TOURIST ASSIST APPLICATION

CS487: Software Engineering

Project Done by;

Name	CWID
Fatema Alteneiji	A20362529
Moosa Hana	A20298732
Saurabh Tiwari	A20356308
Ernesto Garcia	A20364089
Chethan Appaji	A20355296

Table of contents

Application final Design

The application was designed with a certain focus of fulfilling the main requirements, it offers a simple design that the user can easily interact with. The testing plans have been executed in order to make sure that the application can run normally without facing any exceptions, and if any were faced they were caught and handled correctly.

The application UI-Design

Team Performance

The team members have worked and collaborated on achieving this project's requirement and goals. Each member was accountable for their work, and they were able to correct and test the parts they were working on. The work was divided on the members according to their experience, and the available amount of time, each member had completed the work assigned to them in the assigned time slot, and with that the team was able to complete the project in the given time.

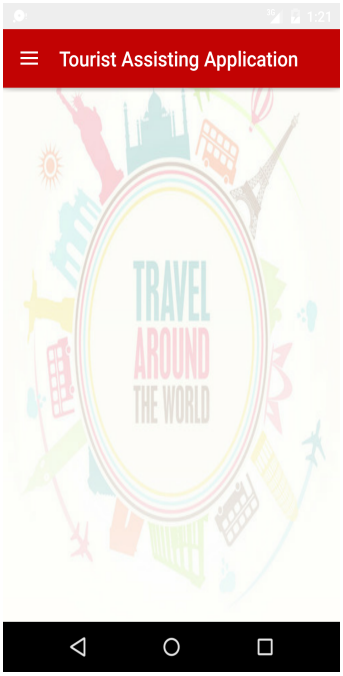
Testing

Main Screen

Code: MainScreenTest00

Case Description: The city name is entered and the submit button is clicked.

Expected Result: A new view shows the navigation screen.



Code: MainScreenTest01

Case Description: no city or wrong city name entered and then the submit button was clicked

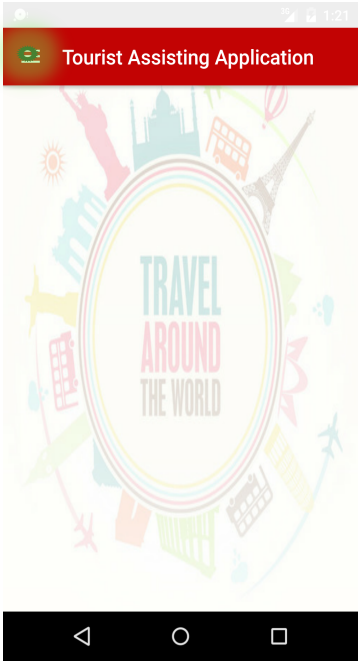
Expected Result: An error message is displayed and the user will have to re-enter the city name

Test Code	Passed	Failed
MainScreenTest00		
MainScreenTest01		

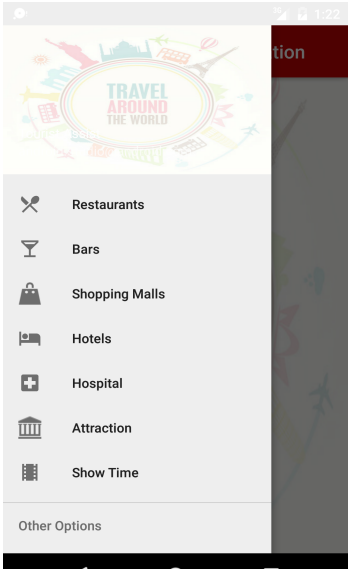
Navigation Screen

Code: NavigationScreenTEst01

Case Description: user clickson the left side of the upper toolbar.



Expected Result: a navigation bar should slide from the left side

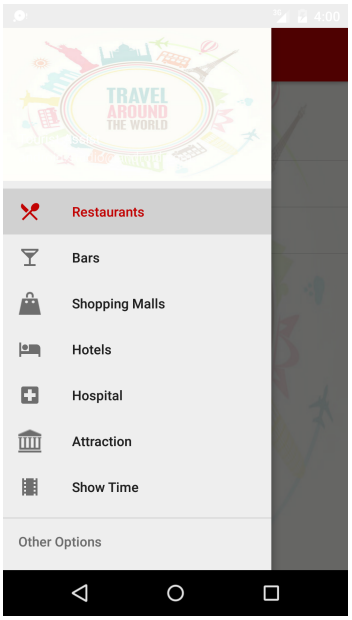


Test Code	Passed	Failed
NavigationScreenTest00		

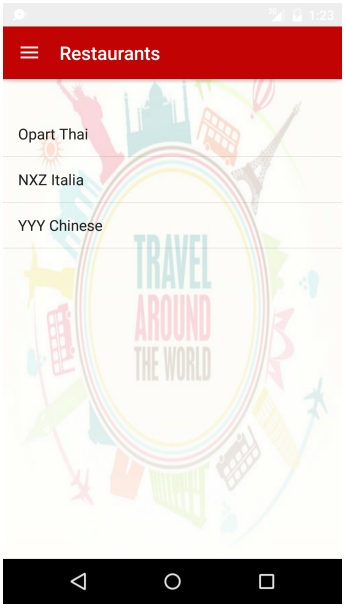
Restaurants

Code: RestauratTest00

Case Description: The Restaurant button is clicked on the menu.

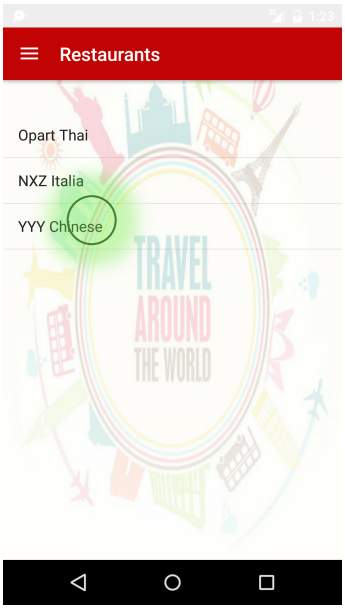


Expected Result: A new view shows the list of restaurants.



Code: RestaurantTest01

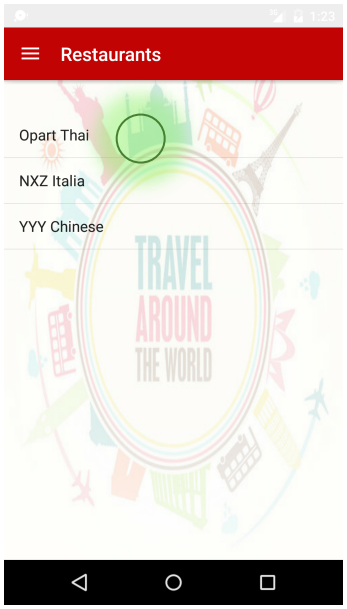
Case Description: One of the rows is clicked in the list of Restaurants. This Restaurant has no reviews.



Expected Result: A new view shows the details of the associated restaurant. The table of reviews is hidden.

Code: RestaurantTest02

Case Description: One of the rows is clicked in the list of restaurants. This restaurant has reviews.

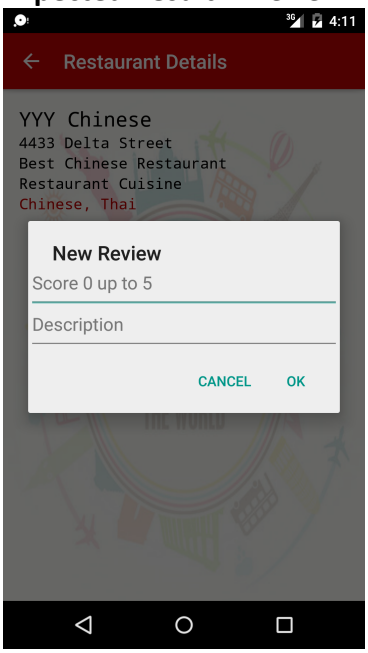


Expected Result: A new view shows the details of the associated restaurant. The table of reviews is included.

Code: RestaurantTest03

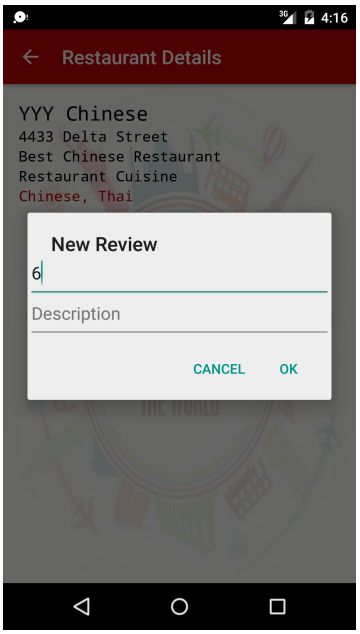
Case Description: Add review button is clicked.

Expected Result: A review input dialog is shown.

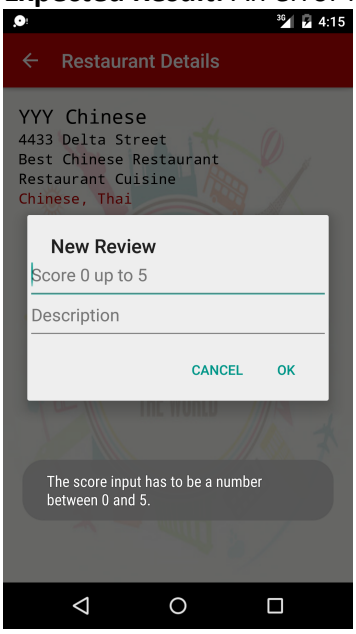


Code: RestaurantTest04

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

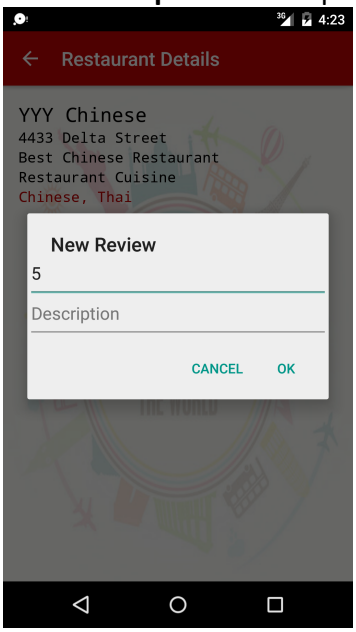


Expected Result: An error message is displayed and the dialog is re-launched.

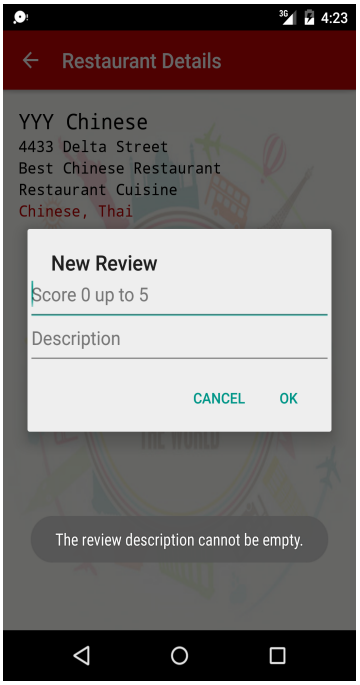


Code: RestaurantTest05

Case Description: Description input box is empty and the user clicks on OK.

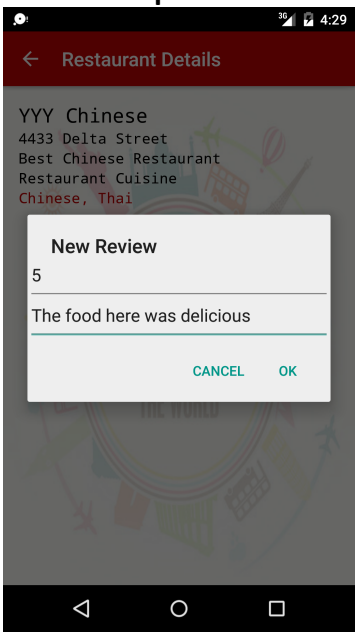


Expected Result: An error message is displayed.



Code: RestaurantTest06

Case Description: Both fields are correctly filled and the user clicks on OK.

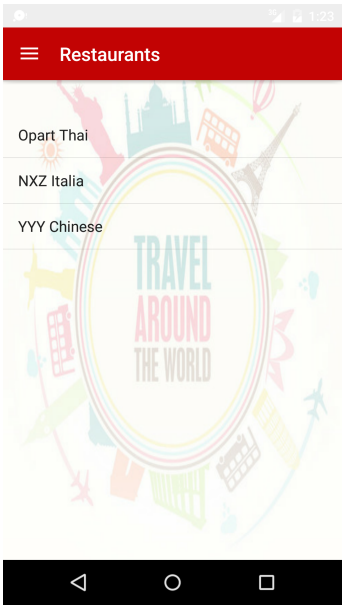


Expected Result: The new review is posted.

