This UI consists with five buttons and one text view along with a background image. From the back-end, inside the class called veryFirstHomeActivity, I have created five on clicks in buttons and intent them separately as Create Another Admin button to the Admin Register interface, View Booked Appointments button to the View Appointments interface, Add/Update/Delete Appointment button to the Admin Appointment interface, Finish With A Bill button to the Finish Billing Appointments interface and View Transaction History button to View Finished Appointment History interface. Thus, it's clear that above shown interface can be access only by an admin whos with a valid admin username and a valid admin password.

2.4 - Admin Registration Interface



The interface shown in the attached screenshot will be appeared only when admin clicks Create Another Admin button from the Admin Menu. Only after the main admin log using default username and password can create another admin. This UI consists with one button, six plain texts and seven text views along with a background image. From the back-end, inside the class called adminRegister, I have created all the validations for the admin registration. The validations used for this admin registration include are empty field validation, password and confirm password matching validation, email validation and username validation which displays the message as a user already exists or not. Further, toast messages have used to show the messages inside the validations. All those coding has done inside the on click of the register button.

The below screenshot shows the methods and database connections that are used to check for existing users where select query has been used and the query that has used for inserting data obtain from the admin register interface.



```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SalonSD [C:\Users\User\AndroidStudioProjects\SalonSD] - DBHelper.java [SalonSD.app]
SalonSD app src main java com example salonsd DHelper addAppointments
adminRegister.java DBHelper.java
public boolean AdminRegister (Admin admin) {
    try {
        ContentValues cv = new ContentValues();
        cv.put("name", admin.getName());
        cv.put("username", admin.getUsername());
        cv.put("password", admin.getPassword());
        cv.put("tel", admin.getTel());
        cv.put("email", admin.getEmail());

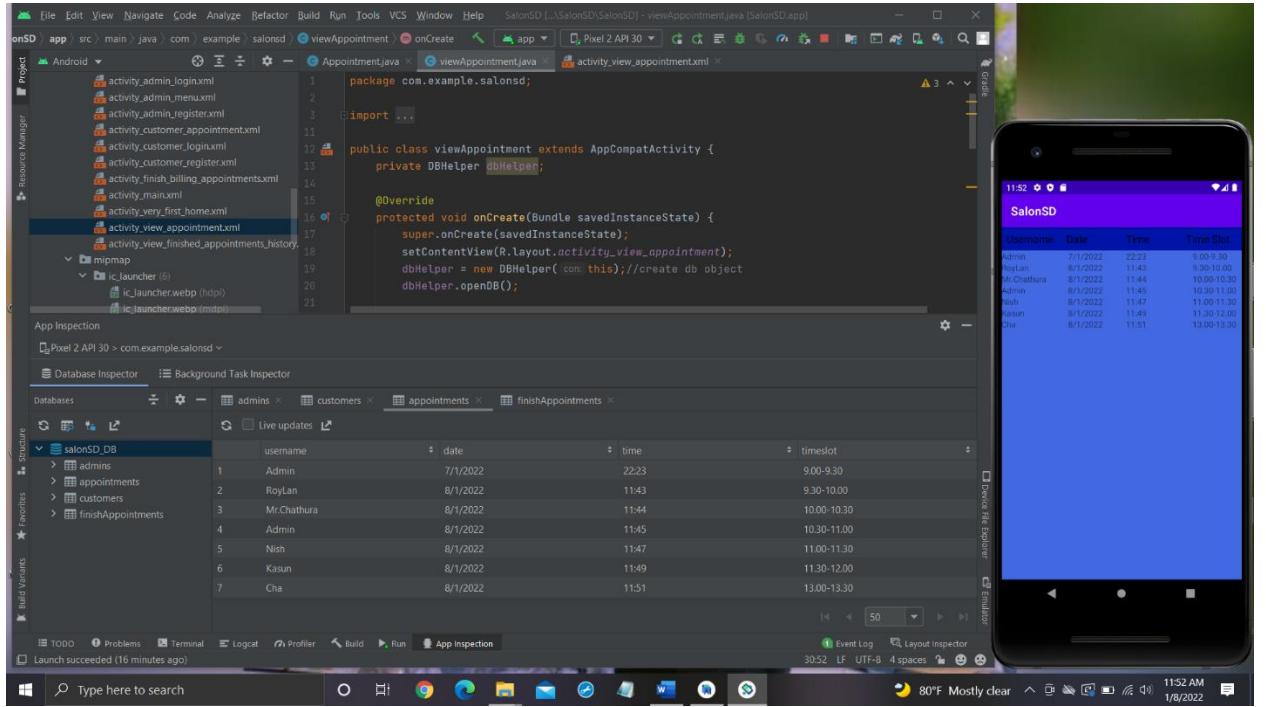
        db.insert("admins", nullColumnHack: null, cv);

        return true;
    } catch (Exception e) {
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

public boolean checkAdminExists(String uname){
    try {
        Cursor cursor = db.rawQuery("select * from admins where username=?", new String[]{uname});
        if(cursor.moveToFirst()){
            return true;
        } else {
            return false;
        }
    } catch (Exception e) {
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

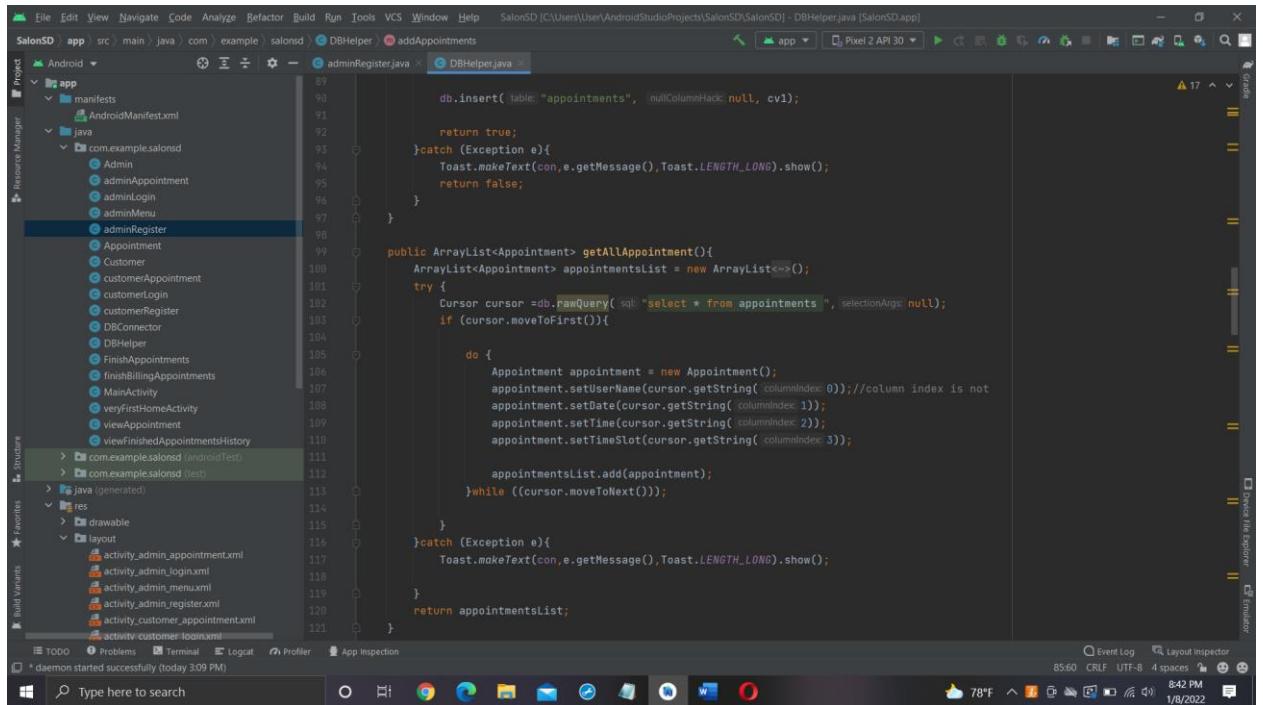
```

2.5 - Admin/Customer View Appointments Interface



This above attached screenshot of the interface will be loaded when an admin clicks on the View Booked Appointments button from the Admin Menu interface. This is a common interface for both admins and customers. Customer can see this interface when making an appointment. The layout used for this interface is a table layout. The table consists of four columns as username, date, time and time slot. There is no attached background image for this interface. This interface is with the blue color in the background. From the back-end, inside the class called `viewAppointments`, I have designed the table layout to display columns and rows along with an ArrayList.

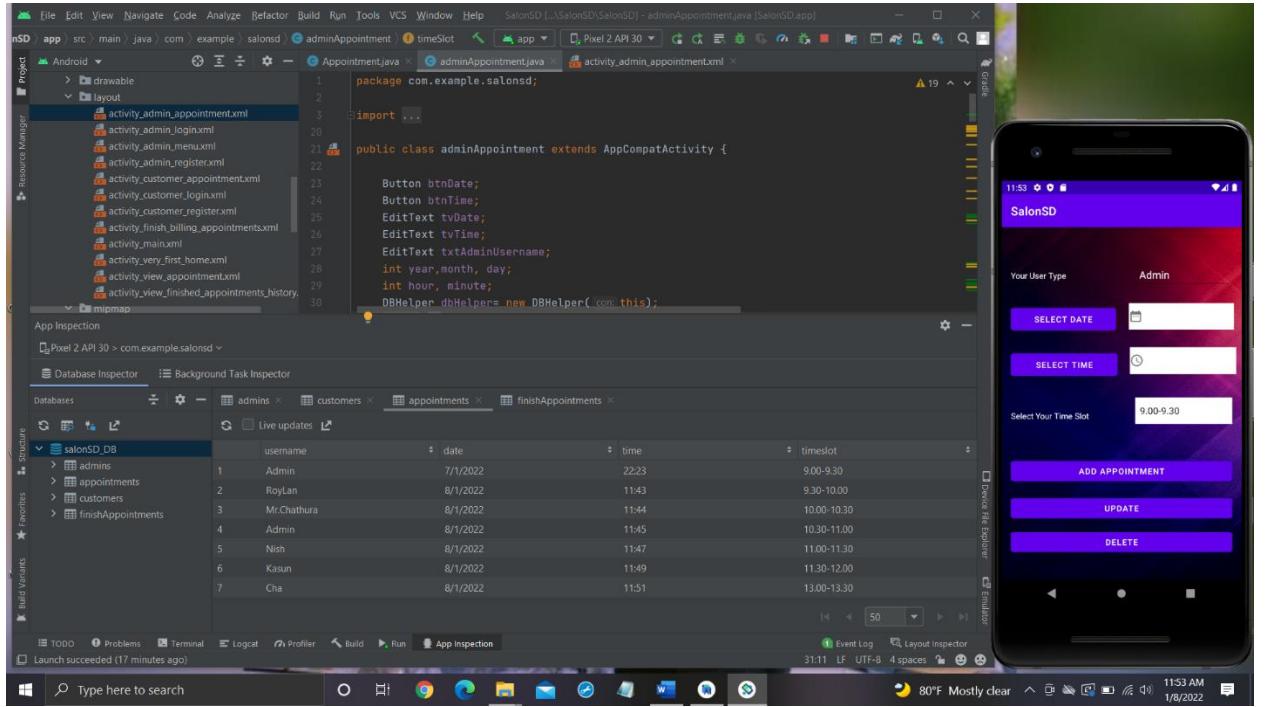
The below screenshot shows the methods and database connections that are used to retrieve booked appointment details with the select query.



The screenshot shows the Android Studio interface with the project 'SalonSD' open. The code editor displays the `DBHelper.java` file, which contains Java code for managing appointments in a database. The code includes methods for inserting and retrieving appointment data. The project structure on the left shows various Java files, XML layouts, and test files. The bottom status bar indicates the device is running at 78°F and the time is 8:42 PM on 1/8/2022.

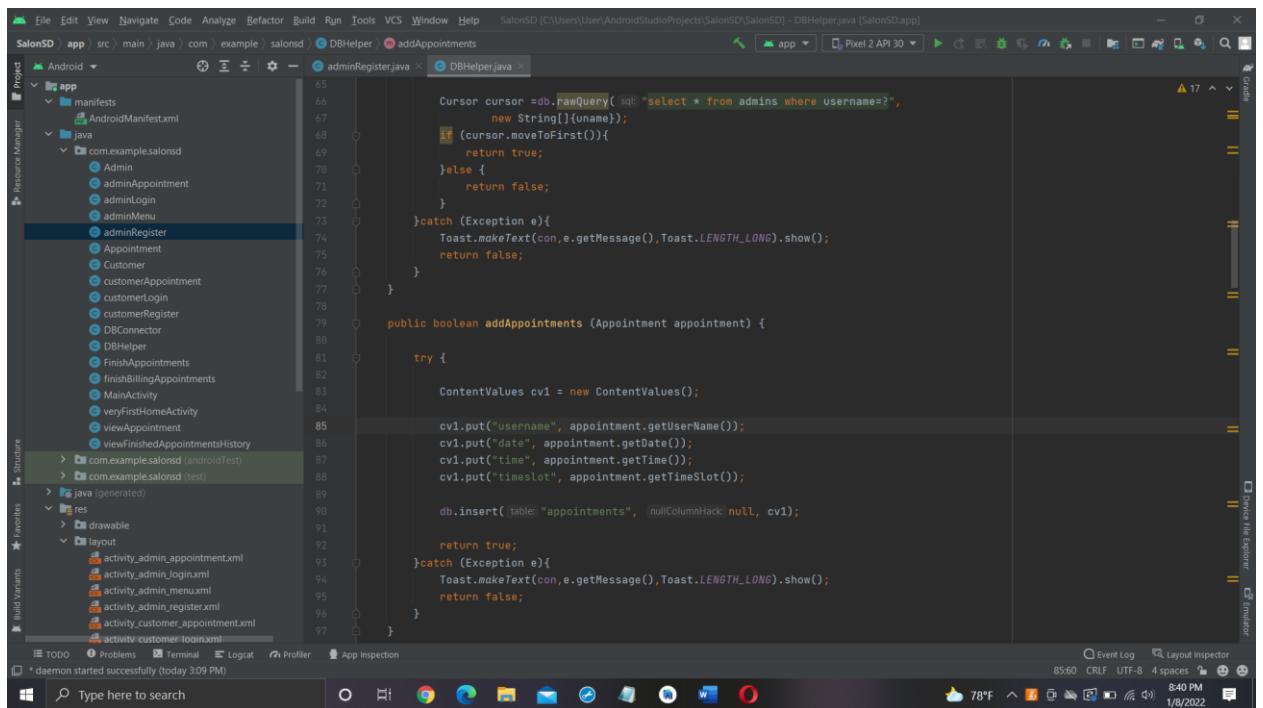
```
89         db.insert("appointments", nullColumnHack, cv1);
90
91     }
92     catch (Exception e){
93         Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
94         return false;
95     }
96 }
97
98 public ArrayList<Appointment> getAllAppointment(){
99     ArrayList<Appointment> appointmentsList = new ArrayList<>();
100    try {
101        Cursor cursor =db.rawQuery("select * from appointments ", selectionArgs);
102        if (cursor.moveToFirst()){
103            do {
104                Appointment appointment = new Appointment();
105                appointment.setUserName(cursor.getString( columnIndex: 0)); //column index is not
106                appointment.setDate(cursor.getString( columnIndex: 1));
107                appointment.setTime(cursor.getString( columnIndex: 2));
108                appointment.setTimeSlot(cursor.getString( columnIndex: 3));
109
110                appointmentsList.add(appointment);
111            }while ((cursor.moveToNext()));
112        }
113        catch (Exception e){
114            Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
115        }
116    }
117    return appointmentsList;
118 }
```

2.6 - Admin Appointment Interface (Add/Update/Delete)



The interface shown in the attached screenshot will be appeared only when admin clicks Add/Update/Delete Appointment button from the Admin Menu. This UI consists with five buttons, four plain texts and two text views along with a background image. There is a date and time picker to get the current date and the time. Time slot can be selected only after viewing the booked appointments and a spinner has used for that. From the back-end, inside the class called `adminAppointment`, I have created on click listeners for date and time picker and three on clicks have been made separately for three buttons called Add, Update, Delete and validations have included inside them. The validations used for this admin appointment include are empty field validation and successfully inserted or not validation. Further, toast messages have used to show the messages inside the validations.

The below screenshot shows the methods and database connections that are used to insert appointment details.



```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SalonSD [C:\Users\User\AndroidStudioProjects\SalonSD] - DBHelper.java [SalonSD.app]
SalonSD app src main java com.example.salonsd DBHelper addAppointments
Project Android
app manifests AndroidManifest.xml
java com.example.salonsd Admin AdminAppointment AdminLogin AdminMenu AdminRegister Appointment Customer customerAppointment customerLogin customerRegister DBConnector DBHelper DBFinishAppointments DBFinishBillings MainActivity veryFirstHomeActivity viewAppointment viewFinishedAppointmentsHistory
com.example.salonsd (androidTest)
com.example.salonsd (test)
java (generated)
res drawable
layout activity_admin_appointment.xml activity_admin_login.xml activity_admin_menu.xml activity_admin_register.xml activity_customer_appointment.xml activity_customer_login.xml
Build Variants Favorites Structure Build Variants
adminRegister.java DBHelper.java
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
Cursor cursor = db.rawQuery("select * from admins where username=?", new String[]{username});
if (cursor.moveToFirst()){
    return true;
} else {
    return false;
}
} catch (Exception e){
    Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
    return false;
}

public boolean addAppointments (Appointment appointment) {
try {
    ContentValues cv1 = new ContentValues();

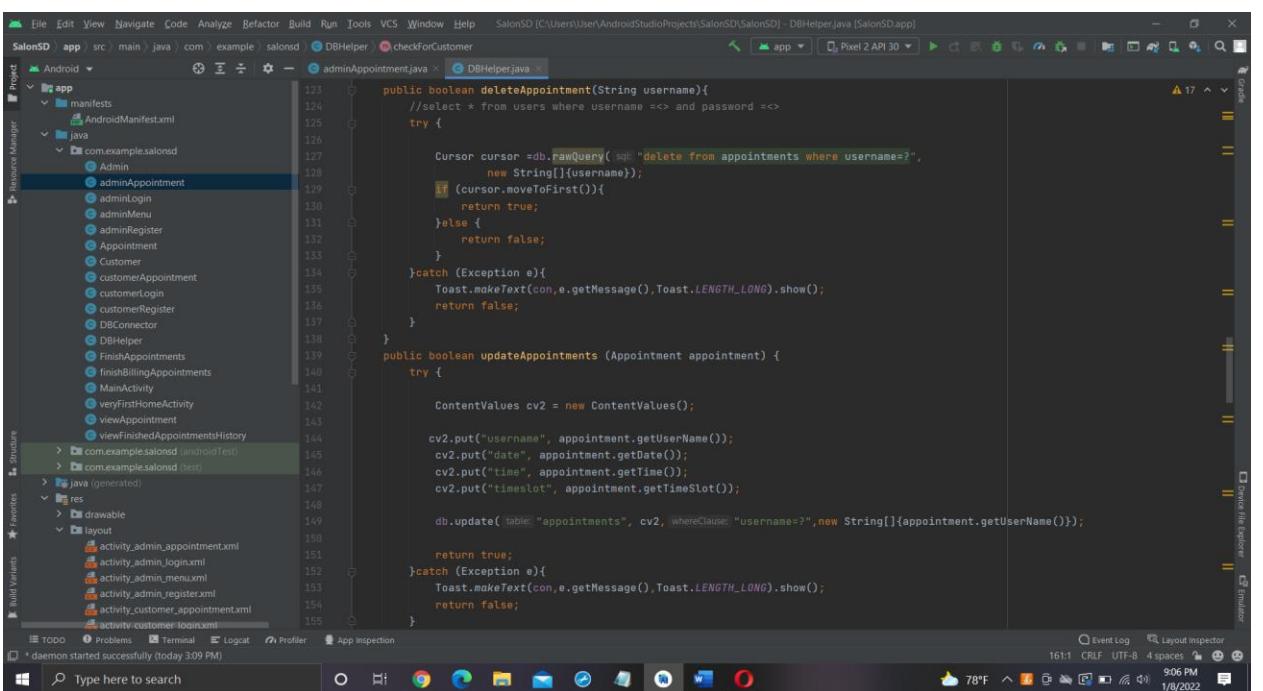
    cv1.put("username", appointment.getUserName());
    cv1.put("date", appointment.getDate());
    cv1.put("time", appointment.getTime());
    cv1.put("timeslot", appointment.getTimeSlot());

    db.insert( Table: "appointments", nullColumnHack: null, cv1);

    return true;
} catch (Exception e){
    Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
    return false;
}
}

```

The below screenshot shows the methods and database connections that are used to update/delete appointment details.



```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SalonSD [C:\Users\User\AndroidStudioProjects\SalonSD] - DBHelper.java [SalonSD.app]
SalonSD app src main java com.example.salonsd DBHelper checkForCustomer
Project Android
app manifests AndroidManifest.xml
java com.example.salonsd Admin AdminAppointment AdminLogin AdminMenu AdminRegister Appointment Customer customerAppointment customerLogin customerRegister DBConnector DBHelper DBFinishAppointments DBFinishBillings MainActivity veryFirstHomeActivity viewAppointment viewFinishedAppointmentsHistory
com.example.salonsd (androidTest)
com.example.salonsd (test)
java (generated)
res drawable
layout activity_admin_appointment.xml activity_admin_login.xml activity_admin_menu.xml activity_admin_register.xml activity_customer_appointment.xml activity_customer_login.xml
adminAppointment.java DBHelper.java
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
public boolean deleteAppointment(String username){
    //select * from users where username >= and password <=
    try {
        Cursor cursor = db.rawQuery("delete from appointments where username=?", new String[]{username});
        if (cursor.moveToFirst()){
            return true;
        } else {
            return false;
        }
    } catch (Exception e){
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

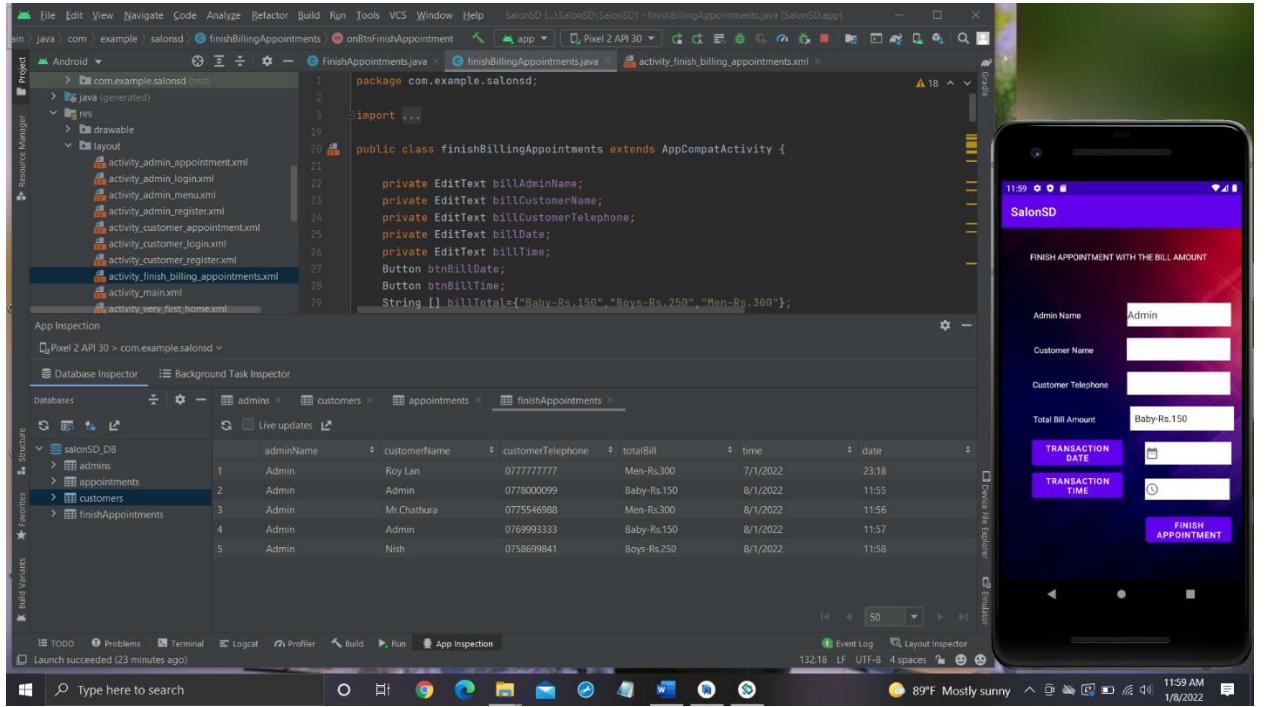
public boolean updateAppointments (Appointment appointment) {
    try {
        ContentValues cv2 = new ContentValues();

        cv2.put("username", appointment.getUserName());
        cv2.put("date", appointment.getDate());
        cv2.put("time", appointment.getTime());
        cv2.put("timeslot", appointment.getTimeSlot());

        db.update( Table: "appointments", cv2, whereClause: "username=?", new String[]{appointment.getUserName()});
        return true;
    } catch (Exception e){
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

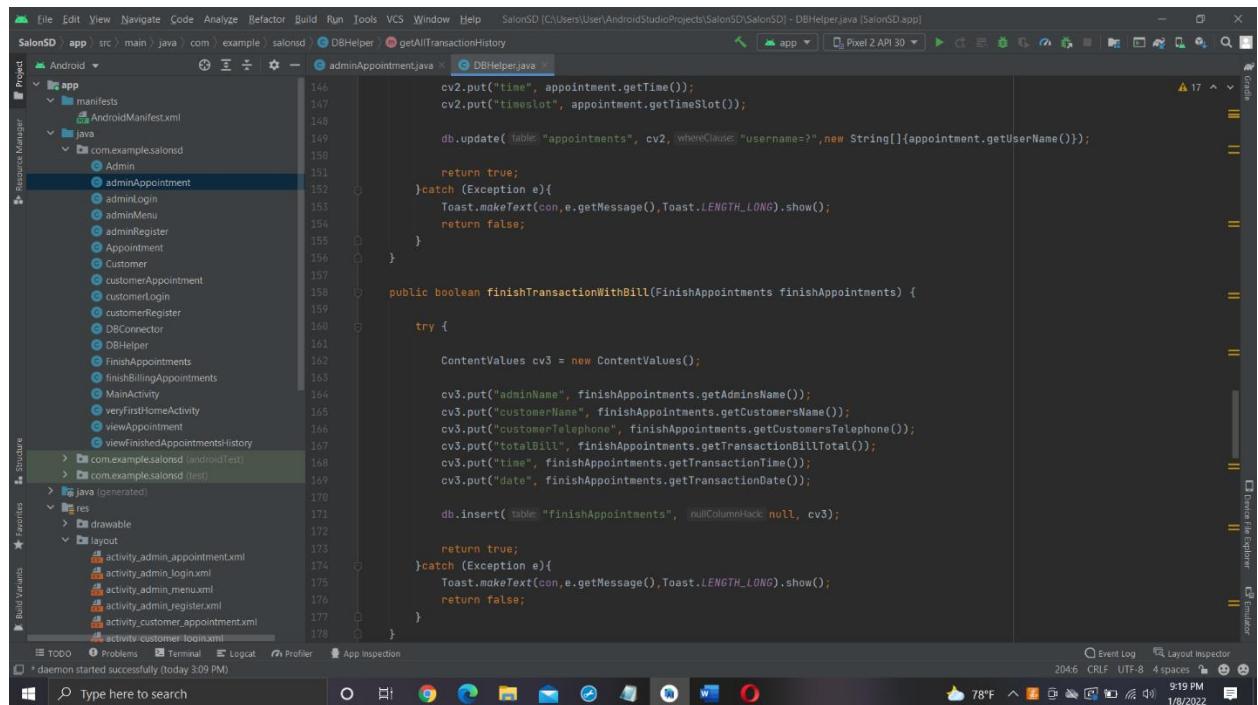
```

2.7 - Admin Finish Billing Appointments Interface



The interface shown in the attached screenshot will be appeared only when admin clicks Finish Appointments button from the Admin Menu. This UI consists with three buttons, six plain texts and five text views along with a background image. There is a date and time picker to get the current date and the time. Total Bill Amount can be selected and a spinner has used for that. From the back-end, inside the class called finishBillingAppointments, I have created on click listeners for date and time picker and a on click has been made for the button called Finish Appointment and validations have included inside it. The validations used for this finish billing appointment include are empty field validation and successfully finished or not validation. Further, toast messages have used to show the messages inside the validations.

The below screenshot shows the methods and database connections that are used to insert finished appointment details.



