T.C.
MARMARA UNIVERSITY
FACULTY of ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

CSE4074 Programming Assignment
Socket Programming

Title of the Project
Trip Planner

Group Members
150115014 – Tolunay Katırcı
150115036 – Hakkı Zahid Kocabey

Content

1 Introduction

In this project, we implemented a trip planner system. A trip plan consists of a hotel reservation in a requested time interval and a flight reservation for requested number of travelers. Specifically: A customer informs the travel agency of the arrival and departure date of his trip, his preferred hotel and airline and the number of travelers. Then travel agency contacts to the hotel and airline and ask for availability for corresponding dates. If there are available rooms for given number of people in the hotel and available seats in the flight for a given number of travelers, then the travel agency finalizes the trip. When the trip is finalized, the updates are reflected in the databases of the hotel and the airline. If the preferred hotel or flight is unavailable, then the databases remain unchanged and the travel agency contacts to alternative hotel(s) or airline(s) and propose an itinerary to the customer.

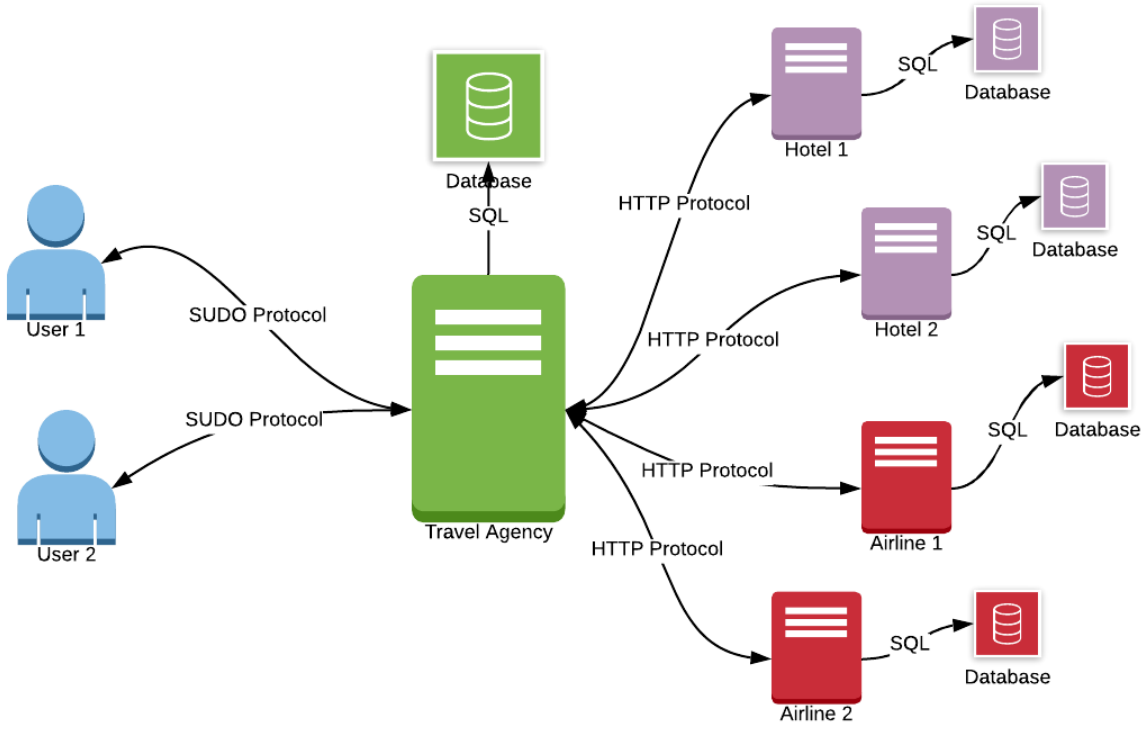
Initially, we created four projects. These are:

- Hotel Project (Java)
- Airline Project (Java)
- Travel Agency Project (Java)
- Customer GUI (Android Studio)

We implemented TCP connection and created multithreaded socket servers. HTTP protocol is used for communication between Travel Agency and Hotels and Airlines. Custom SUDO protocol is implemented for communications between Travel Agency and customers.

We created a jar file for java projects. So, we can easily copy and run the jar to create new hotel or airline.

1.1 Communication Operations



1.2 Bonus Parts

We developed all socket servers as multithreaded. But there is no database synchronization control. So, there may be synchronization issues if customers try to create reservation on same hotels/airlines.

We created jar files for hotels and airlines. To add new hotel or airline, copy jar file and bat file, create or update config and run ‘run-hotel.bat’ or ‘run-airline-bat’ file. It will automatically register to travel agency.