Code: RestaurantTest07

Case Description: The user clicks on back button.

Expected Result: The Restaurants list is shown.

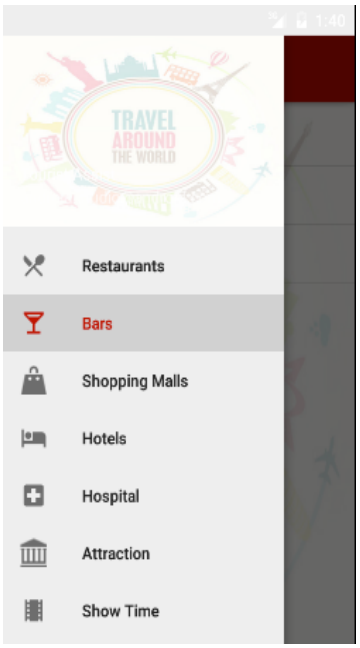


Test Code	Passed	Failed
RestaurantTest00		
RestaurantTest01		
RestaurantTest02		
RestaurantTest03		
RestaurantTest04		
RestaurantTest05		
RestaurantTest06		
RestaurantTest07		

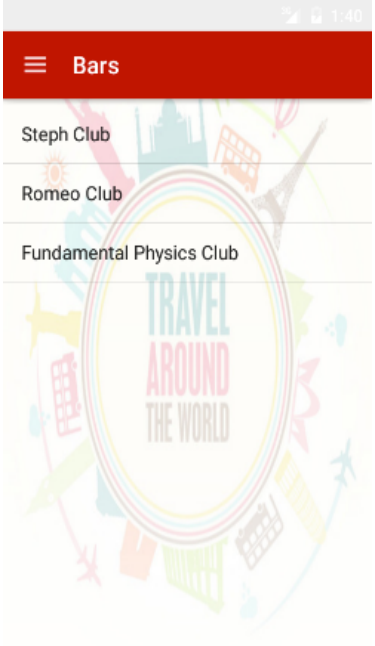
Bars

Code: BarTest00

Case Description: The Bars button is clicked on the menu.

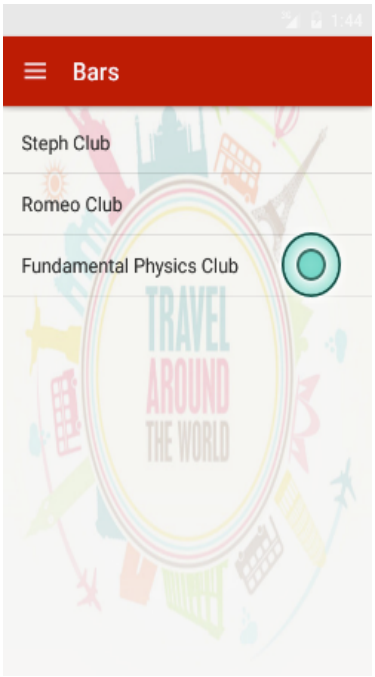


Expected Result: A new view shows the list of bars.

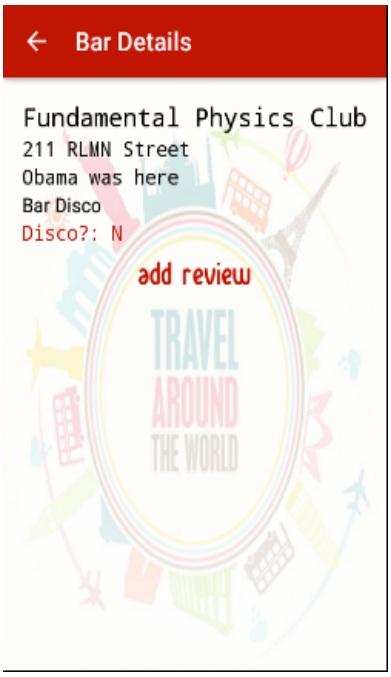


Code: BarTest01

Case Description: One of the rows is clicked in the list of bars. This bar has no reviews.

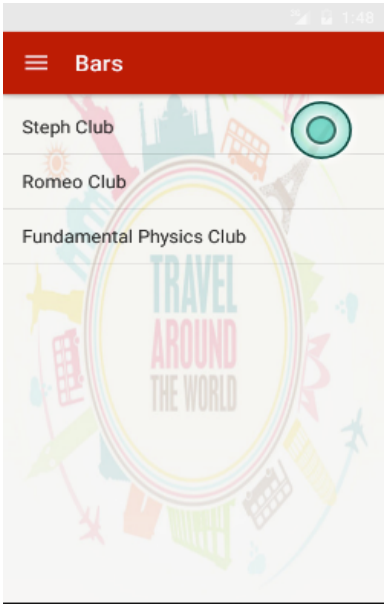


Expected Result: A new view shows the details of the associated bar. The table of reviews is hidden.



Code: BarTest02

Case Description: One of the rows is clicked in the list of bars. This bar has reviews.

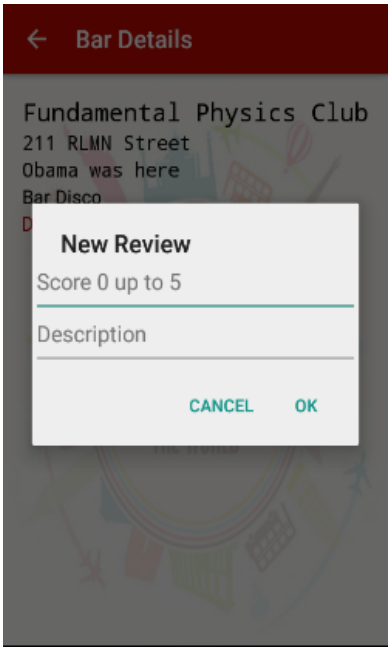


Expected Result: A new view shows the details of the associated bar. The table of reviews is included.

Code: BarTest03

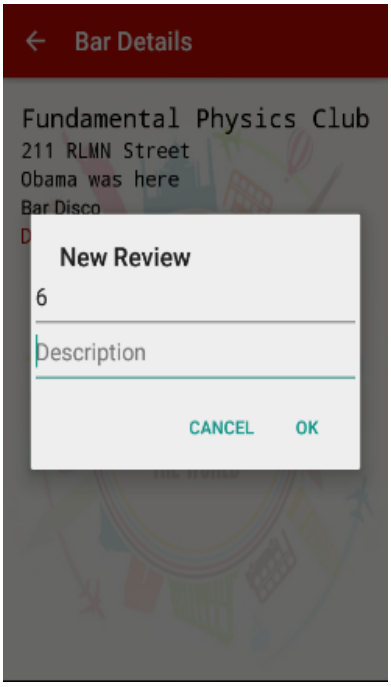
Case Description: Add review button is clicked.

Expected Result: A review input dialog is shown.

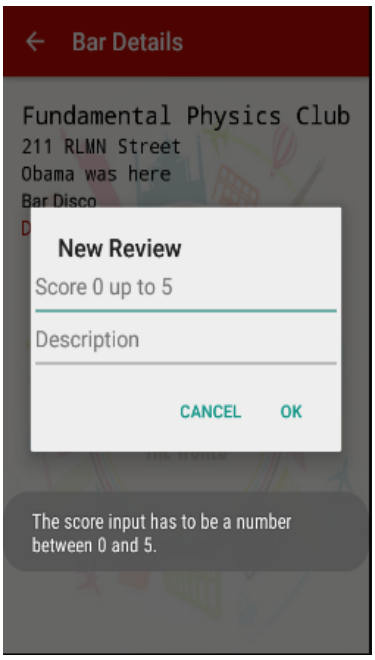


Code: BarTest04

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

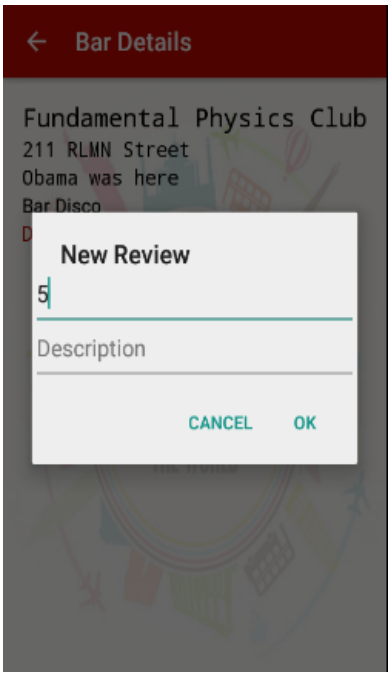


Expected Result: An error message is displayed and the dialog is re-launched.

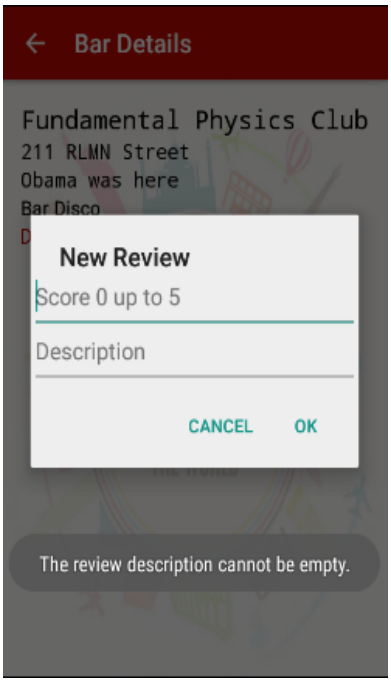


Code: BarTest05

Case Description: Description input box is empty and the user clicks on OK.

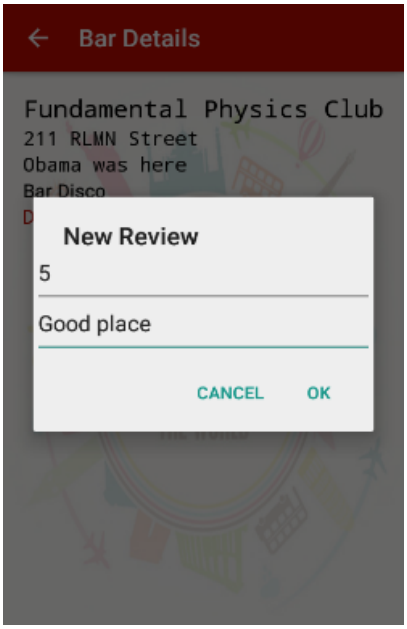


Expected Result: An error message is displayed.



Code: BarTest06

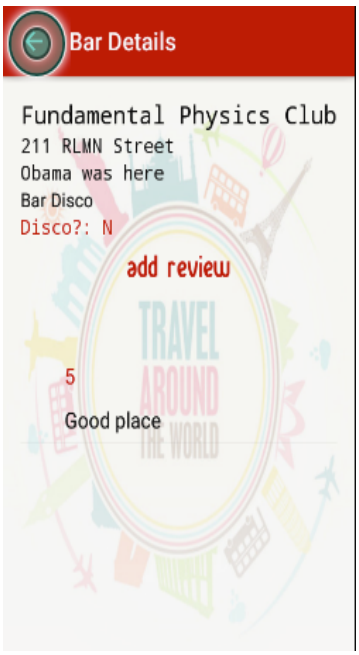
Case Description: Both fields are correctly filled and the user clicks on OK.



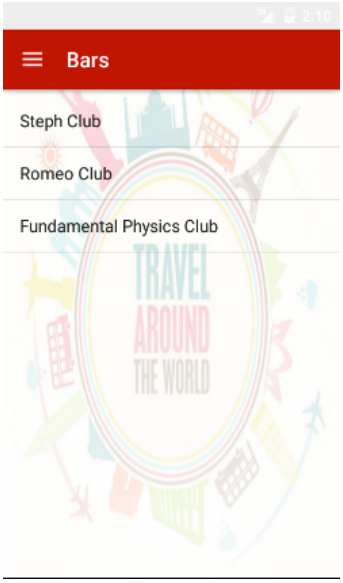
Expected Result: The new review is posted.

Code: BarTest07

Case Description: The user clicks on back button.



Expected Result: The bars list is shown.