The screenshot displays the Android Studio interface with the project 'SalonSD' open. The code editor shows a Java file named 'DBHelper.java' containing methods for managing appointments. The specific method shown is 'finishTransactionWithBill'. The code uses ContentValues and SQLite database operations to insert data into the 'FinishAppointments' table. The code editor includes line numbers, code completion suggestions, and a status bar at the bottom indicating the application has started successfully.

```

    cv2.put("time", appointment.getTime());
    cv2.put("timeslot", appointment.getTimeSlot());

    db.update( table: "appointments", cv2, whereClause: "username=?",
    new String[]{appointment.getUserName()});

    return true;
} catch (Exception e) {
    Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
    return false;
}

public boolean finishTransactionWithBill(FinishAppointments finishAppointments) {
    try {

        ContentValues cv3 = new ContentValues();

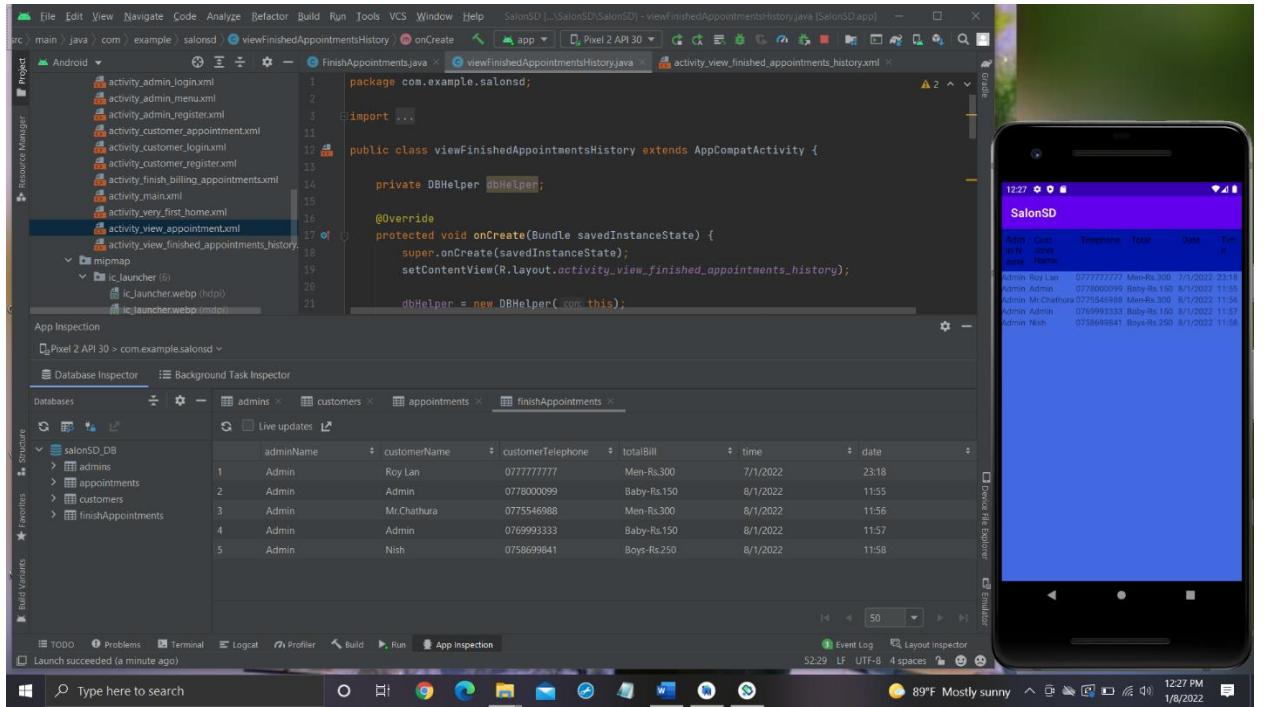
        cv3.put("adminName", finishAppointments.getAdminsName());
        cv3.put("customerName", finishAppointments.getCustomersName());
        cv3.put("customerTelephone", finishAppointments.getCustomersTelephone());
        cv3.put("totalBill", finishAppointments.getTransactionBillTotal());
        cv3.put("time", finishAppointments.getTransactionTime());
        cv3.put("date", finishAppointments.getTransactionDate());

        db.insert( table: "FinishAppointments", nullColumnHack: null, cv3);

        return true;
    } catch (Exception e) {
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

```

2.8 - Admin View Finished Appointments History Interface



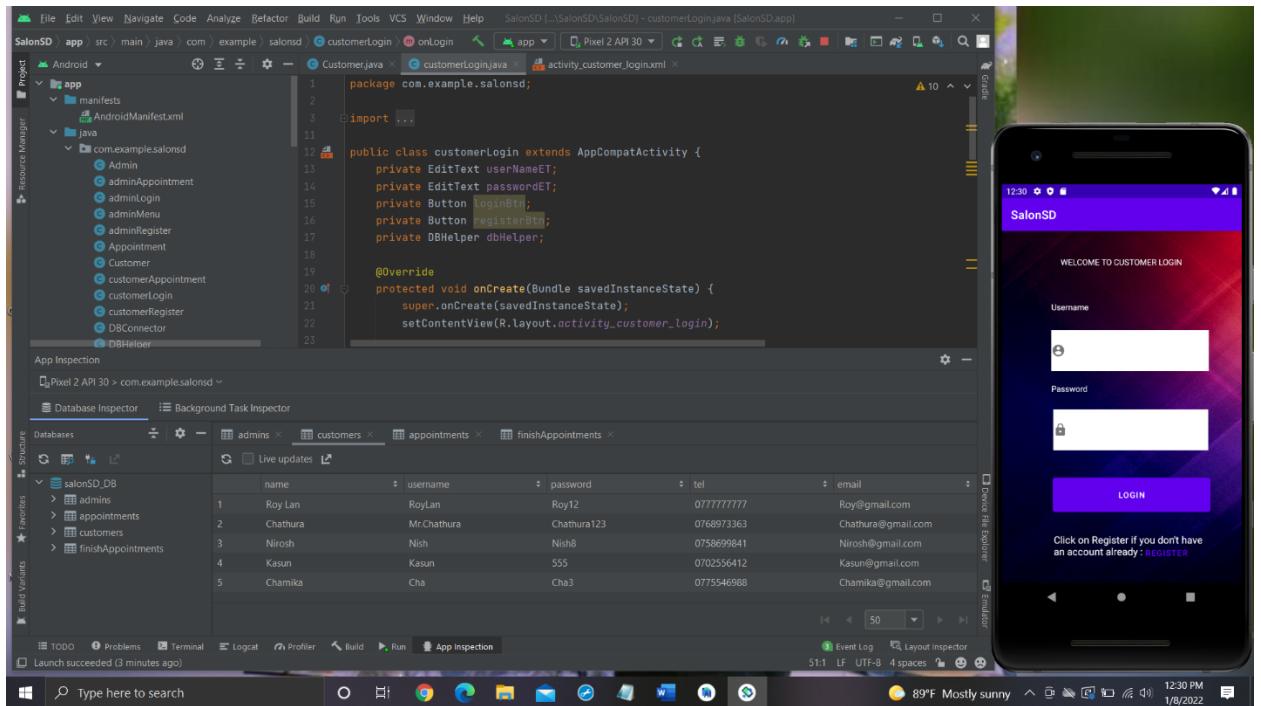
This above attached screenshot of the interface will be loaded when an admin clicks on the View Transaction History button from the Admin Menu interface. This is interface is only accessible for an admin. The layout used for this interface is a table layout. The table consists of six columns as admin name, customer name, telephone, total, date and time. There is no attached background image for this interface. This interface is with the blue color in the background. From the back-end, inside the class called `viewFinishedAppointmentsHistory`, I have designed the table layout to display columns and rows along with an Array List.

The below screenshot shows the methods and database connections that are used to retrieve finished appointment details with the select query.

The screenshot shows the Android Studio interface with the code editor open to the `DBHelper.java` file. The code implements a database helper class with methods for inserting, updating, and querying data related to appointments and transaction history.

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help SalonsD (C:\Users\User\AndroidStudioProjects\SalonsD\SalonSD - DBHelper.java [SalonSD app])  
SalonSD app src main java com.example.example com.example.example.DBHelper checkForCustomer  
Project  
Android  
adminAppointment  
customerLogin  
customerRegister  
DBConnector  
DBHelper  
FinishAppointments  
finishBillingAppointments  
MainActivity  
veryFirstHomeActivity  
viewAppointment  
viewFinishedAppointmentsHistory  
> com.example.example (androidTest)  
> com.example.example (test)  
> java (generated)  
> res  
> drawable  
> layout  
activity_admin_appointment.xml  
activity_admin_login.xml  
activity_admin_menu.xml  
activity_admin_register.xml  
activity_customer_appointment.xml  
activity_customer_login.xml  
activity_customer_register.xml  
activity_finish_billing_appointments.xml  
activity_main.xml  
activity_very_first_home.xml  
activity_view_appointment.xml  
activity_view_finished_appointments_history.xml  
> mipmap  
> ic_launcher (6)  
ic_launcher.webp (hdpi)  
ic_launcher.webp (mdpi)  
ic_launcher.webp (xhdpi)  
ic_launcher.webp (xxhdpi)  
ic_launcher.webp (xxxhdpi)  
customerAppointment  
customerLogin  
customerRegister  
DBConnector  
DBHelper  
FinishAppointments  
finishBillingAppointments  
MainActivityResult  
veryFirstHomeActivity  
viewAppointment  
viewFinishedAppointmentsHistory  
173 return true;  
174 } catch (Exception e){  
175     Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();  
176     return false;  
177 }  
178 }  
179  
180 public ArrayList<FinishAppointments> getAllTransactionHistory(){  
181     ArrayList<FinishAppointments> transactionHistoryList = new ArrayList<>();  
182     try {  
183         Cursor cursor = db.rawQuery("select * from finishAppointments", selectionArgs: null);  
184         if (cursor.moveToFirst()) {  
185             do {  
186                 FinishAppointments finishAppointments = new FinishAppointments();  
187                 finishAppointments.setAdminsName(cursor.getString( columnIndex: 0));  
188                 finishAppointments.setCustomersName(cursor.getString( columnIndex: 1));  
189                 finishAppointments.setCustomersTelephone(cursor.getString( columnIndex: 2));  
190                 finishAppointments.setTransactionBillTotal(cursor.getString( columnIndex: 3));  
191                 finishAppointments.setTransactionDate(cursor.getString( columnIndex: 4));  
192                 finishAppointments.setTransactionTime(cursor.getString( columnIndex: 5));  
193  
194                 transactionHistoryList.add(finishAppointments);  
195             } while ((cursor.moveToNext()));  
196         }  
197     } catch (Exception e){  
198         Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();  
199     }  
200     return transactionHistoryList;  
201 }  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
927  
928  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2079  
2080  
2081  
2082  
2083  
2084  
2085
```

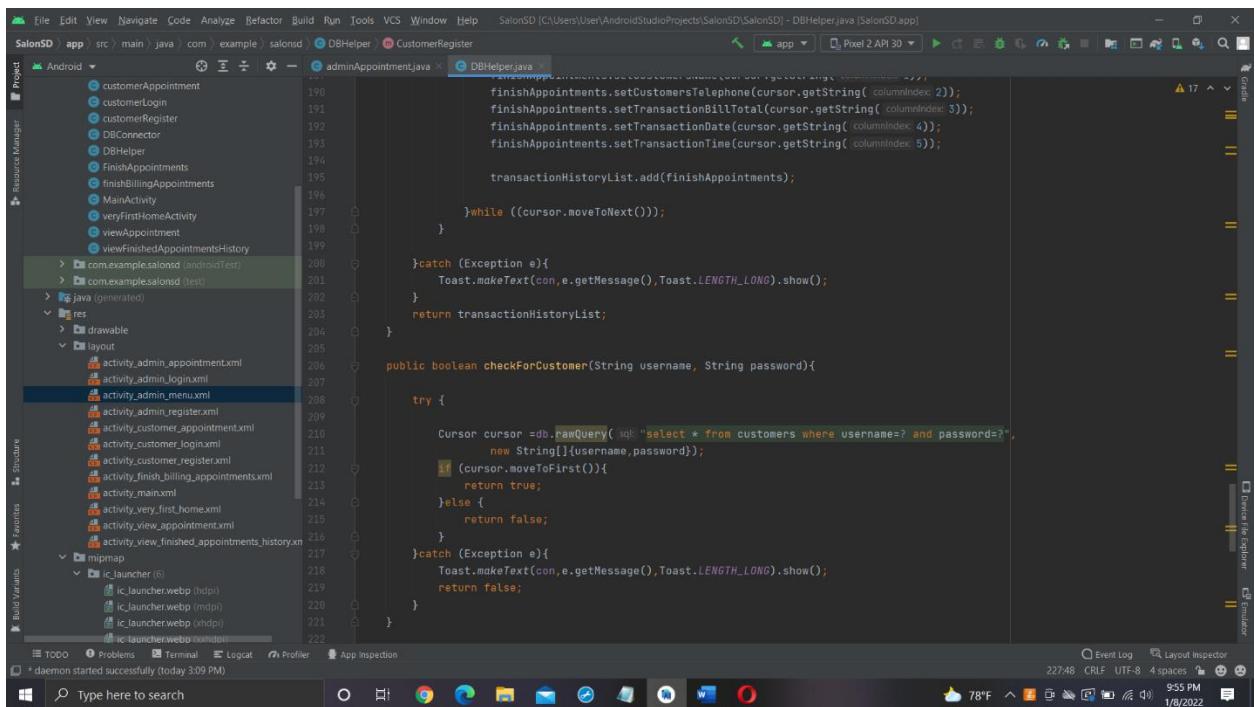
2.9 - Customer Login Interface



The interface shown in the attached screenshot can be seen by only a user who click on the customer button from the very first loaded interface after running the system. This Login interface is only for the Customers who wants to make an appointment. This UI consists with two buttons, two plain texts and three text views along with a background image. From the back-end, inside the class called `customerLogin`, I have created two on clicks as one in register button in which background is transparent and intent it to the customer register interface and in the login button and intent it to the customer appointment interface where a customer can make an appointment and view booked appointments before making an appointment. Here, when a customer sees this interface, he/she has the ability to move to the customer register interface if customer hasn't registered. If a customer processes an already created user account, he/she is eligible to log in. The only difference in this customer login interface and admin interface is customer login has the ability to move to the register interface without being logged in whereas admin has that register functionality inside the admin menu. The validations

used for this customer login interfaces include are empty field validation and correct username/password validation. Further, toast messages have used to show the messages inside the validations. All those coding has done inside the on click of the login button.

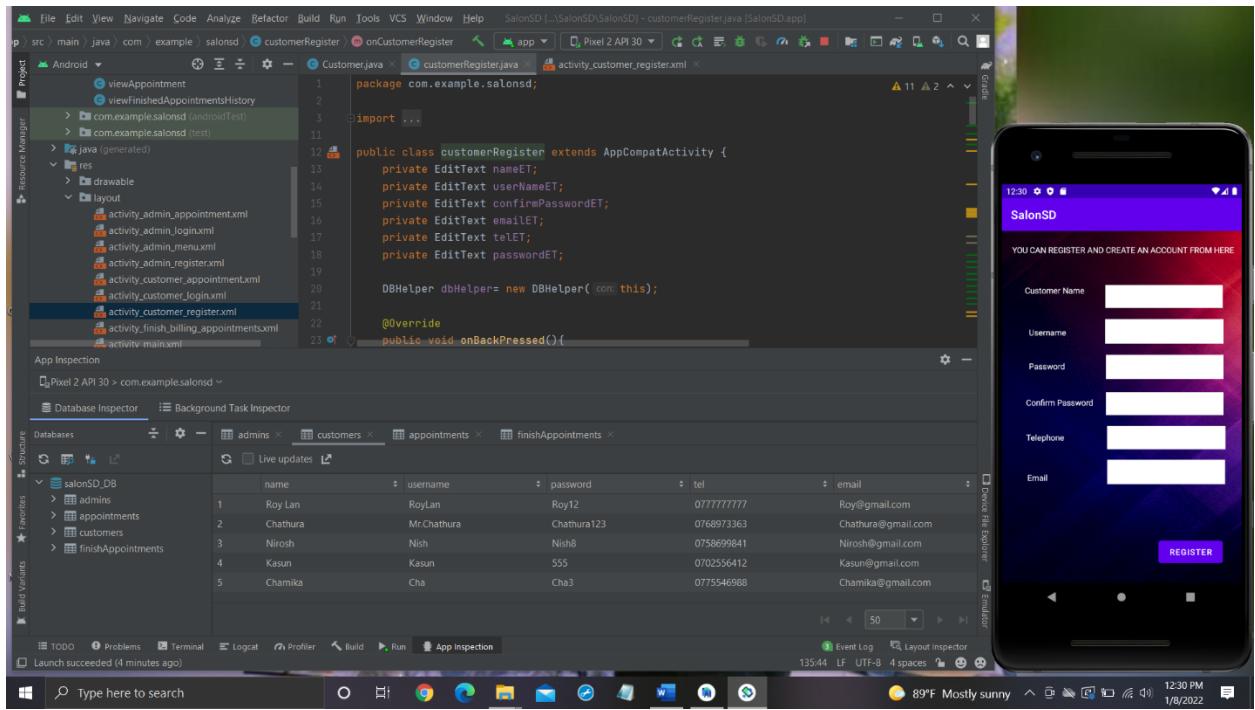
The below screenshot shows the database connection that is used to check the correctness of a customer's username and password. Select query has been used for that.



The screenshot shows the Android Studio interface with the project 'SalonSD' open. The code editor displays the `DBHelper.java` file. The code implements a database helper class with methods for checking customer credentials and retrieving transaction history. A specific method, `checkForCustomer`, contains a SQL query to select customers based on username and password. The code uses a cursor to execute the query and determine if a user exists.

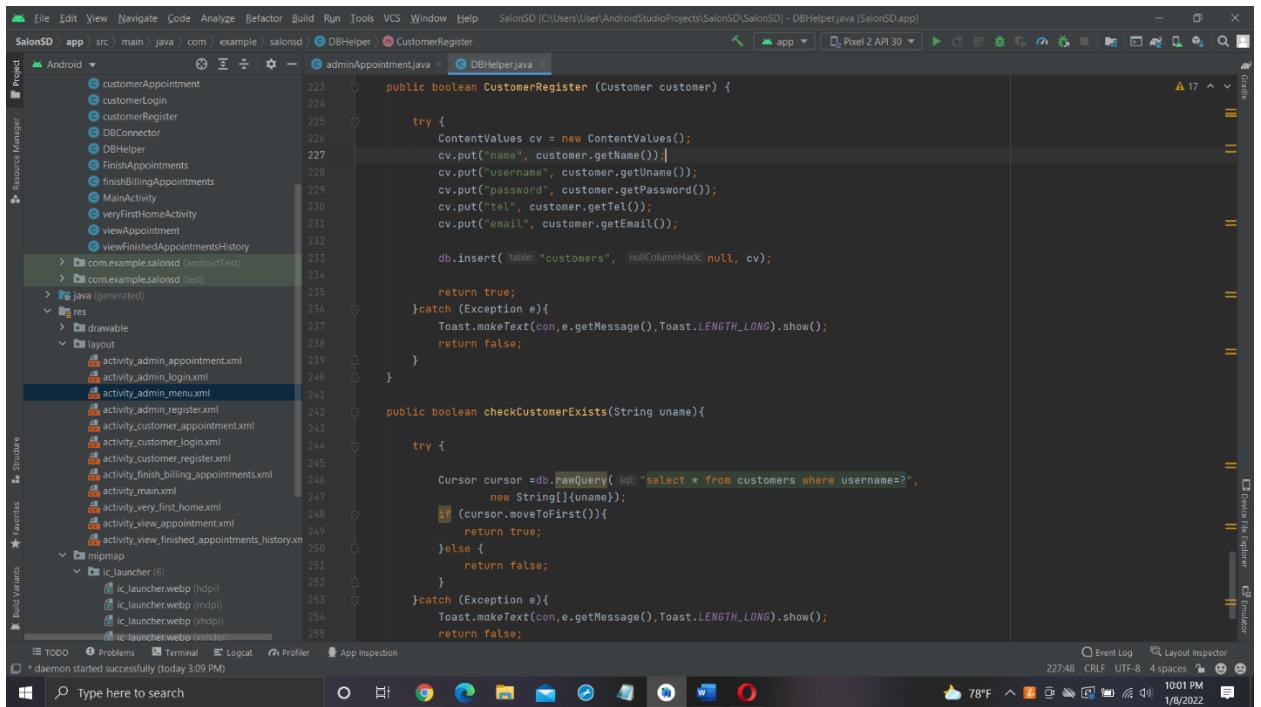
```
public boolean checkForCustomer(String username, String password){  
    try {  
        Cursor cursor = db.rawQuery("select * from customers where username=? and password=?",  
            new String[]{username,password});  
        if(cursor.moveToFirst())  
            return true;  
        else  
            return false;  
    }catch (Exception e){  
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();  
        return false;  
    }  
}
```

2.10 - Customer Registration Interface



The interface shown in the attached screenshot will be appeared only when customer clicks Register button from the Customer Login interface. This UI consists with one button, six plain texts and seven text views along with a background image. From the back-end, inside the class called customerRegister, I have created all the validations for the customer registration. The validations used for this customer registration include are empty field validation, password and confirm password matching validation, email validation and username validation which displays the message as a user already exists or not. Further, toast messages have used to show the messages inside the validations. All those coding has done inside the on click of the register button.

The below screenshot shows the methods and database connections that are used to check for existing users where select query has been used and the query that has used for inserting data obtain from the customer register interface.



The screenshot displays the Android Studio interface with the project 'SalonSD' open. The code editor shows the `DBHelper.java` file, which contains Java code for a database helper class. The code includes methods for registering a customer and checking if a customer exists by their username. The code uses ContentValues and Cursor objects to interact with the database. The Android Studio interface includes toolbars, a navigation bar, and various inspection tools like Layout Inspector and Event Log.

```

public boolean CustomerRegister (Customer customer) {
    try {
        ContentValues cv = new ContentValues();
        cv.put("name", customer.getName());
        cv.put("username", customer.getUsername());
        cv.put("password", customer.getPassword());
        cv.put("tel", customer.getTel());
        cv.put("email", customer.getEmail());

        db.insert( table: "customers", nullColumnHack: null, cv);

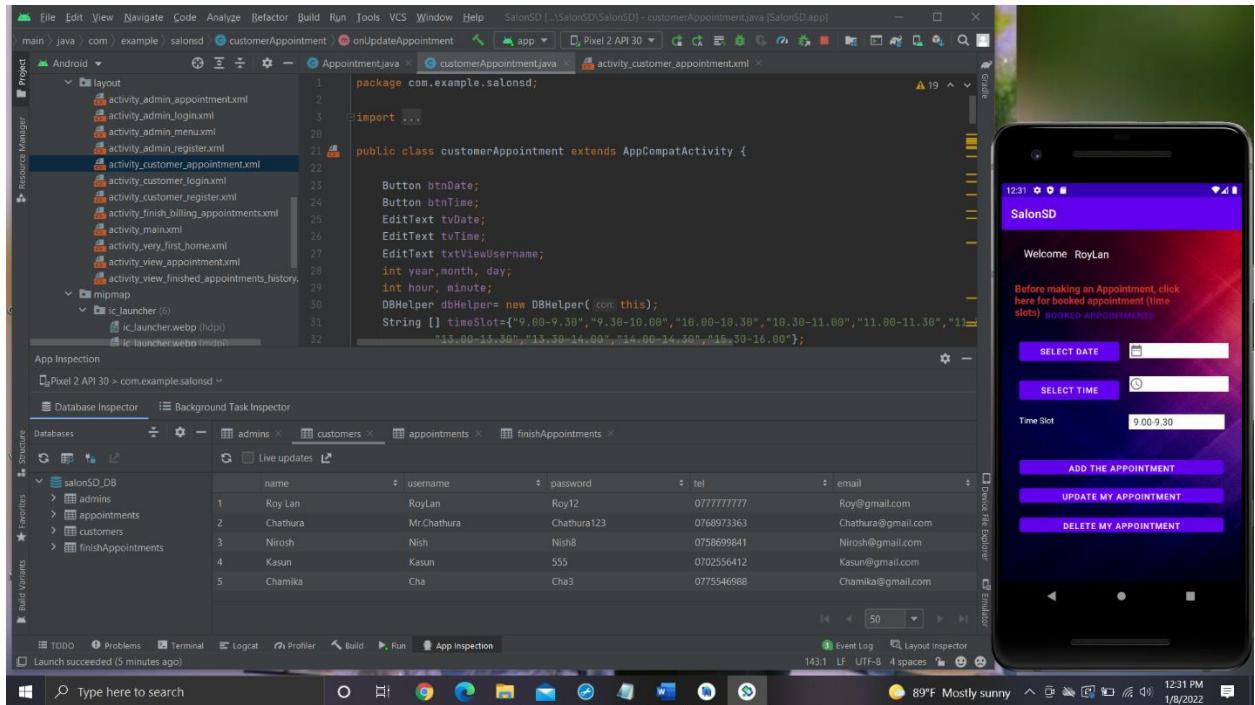
        return true;
    }catch (Exception e){
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

public boolean checkCustomerExists(String uname){
    try {

        Cursor cursor =db.rawQuery( sql:"select * from customers where username=?",
            new String[]{uname});
        if (cursor.moveToFirst()){
            return true;
        }else {
            return false;
        }
    }catch (Exception e){
        Toast.makeText(con,e.getMessage(),Toast.LENGTH_LONG).show();
        return false;
    }
}

```

2.11 - Customer Add Appointments Interface



The interface shown in the attached screenshot will be appeared only when a customer successfully logged in. This UI consists with six buttons as one to View Appointment, one to select date, one to select time, one to Add Appointments, one to Update Appointment and the other to Delete Appointment, four plain texts and four text views along with a background image. In here, a plain text is used to display logged customer's username. There is a date and time picker to get the current date and the time. Time slot can be selected only after viewing the booked appointments from the background transparent button called Booked Appointments. To select the time slot, a spinner has used. From the back-end, inside the class called customerAppointment, I have created on click listeners for date and time picker and four on clicks have been made separately for four buttons called Add, Update, Delete, Booked Appointments and validations have included inside them. The validations used for this customer appointment include are empty field validation and successfully inserted or not validation. Further, toast messages have used to show the messages inside the validations.

Another specialty in customer appointment is it uses the same methods and database connections which were used to insert, update and delete appointment details by the admin. Only the interfaces are changed and intent.

2.12 - Database Structure

The screenshot shows the Android Studio interface with the project 'SalonSD' open. The code editor displays the 'DBConnector.java' file, which extends the 'SQLiteOpenHelper' class. The code defines four database tables: 'admins', 'customers', 'appointments', and 'finishAppointments'. The 'admins' table has columns for name, username, password, tel, and email. The 'customers' table has columns for name, username, password, tel, and email. The 'appointments' table has columns for username, date, time, and timeslot. The 'finishAppointments' table has columns for adminName, customerName, customerTelephone, totalBill, time, and date. The code also includes logic to insert default data into the 'admins' table and handle upgrades.