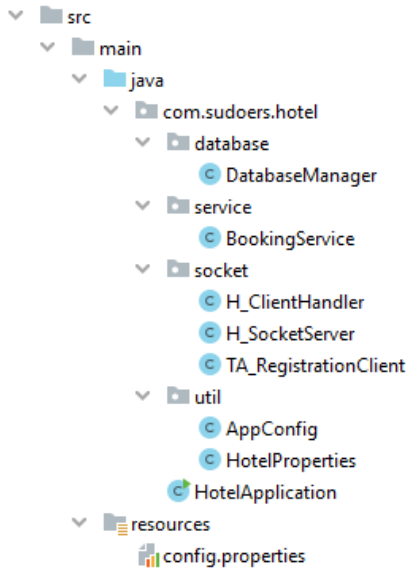
2 Implementation Details

2.1 Hotel Project

Hotel application is built for single hotel. When project started, it initially loads application properties from ‘config.properties’ file. This file consists IP and port information, database name etc. Then, the application connects to database and creates initial tables. It sends a HTTP request

to register itself to Travel Agency service. Finally, runs a socket server for reservation and other HTTP communications with Travel Agency.

```java
public static void main(String[] args) {
    // get application properties from file
    AppConfig.getApplicationProperties();
    // connect to database
    DatabaseManager.connect();
    // create initial tables
    DatabaseManager.createInitialTables();

    // register to travel agency
    TA_RegistrationClient client = new TA_RegistrationClient();
    new Thread(client).start();

    // run socket server
    runSocketServer();

}
```



2.1.1 Load Application Properties

```
hotel.name=Hotel Marina
hotel.port=8081

hotel.database.path=hotel.db
hotel.database.room_count=20

travel-agency.ip = 127.0.0.1
travel-agency.port = 8080
```

```java
public class HotelProperties {

    private String hotelName;
    private int port;

    private String databasePath;
    private int roomCount;

    private String travelAgencyIP;
    private int travelAgencyPort;
}
```

‘config.properties’ file contains information above. We created HotelProperties model and keep initial application properties in this model.

2.1.2 Database Manager

We implemented SQLite database for reservation operations. There are two tables in the database. These are ‘room’ and ‘booking’. Room table holds room id and room name. Booking table holds room id, customer name and reservation date. Reservation date is single day. For longer reservations, it holds multiple record within reservation date interval.

	book_id	room_id	customer_name	reservation_date
1	1	1	Tolunay	1575147600000
2	2	1	Tolunay	1575234000000
3	3	1	Tolunay	1575320400000
4	4	1	Tolunay	1575406800000
5	5	1	Tolunay	1575493200000
6	6	1	Tolunay	1575579600000
7	7	1	Tolunay	1575666000000

For figure above, database contains 7-days reservation record for room 1.

Database Manager includes SQL operation for reservation.

```
// make reservation daily
private void makeReservation(int roomId, String customerName, Date reservationDate){...}

// create reservation with specified values
public void book(int roomId, String customerName, Date startDate, Date endDate) {...}

// check if hotel available on specified date intervals
public List<Integer> isAvailable(Date startDate, Date endDate){...}

// check if room available on specified date
private boolean isRoomAvailable(int roomId, Date startDate, Date endDate) {...}
```

2.1.3 Booking Service

Booking service is singleton class. It has a static object and we call this instance to use this service. Booking service basically connects database manager and socket handler. For database operations, we will use booking service. It has same methods with database manager.

```
// checks if hotel is available in specified date intervals (same with db)
public boolean isAvailable(String customerCount, String startDate, String endDate) {...}

// make reservation with specified values
public boolean makeReservation(String customerName, String customerCount, String startDate, String endDate) {...}
```

2.1.4 Socket Server

We implemented a multithreaded socket server. When requests arrive, it generates a new Thread and sends it to Client Handler class. Server starts initially and listen requests in all running time.

```java
while (!isStopped) {
    Socket clientSocket = null;
    try {
        // accept request and send to handler
        clientSocket = this.serverSocket.accept();
    } catch (IOException e) {
        if(isStopped()) {
            System.out.println("Server Stopped.") ;
            return;
        }
        throw new RuntimeException("Error accepting client connection", e);
    }

    // handle request in background
    new Thread(new H_ClientHandler(clientSocket)).start();
    System.out.println("Server Stopped.");
}
```

2.1.5 Client Handler

When HTTP request comes to Hotel Application, Client Handler handles this request and sends an HTTP response. We used BufferedReader and PrintWriter for socket communication.

```java
in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
out = new PrintWriter(clientSocket.getOutputStream(), autoFlush: true);
```

Handler parses request and finds headers and request path. According to the path, it calls appropriate method to handle request. Then, creates a response and sends it to the Travel Agency client.

```
case "/isRunning":
    // is program running
    respond( statusCode: 200,  msg: "OK!",  body: true);
    break;
case "/getRoomCount":
    // get room count in db
    int roomCount = AppConfig.hotelProperties.getRoomCount();
    respond( statusCode: 200,  msg: "OK!", roomCount);
    break;
case "/isAvailable":
    // is hotel available in specified dates
    startDate = headers.get("startDate").trim();
    endDate = headers.get("endDate").trim();
    customerCount = headers.get("customerCount").trim();
    result = BookingService.getInstance().isAvailable(customerCount, startDate, endDate);

    // respond
    respond( statusCode: 200,  msg: "OK!", result);
    break;
case "/makeReservation":
    // make reservation with specified values
    startDate = headers.get("startDate").trim();
    endDate = headers.get("endDate").trim();
    customerName = headers.get("customerName").trim();
    customerCount = headers.get("customerCount").trim();
```

2.1.6 Registration Client

Registration Client Thread sends hotel information to the specified Travel agency with HTTP request.