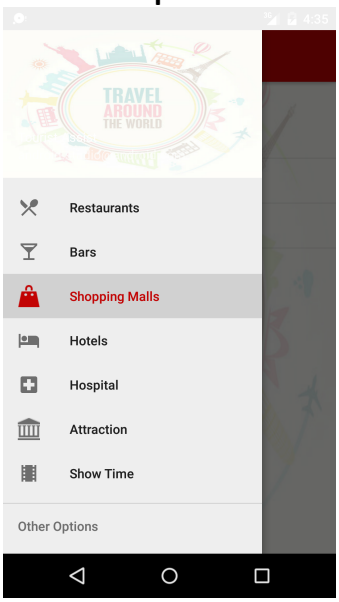


Test Code	Passed	Failed
BarTest00		
BarTest01		
BarTest02		
BarTest03		
BarTest04		
BarTest05		
BarTest06		
BarTest07		

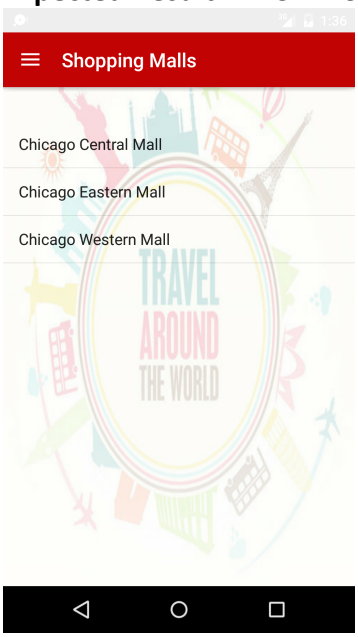
Shopping Malls

Code: MallTest00

Case Description: The Mall button is clicked on the menu.

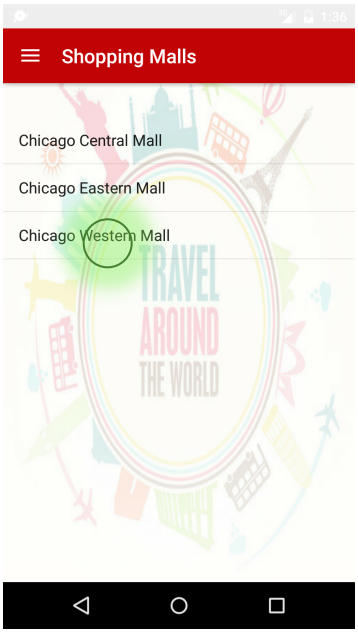


Expected Result: A new view shows the list of malls.



Code: MallTest01

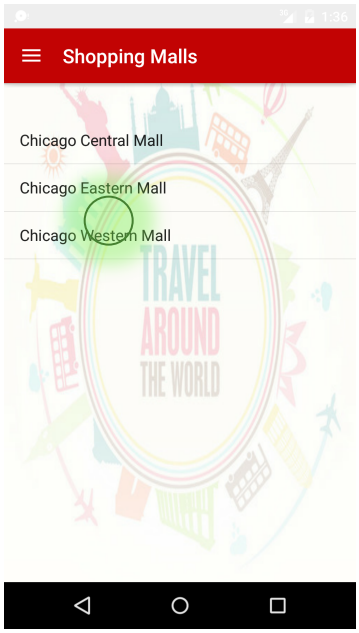
Case Description: One of the rows is clicked in the list of Restaurants. This mall has no reviews.



Expected Result: A new view shows the details of the associated mall. The table of reviews is hidden.

Code: MallTest02

Case Description: One of the rows is clicked in the list of malls. This mall has reviews.

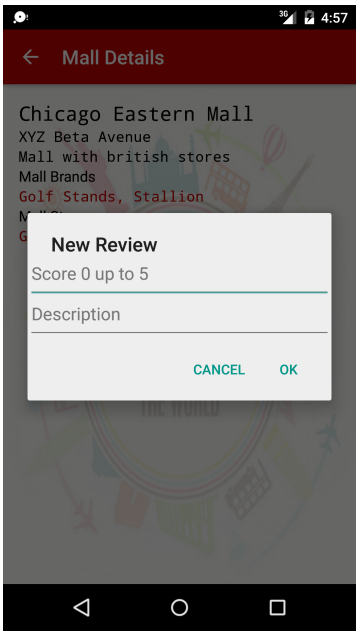


Expected Result: A new view shows the details of the associated mall. The table of reviews is included.

Code: MallTest03

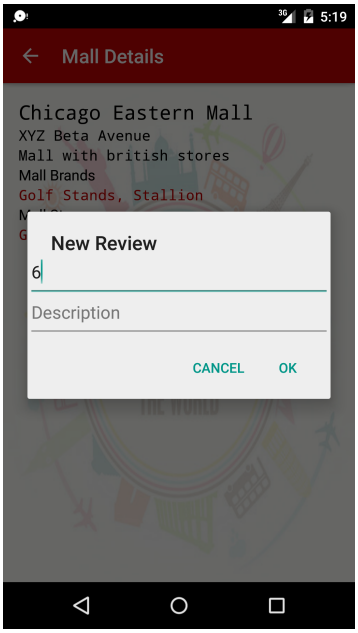
Case Description: Add review button is clicked.

Expected Result: A review input dialog is shown.

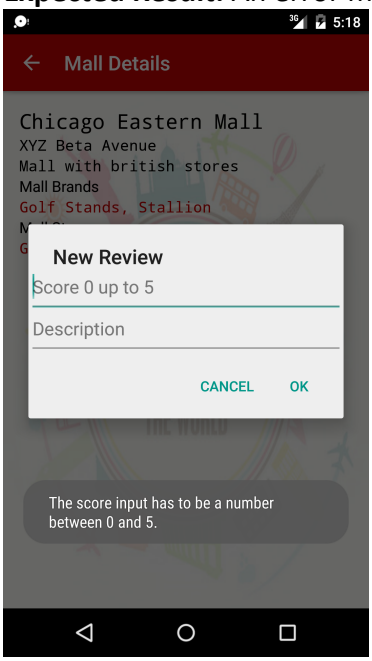


Code: MallTest04

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

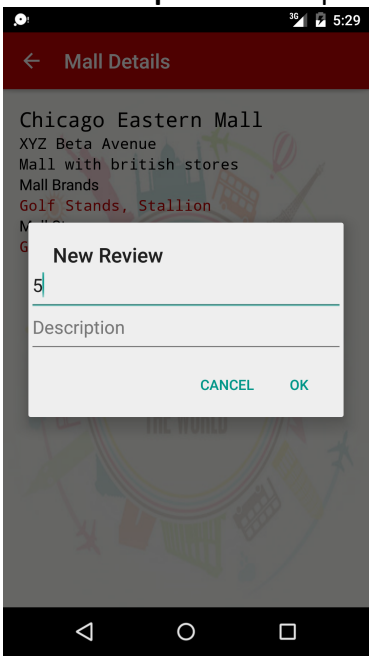


Expected Result: An error message is displayed and the dialog is re-launched.

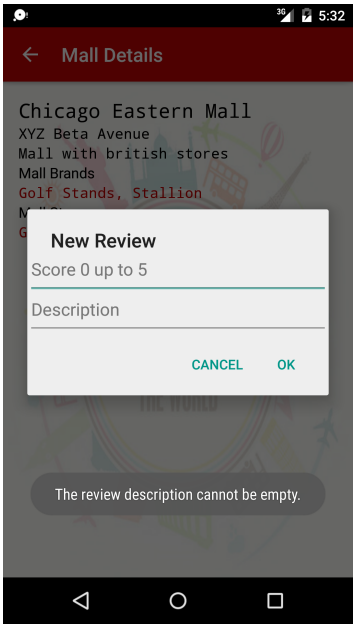


Code: MallTest05

Case Description: Description input box is empty and the user clicks on OK.

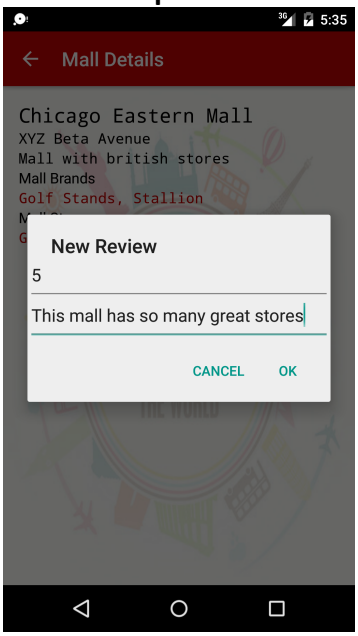


Expected Result: An error message is displayed.

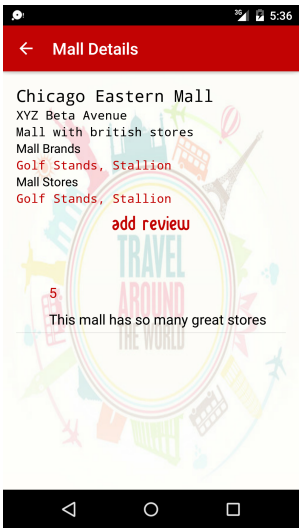


Code: MallTest06

Case Description: Both fields are correctly filled and the user clicks on OK.



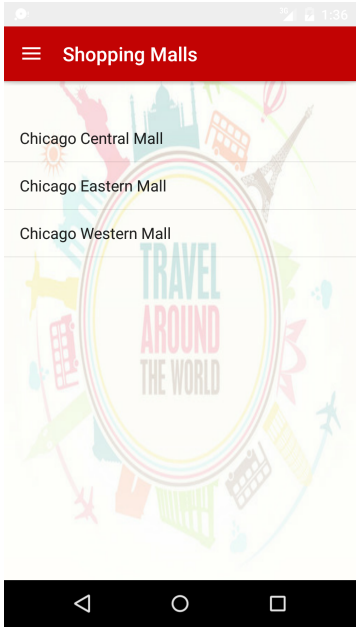
Expected Result: The new review is posted.



Code: MallTest07

Case Description: The user clicks on back button.

Expected Result: The Malls list is shown.

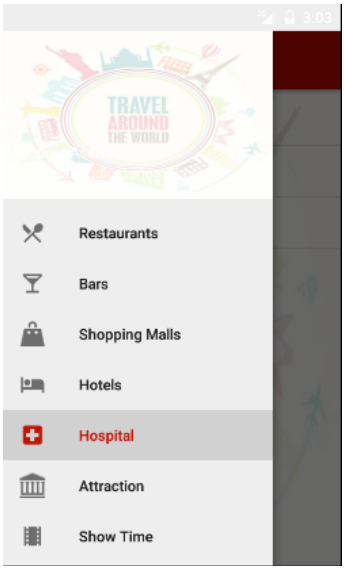


Test Code	Passed	Failed
MallTest00		
MallTest01		
MallTest02		
MallTest03		
MallTest04		
MallTest05		
MallTest06		
MallTest07		

Hospitals

Code: HospTest00

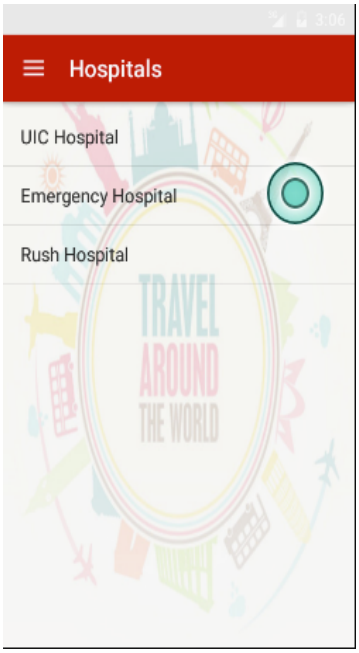
Case Description: The Hospitals button is clicked on the menu.



Expected Result: A new view shows the list of hospitals.

Code: HospTest01

Case Description: One of the rows is clicked in the list of hospitals. This hospital has no reviews.



Expected Result: A new view shows the details of the associated hospital. The table of reviews is hidden.

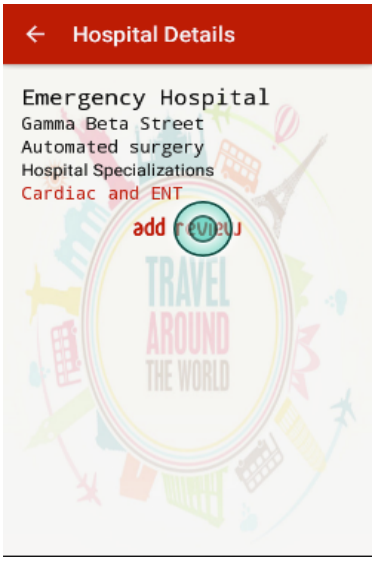
Code: HospTest02

Case Description: One of the rows is clicked in the list of hospitals. This hospital has reviews.

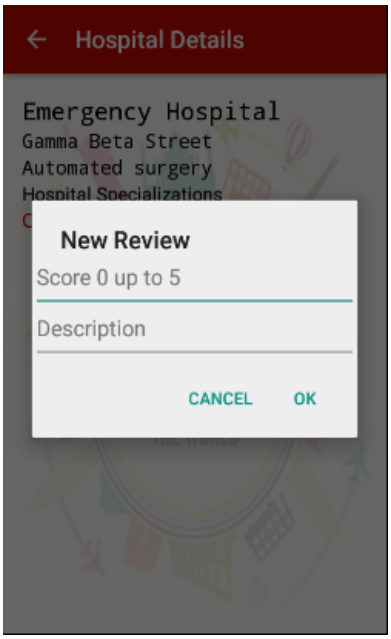
Expected Result: A new view shows the details of the associated hospital. The table of reviews is included.