```
package com.example.salonSD;

import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import android.widget.Toast;

public class DBConnector extends SQLiteOpenHelper {
    //The required database have created here
    public DBConnector(Context context) { super(context, "salonSD_DB", null, 1); }

    @Override
    public void onCreate(SQLiteDatabase db) {

        //All required database tables are here
        //along with the columns in each database table
        db.execSQL("create table admins (name text, username text, password text, tel text, email text)");
        db.execSQL("create table customers (name text, username text, password text, tel text, email text)");
        db.execSQL("create table appointments (username text, date text, time text,timeslot text )");
        db.execSQL("create table finishAppointments (adminName text, customerName text, customerTelephone text, " +
                "totalBill text, time text, date text)");

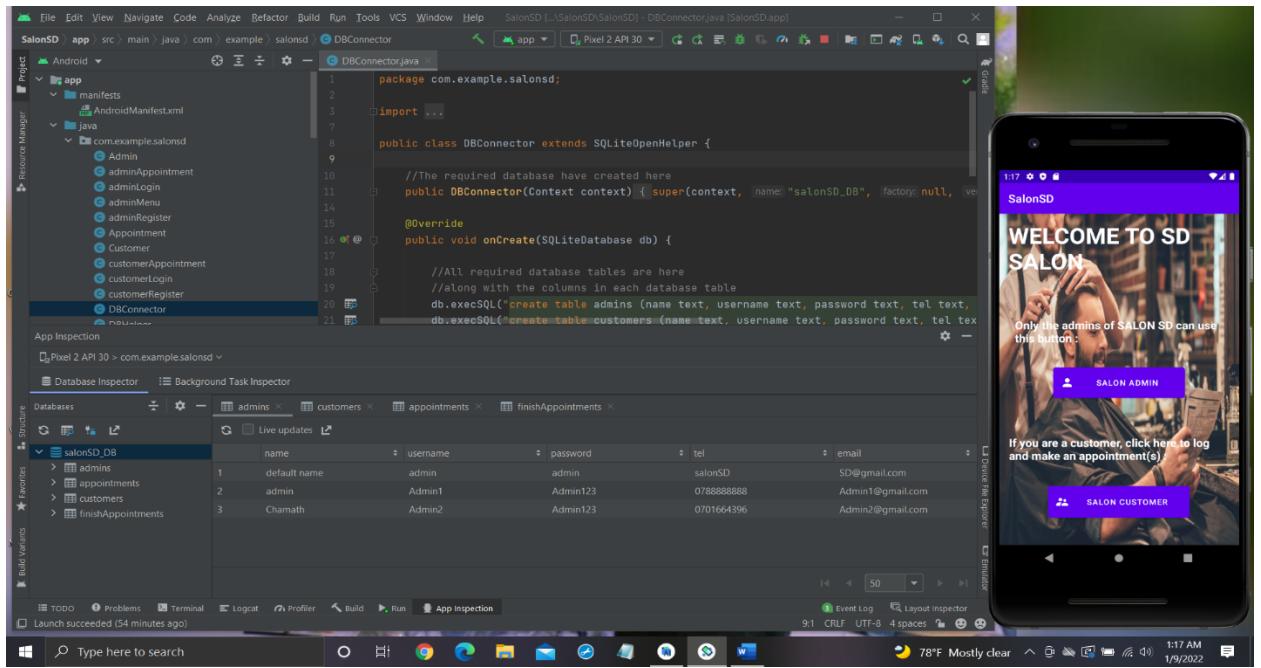
        //add default user to the admins database table
        ContentValues cv = new ContentValues();
        cv.put("name", "default name");
        cv.put("username", "admin");
        cv.put("password", "admin");
        cv.put("tel", "salonSD");
        cv.put("email", "SD@gmail.com");