Code: HospTest03

Case Description: Add review button is clicked.

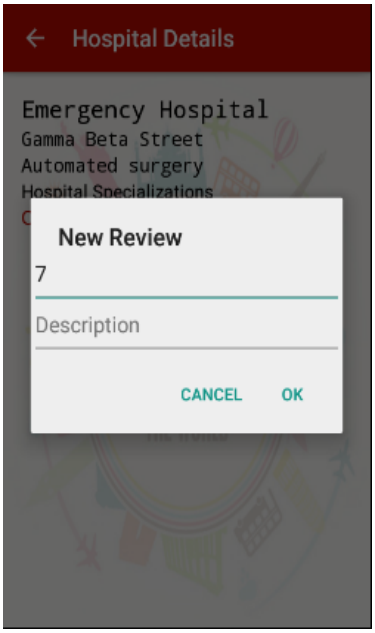


Expected Result: A review input dialog is shown.

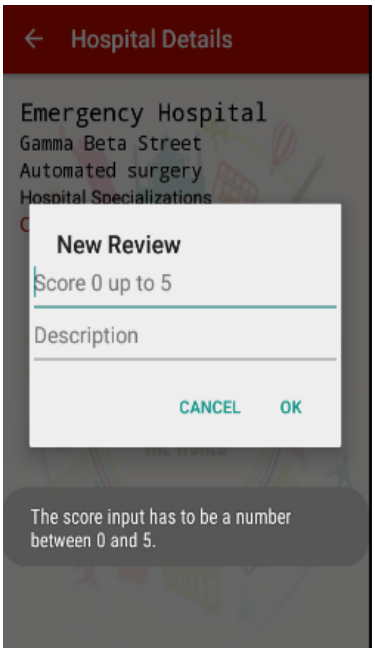


Code: HospTest04

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

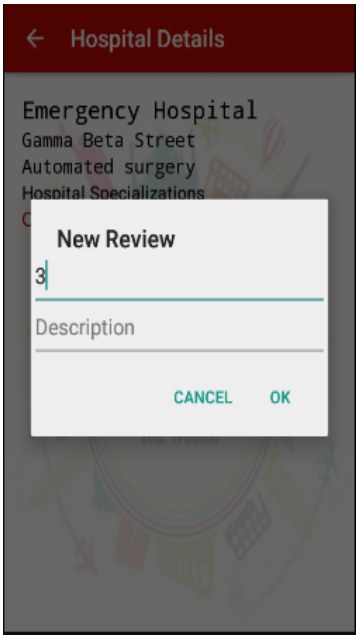


Expected Result: An error message is displayed and the dialog is re-launched.

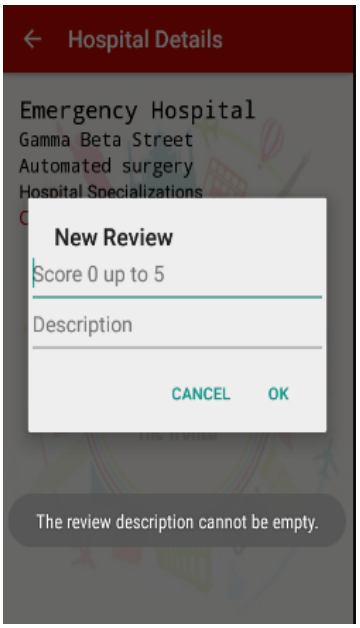


Code: HospTest05

Case Description: Description input box is empty and the user clicks on OK.

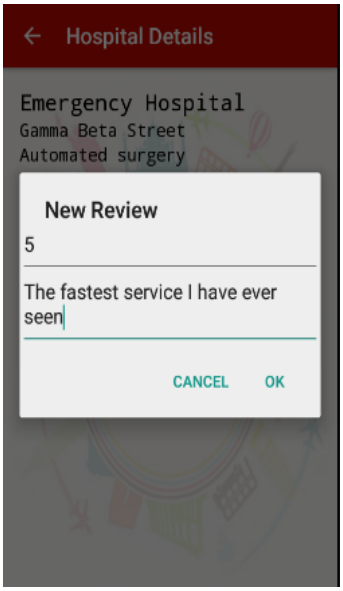


Expected Result: An error message is displayed.

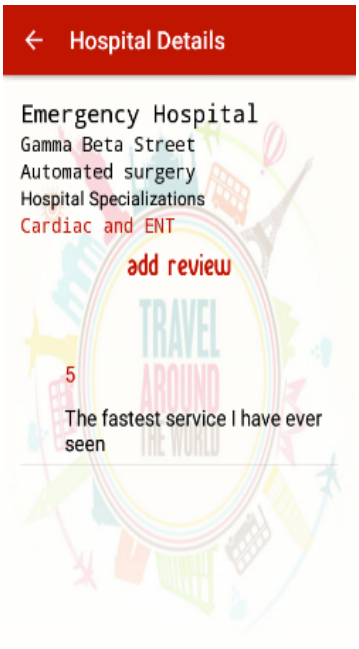


Code: HospTest06

Case Description: Both fields are correctly filled and the user clicks on OK.

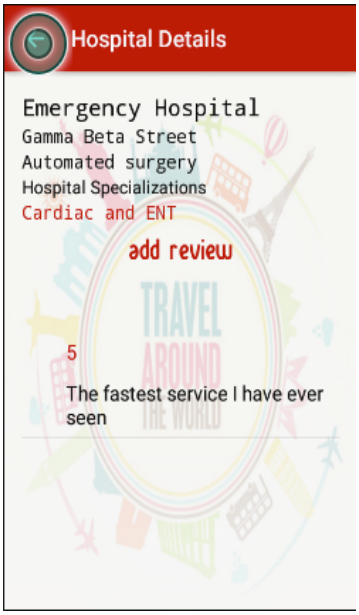


Expected Result: The new review is posted.



Code: HospTest07

Case Description: The user clicks on back button.



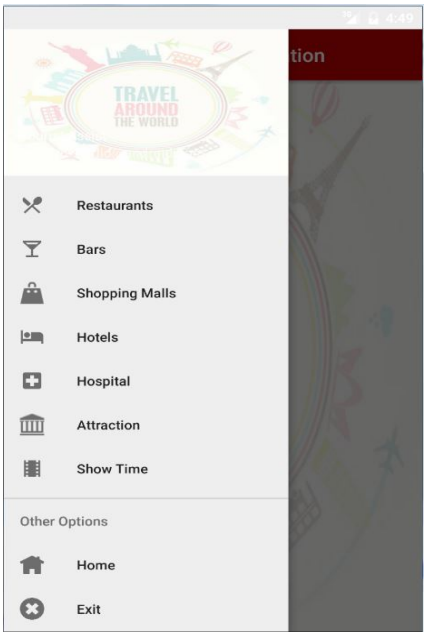
Expected Result: The hospitals list is shown.

Test Code	Passed	Failed
HospTest00		
HospTest01		
HospTest02		
HospTest03		
HospTest04		
HospTest05		
HospTest06		
HospTest07		

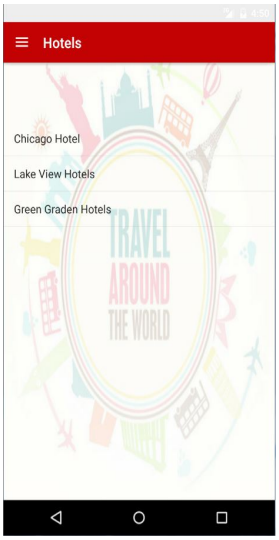
Hotels

Code: HotelTest00

Case Description: The Hotels button is clicked on the menu.

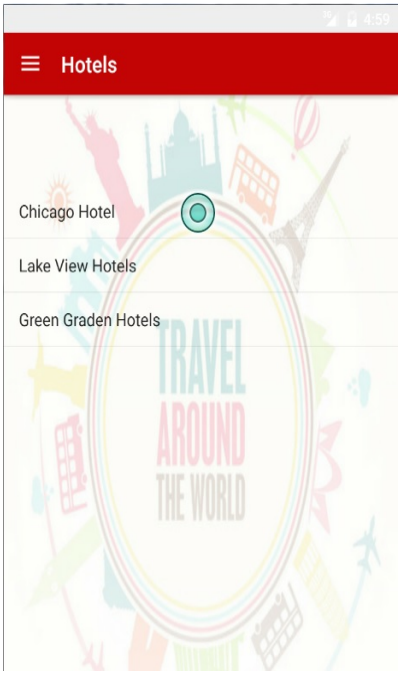


Expected Result: A new view shows the list of Hotels.



Code: HotelTest01

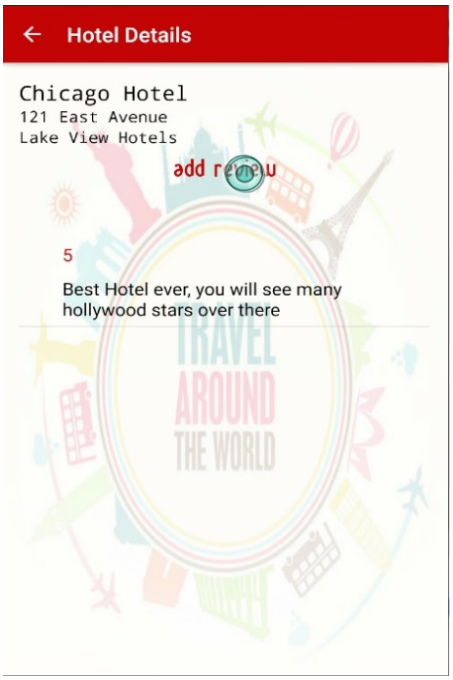
Case Description: One of the rows is clicked in the list of Hotels. This Hotel has reviews.



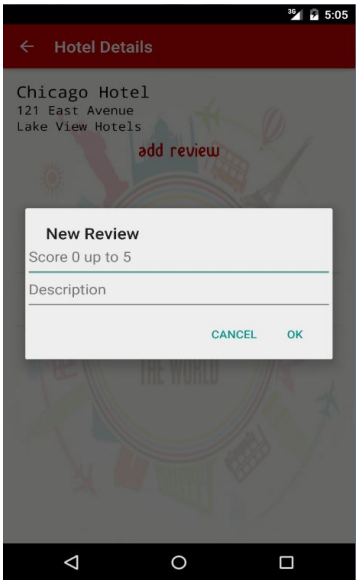
Expected Result: A new view shows the details of the associated Hotel. The table of reviews is included.

Code: HotelTest02

Case Description: Add review button is clicked.

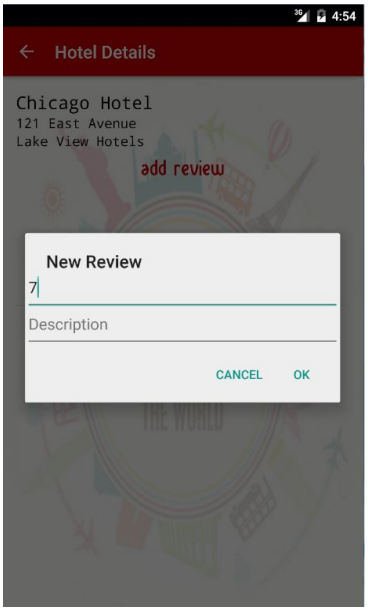


Expected Result: A review input dialog is shown.

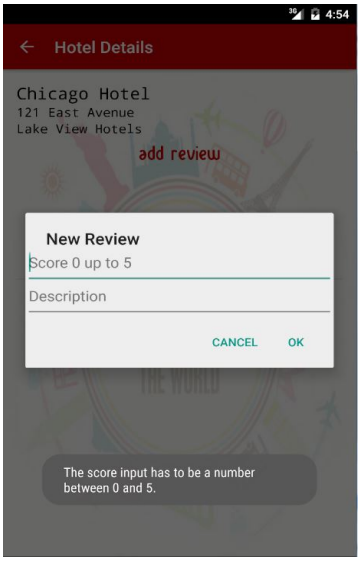


Code: HotelTest03

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

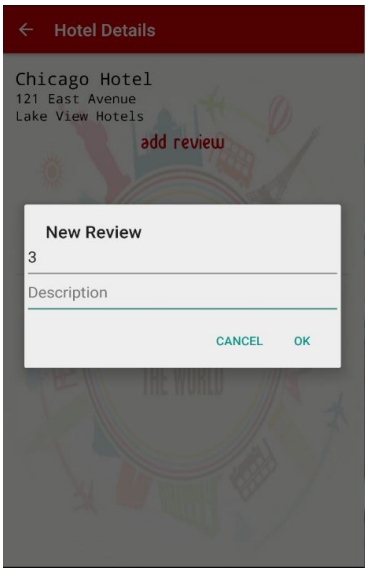


Expected Result: A toast message with error is displayed and the dialog is re-launched.

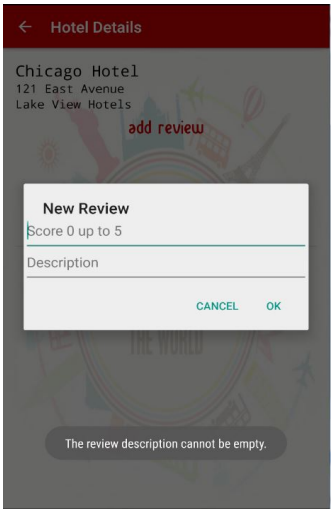


Code: HotelTest04

Case Description: The score is in the interval but the description input box is empty and the user clicks on OK.

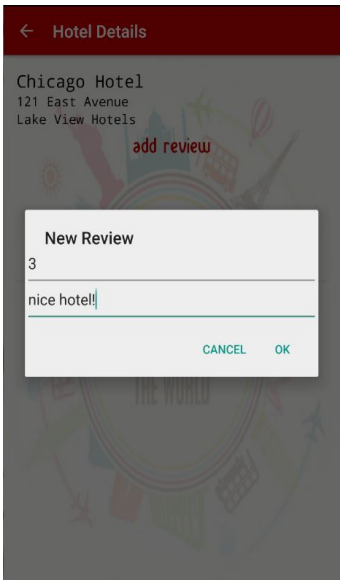


Expected Result: A toast message with error is displayed.



Code: HotelTest05

Case Description: Both fields are correctly filled and the user clicks on OK.

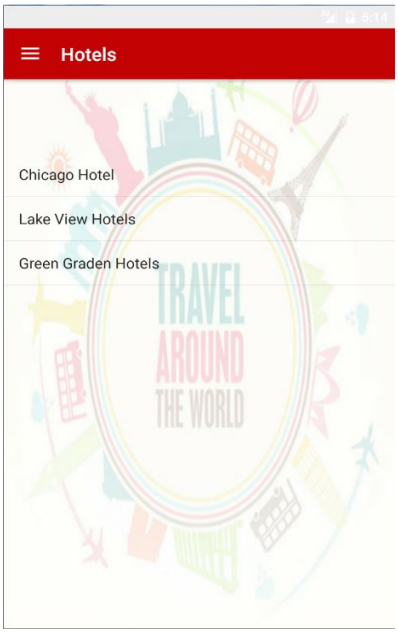


Expected Result: The new review is posted.

Code: HotelTest06

Case Description: The user clicks on back button.

Expected Result: The Hotels list is shown.

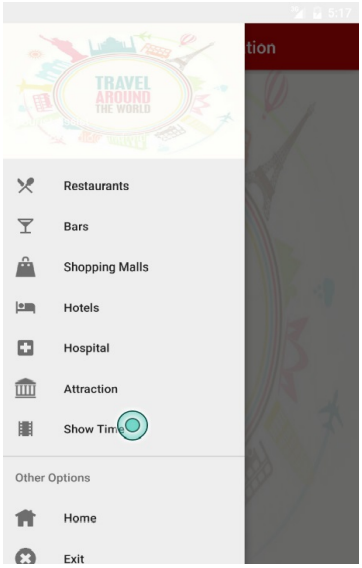


Test Code	Passed	Failed
HotelTest00		
HotelTest01		
HotelTest02		
HotelTest03		
HotelTest04		
HotelTest05		
HotelTest06		

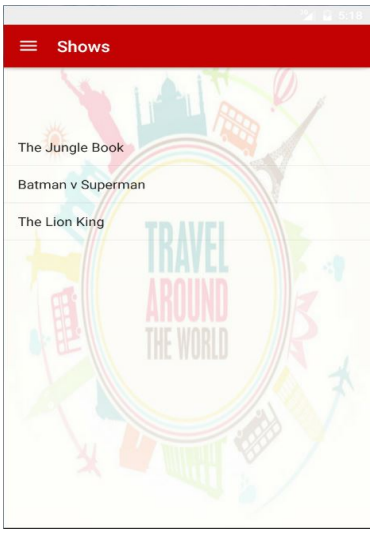
Show times

Code: ShowTest00

Case Description: The Showtimes button is clicked on the menu.



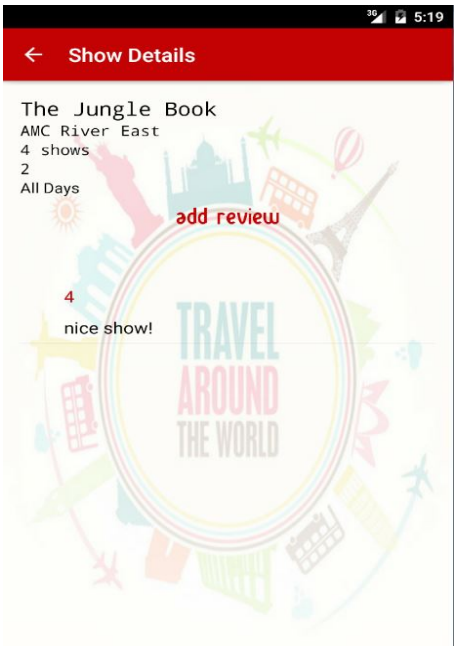
Expected Result: A new view shows the list of Showtimes.



Code: ShowTest01

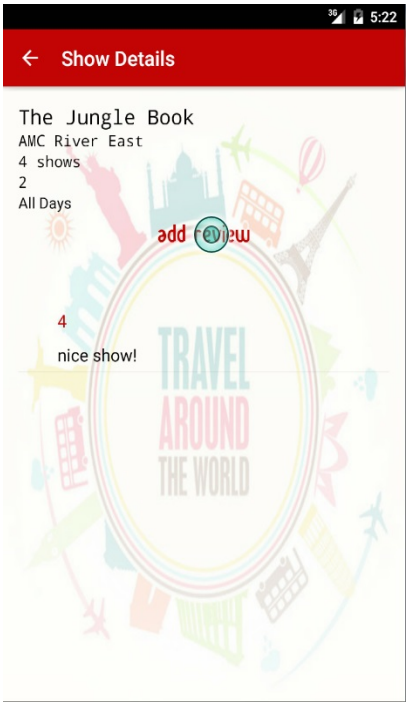
Case Description: One of the rows is clicked in the list of Showtimes. This Showtime has reviews.

Expected Result: A new view shows the details of the associated Showtime. The table of reviews is included.

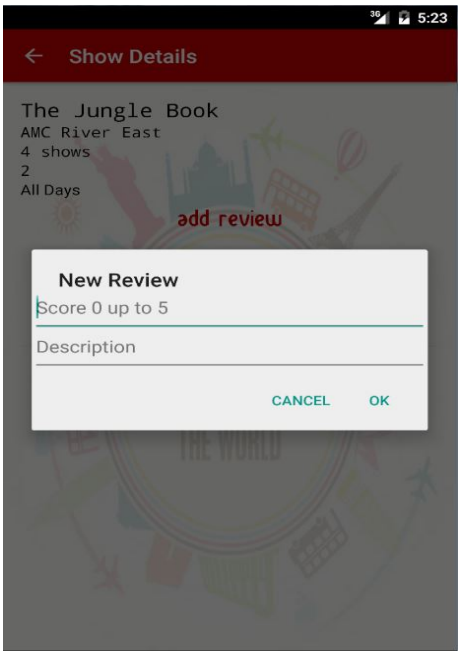


Code: ShowTest02

Case Description: Add review button is clicked.

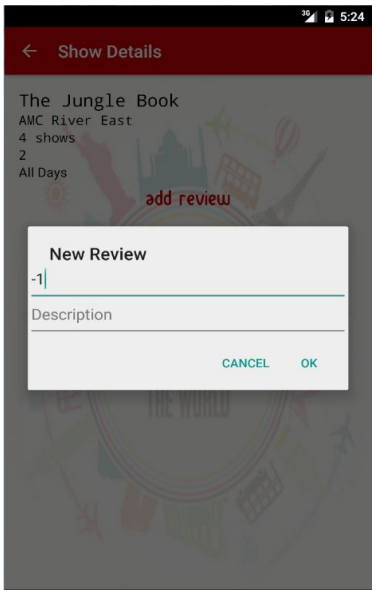


Expected Result: A review input dialog is shown.

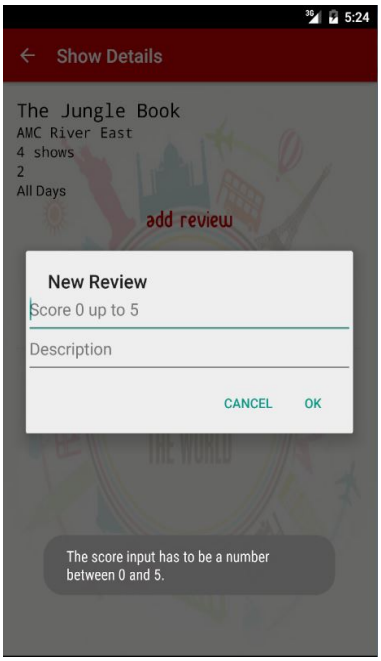


Code: ShowTest03

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

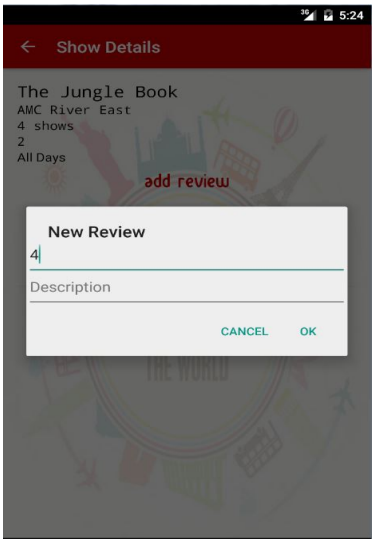


Expected Result: An error message is displayed and the dialog is re-launched.

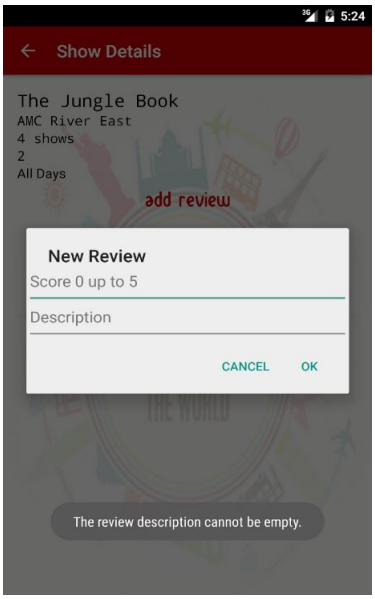


Code: ShowTest04

Case Description: Description input box is empty and the user clicks on OK.

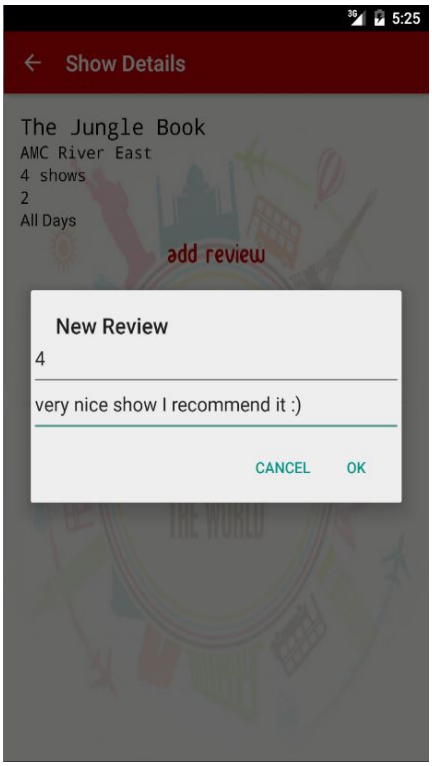


Expected Result: An error message is displayed.



Code: ShowTest05

Case Description: Both fields are correctly filled and the user clicks on OK.



Expected Result: The new review is posted.

Code: ShowTest06

Case Description: The user clicks on back button.

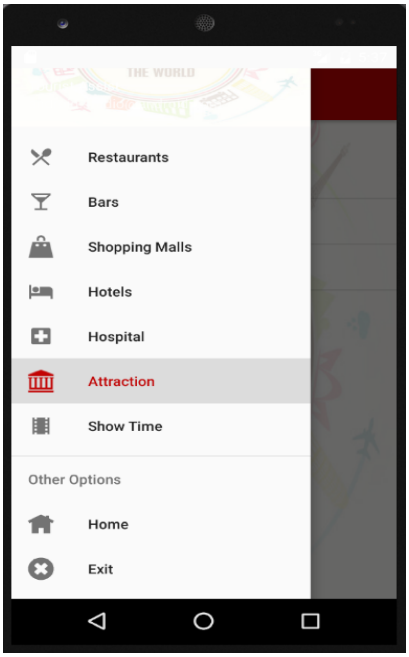
Expected Result: The Showtimes list is shown again.