        db.insert("admins", null, cv);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int i, int i1) {
    }
}
```

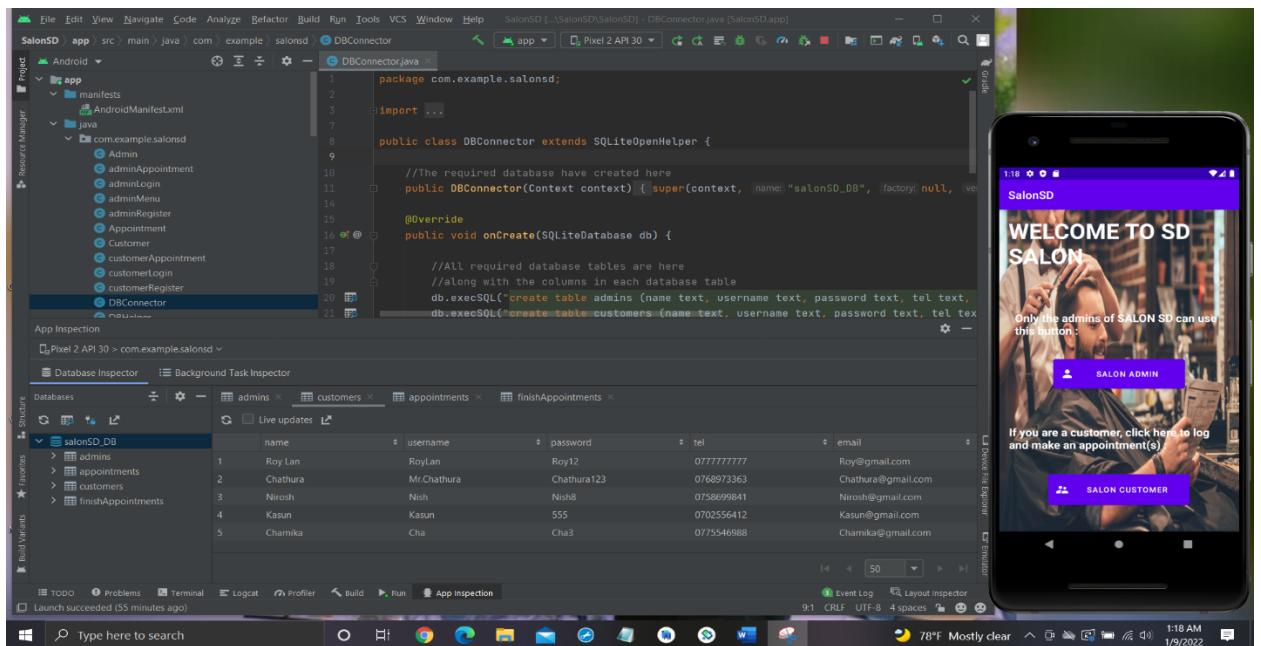
In the above screenshot, it depicts how the coding has been done to make a database and database table along with columns. All those coding has been coded inside an activity called DBConnector. Its visible in the screenshot that the database name is coded as salonSD_DB. Moreover, there are four database tables named as admins, customers, appointments and finishAppointments. The admins database table consists with the columns called name, username, password, telephone and email. The customers database table consists with the columns called name, username, password, telephone and email. The appointments database table consists with the columns called username, date, time and time slot. The finishAppointments database table consists with the columns called admin name, customer name, customer telephone, total bill, time and date.

2.12.1 - Admins Database Table

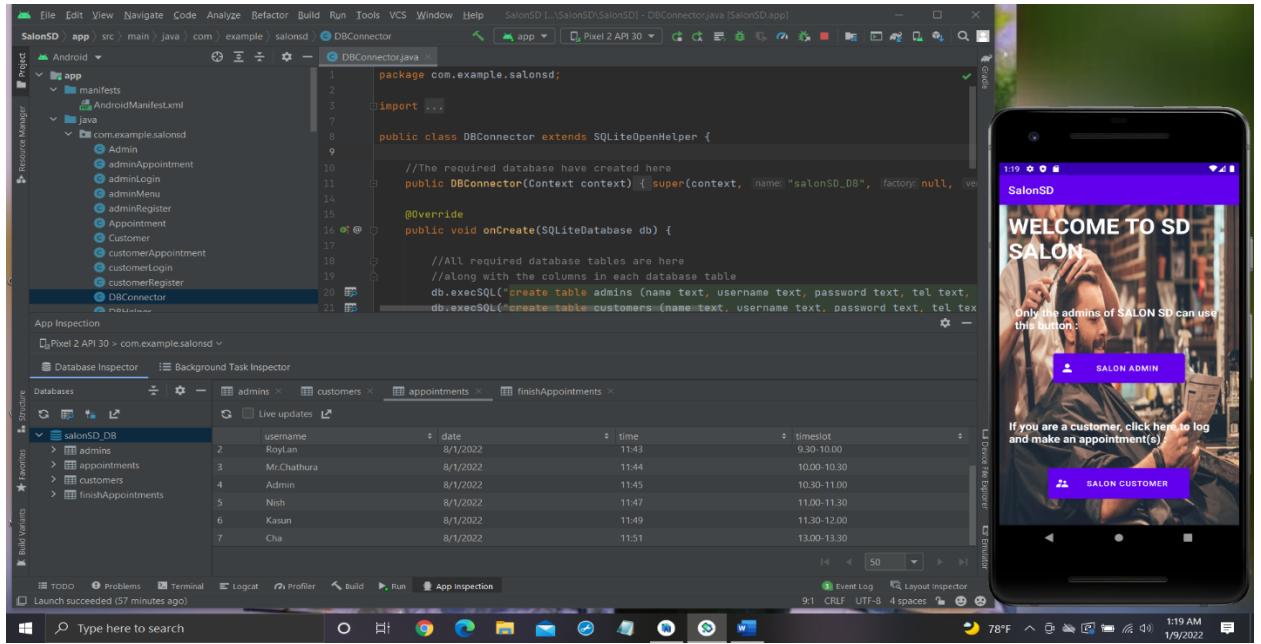


This above screenshot depicts the graphical representation of the database table which has named as “admins”. Further, it shows three records inside the table as table rows.

2.12.2 - Customers Database Table

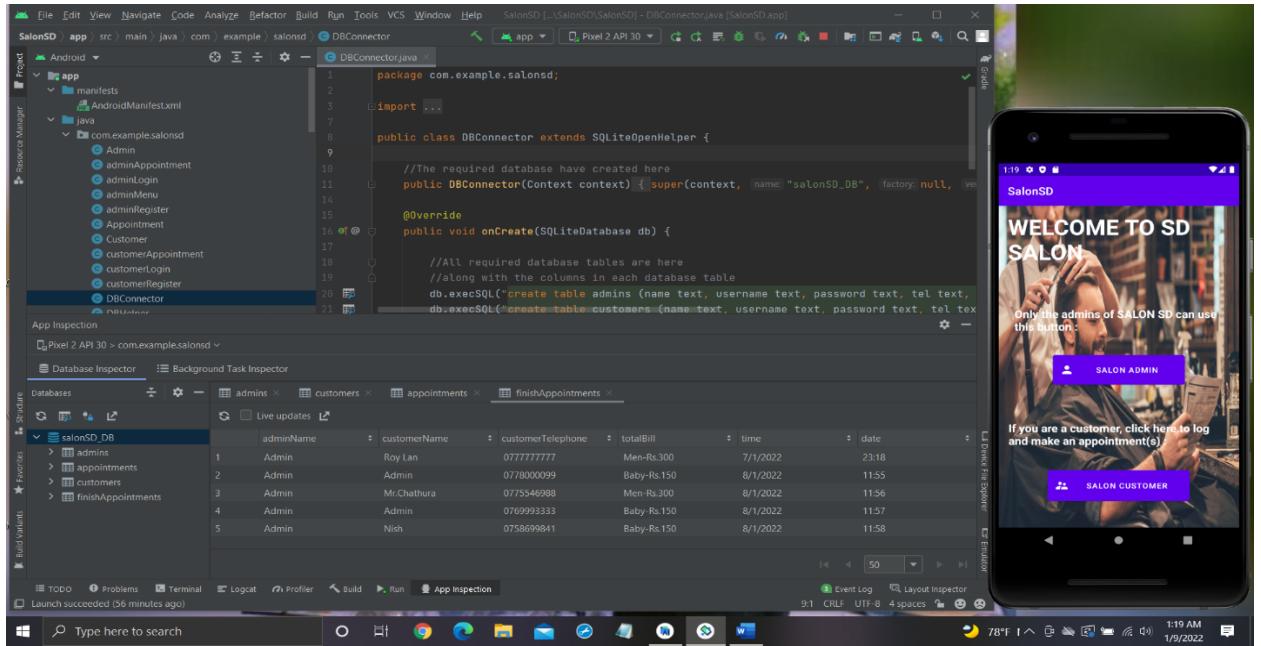


2.12.3 - Appointments Database Table



This above screenshot depicts the graphical representation of the database table which has named as “appointments”. Further, it shows seven records inside the table as table rows.

2.12.4 - Finish Appointments Database Table



This above screenshot depicts the graphical representation of the database table which has named as “finishAppointments”. Further, it shows five records inside the table as table rows.

Task 3 - Testing

Testing a system involves in testing the existing overall system. All most all the functions are linked together to see if the system performs as planned. The main purpose of system testing is to produce a high-quality system with well-functioning outputs.

Therefore, Black-Box system testing method will be used for this Mobile Application of Salon SD. Because any internal knowledge about the coding is not required for Black-Box testing whereas internal knowledge about the coding is required for White-Box testing. System testing covers both functional and non-functional functions and many other aspects of testing. Thus, they are tested using a Black-Box approach as its easier just to submit the input into the system and confirm the result.

Many terminologies are used to describe software quality assurance through system testing. People who are specialized in this industry uses various software terminologies for software quality assurance. Among those terminologies, Test Plans and Test Cases are widely used for system testing in order to assure the quality of a software. (Dondeti, 2012)

Thus, below shows the Test Plan and the Test Cases that are drawn for testing the Mobile Application of the Salon SD which plays a key part in providing a high-quality product. There I've used the Android Virtual Device called Pixel 2 API 30 which runs the Android 11.0 and the resolution is 1080*1920: 420 dpi.

3.1 - Test Plan

TEST CASE ID	TEST CASE NAME	DESCRIPTION	EXPECTED RESULT
TC1	Admin Login		
TC1.1	Check Login empty fields	Have to press on login button without filling any field	Should display an Error Message as Fields can't be blank
TC1.2	Check fields by entering values one by one into fields		
TC1.2.1	Check by filling only username field	Have enter a username inside the username field and press login button.	Should display the same Error Message which displayed previously as Fields can't be blank
TC1.2.2	Check by filling only password field	Have to enter only a password and press login button separately	Should display the same Error Message which displayed previously as Fields can't be blank
TC1.3	Check by wrong username or password		
TC1.3.1	Check with wrong username and correct password	Have to enter a wrong username along with the correct password and press login button	Should display an Error Message as Please enter valid username & password
TC1.3.2	Check with wrong password and correct username	Have to enter a wrong password along with the correct username and press login button	Should display an Error Message as Please enter valid username & password
TC1.4	Check Login working without any empty field and with default (hard coded) login credentials	Have to enter a default username(admin) and password(admin) correctly	Admin menu interface should be loaded along with a message as Login granted successfully
TC1.5	Check Login working without any empty field and with saved &	Have to enter the admin registered and saved username and password correctly	Admin menu interface should be loaded along with a message as Login granted successfully

	correct login credentials		
TC2	Admin Registration		
TC2.1	Check Register empty fields	Have to press on Register button without filling any fields	Should display an Error Message as Fields can't be blank
TC2.2	Check by entering values one by one into fields		
TC2.2.1	Check by entering only admin name	Have to enter an admin name inside the username field and press register button	Should display an Error Message as Fields can't be blank
TC2.2.2	Check by entering only username	Have to enter a username inside the username field and press register button	Should display an Error Message as Fields can't be blank
TC2.2.3	Check by entering only password	Have to enter only a password and press register button	Should display an Error Message as Fields can't be blank
TC2.2.4	Check by entering only confirm password	Have to enter a confirm password and press register button separately	Should display an Error Message as Fields can't be blank
TC2.2.5	Check by entering only telephone	Have to enter a telephone and press register button separately	Should display an Error Message as Fields can't be blank
TC2.2.6	Check by entering only email	Have to enter an email and press register button separately	Should display an Error Message as Fields can't be blank
TC2.3	Check by entering already saved username into the username field	Have to fill all other fields, enter an already registered admin's username and press register button	Should display an Error Message as Entered username already exists
TC2.4	Check password and confirm password matching	Have to fill all other fields, enter a password and a different confirm password and press register button	Should display an Error Message as Password & confirm password should match
TC2.5	Check email format	Have to fill all other fields correctly, enter an email address with the @ symbol and press register button	Should display an Error Message as Invalid Email format

TC2.6	Check Registration working without any empty field and with new correct data	Have to enter a new username, matching password with confirm password, correct email format, without empty fields and press register button	Should display a message as User account created successfully
TC2.7	Check on back when some fields are filled		
TC2.7.1	Check back button notification when some fields are filled	Fill only few fields along with some empty fields and press on back button of the phone	A notification should pop up and ask whether user wants to move back and get user's response.
TC2.7.2	Check back button notification when user's response is YES	Press on YES	If user provided YES, the previous interface will be displayed.
TC2.7.3	Check back button notification when user's response is NO	Press on NO	If user provided NO, that notification will fade away and a message will pop up mentioning that Continue Editing
TC3	Admin Appointment		
TC3.1	Check Add Appointment with empty fields	Have to press on Add Appointment button without filling any fields	Should display an Error Message as Fields can't be blank
TC3.2	Check Add Appointment working without any empty field	Have to fill all fields and press on Add Appointment button	Should display a message as Appointment added successfully
TC3.3	Check Update Appointment working without any empty and updated fields	Have to change the necessary fields without any empty field by entering the update needed user's username on the user type field and press on Update Appointment button	Should display a message as Appointment updated successfully

TC3.4	Check Delete Appointment working	Press on Delete Appointment button by entering the delete needed user's username on the user type field and press on Delete Appointment button	Should display a message as Appointment deleted successfully
TC4	Finish Billing Appointments		
TC4.1	Check Finish Appointment with empty fields	Have to press on Finish Appointment button without filling any fields	Should display an Error Message as Fields can't be blank
TC4.2	Check Finish Appointment working without any empty field	Have to fill all fields and press on Finish Appointment button	Should display a message as Appointment Finished successfully
TC5	Customer Registration		
TC5.1	Check Register empty fields	Have to press on Register button without filling any fields	Should display an Error Message as Fields can't be blank
TC5.2	Check by entering values one by one into fields		
TC5.2.1	Check by entering only customer name	Have to enter a customer name inside the username field and press register button	Should display an Error Message as Fields can't be blank
TC5.2.2	Check by entering only username	Have to enter a username inside the username field and press register button	Should display an Error Message as Fields can't be blank
TC5.2.3	Check by entering only password	Have to enter only a password and press register button	Should display an Error Message as Fields can't be blank
TC5.2.4	Check by entering only confirm password	Have to enter a confirm password and press register button separately	Should display an Error Message as Fields can't be blank
TC5.2.5	Check by entering only telephone	Have to enter a telephone and press register button separately	Should display an Error Message as Fields can't be blank

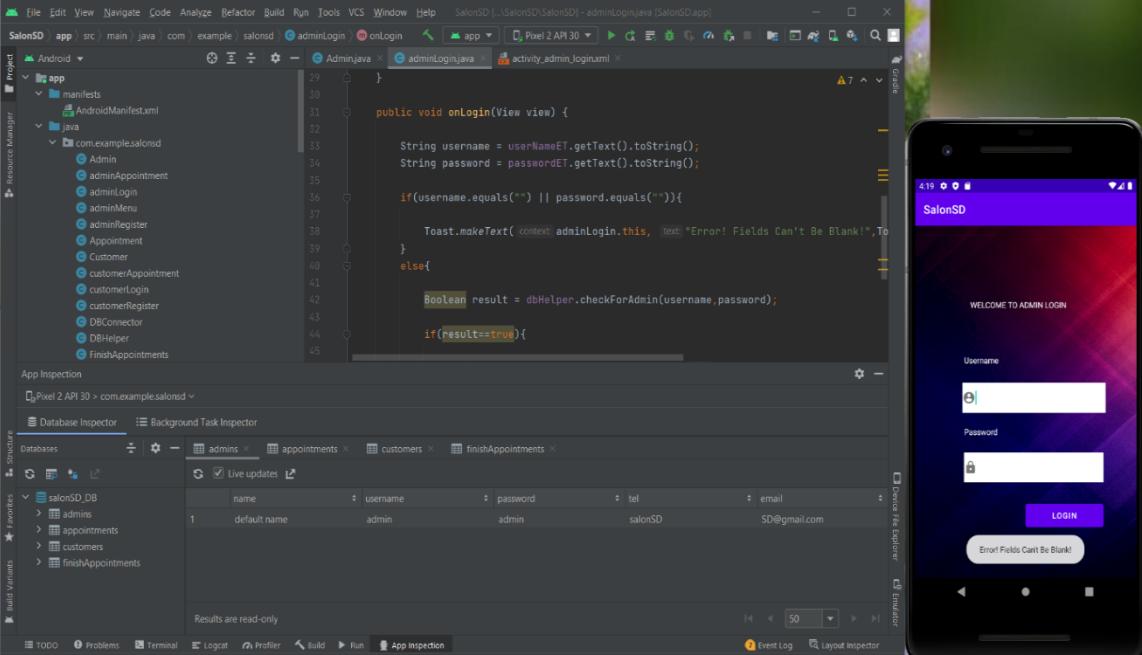
TC5.2.6	Check by entering only email	Have to enter an email and press register button separately	Should display an Error Message as Fields can't be blank
TC5.3	Check by entering already saved username into the username field	Have to fill all other fields, enter an already registered customer's username and press register button	Should display an Error Message as Entered username already exists
TC5.4	Check password and confirm password matching	Have to fill all other fields, enter a password and a different confirm password and press register button	Should display an Error Message as Password & confirm password should match
TC5.5	Check email format	Have to fill all other fields correctly, enter an email address with the @ symbol and press register button	Should display an Error Message as Invalid Email format
TC5.6	Check Registration working without any empty field and with new correct data	Have to enter a new username, matching password with confirm password, correct email format, without empty fields and press register button	Should display a message as User account created successfully
TC5.7	Check on back when some fields are filled		
TC5.7.1	Check back button notification when some fields are filled	Fill only few fields along with some empty fields and press on back button of the phone	A notification should pop up and ask whether user wants to move back and get user's response.
TC5.7.2	Check back button notification when user's response is YES	Press on YES	If user provided YES, the previous interface will be displayed.
TC5.7.3	Check back button notification when user's response is NO	Press on NO	If user provided NO, that notification will fade away and a message will pop up mentioning that Continue Editing

TC6	Customer Login		
TC6.1	Check Login empty fields	Have to press on login button without filling any field	Should display an Error Message as Fields can't be blank
TC6.2	Check fields by entering values one by one into fields		
TC6.2.1	Check by filling only username field	Have enter a username inside the username field and press login button.	Should display the same Error Message which displayed previously as Fields can't be blank
TC6.2.2	Check by filling only password field	Have to enter only a password and press login button separately	Should display the same Error Message which displayed previously as Fields can't be blank
TC6.3	Check by wrong username or password		
TC6.3.1	Check with wrong username and correct password	Have to enter a wrong username along with the correct password and press login button	Should display an Error Message as Please enter valid username & password
TC6.3.2	Check with correct username and wrong password	Have to enter a wrong password along with the correct username and press login button	Should display an Error Message as Please enter valid username & password
TC6.4	Check Login working without any empty field and with saved & correct login credentials	Have to enter the customer registered and saved username and password correctly	Customer Appointment interface should be loaded along with a message as Login granted successfully
TC7	Customer Appointment		
TC7.1	Check Add Appointment with empty fields	Have to press on Add Appointment button without filling any fields	Should display an Error Message as Fields can't be blank
TC7.2	Check Add Appointment working without any empty field	Have to fill all fields and press on Add Appointment button	Should display a message as Appointment added successfully

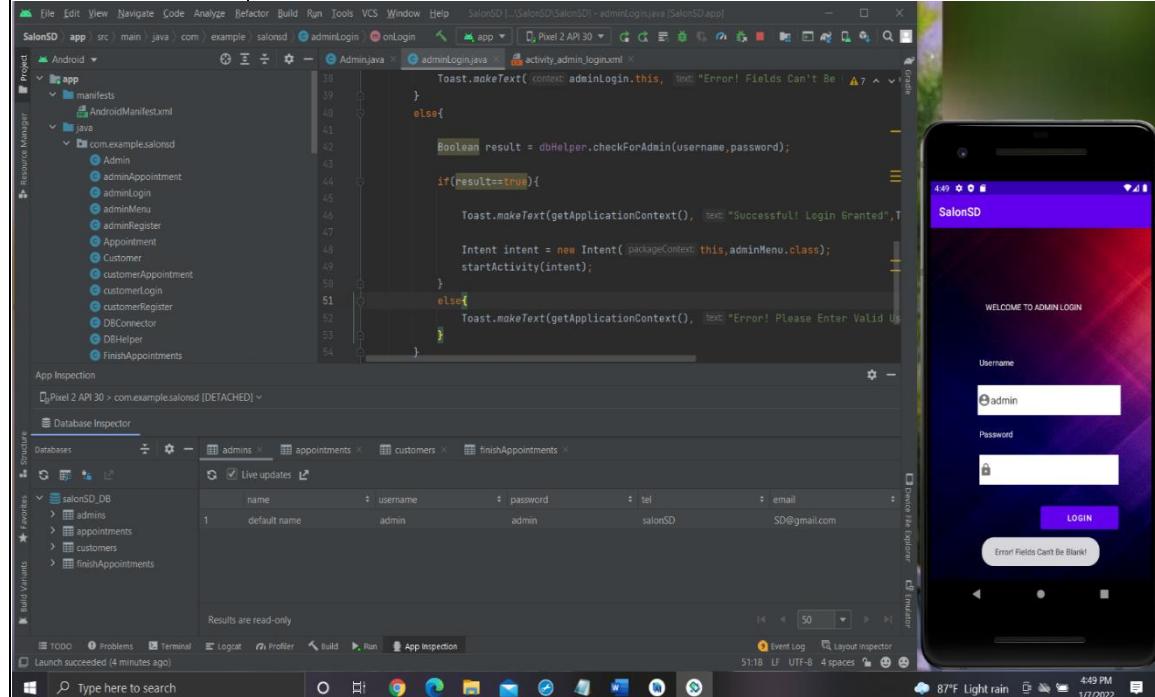
TC7.3	Check Update Appointment working without any empty and updated fields	Have to change the necessary fields without any empty field and press on Update Appointment button	Should display a message as Appointment updated successfully
TC7.4	Check Delete Appointment working	Press on Delete Appointment button where particular logged user's username is displayed at the right top of the interface	Should display a message as Appointment deleted successfully

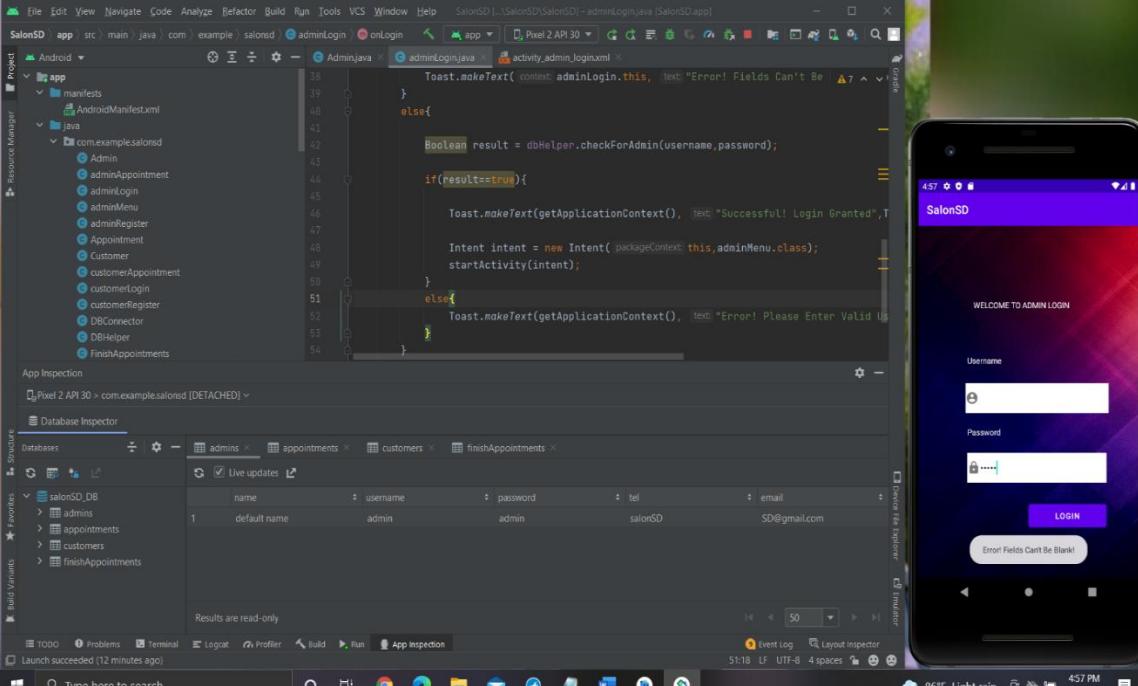
3.2 - Test Cases

TC1 - Admin Login

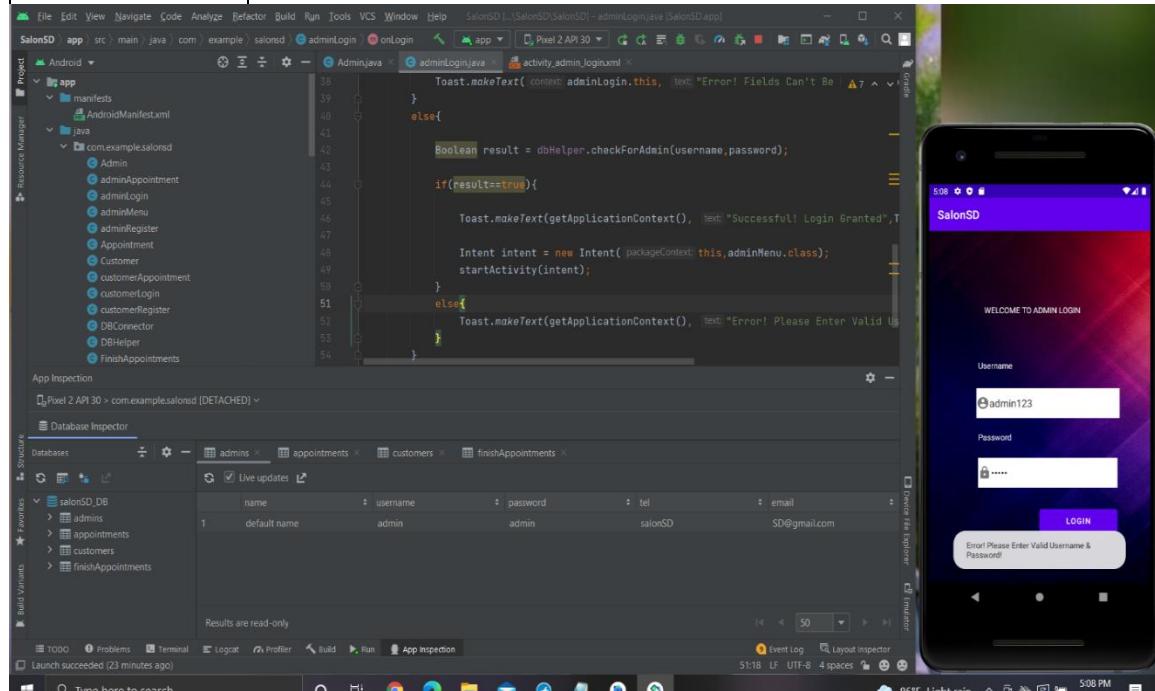
TEST CASE ID	TC1.1
TEST CASE NAME	Check Login empty fields
TEST SUMMARY	Checking the empty fields validation in Admin Login interface
TEST STEPS	Press on login button (without filling any field)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

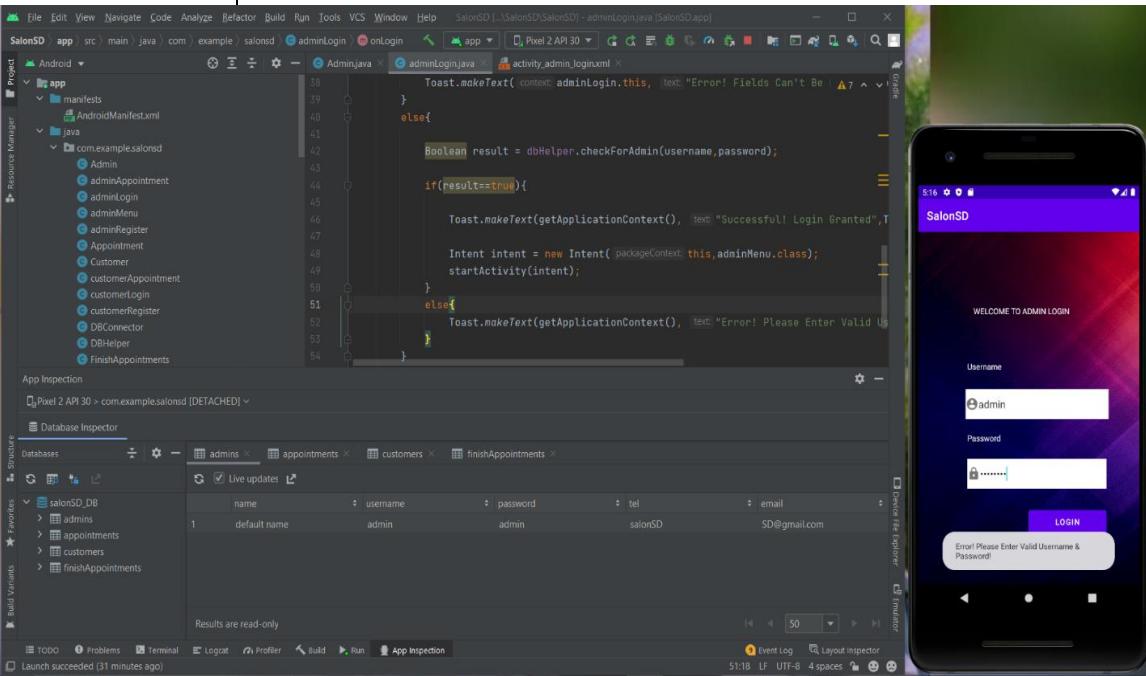
TC1.2 - Check fields by entering values one by one into fields

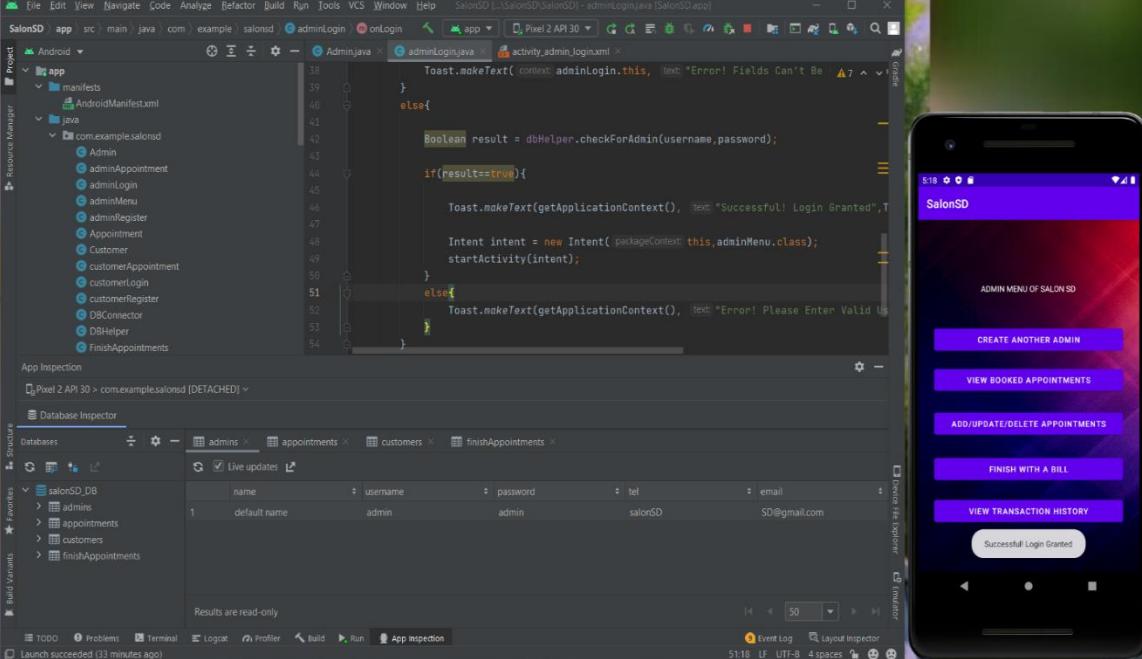
TEST CASE ID	TC1.2.1
TEST CASE NAME	Check by filling only username field
TEST SUMMARY	Checking empty field validation by filling only one field
TEST STEPS	<ol style="list-style-type: none"> Fill only username field with a valid admin username Press on Login button
TEST DATA	Username: admin Password: empty
EXPECTED RESULT	Should display the same Error Message which displayed previously as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error messages displayed as a toast message with very clear outcomes and the same error displayed in the above test case has received here also.
STATUS (PASS/FAIL)	Pass

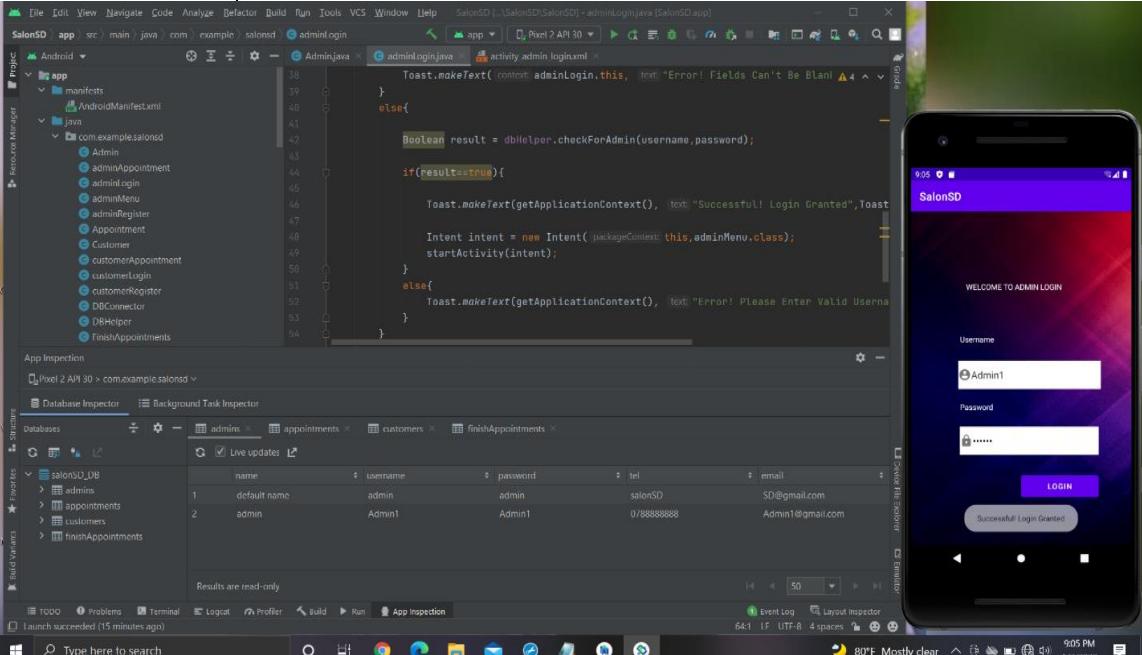
TEST CASE ID	TC1.2.2
TEST CASE NAME	Check by filling only password field
TEST SUMMARY	Checking empty field validation by filling only one field
TEST STEPS	<ol style="list-style-type: none"> Fill only password field with a valid admin password Press on Login button
TEST DATA	Username: empty Password: admin
EXPECTED RESULT	Should display the same Error Message which displayed previously as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error messages displayed as a toast message with very clear outcomes and the same error displayed in the above test case has received here also.
STATUS (PASS/FAIL)	Pass

TC1.3 - Check by wrong username or password

TEST CASE ID	TC1.3.1
TEST CASE NAME	Check with wrong username and correct password
TEST SUMMARY	Checking correctly filled password field validation whereas username is wrong
TEST STEPS	<ol style="list-style-type: none"> Fill only password field with a valid admin password Fill username field with a wrong username Press on Login button
TEST DATA	Username: admin123 Password: admin
EXPECTED RESULT	Should display an Error Message as Please enter valid username & password
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

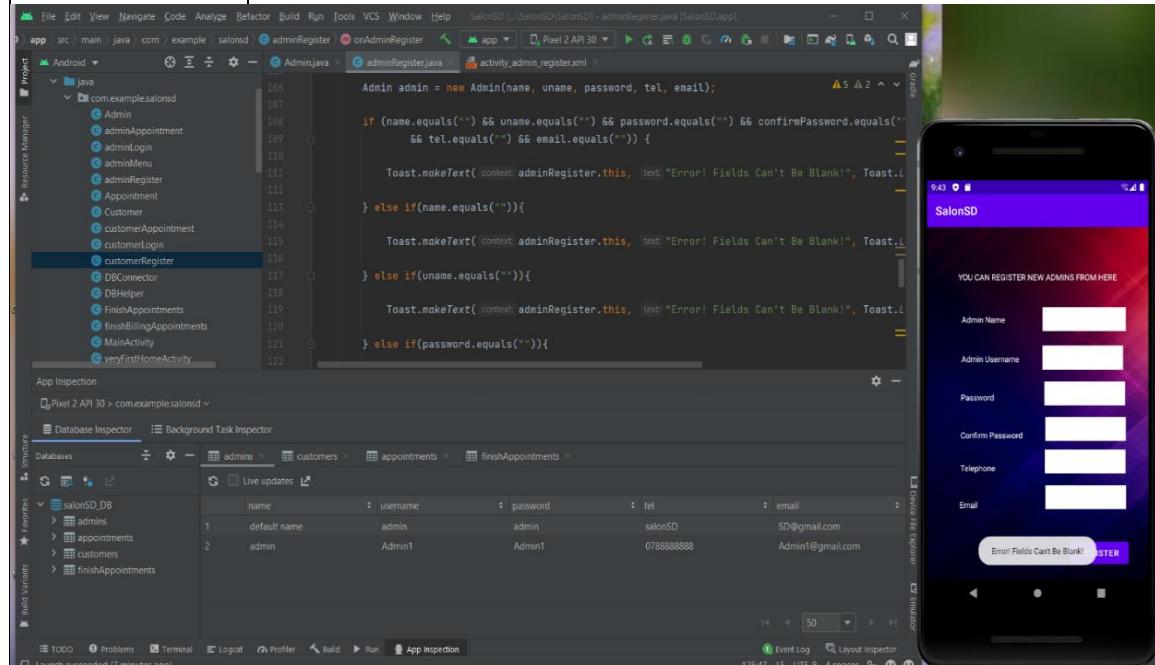
TEST CASE ID	TC1.3.2
TEST CASE NAME	Check with correct username and wrong password
TEST SUMMARY	Checking correctly filled username field validation whereas password is wrong
TEST STEPS	<ol style="list-style-type: none"> Fill only username field with a valid admin username Fill password field with a wrong password Press on Login button
TEST DATA	Username: admin Password: admin123
EXPECTED RESULT	Should display an Error Message as Please enter valid username & password
ACTUAL RESULT	
CONCLUSION	The expected error messages displayed as a toast message with very clear outcomes and the same error displayed in the above test case has received here also.
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC1.4
TEST CASE NAME	Check Login working without any empty field and with default (hard coded) login credentials
TEST SUMMARY	Checking whether correctly filled username and password is working or not
TEST STEPS	<ol style="list-style-type: none"> Fill username field with a valid(hardcoded) admin username Fill password field with a valid(hardcoded) password Press on Login button
TEST DATA	Username: admin Password: admin
EXPECTED RESULT	Admin menu interface should be loaded along with a message as Login granted successfully
ACTUAL RESULT	
CONCLUSION	The expected successful message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

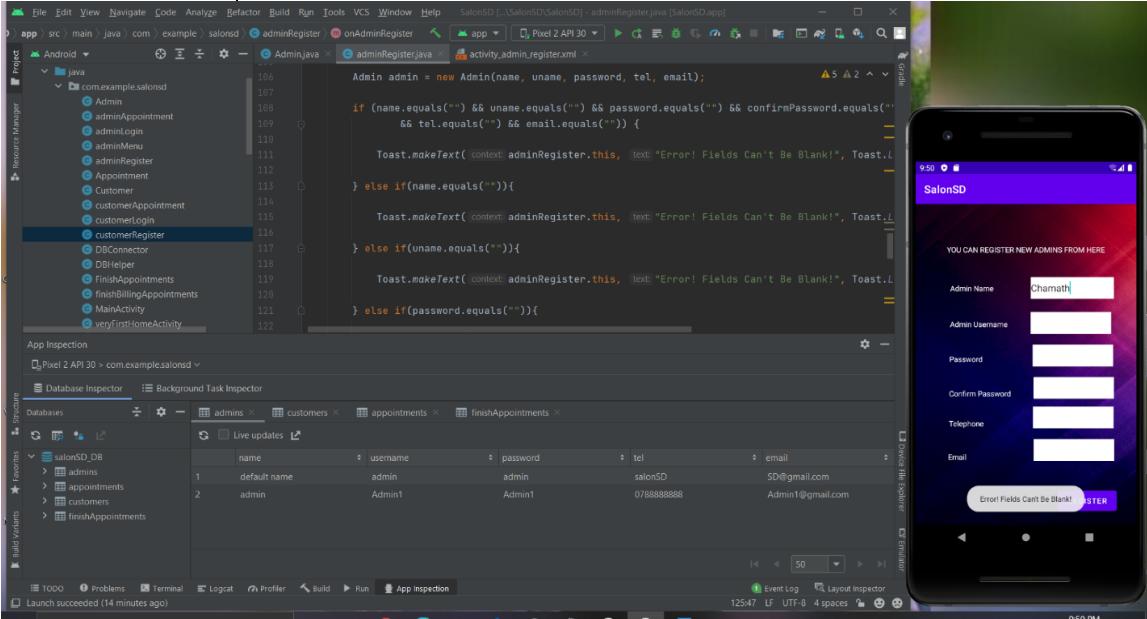
TEST CASE ID	TC1.5
TEST CASE NAME	Check Login working without any empty field and with saved & correct login credentials
TEST SUMMARY	Checking whether correctly filled username and password is working or not using newly registered username and password
TEST STEPS	<ol style="list-style-type: none"> Fill username field with a valid newly registered admin username Fill password field with a valid newly registered password Press on Login button
TEST DATA	Username: Admin1 Password: Admin123
EXPECTED RESULT	Admin menu interface should be loaded along with a message as Login granted successfully
ACTUAL RESULT	

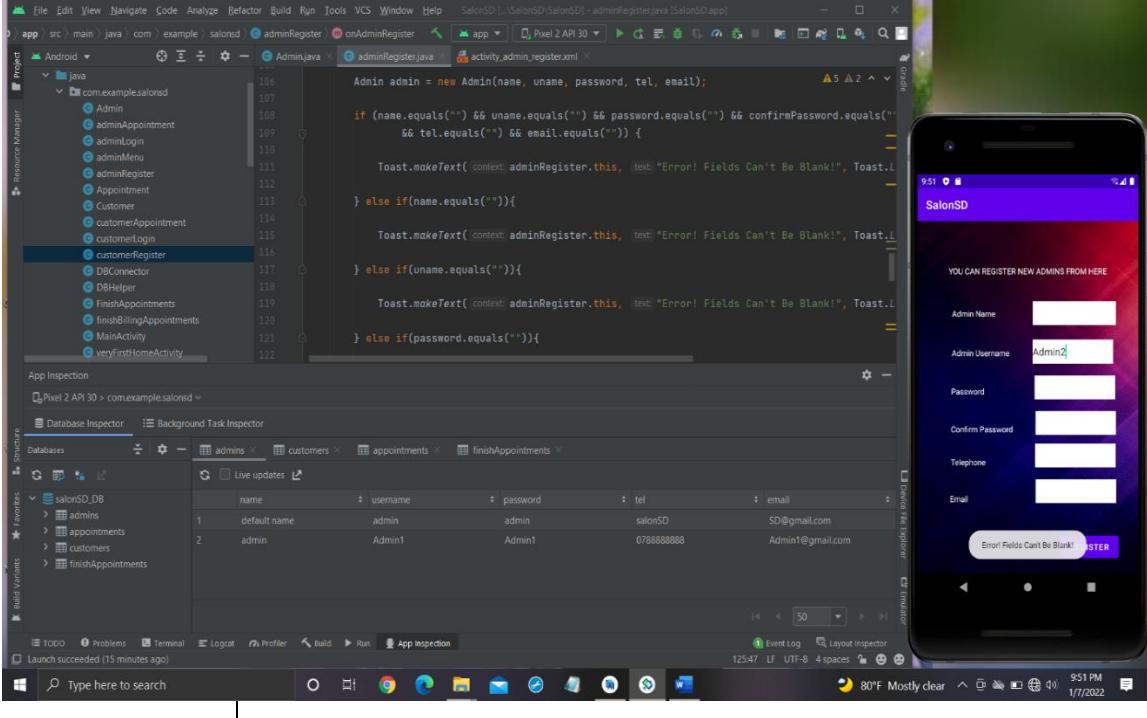
CONCLUSION	The expected login successful message displayed as a toast message and outcomes are very clear	
STATUS (PASS/FAIL)	Pass	

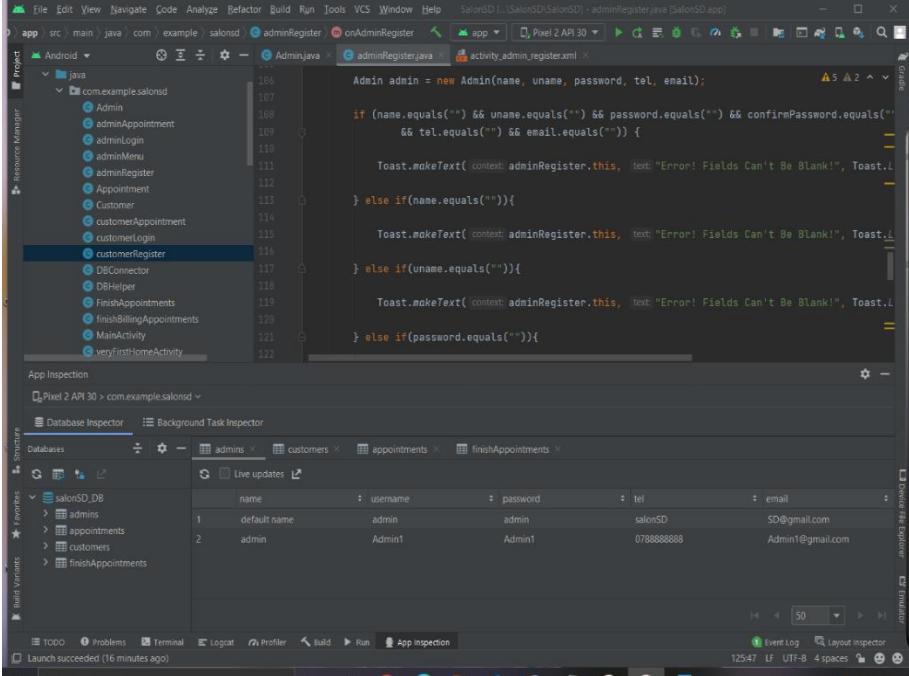
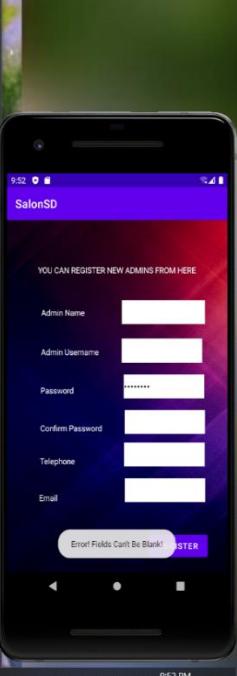
TC2 - Admin Registration

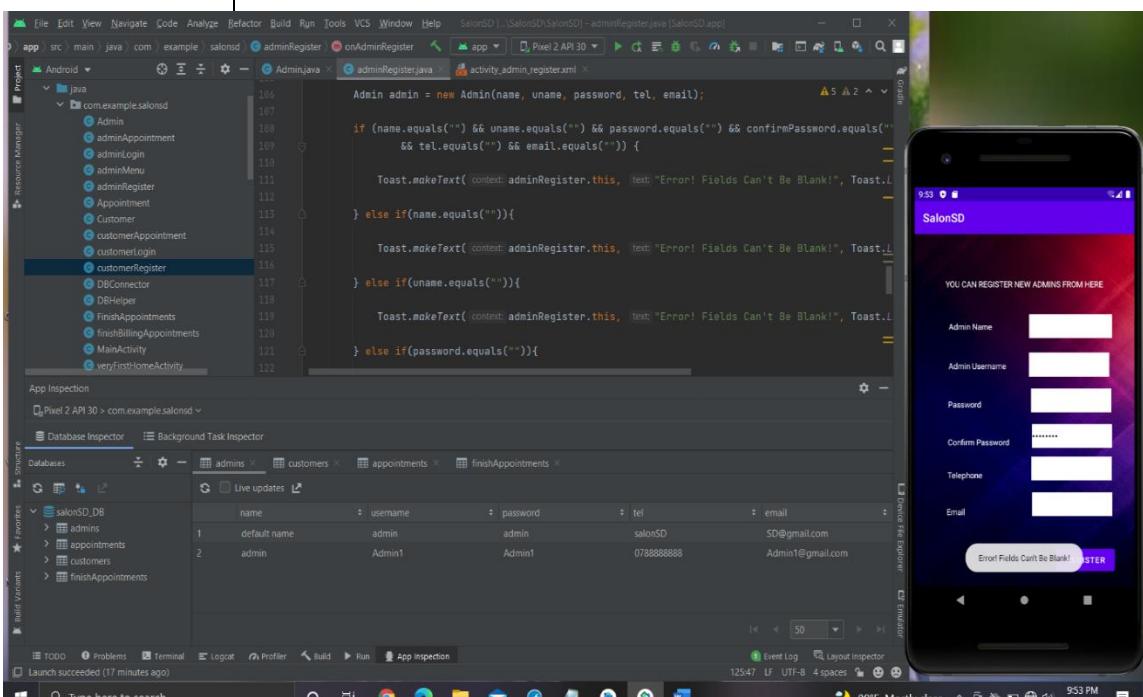
TEST CASE ID	TC2.1
TEST CASE NAME	Check Register empty fields
TEST SUMMARY	Checking the empty fields validation in Admin Register interface
TEST STEPS	Press on register button (without filling any field)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TC2.2 - Check by entering values one by one into fields

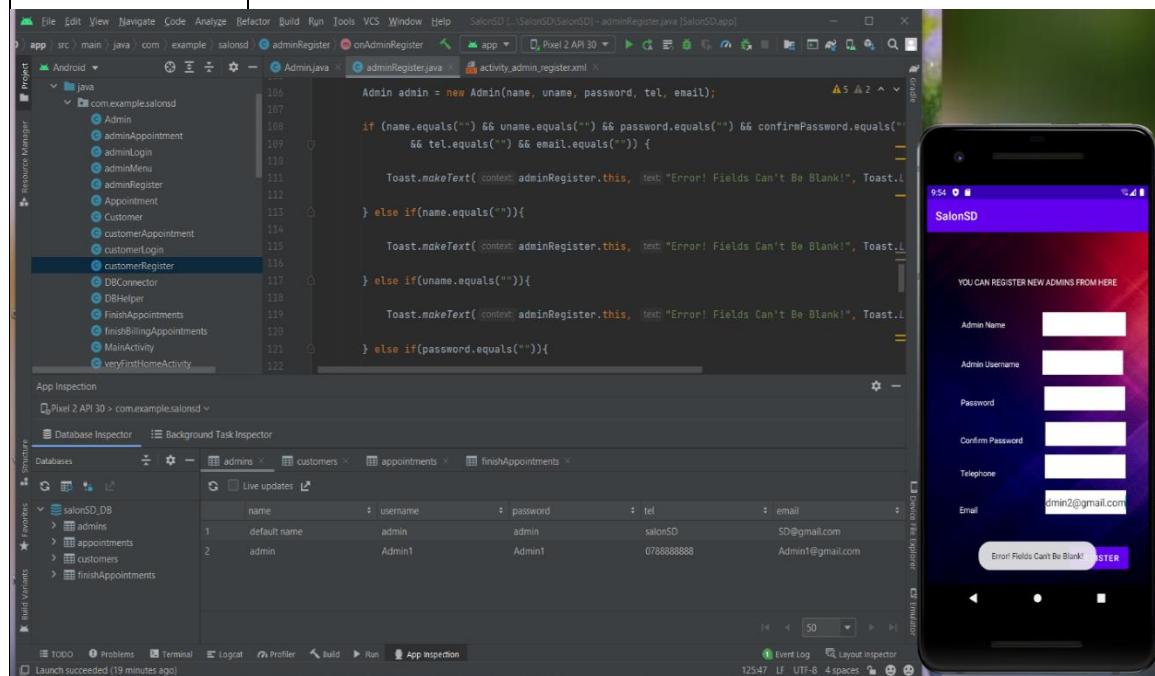
TEST CASE ID	TC2.2.1
TEST CASE NAME	Check by entering only admin name
TEST SUMMARY	Checking the empty fields validation in Admin Register by filling only admin name
TEST STEPS	<ol style="list-style-type: none"> Fill only admin name field Press on register button
TEST DATA	Admin name: Chamath
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

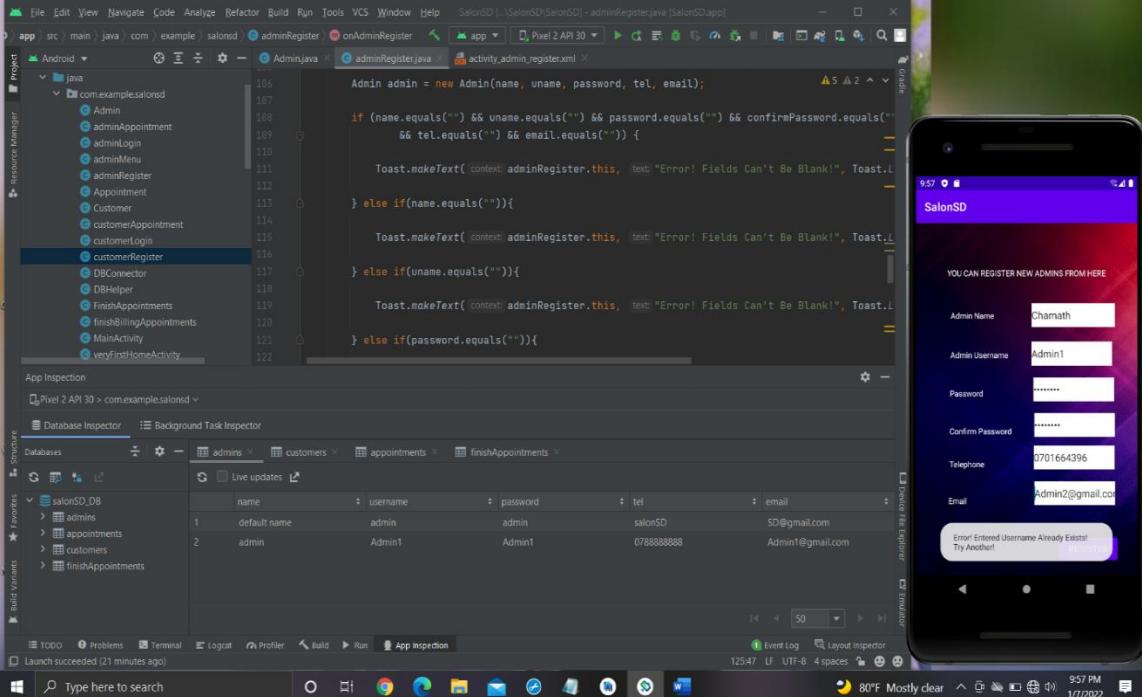
TEST CASE ID	TC2.2.2
TEST CASE NAME	Check by entering only username
TEST SUMMARY	Checking the empty fields validation in Admin Register by filling only username
TEST STEPS	<ol style="list-style-type: none"> Fill only username field Press on register button
TEST DATA	Username: Admin2
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

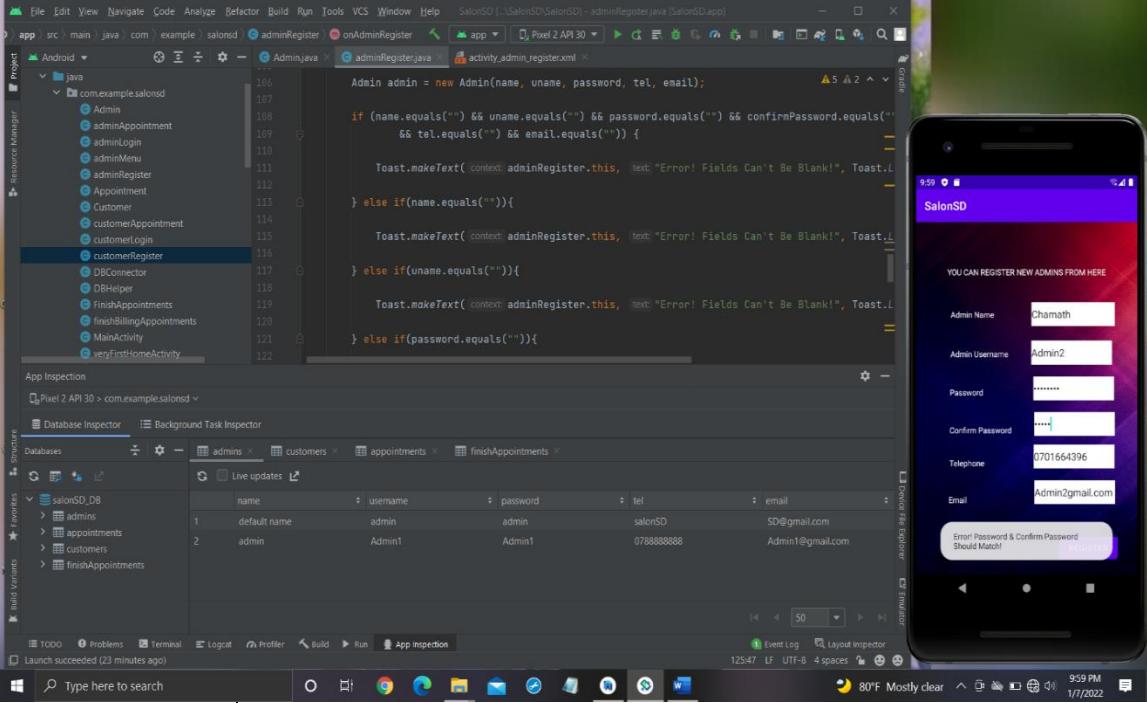
TEST CASE ID	TC2.2.3
TEST CASE NAME	Check by entering only password
TEST SUMMARY	Checking the empty fields validation in Admin Register by filling only password
TEST STEPS	<ol style="list-style-type: none"> Fill only password field Press on register button
TEST DATA	Password: Admin123
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	 
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

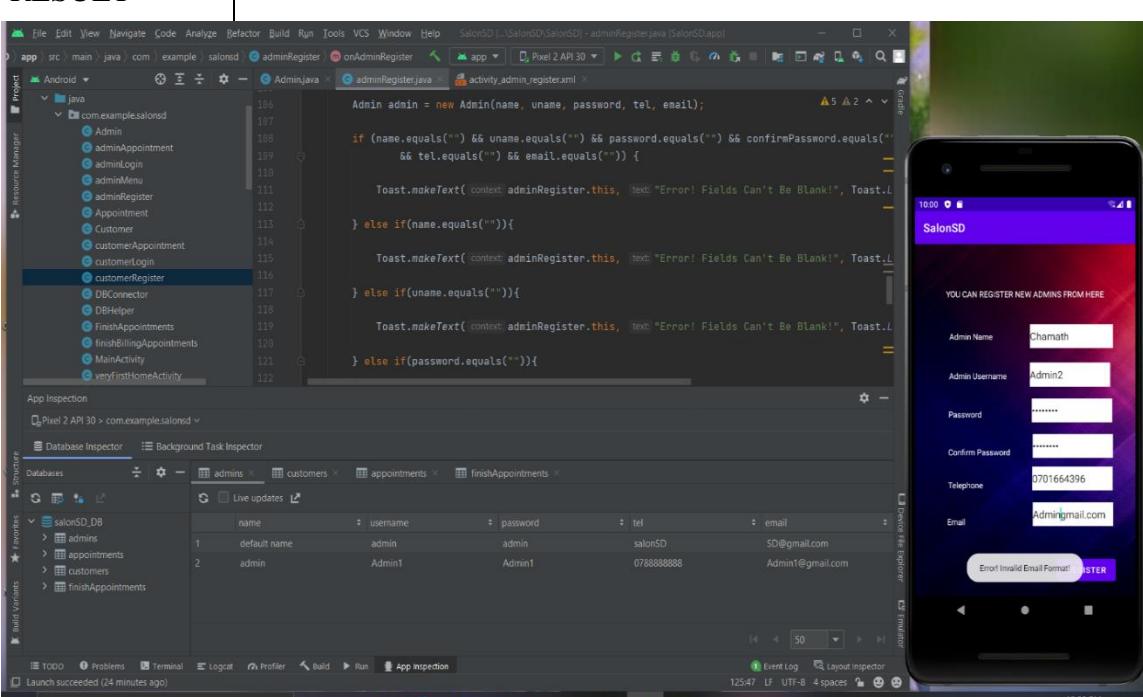
TEST CASE ID	TC2.2.4
TEST CASE NAME	Check by entering only confirm password
TEST SUMMARY	Checking the empty fields validation in Admin Register by filling only confirm password
TEST STEPS	<ol style="list-style-type: none"> Fill only confirm password field Press on register button
TEST DATA	Confirm password: Admin123
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

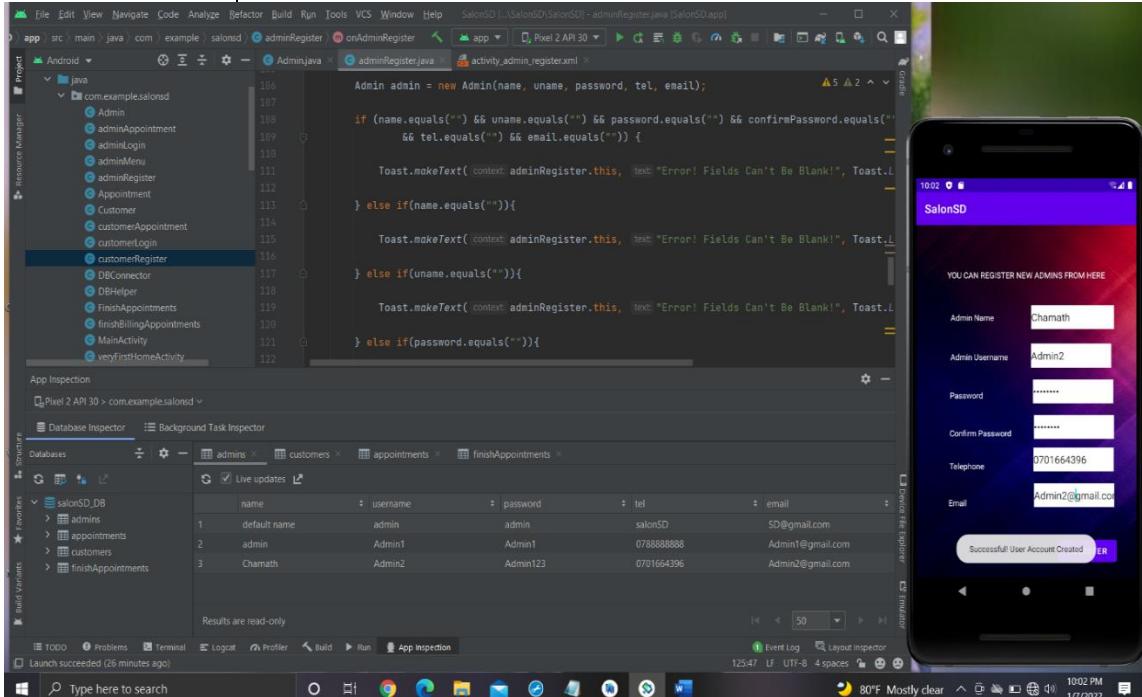
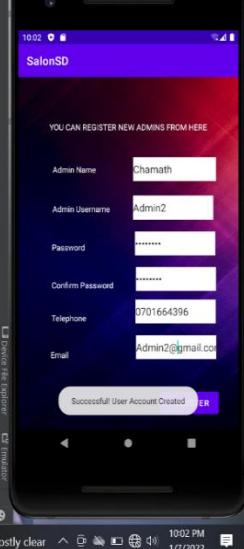
TEST CASE ID	TC2.2.5
TEST CASE NAME	Check by entering only telephone
TEST SUMMARY	Checking the empty fields validation in Admin Register by filling only telephone
TEST STEPS	<ol style="list-style-type: none"> Fill only telephone field Press on register button
TEST DATA	Telephone: 0701664396
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC2.2.6
TEST CASE NAME	Check by entering only email
TEST SUMMARY	Checking the empty fields validation in Admin Register by filling only email
TEST STEPS	<ol style="list-style-type: none"> Fill only email field Press on register button