Test Code	Passed	Failed
ShowTest00		
ShowTest01		
ShowTest02		
ShowTest03		
ShowTest04		
ShowTest05		
ShowTest06		

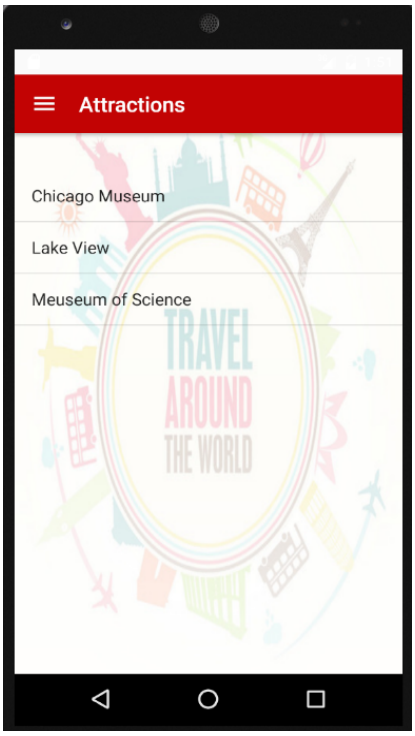
Attractions

Code: AttractionTest00

Case Description: The Attraction button is clicked on the menu.

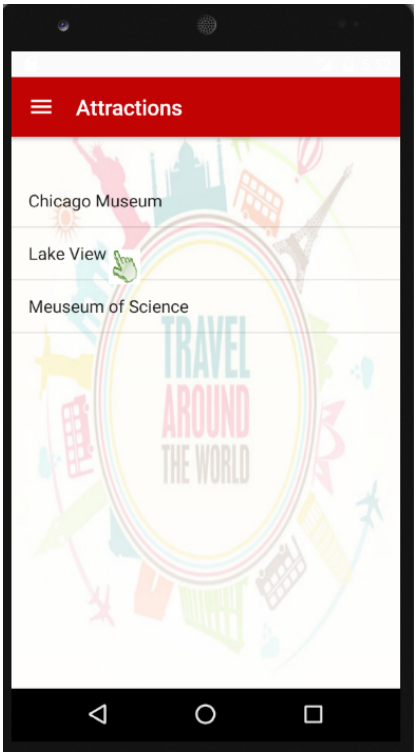


Expected Result: A new view shows the list of attractions.

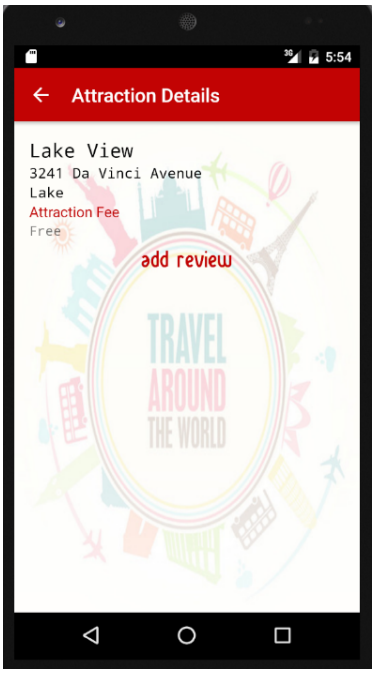


Code: AttractionTest01

Case Description: One of the rows is clicked in the list of attractions. This attraction has no reviews.

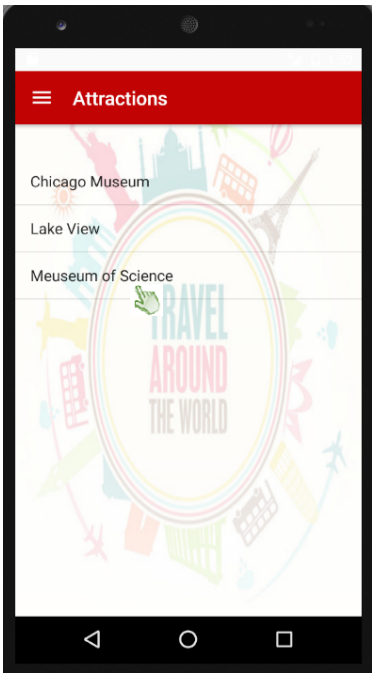


Expected Result: A new view shows the details of the associated attraction. The table of reviews is hidden.



Code: AttractionTest02

Case Description: One of the rows is clicked in the list of attractions. This attraction has reviews.

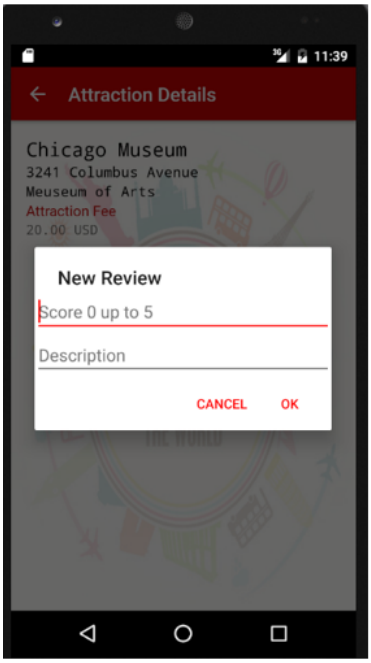


Expected Result: A new view shows the details of the associated attraction. The table of reviews is included.

Code: AttractionTest03

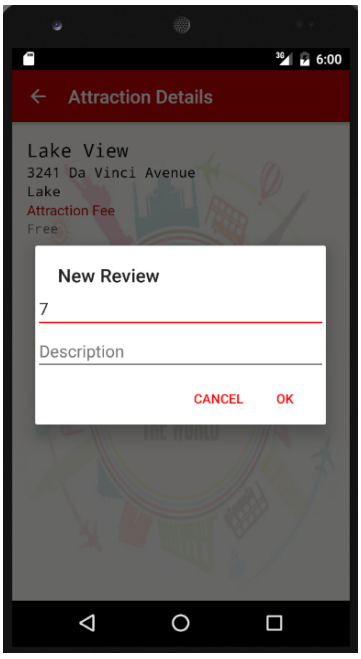
Case Description: Add review button is clicked.

Expected Result: A review input dialog is shown.

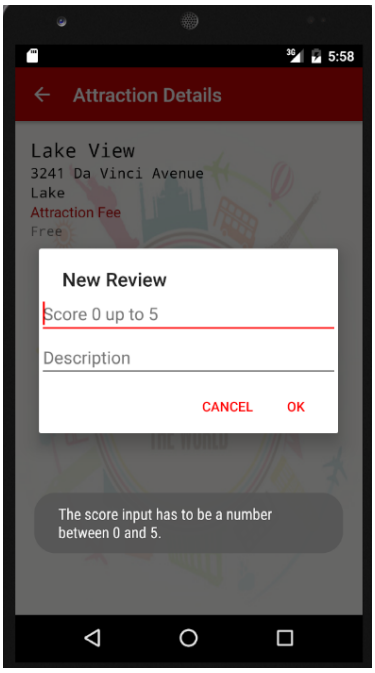


Code: AttractionTest04

Case Description: the score introduced by the user is out of the supported interval and the user clicks on OK.

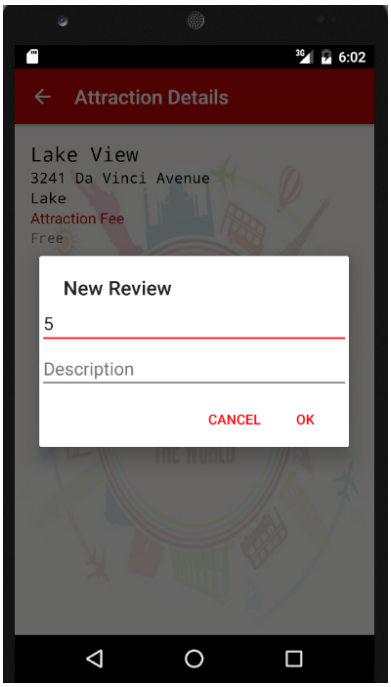


Expected Result: An error message is displayed and the dialog is re-launched.

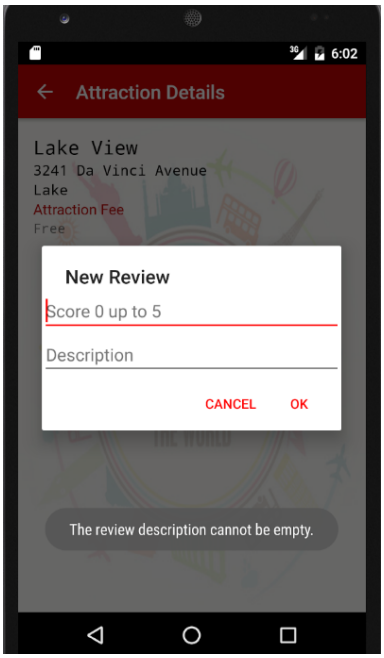


Code: AttractionTest05

Case Description: Description input box is empty and the user clicks on OK.

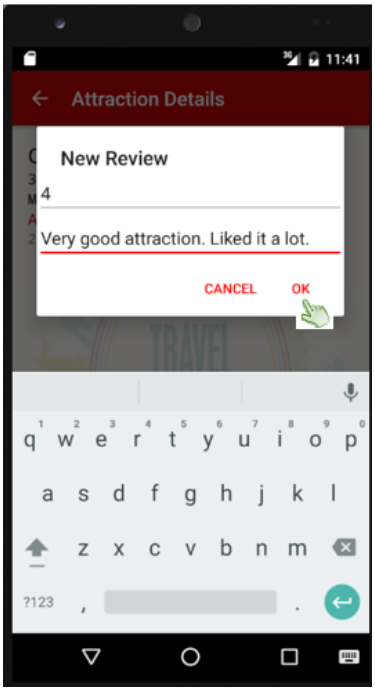


Expected Result: An error message is displayed.

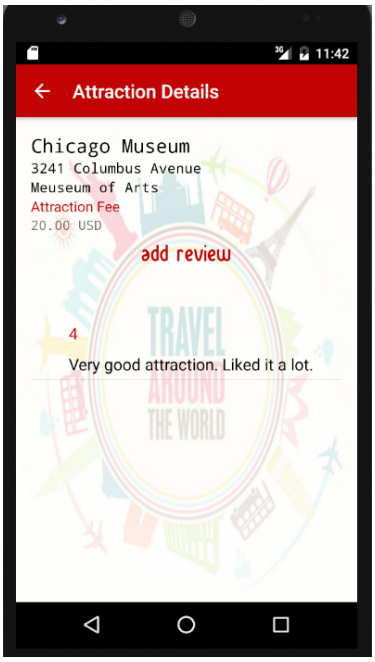


Code: AttractionTest06

Case Description: Both fields are correctly filled and the user clicks on OK.

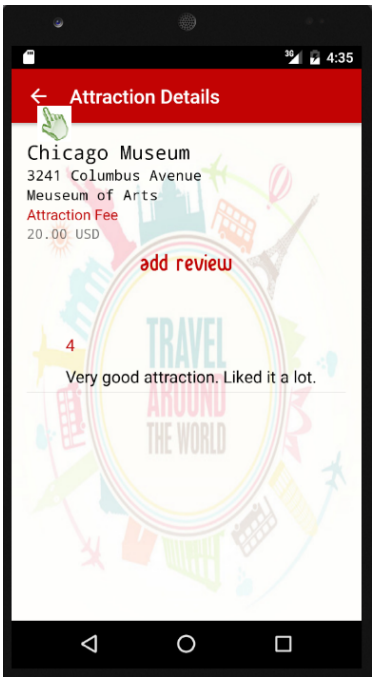


Expected Result: The new review is posted.

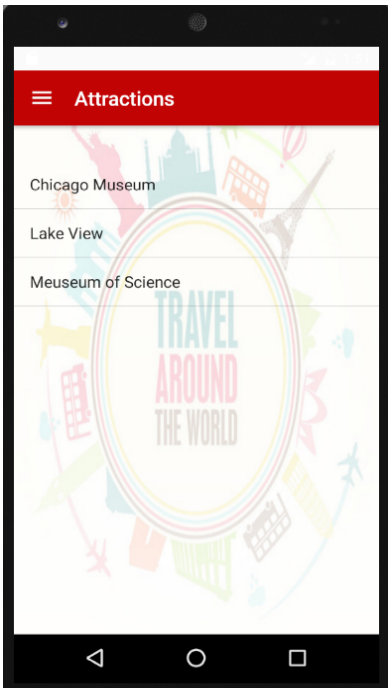


Code: AttractionTest07

Case Description: The user clicks on back button.



Expected Result: The attractions list is shown.