TEST DATA	Email: Admin2@gmail.com
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

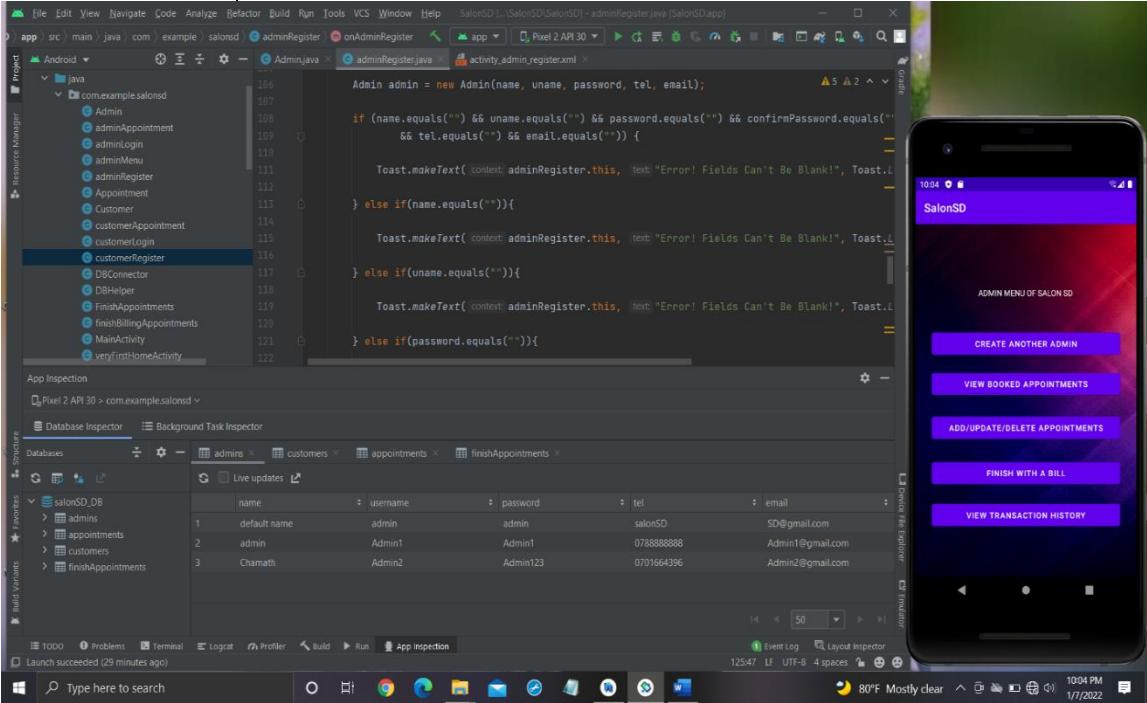
TEST CASE ID	TC2.3																		
TEST CASE NAME	Check by entering already saved username into the username field																		
TEST SUMMARY	Checking the already saved username validation in Admin Register by filling all other fields with a existing username in the username field																		
TEST STEPS	<ol style="list-style-type: none"> Fill all the fields Fill username field with already existing username Press on register button 																		
TEST DATA	Admin Name: Chamath Username: Admin1 Password: Admin123 Confirm Password: Admin123 Telephone: 0701664396 Email: Admin2@gmail.com																		
EXPECTED RESULT	Should display an Error Message as Entered username already exists																		
ACTUAL RESULT																			
 <p>The screenshot shows the Android Studio interface. On the left, the Project structure is visible with packages like com.example.salonSD and sub-packages like java containing various Java files. The main code editor shows AdminRegister.java with code related to user registration validation. The Database Inspector on the right shows the 'admins' table with two rows of data:</p> <table border="1"> <thead> <tr> <th></th> <th>name</th> <th>username</th> <th>password</th> <th>tel</th> <th>email</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>default name</td> <td>admin</td> <td>admin</td> <td>salonSD</td> <td>SD@gmail.com</td> </tr> <tr> <td>2</td> <td>admin</td> <td>Admin1</td> <td>Admin1</td> <td>0788888888</td> <td>Admin1@gmail.com</td> </tr> </tbody> </table>			name	username	password	tel	email	1	default name	admin	admin	salonSD	SD@gmail.com	2	admin	Admin1	Admin1	0788888888	Admin1@gmail.com
	name	username	password	tel	email														
1	default name	admin	admin	salonSD	SD@gmail.com														
2	admin	Admin1	Admin1	0788888888	Admin1@gmail.com														
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear																		
STATUS (PASS/FAIL)	Pass																		

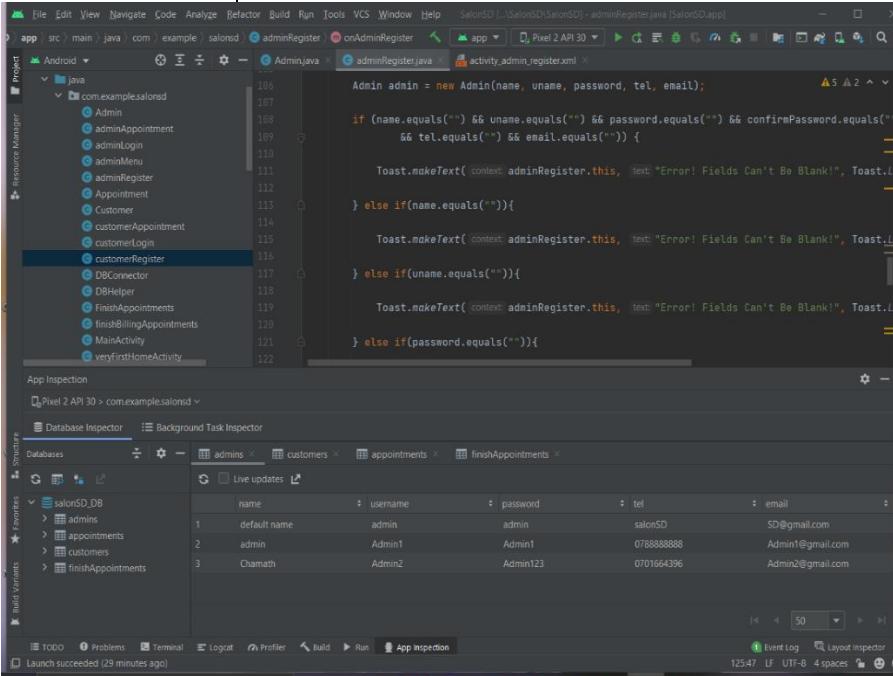
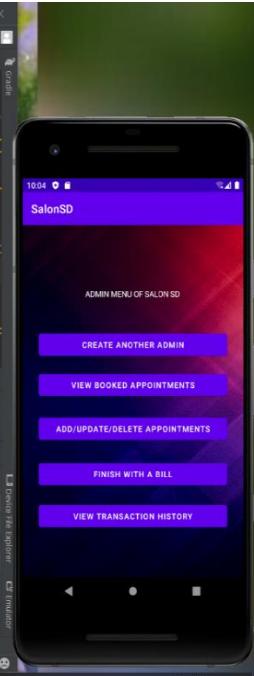
TEST CASE ID	TC2.4
TEST CASE NAME	Check password and confirm password matching
TEST SUMMARY	Checking the password and confirm password matching validation in Admin Register
TEST STEPS	<ol style="list-style-type: none"> Fill all the fields Fill username field with a new username Enter a password Enter a different confirm password Press on register button
TEST DATA	<p>Admin Name: Chamath Username: Admin2 Password: Admin123 Confirm Password: Admin Telephone: 0701664396 Email: Admin2@gmail.com</p>
EXPECTED RESULT	Should display an Error Message as Password & confirm password should match
ACTUAL RESULT	
	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

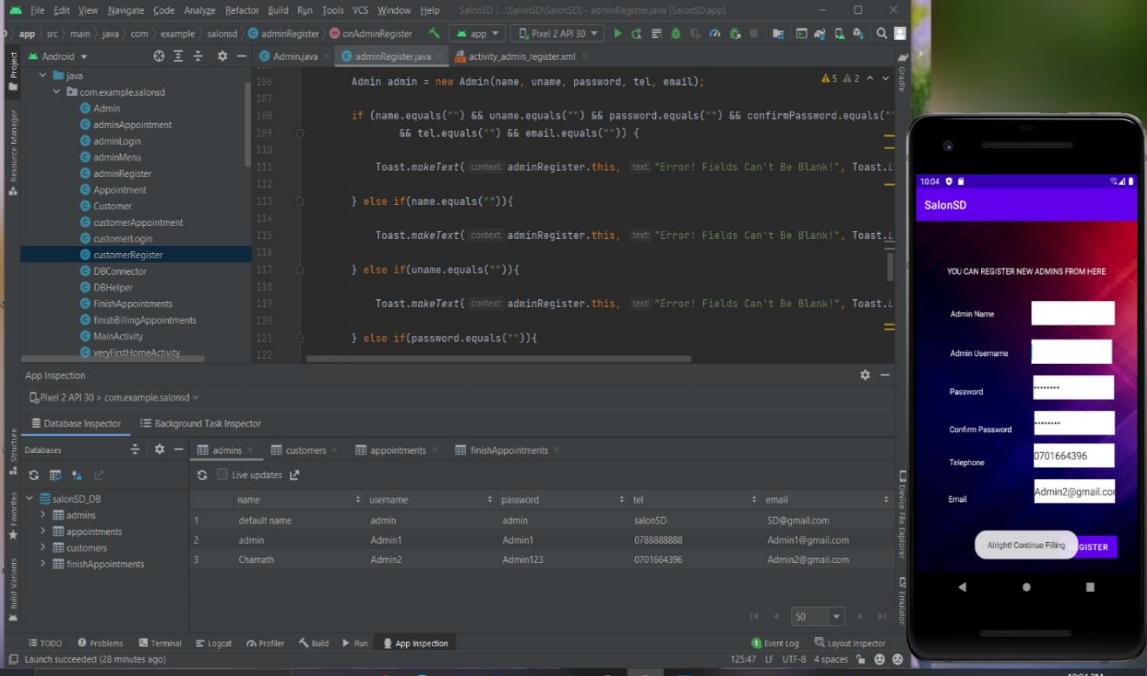
TEST CASE ID	TC2.5
TEST CASE NAME	Check email format
TEST SUMMARY	Checking the email format validation in Admin Register
TEST STEPS	<ol style="list-style-type: none"> Fill all the fields Fill username field with a new username Enter matching password and confirm password Enter email without the @ symbol Press on register button
TEST DATA	Admin Name: Chamath Username: Admin2 Password: Admin123 Confirm Password: Admin123 Telephone: 0701664396 Email: Admin2@gmail.com
EXPECTED RESULT	Should display an Error Message as Invalid Email format
ACTUAL RESULT	
 <p>The screenshot shows the Android Studio interface. On the left, the Project structure is visible with files like Admin.java, AdminRegister.java, and activity_admin_register.xml. In the center, the code editor displays Java code for an Admin object and various validation logic. On the right, an emulator for a Pixel 2 API 30 device shows a registration screen. The user has entered 'Chamath' for Admin Name, 'Admin2' for Admin Username, 'Admin123' for Password, 'Admin123' for Confirm Password, '0701664396' for Telephone, and 'Admin@gmail.com' for Email. A toast message at the bottom of the screen reads 'Error! Invalid Email Format!', indicating the validation logic worked as expected.</p>	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC2.6
TEST CASE NAME	Check Registration working without any empty field and with new correct data
TEST SUMMARY	Checking the registration with all correct values inside fields in Admin Register
TEST STEPS	<ol style="list-style-type: none"> Fill all the fields Press on register button
TEST DATA	<p>Admin Name: Chamath Username: Admin2 Password: Admin123 Confirm Password: Admin123 Telephone: 0701664396 Email: Admin2@gmail.com</p>
EXPECTED RESULT	Should display a message as User account created successfully
ACTUAL RESULT	 
CONCLUSION	The expected successful message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

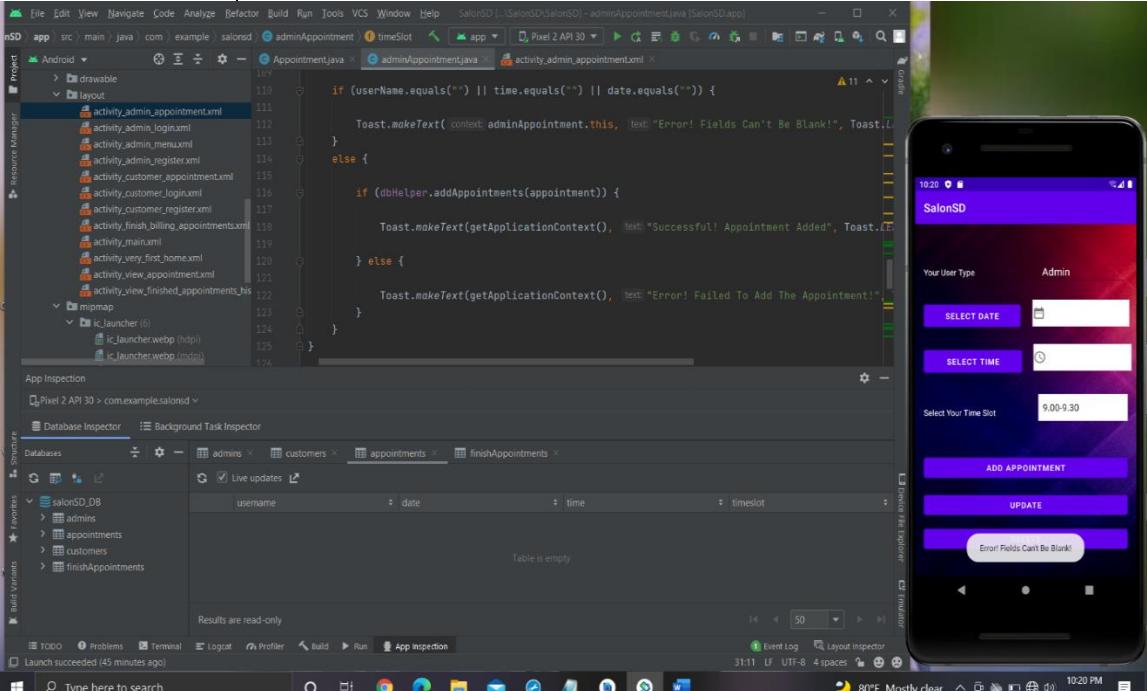
TC2.7 - Check on back when some fields are filled

TEST CASE ID	TC2.7.1
TEST CASE NAME	Check back button notification when some fields are filled
TEST SUMMARY	Checking Admin Registration back button with some empty fields
TEST STEPS	<ol style="list-style-type: none"> 1. Fill only few fields 2. Keep some empty fields 3. Press on back button of the phone
TEST DATA	Admin Name: Empty Username: Empty Password: Admin123 Confirm Password: Admin123 Telephone: 0701664396 Email: Admin2@gmail.com
EXPECTED RESULT	A notification should pop up and ask whether user wants to move back and get user's response.
ACTUAL RESULT	
 <p>The screenshot shows the Android Studio interface with the code for <code>adminRegister.java</code>. The code contains logic to check if all fields are empty and display an error toast. To the right, a screenshot of the app running on a Pixel 2 API 30 device shows the Admin Menu of SALON SD with several purple-themed buttons for managing appointments and transactions.</p>	
CONCLUSION	The expected notification displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass
TEST CASE ID	TC2.7.2

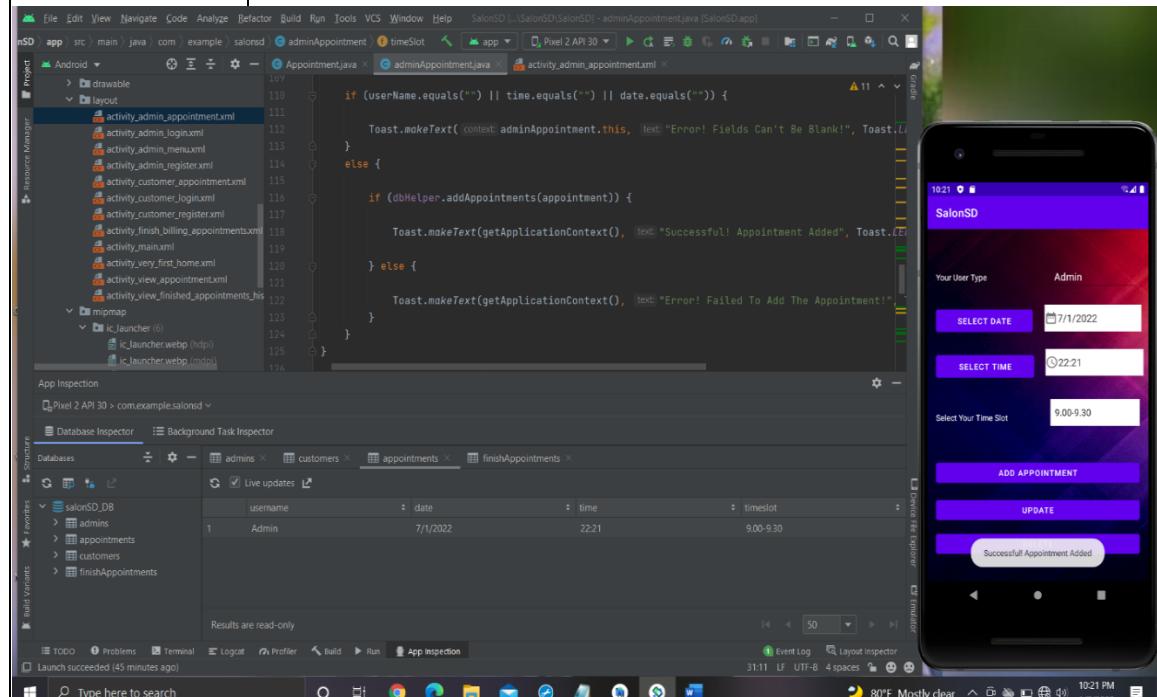
TEST CASE NAME	Check back button notification when user's response is YES
TEST SUMMARY	Checking YES response
TEST STEPS	Press on YES
TEST DATA	YES response
EXPECTED RESULT	If user provided YES, the previous interface will be displayed.
ACTUAL RESULT	 
CONCLUSION	The expected interface displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC2.7.2
TEST CASE NAME	Check back button notification when user's response is NO
TEST SUMMARY	Checking NO response
TEST STEPS	Press on NO
TEST DATA	NO response
EXPECTED RESULT	If user provided NO, that notification will fade away and a message will pop up mentioning that Continue Editing
ACTUAL RESULT	 The screenshot shows the Android Studio interface. On the left, the Project structure is visible with packages like com.example.salonSD containing classes such as Admin, AdminAppointment, AdminLogin, AdminMenu, AdminRegister, Appointment, Customer, customerAppointment, customerLogin, customerRegister, DBConnector, DBHelper, finishAppointments, finishBillingAppointments, MainActivity, and veryFirstHomeActivity. The main editor window displays AdminRegister.java with code for validating registration fields. On the right, the Database Inspector shows the salonSD database with a single table named admins containing three rows of data. The phone emulator on the right displays a registration screen with fields for Admin Name, Admin Username, Password, Confirm Password, Telephone, and Email. A toast message "Alright Continue Filling" is visible at the bottom of the screen.
CONCLUSION	The expected toast message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TC3 - Admin Appointment

TEST CASE ID	TC3.1
TEST CASE NAME	Check Add Appointment with empty fields
TEST SUMMARY	Checking empty fields validation in Admin Appointment interface
TEST STEPS	Press on Add Appointment button (without filling any field)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

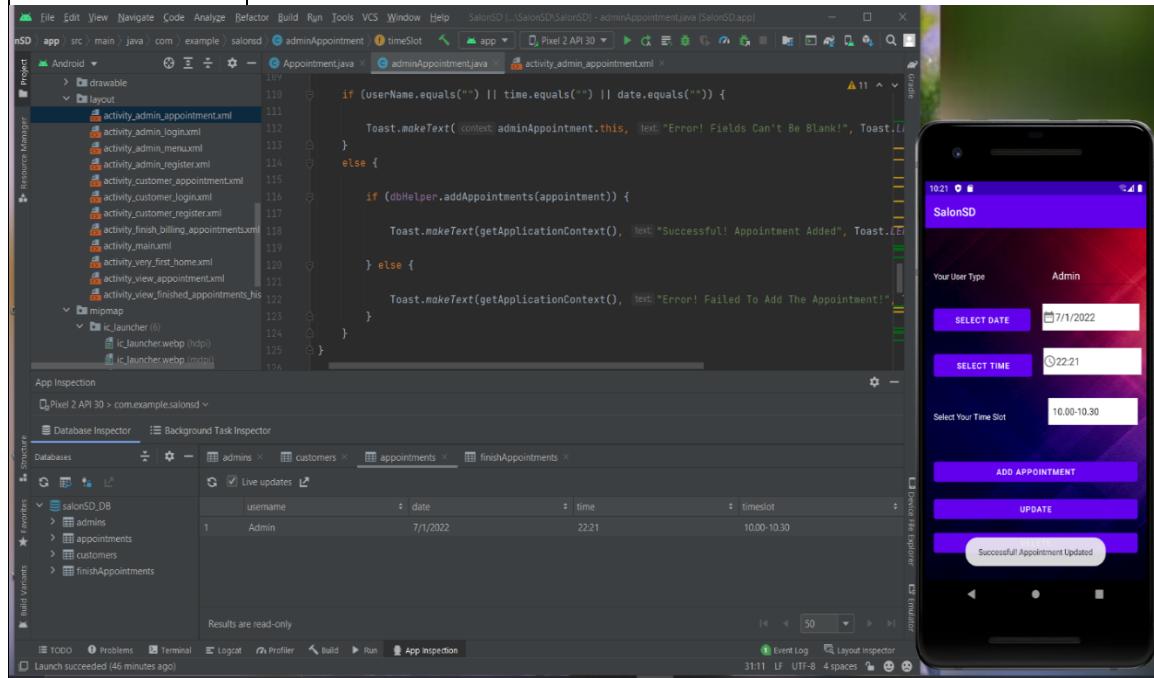
TEST CASE ID	TC3.2
TEST CASE NAME	Check Add Appointment working without any empty field
TEST SUMMARY	Checking correctly working of Add Appointment in Admin Appointment interface
TEST STEPS	<ol style="list-style-type: none"> 1. Have to fill all fields 2. Press on Add Appointment button
TEST DATA	User Type: Admin Date: Current Date Time: Current Time Time Slot: Select Time Slot (9.00-9.30) from the Spinner
EXPECTED RESULT	Should display a message as Appointment added successfully
ACTUAL RESULT	
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

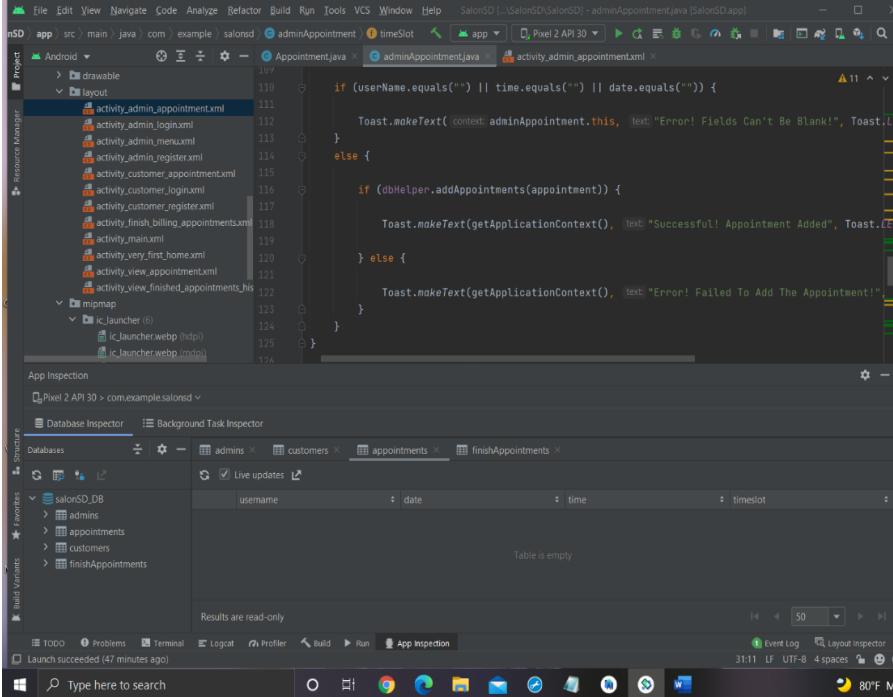
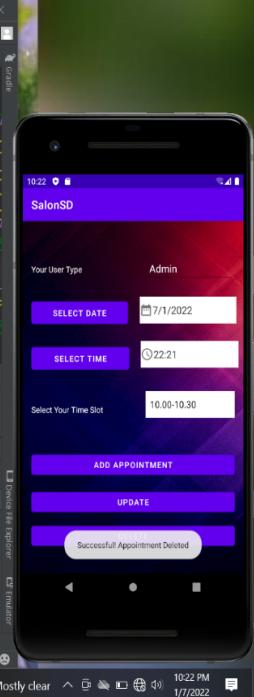


The screenshot shows the Android Studio environment with the following components:

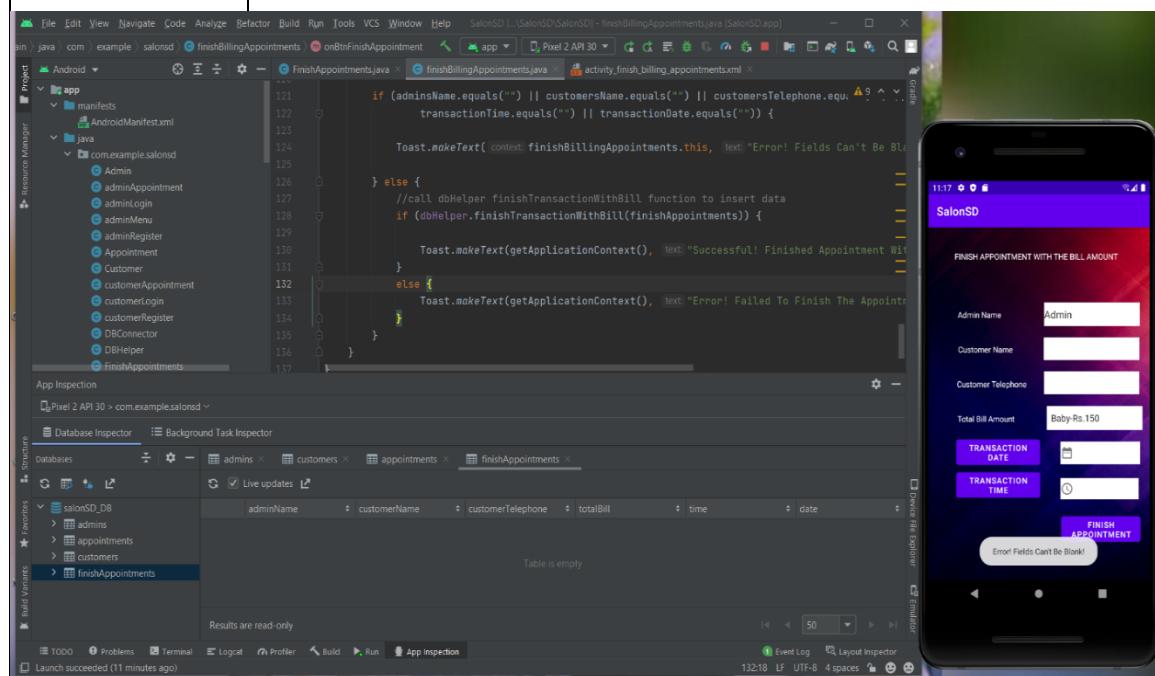
- Code Editor:** Displays the `adminAppointment.java` file containing Java code for handling appointment submissions.
- Emulator:** Shows a smartphone screen with the app's UI. It has fields for "SELECT DATE" (7/1/2022), "SELECT TIME" (22:21), and "Select Your Time Slot" (9.00-9.30). Buttons for "ADD APPOINTMENT", "UPDATE", and a success message "Successful Appointment Added" are visible.
- Database Inspector:** Shows the `salonSD_DB` database with a table named `appointments`. A single row is present with the values: username (Admin), date (7/1/2022), time (22:21), and timeslot (9.00-9.30).

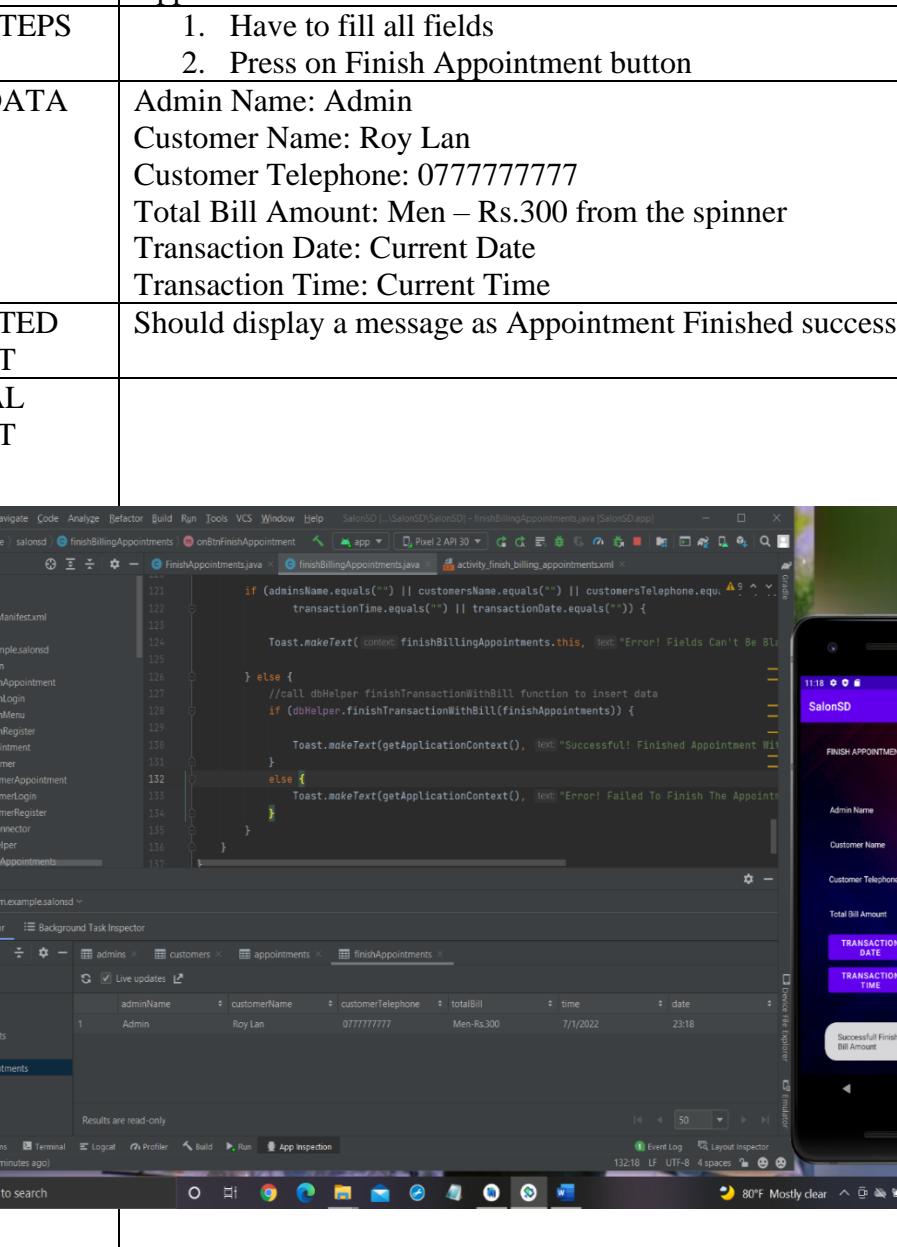
TEST CASE ID	TC3.3
TEST CASE NAME	Check Update Appointment working without any empty and updated fields
TEST SUMMARY	Checking correctly working of Update Appointment in Admin Appointment interface
TEST STEPS	<ol style="list-style-type: none"> Have to fill all fields Change the fields that are want to be changed Press on Update Appointment button
TEST DATA	<p>User Type: Admin Date: Change the date Time: Change the time Time Slot: Change the Time Slot(9.00-9.30) to the 10.00-10.30 Time Slot from the Spinner</p>
EXPECTED RESULT	Should display a message as Appointment updated successfully
ACTUAL RESULT	
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass



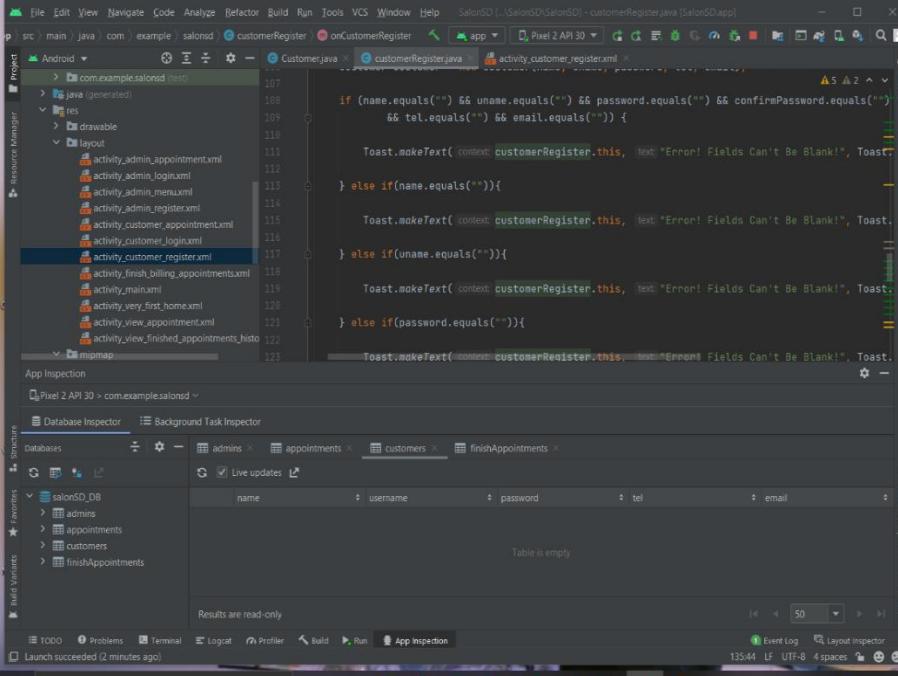
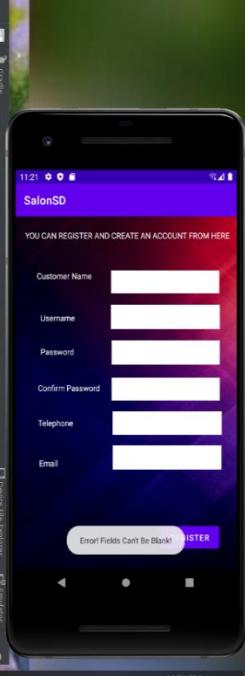
TEST CASE ID	TC3.4
TEST CASE NAME	Check Delete Appointment working
TEST SUMMARY	Checking correctly working of Delete Appointment in Admin Appointment interface
TEST STEPS	<ol style="list-style-type: none"> 1. Have to fill all fields (Specially the username field) 2. Press on Delete Appointment button
TEST DATA	Username: Admin
EXPECTED RESULT	Should display a message as Appointment deleted successfully
ACTUAL RESULT	 
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TC4 - Finish Billing Appointments

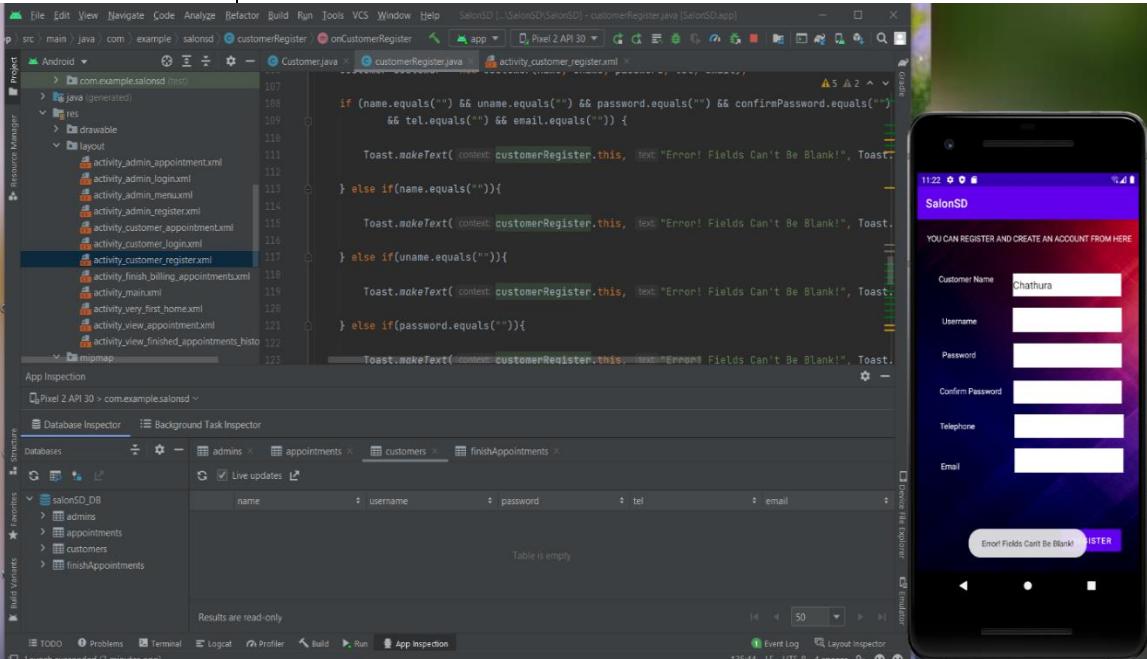
TEST CASE ID	TC4.1
TEST CASE NAME	Check Finish Appointment with empty fields
TEST SUMMARY	Checking empty fields validation in Finish Appointment interface
TEST STEPS	Press on Finish Appointment button (without filling any fields)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

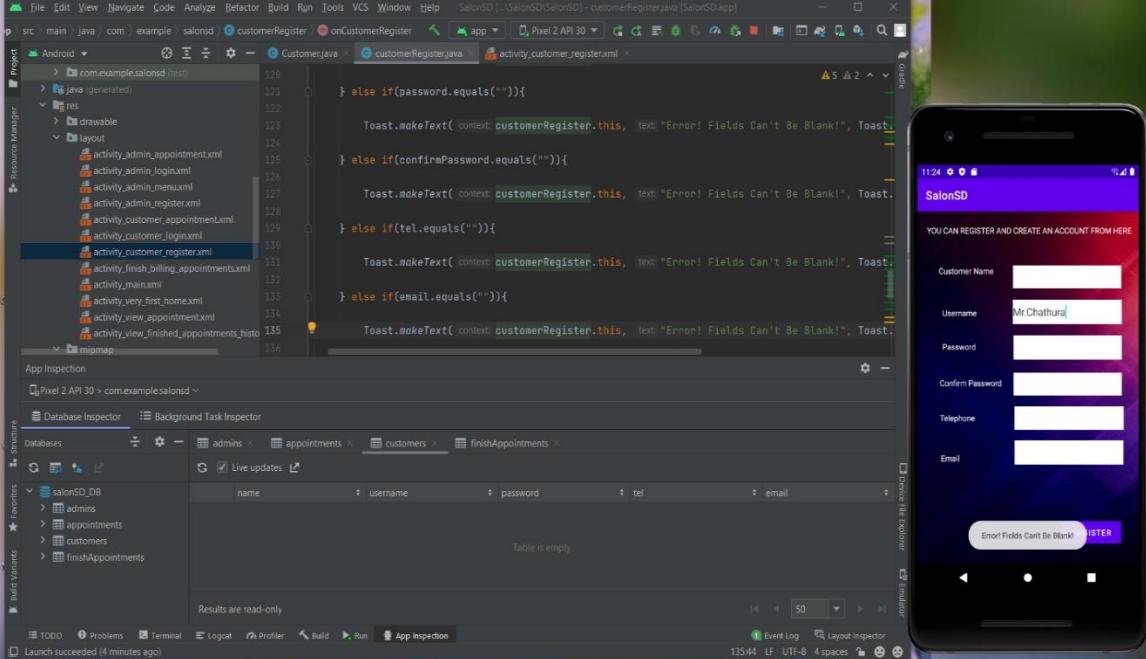
TEST CASE ID	TC4.2
TEST CASE NAME	Check Finish Appointment working without any empty field
TEST SUMMARY	Checking correctly working of Finish Appointment in Admin Finish Appointment interface
TEST STEPS	<ol style="list-style-type: none"> 1. Have to fill all fields 2. Press on Finish Appointment button
TEST DATA	<p>Admin Name: Admin Customer Name: Roy Lan Customer Telephone: 0777777777 Total Bill Amount: Men – Rs.300 from the spinner Transaction Date: Current Date Transaction Time: Current Time</p>
EXPECTED RESULT	Should display a message as Appointment Finished successfully
ACTUAL RESULT	
 <p>The screenshot shows the Android Studio interface. On the left, the Project structure is visible with packages like com.example.salonSD and sub-directories like app, manifests, and java containing various Java files such as Admin, adminAppointment, adminLogin, adminMenu, adminRegister, Appointment, Customer, customerAppointment, customerLogin, customerRegister, DBConnector, DBHelper, and FinishAppointments. The main code editor shows the implementation of the finishAppointments() method. It checks if all fields (adminName, customersName, customerTelephone, totalBill) are filled. If so, it calls the DBHelper's finishTransactionWithBill() method. A success message is displayed if the transaction is successful, or an error message if it fails. Below the code editor is a Database Inspector showing a table named appointments with one row: adminName: Admin, customerName: Roy Lan, customerTelephone: 0777777777, totalBill: Men-Rs300, time: 7/1/2022, date: 23:18.</p>	
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

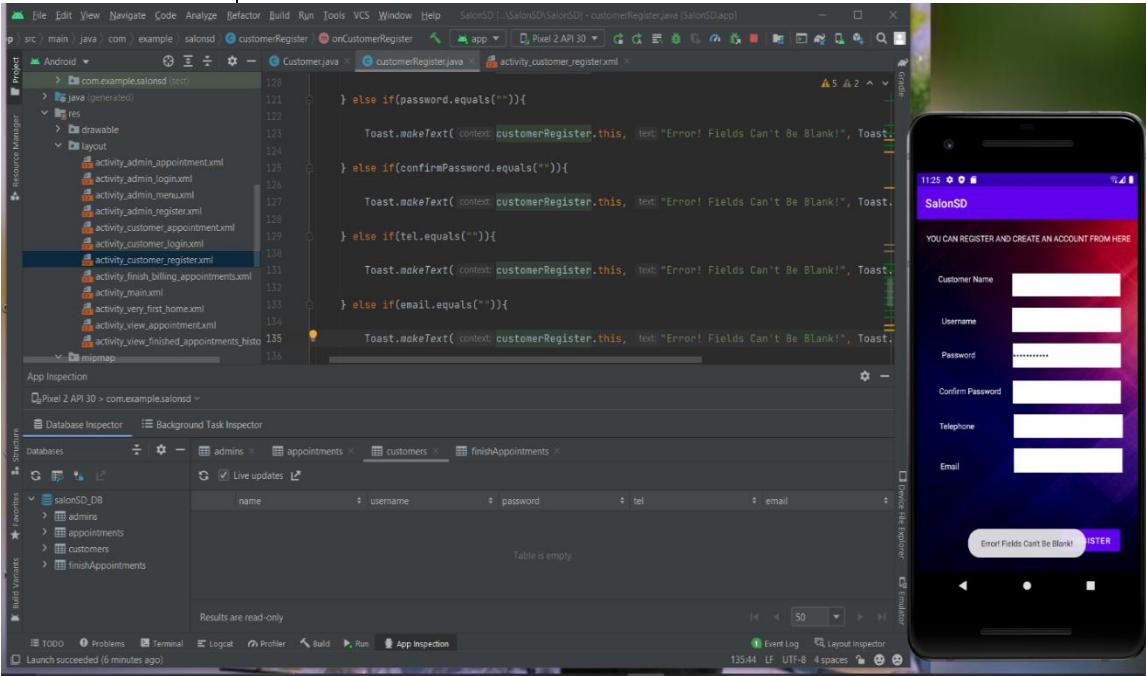
TC5 - Customer Registration

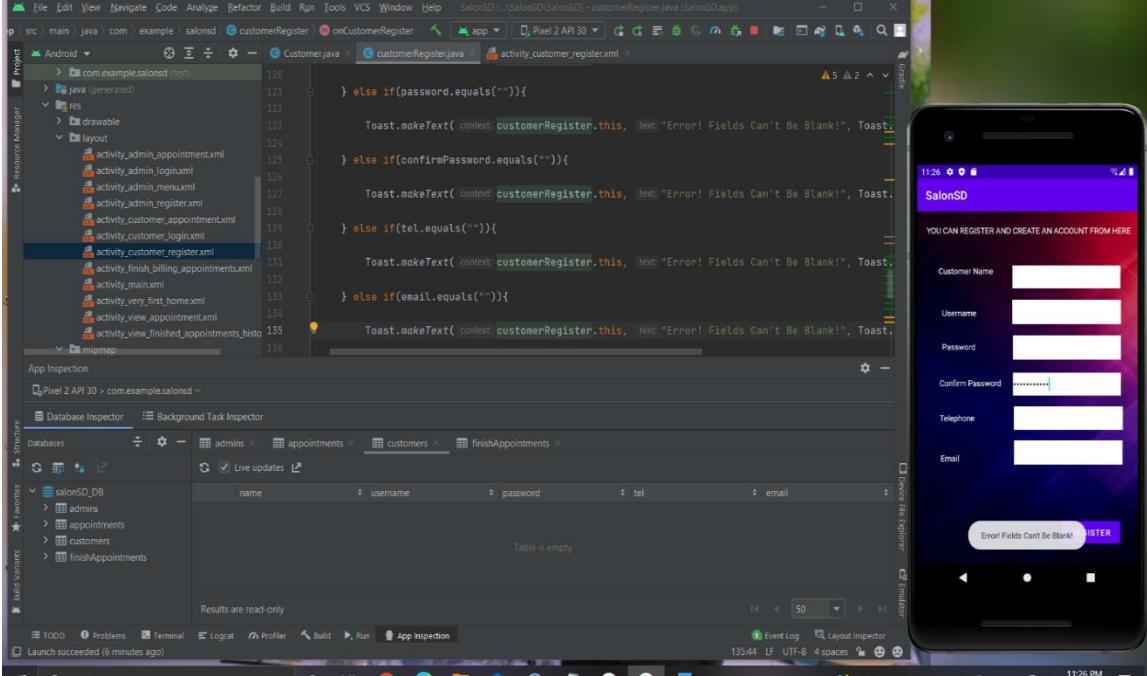
TEST CASE ID	TC5.1
TEST CASE NAME	Check Register empty fields
TEST SUMMARY	Checking the empty fields validation in Customer Register interface
TEST STEPS	Press on register button (without filling any field)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	 
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

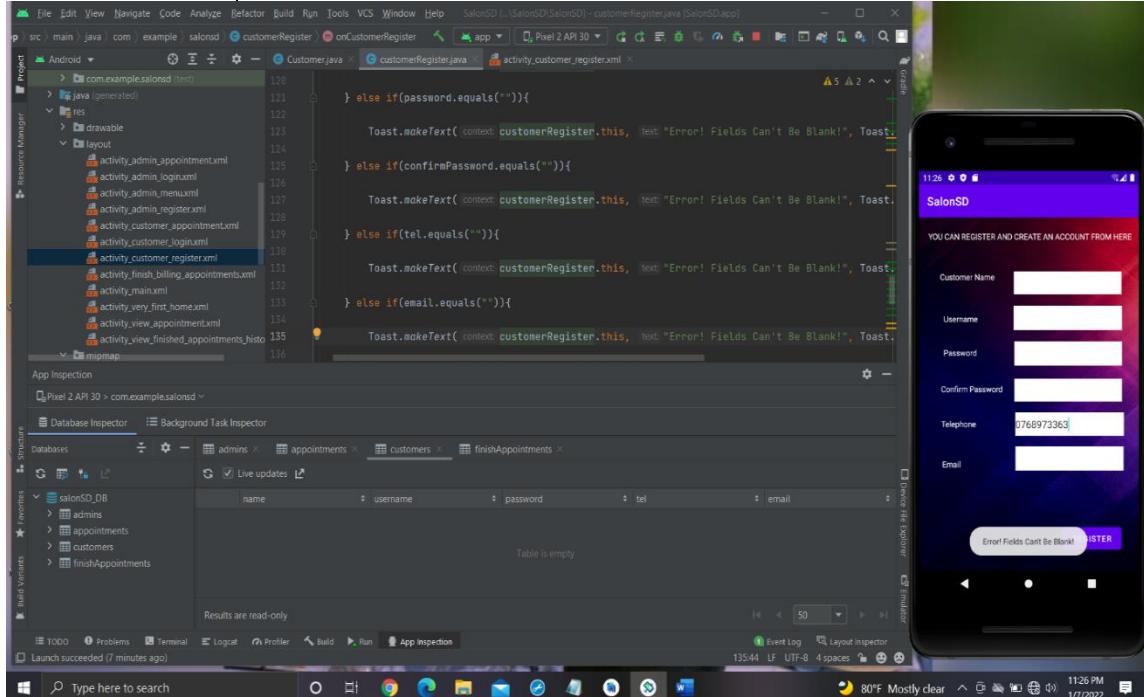
TC5.2 - Check by entering values one by one into fields

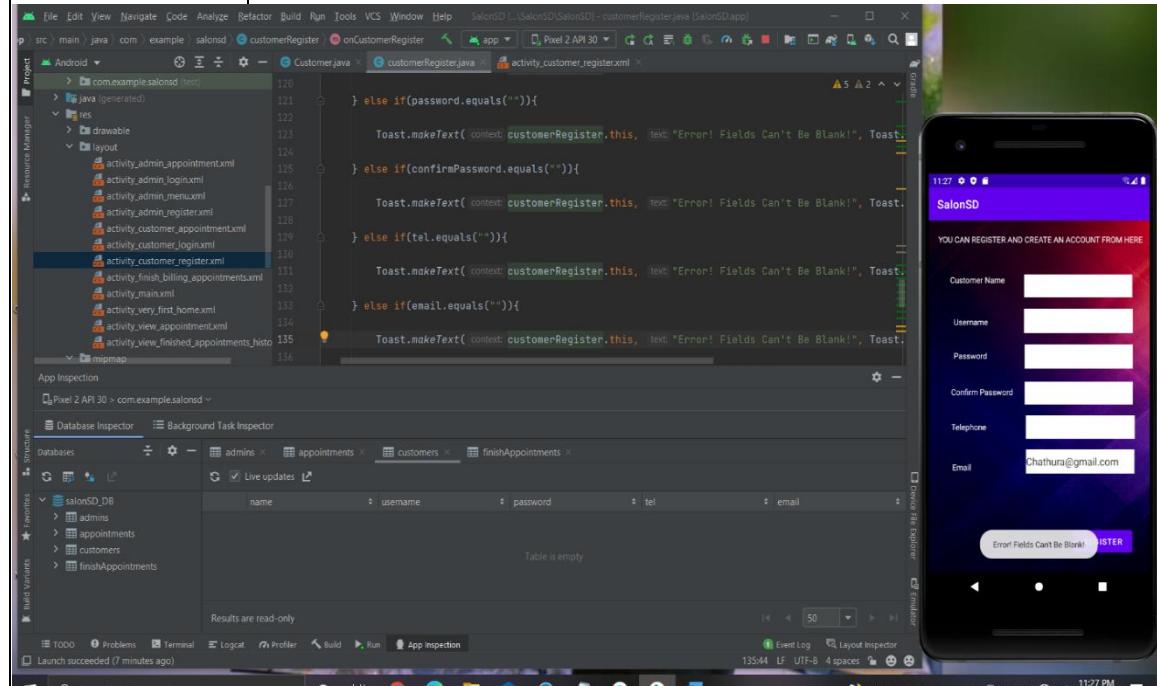
TEST CASE ID	TC5.2.1
TEST CASE NAME	Check by entering only customer name
TEST SUMMARY	Checking the empty fields validation in Customer Register by filling only customer name
TEST STEPS	<ol style="list-style-type: none"> 3. Fill only customer name field 4. Press on register button
TEST DATA	Customer name: Chathura
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC5.2.2
TEST CASE NAME	Check by entering only username
TEST SUMMARY	Checking the empty fields validation in Customer Register by filling only username
TEST STEPS	<p>3. Fill only username field 4. Press on register button</p>
TEST DATA	Username: Mr.Chathura
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

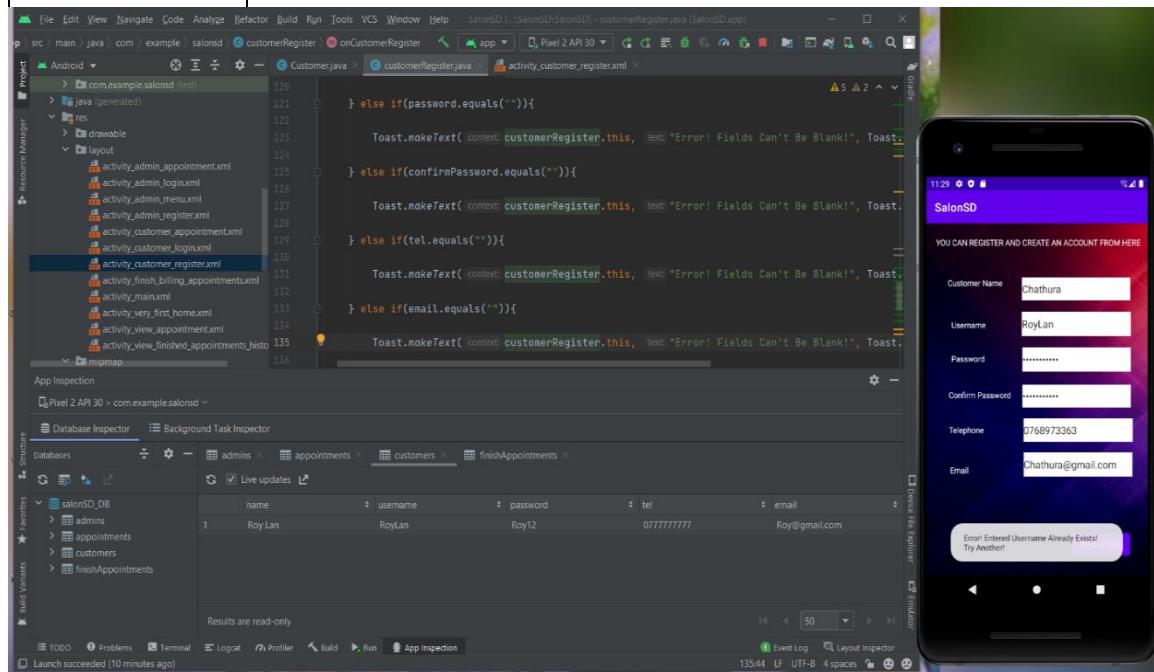
TEST CASE ID	TC5.2.3
TEST CASE NAME	Check by entering only password
TEST SUMMARY	Checking the empty fields validation in Customer Register by filling only password
TEST STEPS	<p>3. Fill only password field 4. Press on register button</p>
TEST DATA	Password: Chathura123
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC5.2.4
TEST CASE NAME	Check by entering only confirm password
TEST SUMMARY	Checking the empty fields validation in Customer Register by filling only confirm password
TEST STEPS	<ol style="list-style-type: none"> 3. Fill only confirm password field 4. Press on register button
TEST DATA	Confirm password: Chathura123
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

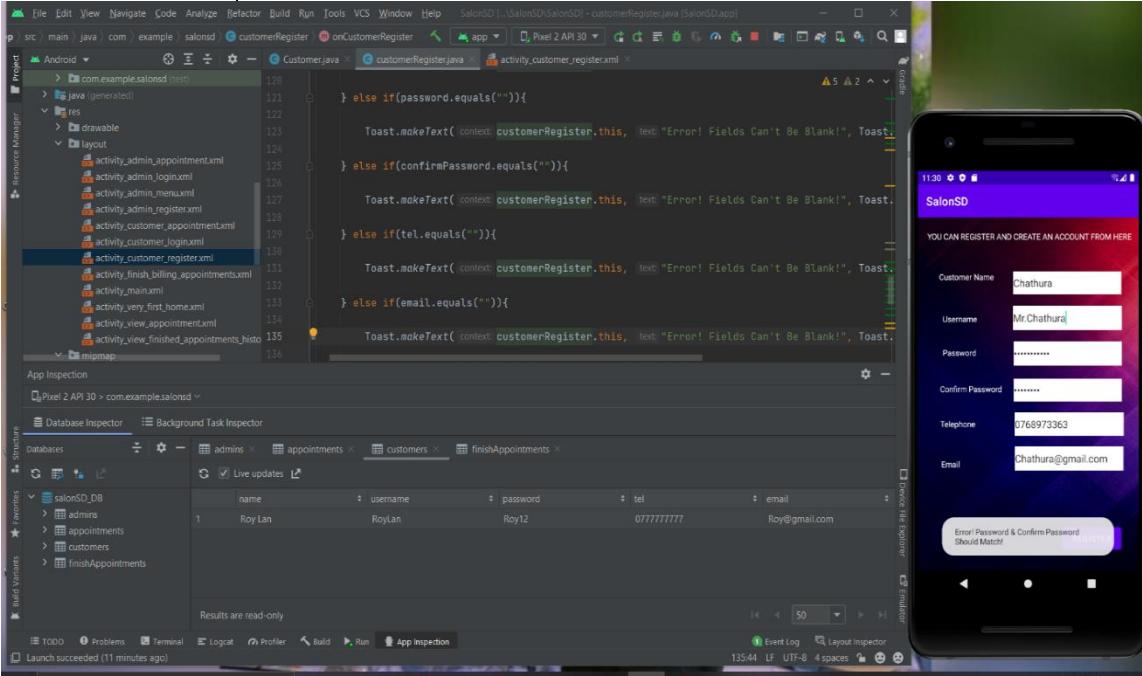
TEST CASE ID	TC5.2.5
TEST CASE NAME	Check by entering only telephone
TEST SUMMARY	Checking the empty fields validation in Customer Register by filling only telephone
TEST STEPS	<ol style="list-style-type: none"> 3. Fill only telephone field 4. Press on register button
TEST DATA	Telephone: 0768973363
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

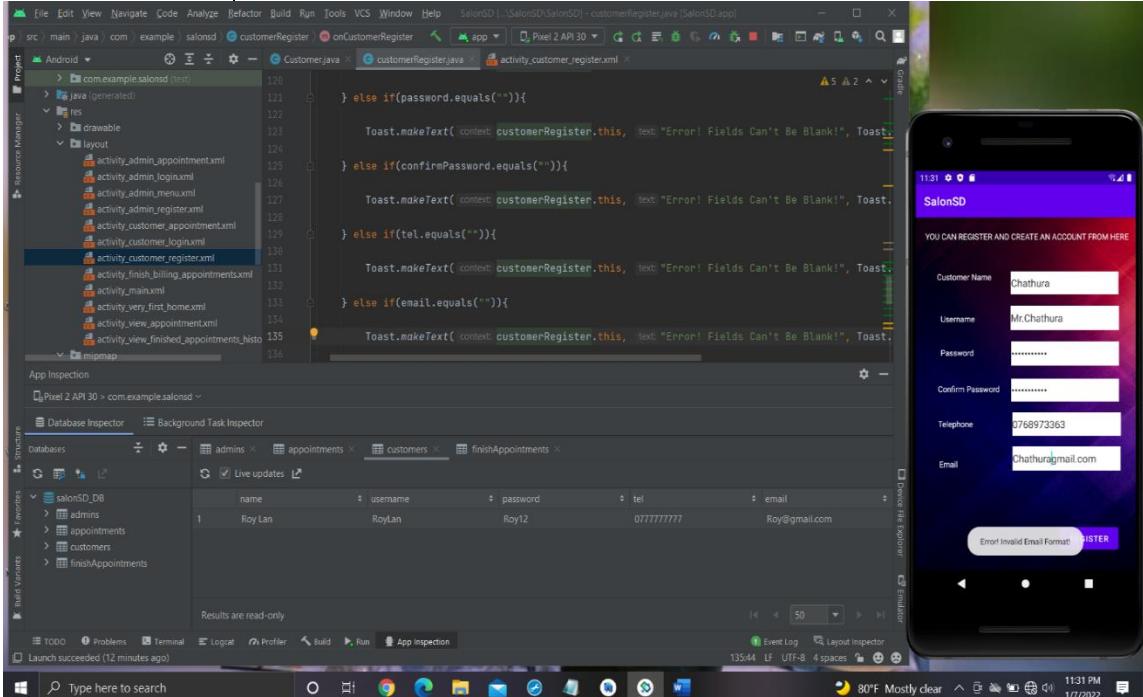
TEST CASE ID	TC5.2.6
TEST CASE NAME	Check by entering only email
TEST SUMMARY	Checking the empty fields validation in Customer Register by filling only email
TEST STEPS	<ol style="list-style-type: none"> 3. Fill only email field 4. Press on register button
TEST DATA	Email: Chathura@gmail.com
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

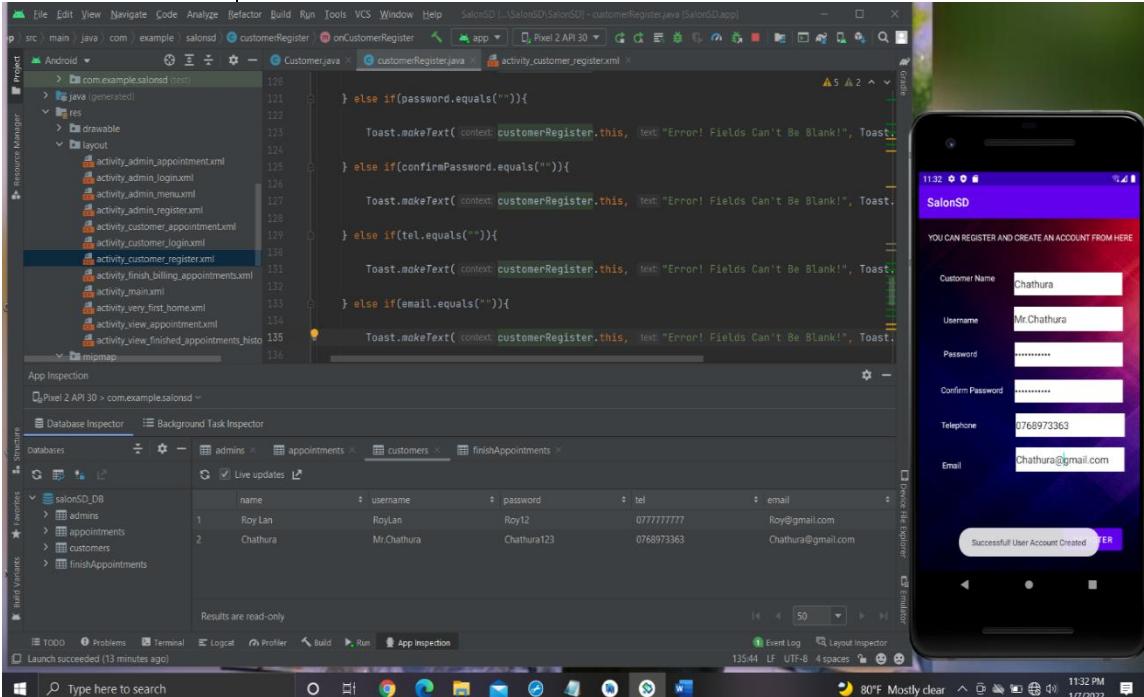
TEST CASE ID	TC2.3
TEST CASE NAME	Check by entering already saved username into the username field
TEST SUMMARY	Checking the already saved username validation in Customer Register by filling all other fields with an existing username in the username field
TEST STEPS	<ol style="list-style-type: none"> 4. Fill all the fields 5. Fill username field with already existing username 6. Press on register button
TEST DATA	Admin Name: Chathura Username: RoyLan Password: Chathura123 Confirm Password: Chathura123 Telephone: 0768973363 Email: Chathura@gmail.com
EXPECTED RESULT	Should display an Error Message as Entered username already exists
ACTUAL RESULT	



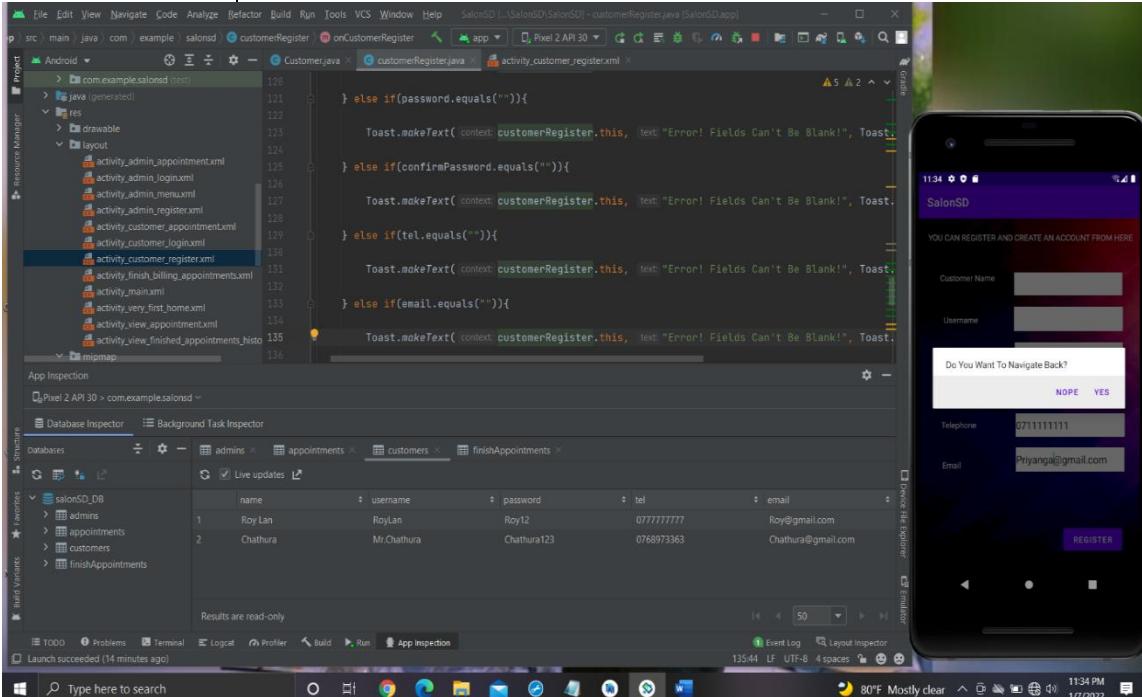
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

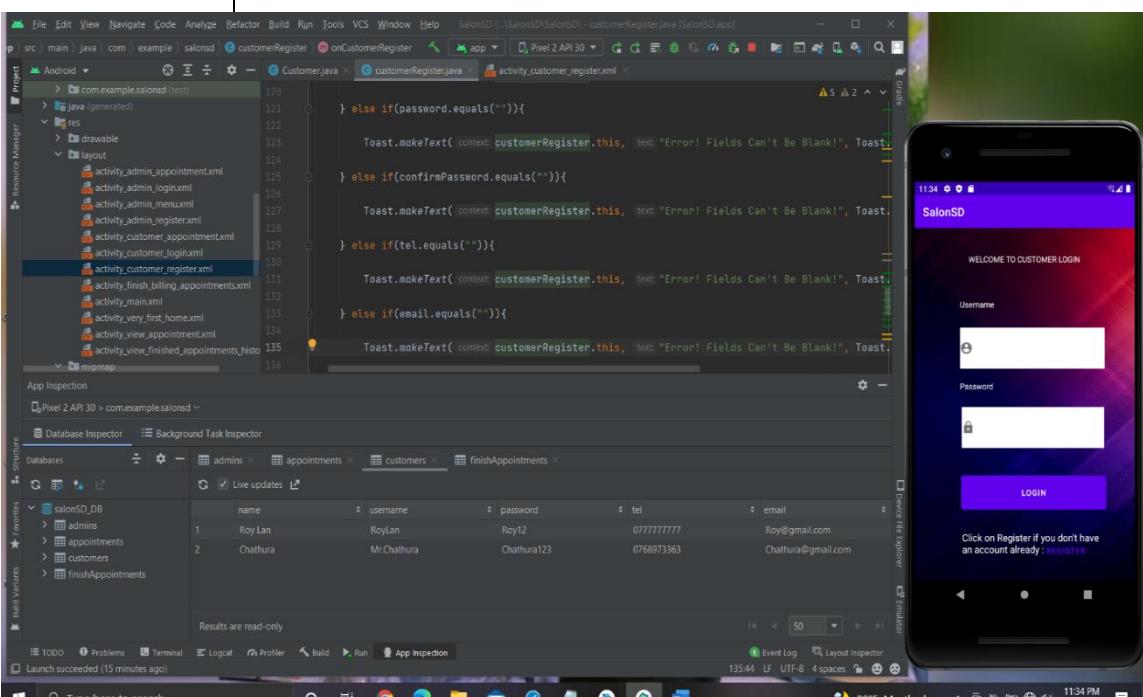
TEST CASE ID	TC5.4
TEST CASE NAME	Check password and confirm password matching
TEST SUMMARY	Checking the password and confirm password matching validation in Customer Register
TEST STEPS	<ol style="list-style-type: none"> 6. Fill all the fields 7. Fill username field with a new username 8. Enter a password 9. Enter a different confirm password 10. Press on register button
TEST DATA	Customer Name: Chathura Username: Mr.Chathura Password: Chathura123 Confirm Password: Chathura Telephone: 0768973363 Email: Chathura@gmail.com
EXPECTED RESULT	Should display an Error Message as Password & confirm password should match
ACTUAL RESULT	
 <p>The screenshot shows the Android Studio interface. On the left, the Project and Resource Manager panes are visible, showing Java files like com.example.salonSD.MainActivity and XML layouts for various activities. In the center, the code editor displays the Java code for customerRegister.java, specifically the logic for validating password and confirm password. On the right, the Device Preview pane shows a smartphone screen with a registration form. The form has fields for Customer Name, Username, Password, Confirm Password, Telephone, and Email. A toast message at the bottom of the screen reads "Error! Password & Confirm Password Should Match!". Below the preview, the Database Inspector shows the salonSD_DB with tables for admins, appointments, customers, and finishAppointments, containing sample data.</p>	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

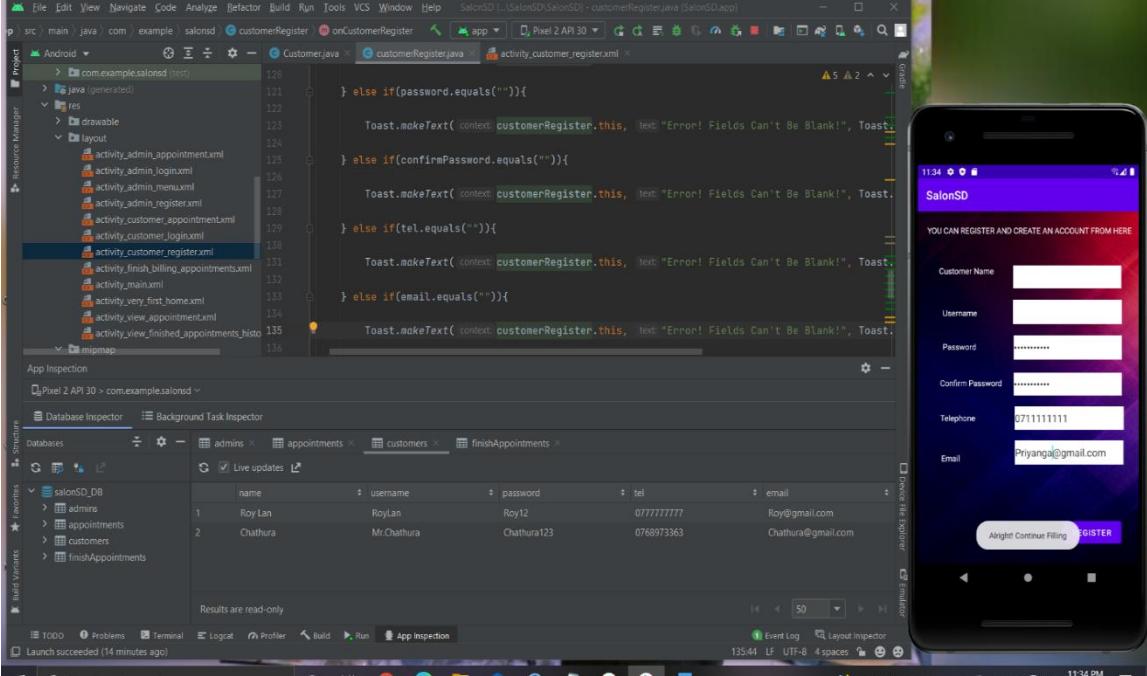
TEST CASE ID	TC5.5
TEST CASE NAME	Check email format
TEST SUMMARY	Checking the email format validation in Customer Register
TEST STEPS	<ol style="list-style-type: none"> 6. Fill all the fields 7. Fill username field with a new username 8. Enter matching password and confirm password 9. Enter email without the @ symbol 10. Press on register button
TEST DATA	Customer Name: Chathura Username: Mr.Chathura Password: Chathura123 Confirm Password: Chathura123 Telephone: 0768973363 Email: Chathuragmail.com
EXPECTED RESULT	Should display an Error Message as Invalid Email format
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC5.6
TEST CASE NAME	Check Registration working without any empty field and with new correct data
TEST SUMMARY	Checking the registration with all correct values inside fields in Customer Register
TEST STEPS	<ol style="list-style-type: none"> 3. Fill all the fields 4. Press on register button
TEST DATA	Customer Name: Chathura Username: Mr.Chathura Password: Chathura123 Confirm Password: Chathura123 Telephone: 0768973363 Email: Chathura@gmail.com
EXPECTED RESULT	Should display a message as User account created successfully
ACTUAL RESULT	
CONCLUSION	The expected successful message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

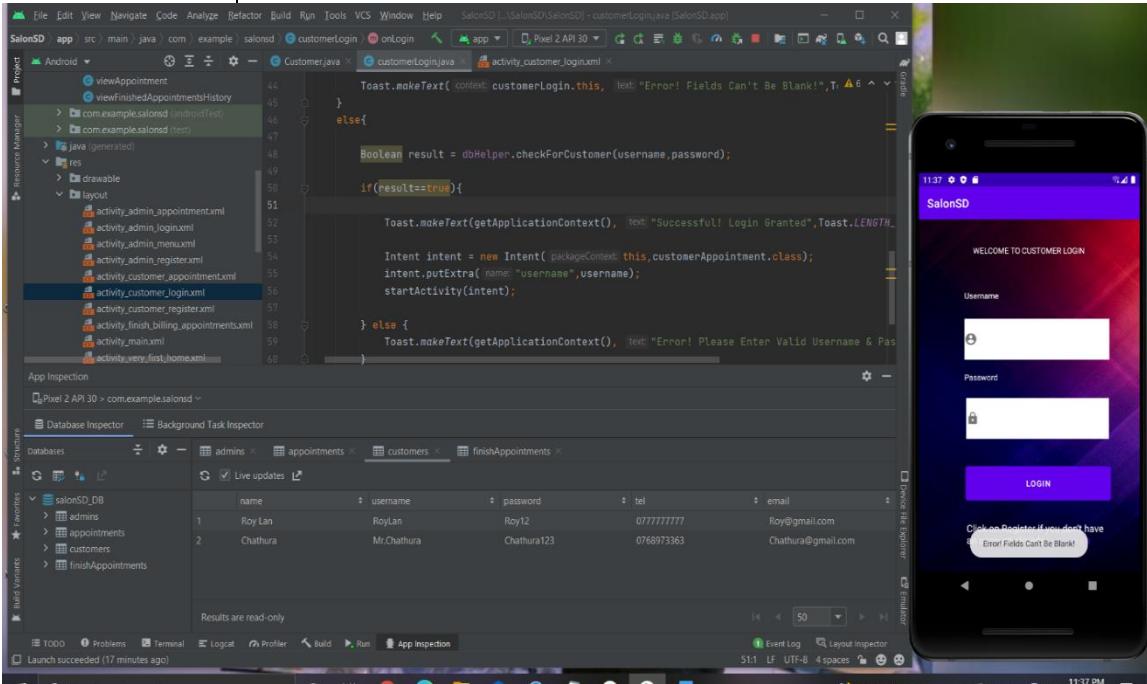
TC5.7 - Check on back when some fields are filled

TEST CASE ID	TC5.7.1
TEST CASE NAME	Check back button notification when some fields are filled
TEST SUMMARY	Checking Customer Registration back button with some empty fields
TEST STEPS	<ol style="list-style-type: none"> 4. Fill only few fields 5. Keep some empty fields 6. Press on back button of the phone
TEST DATA	Customer Name: Empty Username: Empty Password: Priyanga456 Confirm Password: Priyanga456 Telephone: 0711111111 Email: Priyanga@gmail.com
EXPECTED RESULT	A notification should pop up and ask whether user wants to move back and get user's response.
ACTUAL RESULT	
 <p>The screenshot shows the Android Studio interface with the code for <code>customerRegister.java</code>. The code includes logic to check if fields are empty and display a toast message if they are. To the right, a screenshot of the app on an iPhone X shows the registration screen. Several fields (Customer Name, Username, Password, Confirm Password, Telephone, Email) are empty. A toast message 'Do You Want To Navigate Back?' is displayed at the top of the screen, with 'NOPE' and 'YES' buttons below it.</p>	
CONCLUSION	The expected notification displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

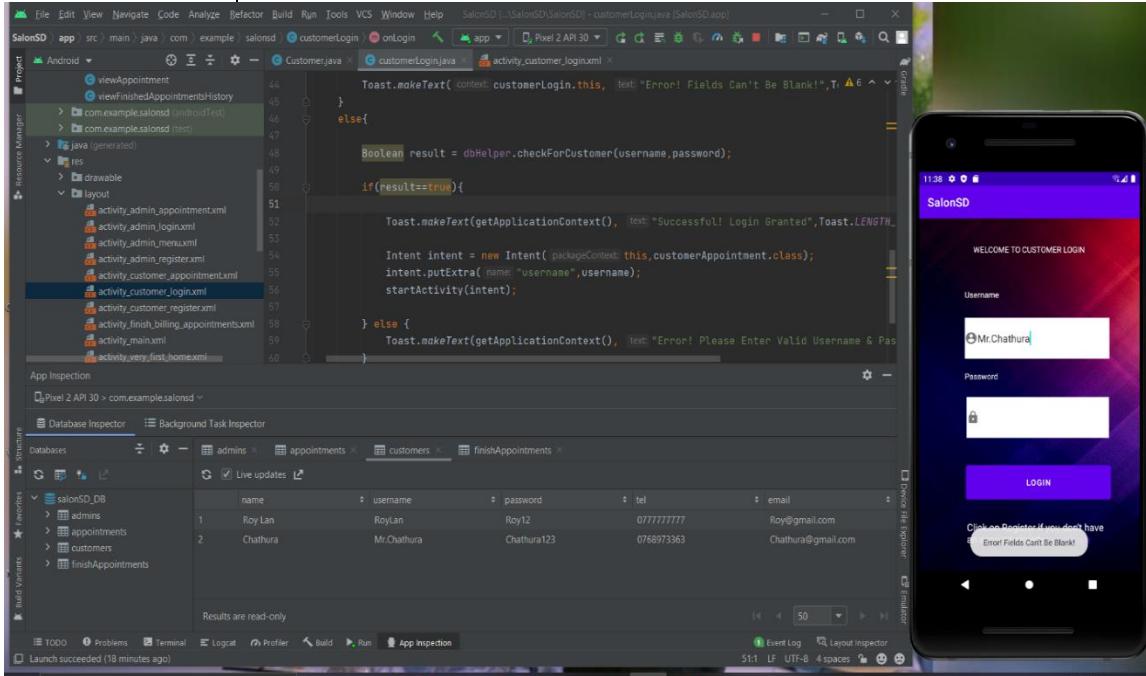
TEST CASE ID	TC5.7.2
TEST CASE NAME	Check back button notification when user's response is YES
TEST SUMMARY	Checking YES response
TEST STEPS	Press on YES
TEST DATA	YES response
EXPECTED RESULT	If user provided YES, the previous interface will be displayed.
ACTUAL RESULT	
CONCLUSION	The expected interface displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

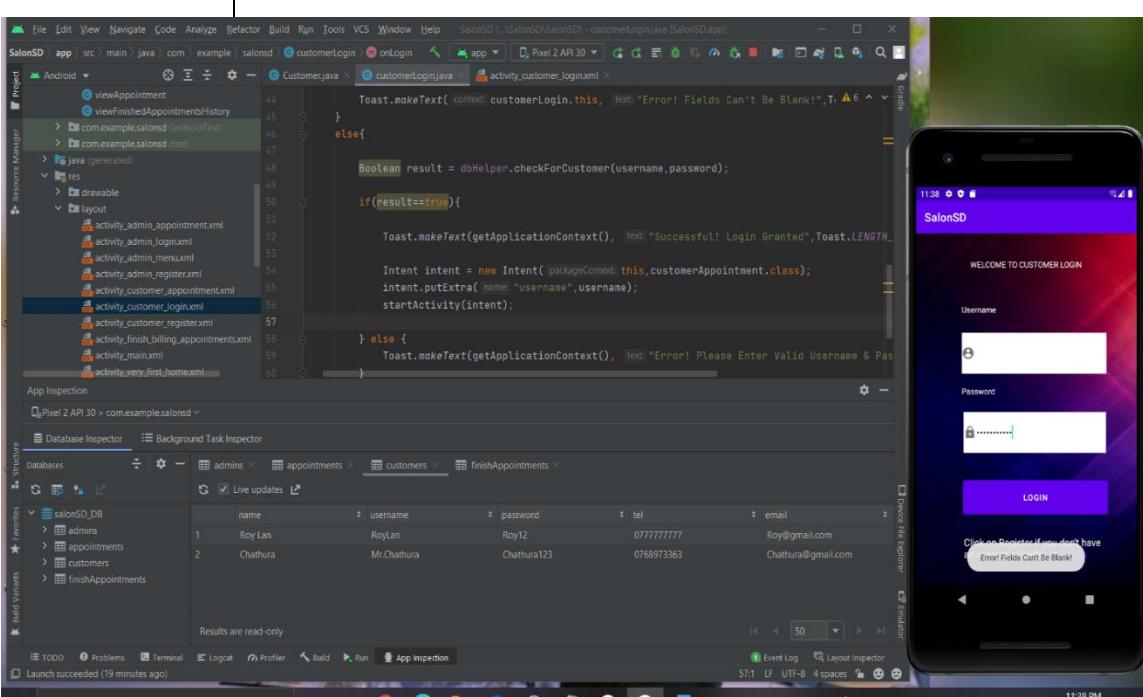
TEST CASE ID	TC5.7.2
TEST CASE NAME	Check back button notification when user's response is NO
TEST SUMMARY	Checking NO response
TEST STEPS	Press on NO
TEST DATA	NO response
EXPECTED RESULT	If user provided NO, that notification will fade away and a message will pop up mentioning that Continue Editing
ACTUAL RESULT	
CONCLUSION	The expected toast message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TC6 - Customer Login

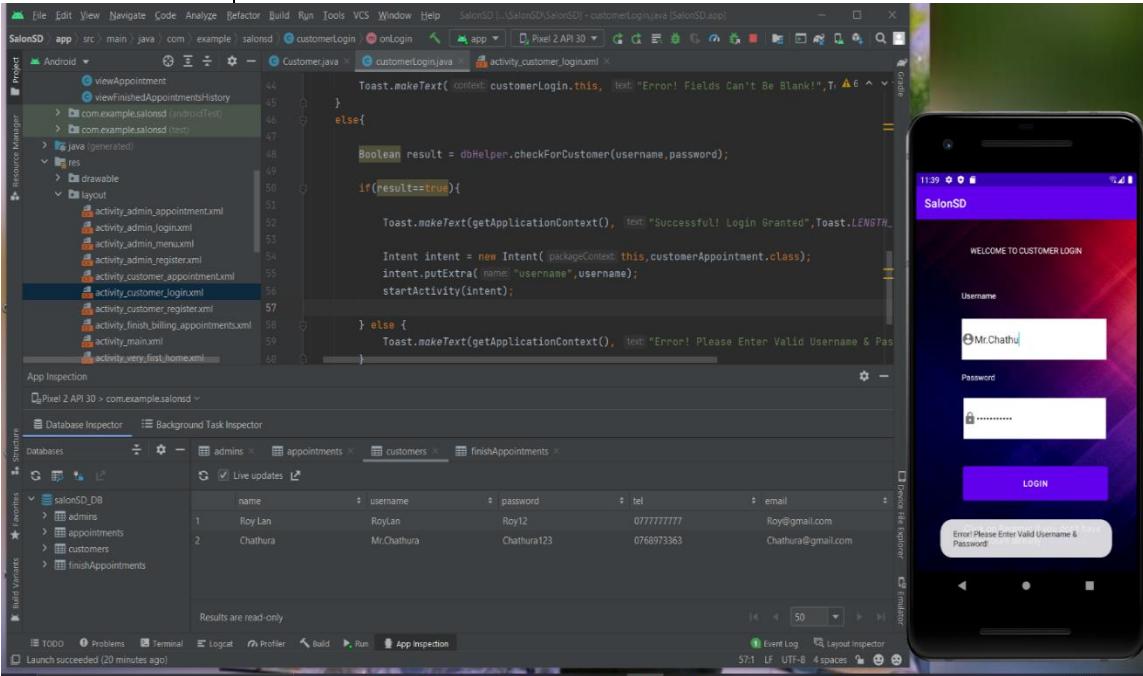
TEST CASE ID	TC6.1
TEST CASE NAME	Check Login empty fields
TEST SUMMARY	Checking the empty fields validation in Customer Login interface
TEST STEPS	Press on login button (without filling any field)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