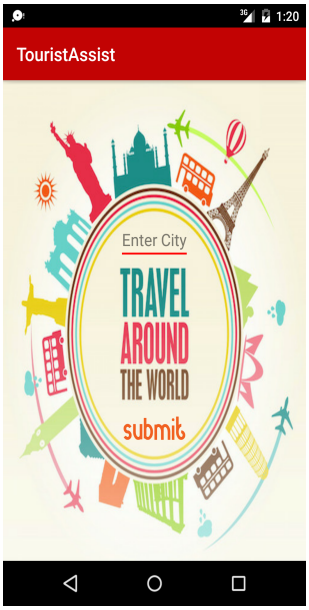


Test Code	Passed	Failed
AttractionTest00		
AttractionTest01		
AttractionTest02		
AttractionTest03		
AttractionTest04		
AttractionTest05		
AttractionTest06		
AttractionTest07		

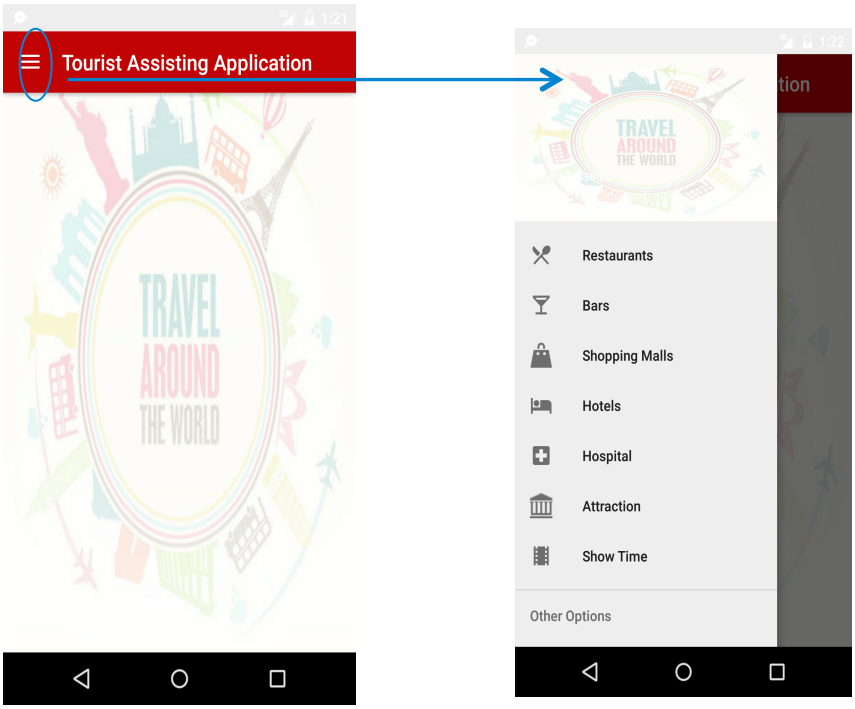
User Guide

Main Screen

When the Application is first opened The users will have to enter the name of the they are for information about and then click on submit



The user would then be taken to a screen when he could click on the left side of the toolbar and navigate through places



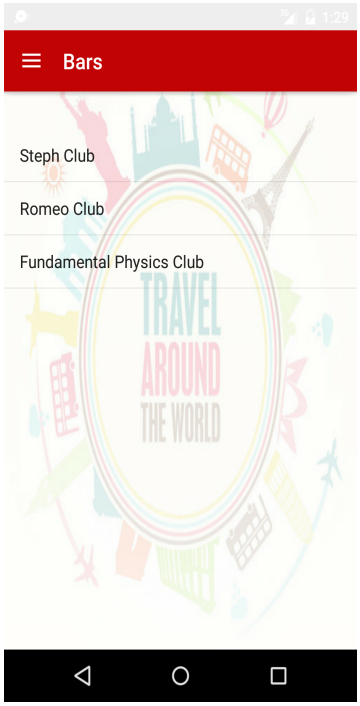
Restaurant

In the Restaurants menu the user can find a list of the restaurant available in the selected city.

If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected restaurant: address, information, Cuisines and reviews.

Bar

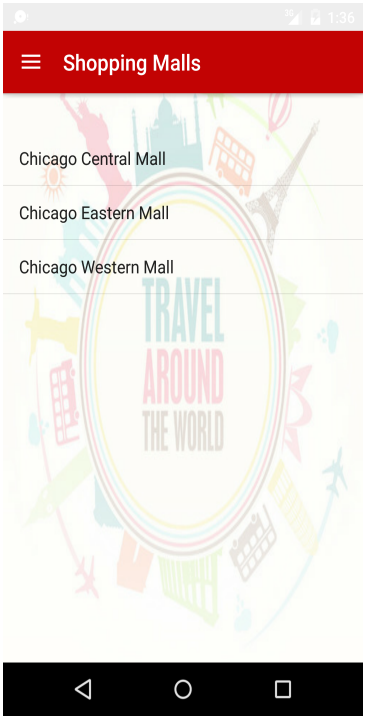
In the Bars menu the user can find a list of the bars and clubs available in the selected city.



If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected bar: address, description, disco/no disco and reviews.

Shopping Malls

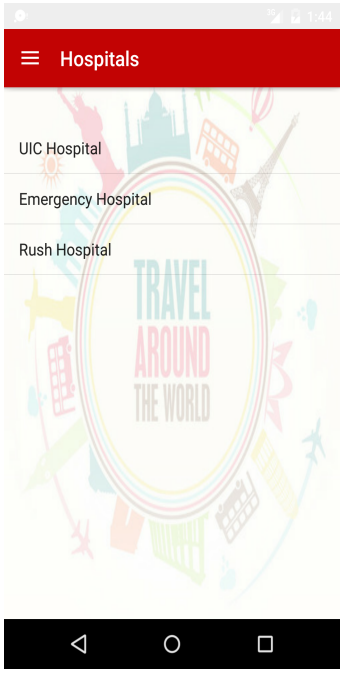
In the Shopping Malls menu the user can find a list of the malls available in the selected city.



If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected mall: address, information, brands, stores and reviews.

Hospitals

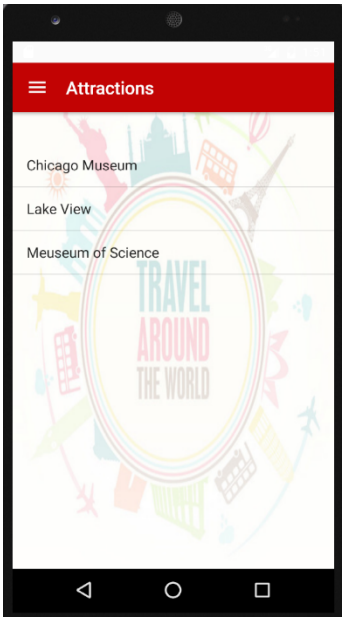
In the Hospitals menu the user can find a list of the hospitals available in the selected city.



If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected hospital: address, description, specializations and reviews.

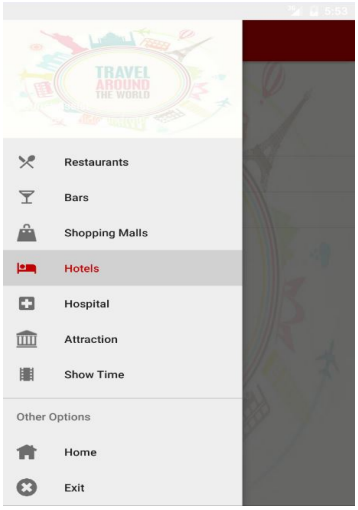
Attraction

In the Attraction menu the user can find a list of the Attraction and clubs available in the selected city.



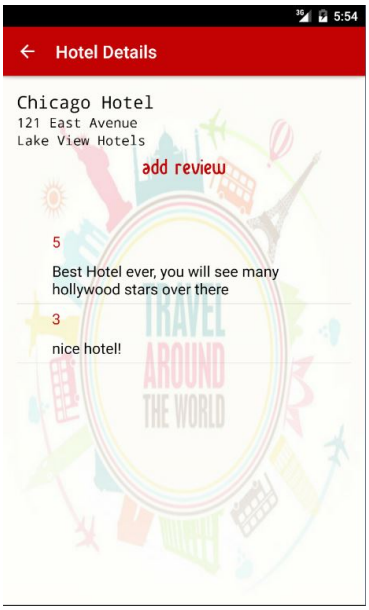
If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected attraction: address, description, fee and reviews.

Hotels

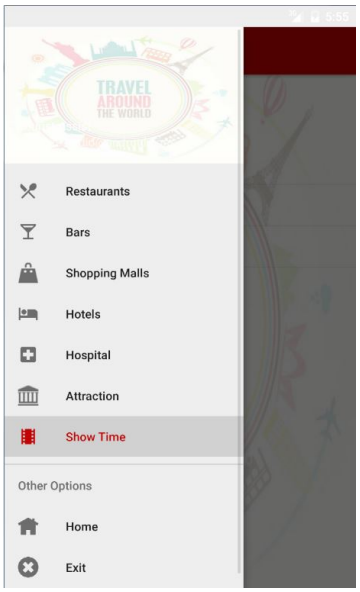


In the Hotels menu the user can find a list of the Hotels available in the selected city.

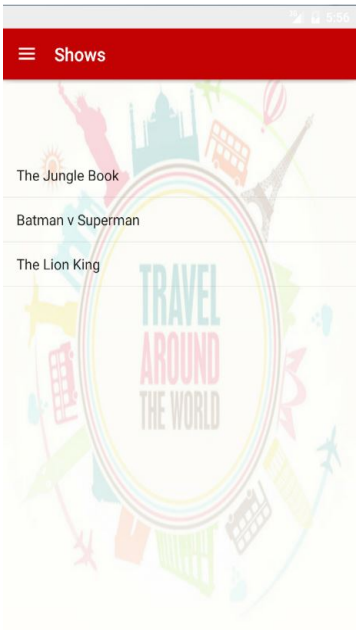
If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected hotel: address, description and reviews.


5:54
Hotel Details
Chicago Hotel
121 East Avenue
Lake View Hotels
add review
5
Best Hotel ever, you will see many hollywood stars over there
3
nice hotel!
TRAVEL
AROUND
THE WORLD

Show Times



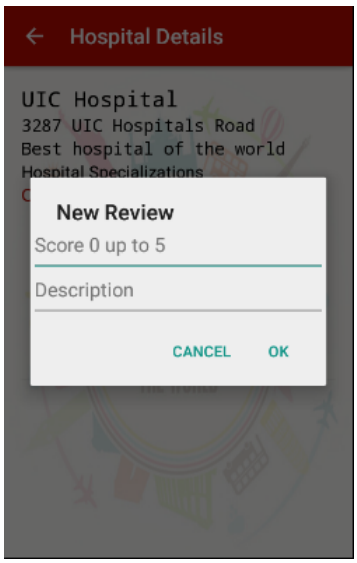
In the Show times menu the user can find a list of the shows and movies available in the selected city.



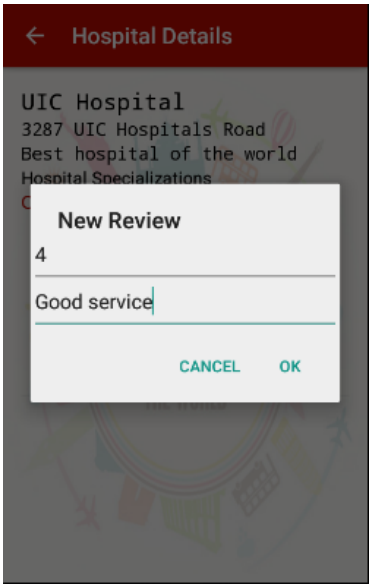
If one of the rows is clicked on, the application will move to a new screen that shows a list of details about the selected show time: address, description, duration, days of show and reviews.

Reviews

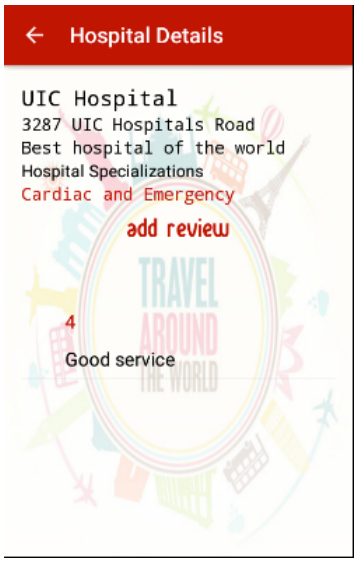
By clicking on the button “ADD REVIEW” it is possible to rate the hospital and write a review in any of the categories.



The inputted rating score has to be a number between 0 and 5 and the description cannot be left blank.