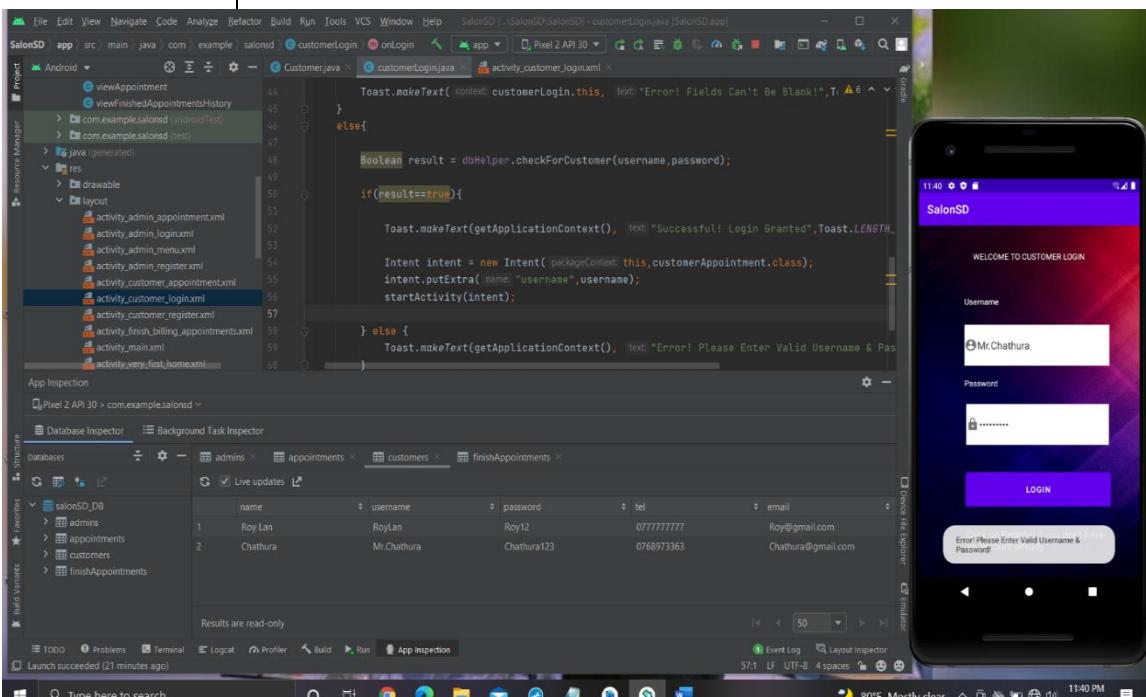
TC6.2 - Check fields by entering values one by one into fields

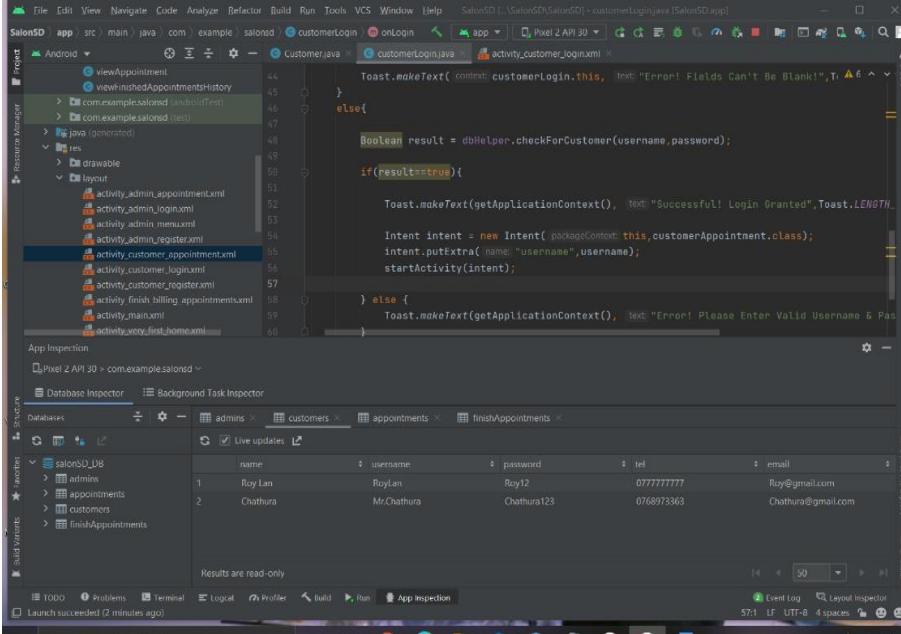
TEST CASE ID	TC6.2.1
TEST CASE NAME	Check by filling only username field
TEST SUMMARY	Checking empty field validation by filling only one field
TEST STEPS	<ol style="list-style-type: none"> 3. Fill only username field with a valid customer username 4. Press on Login button
TEST DATA	Username: Mr.Chathura Password: empty
EXPECTED RESULT	Should display the same Error Message which displayed previously as Fields can't be blank
ACTUAL RESULT	 <p>The screenshot shows the Android Studio interface with the code for CustomerLogin.java. The code checks if both username and password are provided. If either is empty, it displays a toast message: "Error! Fields Can't Be Blank!". The screenshot also shows the mobile application's login screen where the user has entered a valid username ("Mr.Chathura") but left the password field empty. A toast message at the bottom of the screen reads: "Error! Please Enter Valid Username & Password".</p>
CONCLUSION	The expected error messages displayed as a toast message with very clear outcomes and the same error displayed in the above test case has received here also.
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC6.2.2
TEST CASE NAME	Check by filling only password field
TEST SUMMARY	Checking empty field validation by filling only one field
TEST STEPS	<ol style="list-style-type: none"> 3. Fill only password field with a valid customer password 4. Press on Login button
TEST DATA	Username: empty Password: Chathura123
EXPECTED RESULT	Should display the same Error Message which displayed previously as Fields can't be blank
ACTUAL RESULT	
CONCLUSION	The expected error messages displayed as a toast message with very clear outcomes and the same error displayed in the above test case has received here also.
STATUS (PASS/FAIL)	Pass

TC6.3 - Check by wrong username or password

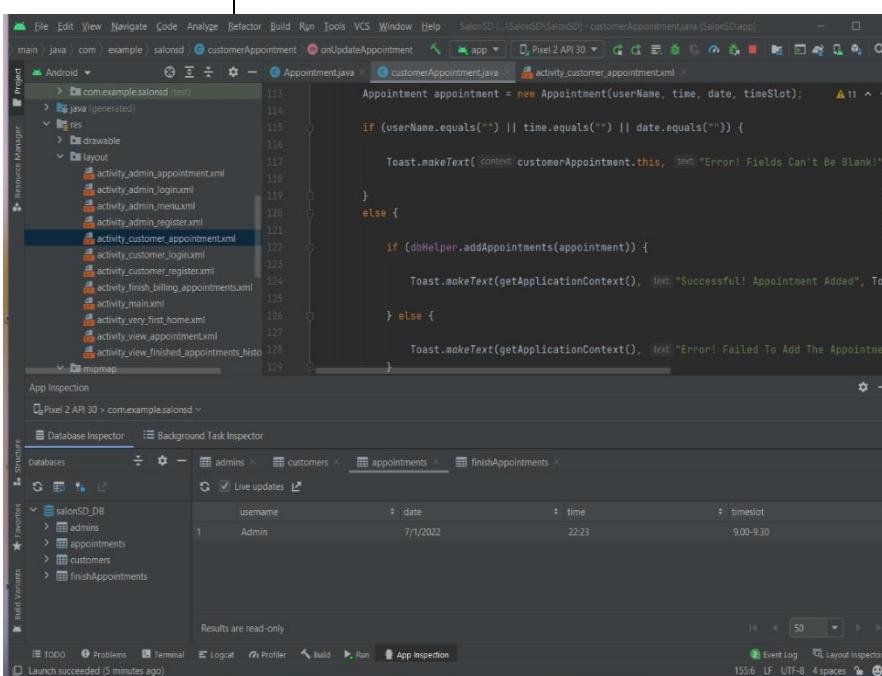
TEST CASE ID	TC6.3.1
TEST CASE NAME	Check with wrong username and correct password
TEST SUMMARY	Checking correctly filled password field validation whereas username is wrong
TEST STEPS	<ol style="list-style-type: none"> 4. Fill only password field with a valid customer password 5. Fill username field with a wrong username 6. Press on Login button
TEST DATA	Username: Mr.Chathu Password: Chathura123
EXPECTED RESULT	Should display an Error Message as Please enter valid username & password
ACTUAL RESULT	
CONCLUSION	The expected error message displayed as a toast message and outcomes are very clear
STATUS (PASS/FAIL)	Pass

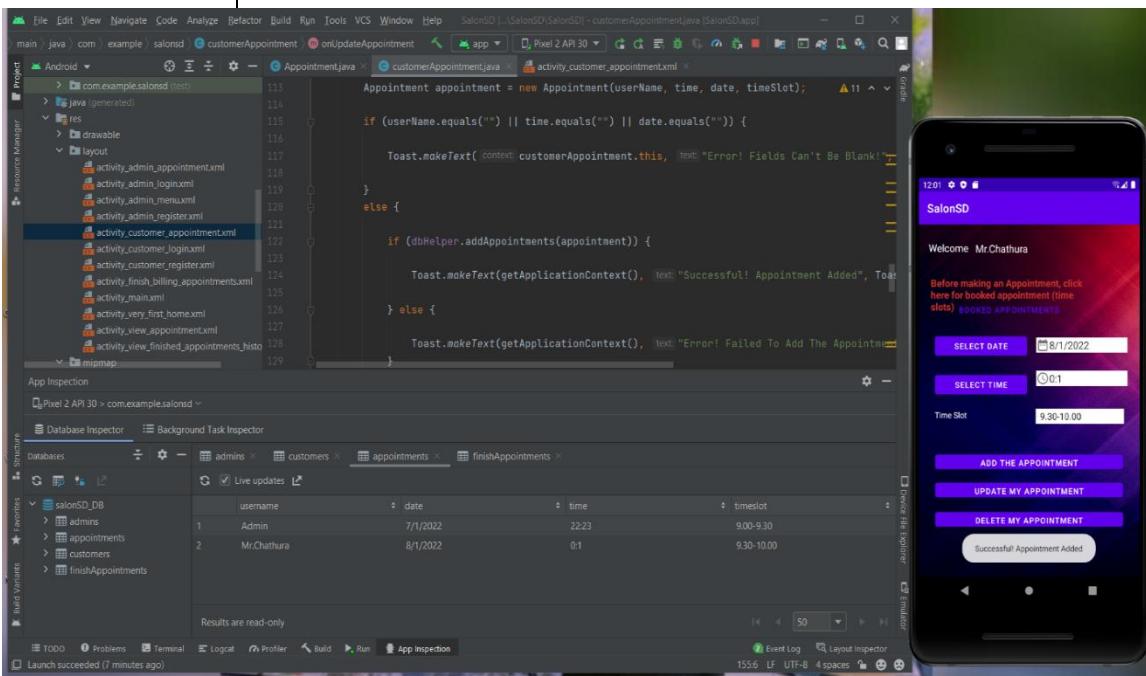
TEST CASE ID	TC6.3.2
TEST CASE NAME	Check with correct username and wrong password
TEST SUMMARY	Checking correctly filled username field validation whereas password is wrong
TEST STEPS	<ol style="list-style-type: none"> 4. Fill only username field with a valid customer username 5. Fill password field with a wrong password 6. Press on Login button
TEST DATA	Username: Mr.Chathura Password: Chathura1
EXPECTED RESULT	Should display an Error Message as Please enter valid username & password
ACTUAL RESULT	
CONCLUSION	The expected error messages displayed as a toast message with very clear outcomes and the same error displayed in the above test case has received here also.
STATUS (PASS/FAIL)	Pass

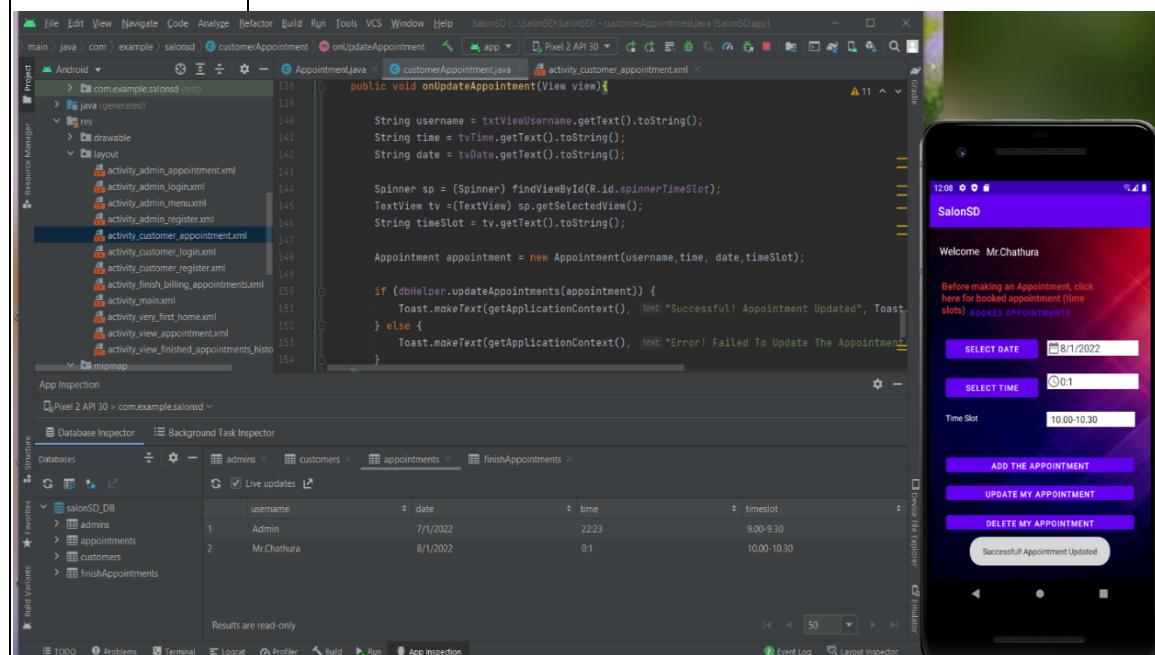
TEST CASE ID	TC1.4															
TEST CASE NAME	Check Login working without any empty field and with saved & correct login credentials															
TEST SUMMARY	Checking whether correctly filled username and password is working or not using registered username and password															
TEST STEPS	<ol style="list-style-type: none"> 4. Fill username field with a valid customer username 5. Fill password field with a valid customer password 6. Press on Login button 															
TEST DATA	<p>Username: Mr.Chathura Password: Chathura123</p>															
EXPECTED RESULT	Customer Appointment interface should be loaded along with a message as Login granted successfully															
ACTUAL RESULT	  <table border="1"> <thead> <tr> <th>name</th> <th>username</th> <th>password</th> <th>tel</th> <th>email</th> </tr> </thead> <tbody> <tr> <td>Roy Lan</td> <td>RoyLan</td> <td>Roy12</td> <td>0777777777</td> <td>Roy@gmail.com</td> </tr> <tr> <td>Chathura</td> <td>Mr.Chathura</td> <td>Chathura123</td> <td>0768973363</td> <td>Chathura@gmail.com</td> </tr> </tbody> </table>	name	username	password	tel	email	Roy Lan	RoyLan	Roy12	0777777777	Roy@gmail.com	Chathura	Mr.Chathura	Chathura123	0768973363	Chathura@gmail.com
name	username	password	tel	email												
Roy Lan	RoyLan	Roy12	0777777777	Roy@gmail.com												
Chathura	Mr.Chathura	Chathura123	0768973363	Chathura@gmail.com												

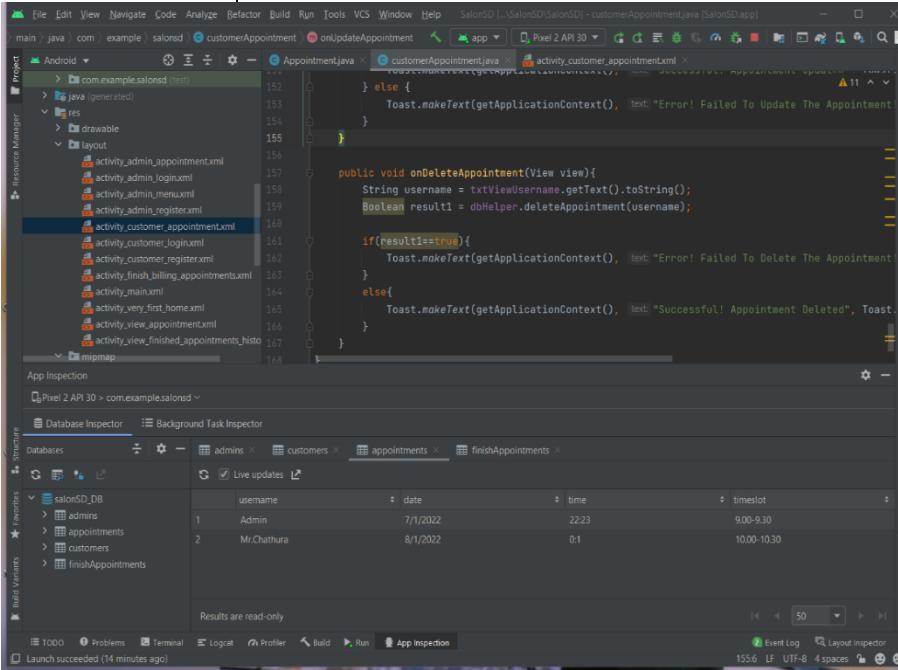
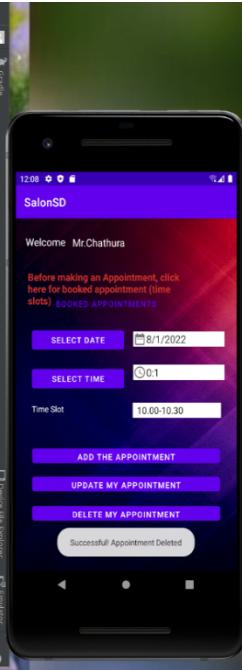
CONCLUSION	The expected successful message displayed as a toast message and outcomes are very clear		
STATUS (PASS/FAIL)	Pass		

TC7 - Customer Appointment

TEST CASE ID	TC7.1
TEST CASE NAME	Check Add Appointment with empty fields
TEST SUMMARY	Checking empty fields validation in Customer Appointment interface
TEST STEPS	Press on Add Appointment button (without filling any field)
TEST DATA	Empty fields
EXPECTED RESULT	Should display an Error Message as Fields can't be blank
ACTUAL RESULT	 
CONCLUSION	The expected error message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC7.2
TEST CASE NAME	Check Add Appointment working without any empty field
TEST SUMMARY	Checking correctly working of Add Appointment in Customer Appointment interface
TEST STEPS	<ol style="list-style-type: none"> 3. Have to fill all fields 4. Press on Add Appointment button
TEST DATA	<p>Date: Current Date Time: Current Time Time Slot: Select 9.30-10.00 Time Slot from the Spinner</p>
EXPECTED RESULT	Should display a message as Appointment added successfully
ACTUAL RESULT	
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC7.3
TEST CASE NAME	Check Update Appointment working without any empty and updated fields
TEST SUMMARY	Checking correctly working of Update Appointment in Customer Appointment interface
TEST STEPS	<ol style="list-style-type: none"> 4. Have to fill all fields 5. Change the fields that are want to be changed 6. Press on Update Appointment button
TEST DATA	<p>Date: Change the date Time: Change the time Time Slot: Change the Time Slot(9.30-10.00) to the Time Slot(10.00-10.30) from the Spinner</p>
EXPECTED RESULT	Should display a message as Appointment updated successfully
ACTUAL RESULT	 A screenshot of the Android Studio interface. On the left, the Project and Database Inspector panes are visible. The Project pane shows the file structure with 'AppointmentJava' selected. The Database Inspector pane shows a table named 'appointments' with two rows of data: Admin at 7/1/2022 22:23 and Mr.Chathura at 8/1/2022 0:1. On the right, there is a screenshot of an Android emulator displaying a booking application. The app has a purple header 'SalonSD' and a welcome message 'Welcome Mr.Chathura'. It asks to click for booked appointment time slots. Below are 'SELECT DATE' (8/1/2022), 'SELECT TIME' (0:1), and 'Time Slot' (10.00-10.30). Buttons for 'ADD THE APPOINTMENT', 'UPDATE MY APPOINTMENT', and 'DELETE MY APPOINTMENT' are present. A success message 'Successful Appointment Updated' is shown.
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

TEST CASE ID	TC7.4
TEST CASE NAME	Check Delete Appointment working
TEST SUMMARY	Checking correctly working of Delete Appointment in Customer Appointment interface
TEST STEPS	<p>3. Have to fill all fields (Not Necessary)</p> <p>4. Press on Delete Appointment button</p>
TEST DATA	Delete button
EXPECTED RESULT	Should display a message as Appointment deleted successfully
ACTUAL RESULT	 
CONCLUSION	The expected successful message was displayed and outcomes are very clear
STATUS (PASS/FAIL)	Pass

References

- Arnicans, G., 1998. *Application generation for the simple*. [Online] Available at: https://www.researchgate.net/profile/Guntis-Arnicans-2/publication/268274019_Application_generation_for_the_simple_database_browser_based_on_the_ER_diagram/links/550469310cf2d60c0e67860d/Application-generation-for-the-simple-database-browser-based-on-the- [Accessed 3 December 2021].
- Bell, D., 2004. *UML basics: The sequence diagram*. [Online] Available at: <http://www.csun.edu/~twang/380/Slides/SequenceDiagram.pdf> [Accessed 10 December 2021].
- Dondeti, S. N. & J., 2012. *BLACK BOX AND WHITE BOX TESTING*. [Online] Available at: https://www.researchgate.net/profile/S-Nidhra/publication/276198111_Black_Box_and_White_Box_Testing_Techniques_-_A_Literature_Review/links/570e313f08ae2b772e46aa40/Black-Box-and-White-Box-Testing-Techniques-A-Literature-Review.pdf [Accessed 15 December 2021].
- JavaTpoint., 2011-2021. *JavaTpoint - UML Class Diagram*. [Online] Available at: <https://www.javatpoint.com/uml-class-diagram> [Accessed 8 December 2021].
- smartdraw, 1994-2022. *Smart Draw - Use Case Diagram*. [Online] Available at: <https://www.smartdraw.com/use-case-diagram/> [Accessed 5 December 2021].

Conclusion

However, I could get many advantages from this course work such as how to solve doubts and issues, how to gather requirements needed, how to do the analyzation properly, how to manage time effectively, how to overcome from errors which are arising when writing the code and so on. Finally...with all above mentioned details and me proper time management, I could prepare a successful document to complete this assignment. Thus, I hope this assignment been a great help for me to get learnt about Mobile Application Development and to prove that I have successfully completed my seventeenth assignment in my HD Program. Moreover, during this Mobile Application Development assessment, I learnt about many more things which are essential in designing and about coding while learning Java programming language more and many more other things. Developing of this mobile application for SD Saloon gave me the chance to try my new skills in practice. Specially this recalled my knowledge for drawing UML diagrams and Java coding. So, I have built the application in a user-friendly manner including admin registration and admin login, customer registration and login, admin menu, make appointment, view appointment, finish appointments, etc. While doing this project I also gained deep understanding on programming/coding and how it can be implemented in real life situations as now I have the experience in creating a Mobile Application for SD Salon. Thus, I believe that this was a great chance for me to improve my mobile application developing and computer programming skills.