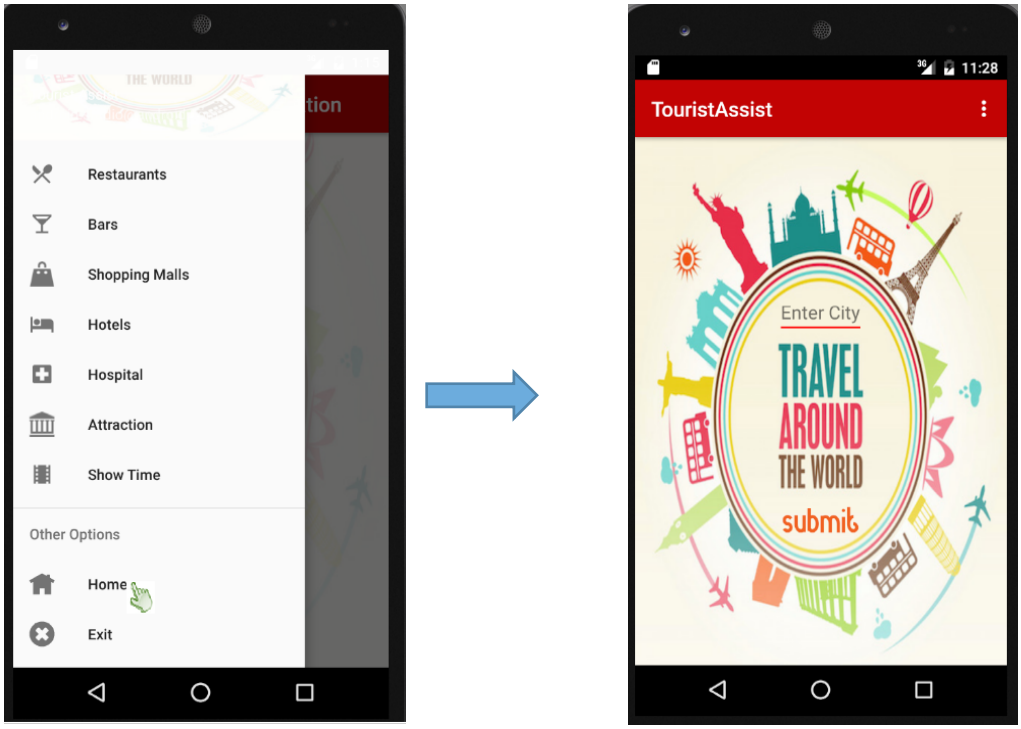


If both requisites are met, the review will be sent to the server and immediately posted:

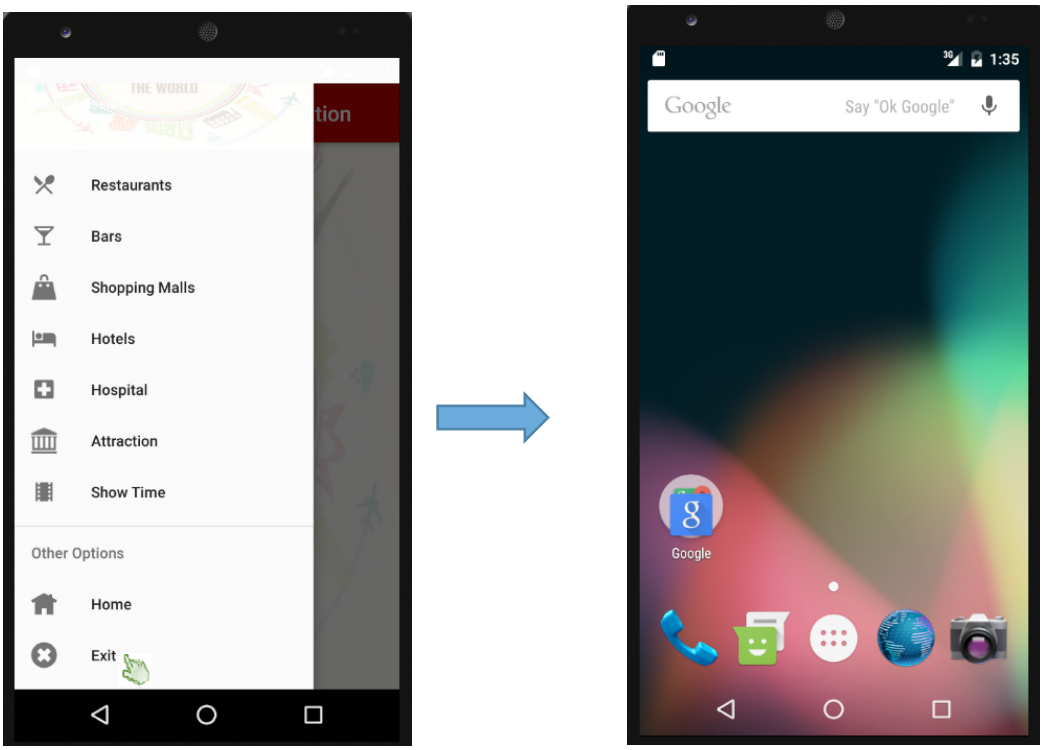


Other Options

On scrolling down in the Navigation bar user can get in to Main Page of the application by clicking on ‘Home’ Button



Similarly on user can exit from the application by clicking on ‘Exit’ Button which takes to the home screen of the phone.



Developer Guide

Setting up the Database and Backend Services.

Database Setup Guide

Clone the repository using the git clone and use the repository url to get the latest code.
You can either create your own database, MySQL or use the RDS provided by the AWS.
In this project we have used RDS services of AWS to host a MySQL server. If you are setting up your own server and not using the project default server, you will need to run the DB scripts which are located in the TA_DB_SCRIPTS. The TA_DB_SCRIPT folder further contains two other folders which contain the Table creation scripts and the Seed data script.

It is advisable to create the tables using the scripts provided in the TA_DB_SCRIPT /TABLES folder and then seed your tables with the data using TA_DB_SCRIPT/SEED_SCRIPTS. The process discussed should be enough to run your database.

If you are going to be using the schema which is created by the team from the AWS, you will need following connection details.

Database URL: stiwari5-cluster.cluster-cltssb30vxlo.us-east-1.rds.amazonaws.com

Port: 3306
Username: ta_user
Schema: tourist_assist

These details will also be used in DBConnectionManager.java file which we are going to discuss in further sections of the report.

For Entity Relationship diagram please refer to the DB diagram drawn in the Design report.

Backend Services:

Overview and Running Services

The services are exposed as SOAP, by its nature SOAP uses XML information set to send and receive the message. The mobile clients use the "org.ksoap2" libraries to make sure they send and receive the information to the services as per the specification.

In our services we use JAX-WS specification which is a specification laid out by the Java API for XML web services.

To run the web services, the developers can make use of Weblogic or the tomcat server. For our development purpose the team has used jdeveloper which is an IDE that comes up with an integrated weblogic server. The version of the IDE is 12c and is free to download using Oracle developer download program.

If you are using the jdeveloper, just rightclick on the ToursitAssist.java file and run them as webservice. The process will automatically start the weblogic server and will deploy the war file that includes your services.

The same war file can be used in deploying it to the tomcat server with minimal changes in WEB-INF/weblogic.xml and replacing the content with WEB-INF/web.xml

Description of Backend Services

This section explains the usage of each file in the order of their appearance in the flow of code as mobile devices and emulators make request to the server.

As soon as the client makes a call to the server, the weblogic server redirects the request XML and parses it in the form of objects and sends it to the exposed file which goes by the name of TouristAssistService.java

TouristAssistService

This file is responsible for accepting request and response from the clients. It has all of its method which are exposed public and they directly deal with the webservice infrastructure of the weblogic server.

Sample web service method in the code.

```
@WebMethod
public CityResponse getCityFromCityName(@WebParam(name = "arg0") String cityName) {
    ResponseBuilder responseBuilder = new ResponseBuilder();
    return responseBuilder.getCityFromCityName(cityName);
}
```

ResponseBuilder

The next file in the flow of code is ResponseBuilder.java which takes care of the response and helps in building responses which are pulled from the database. ResponseBuilder class holds different types of Response which further hold the array of objects.

For example, ResponseBuilder will return RestaurantResponse and RestaurantResponse class will have array of Restaurants with their details in it.

Following is an example of ResponseBuilderMethod

```
public RestaurantResponse getRestaurants(String cityId) {
    RestaurantResponse restaurantResponse = new RestaurantResponse();
    restaurantResponse.setRestaurants(genResponse.getRestaurants(cityId));
    return restaurantResponse;
}
```

GenResponses

The file GenResponses.java is responsible for making a call to the DBConnectionManager and parse the result into the Array Objects and return the array to the ResponseBuilder class. The GenResponses class also has a facility in which it returns dummy data in case no data is available. This is controlled by a variable, isDBDataAvailable, which is Boolean in nature, and to be set false in case of no database availability. If the DB is not available, the services will return a dummy data.

Following is an excerpt from the GenResponses class.

```
public Review[] getReviews(String entityId) {
    ArrayList<Review> reviews = new ArrayList<Review>();
    if (!isDBDataAvailable) {
        reviews.add(new Review("020", "004", "3.5", "Very nice Place"));
        reviews.add(new Review("021", "006", "4", "Awesome Place"));
        reviews.add(new Review("022", "010", "5", "Best Place"));
    } else {
        SQLAggregatedConnectionObjects sqlObjects = new SQLAggregatedConnectionObjects();
        String query = "select * from REVIEWS where ET_ID=" + entityId;
        sqlObjects = connectionMgr.fetchResultSetFromQuery(query, sqlObjects);
        ResultSet resultSet = sqlObjects.getResultSet();
        try {
            while (resultSet.next()) {
                String REVIEW_id = resultSet.getString("REVIEW_id"), ET_ID =
                    resultSet.getString("ET_ID"), REVIEW_RVW = resultSet.getString("REVIEW_RVW"), REVIEW_RATING =
                    resultSet.getString("REVIEW_RATING");
                reviews.add(new Review(REVIEW_id, ET_ID, REVIEW_RATING, REVIEW_RVW));

            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                connectionMgr.closeConnectionObjects(sqlObjects);
```

DBConnectionMgr

This is the class which is responsible for making Database connection and return the SQLAggregatedObjects to the caller by connecting to the DB and calling DB connections. It has two methods one which inserts and updates the data and the other that just fetches certain data based on the query.

The two methods:

1. fetchResultSetFromQuery
2. insertOrUpdateObjectsUsingQuery

The reason for making these two methods different was because fetching a data involves result set while the other just returns a value in the Statement object of SQL Aggregated objects.

The following is the screenshot of the fetchResultSetFromQuery method.

```
public static SQLAggregatedConnectionObjects fetchResultSetFromQuery(String query, SQLAggregatedConnectionObjects
    try {
        Class.forName(JDBC_DRIVER);

        sqlObjects.setConnection(DriverManager.getConnection(DB_URL, USER, PASS));
        sqlObjects.setStatement(sqlObjects.getConnection().createStatement());
        sqlObjects.setResultSet(sqlObjects.getStatement().executeQuery(query));

    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return sqlObjects;
}
```

JUnits

TouristAssistServiceTest

The project has an extensive coverage of JUnits. The JUnits test whether a certain service is working independentl of the servers or not. The excerpt of the code shows a sample method of the JUnit test case.

```
/**
 * @see com.cs.se.ta.main.TouristAssistService#getReviews(String)
 */
@Test
public void testGetReviews() {
    try {
        TouristAssistService touristAssistService = new TouristAssistService();
        ReviewResponse reviewResponse = touristAssistService.getReviews("017");
        if(reviewResponse.getReviews().length != 0){
                assertTrue("Success", true);
        } else {
                fail("No records found");
            }

    } catch (Exception e) {
        e.printStackTrace();
        fail("Failure");
    }
}
```

It makes a call to the service and checks whether for a given entity id the response is there or not. If the array returned is of the size zero, then it returns failure otherwise success is displayed.

Other Supporting Files.

Response Holders: Response Holders are specific to each module like RestaurantResponse, ClubResponse, HotelResponse, City etc. They contain the array of ResponseObjects retrieved from the database or static implementation.

Response Objects: Response Objects fill the response holders and they are generally returned in the form of Arrays and are set in the Response Holders. They are mostly by the name of Hotel, Restaurant, Club, City etc.

The Users making changes in the services are requested to follow the same pattern of development as it is explained in this developer guide. To make any architectural changes please discuss it with the team before implementing it.

Please refer to the sequence diagram in the project report for more details.

Future Work

This project has resulted on a working prototype of the application, and some of the future work that might need to be done on the application would be done on two parts:

On the WebServices:

- Integrate the Model Objects with ORM to remove the text based queries.
- Add services for finding city by coordinates
- Add services for finding city by cityId
- Auto-suggest services which fetch likely spelled cities.
- Add SQL sanitization.
- Write Manual Exception mechanism to propagate business exceptions.
- Store the Average of all the reviews.
- Add support for User level data like personal reviews and ratings and cities/places visited.

On the User Interface:

- Enable auto suggestion of city names
- Adding a map based search and find support
- Adding user profile and security features
- Personalized reviews and arrange them by descending order of rating.
- Maintaining history of a user data