2.2 Airline Project

Airline application is similar to Hotel application. Main operations are same, the differences are config file and some variable/parameter names.

```java
public static void main(String[] args) {
    // get application properties from file
    AppConfig.getApplicationProperties();
    // connect to database
    DatabaseManager.connect();
    // create initial tables
    DatabaseManager.createInitialTables();

    // register to travel agency
    TA_RegistrationClient client = new TA_RegistrationClient();
    new Thread(client).start();

    // run socket server
    runSocketServer();

}
```

2.3 Travel Agency Project

Travel Agency is also similar to Hotel Project. It has multithreaded socket server and it can handle HTTP requests and also custom SUDO protocol we created.

2.3.1 Database Manager

We implemented SQLite database to hold hotel and airline information. It has two tables and these are ‘hotel’ and ‘airline’ tables. These tables holds name, IP, port, capacity and running status information of hotels or airlines.

	hotel_id	name	ip	port	capacity	status
1	1	Silence Hotel	127.0.0.1	8081	8	☐
2	2	Beyaz Otel	127.0.0.1	8082	12	☑
3	3	Hotel Marina	127.0.0.1	8083	15	☐

Database manager includes SQL operations for registered hotels and airlines.

```java
// add hotel to database, if exists, update properties
public boolean addOrUpdateHotel(Hotel hotel) {...}

// find hotel by ip and port
public Hotel findHotel(String ip, int port) {...}

// find hotel by name
public Hotel findHotelByName(String hotelName) {...}

public List<Hotel> findAllHotels() {...}

public List<Hotel> findAvailableHotels() {...}
```

2.3.2 SUDO Protocol

We created a custom protocol named SUDO. It is proper for list requests and string map requests. It has host, port, version and function information. For list objects we need to add 'DATA:' prefix for each item in list. For map objects, we need to add 'DATA>' prefix and ':' character between map key and value. We used function property to understand request.

<u>Request:</u>

```
SUDO
HOST:192.168.1.37
PORT:8080
VERSION:VER_1.0
FUNCTION:getHotelList
DATA>testkey:testvalue
DATA>testkey2:testvalue2
DATA:testList
```

<u>Response:</u>

```
SUDO
STATUS:20
MESSAGE:Hotels successfully listed
VERSION:VER_1.0
DATA:Beyaz Otel
DATA:Hotel Marina
```

Our code also includes request and response parser. So, we easily communicated with Travel Agency and Customer.

```
if(currentLine.startsWith("HOST:"))
    sudoRequest.setHost(currentLine.substring(5));
else if (currentLine.startsWith("PORT:"))
    sudoRequest.setPort(Integer.parseInt(currentLine.substring(5)));
else if (currentLine.startsWith("VERSION:"))
    sudoRequest.setVersion(currentLine.substring(8));
else if (currentLine.startsWith("FUNCTION:"))
    sudoRequest.setFunction(currentLine.substring(9));
else if (currentLine.startsWith("DATA:")) {
    String dataValue = currentLine.substring(5);
    tempDataList.add(dataValue);
}
else if (currentLine.startsWith("DATA>")){
    String dataKeyValue = currentLine.substring(5);
    String[] splitted = dataKeyValue.split( regex: ":");
    String dataKey = splitted[0];
    String dataValue = splitted[1];
    tempDataMap.put(dataKey, dataValue);
}
```

2.3.3 Services

We have Airline Service and Hotel Service. Airline Service handles airline requests and Hotel Service handles hotel requests. These services are also singleton class. We need to call instance object to use it.

```
// add or update airline (same with db)
public boolean addOrUpdateAirline(String name, String ip, String port, String capacity) {...}

// get active Airlines (where status = true)
public List<Airline> getAvailableAirlines() { return databaseManager.findAvailableAirlines(); }

// get all Airlines
public List<Airline> getAllAirlines() { return databaseManager.findAllAirlines(); }

// make airline reservation with specified properties
public boolean makeReservation(String startDate, String endDate, String customerName, int customerCount, String airlineName) {...}
```

2.3.4 Socket Server

We implemented multithreaded socket server. As in Hotel Projects, it accepts requests and creates a new handler thread to handle request. Socket server handles multiple clients but there will be synchronization issues.

2.3.5 Client Handler

We need to handle both HTTP requests and SUDO requests. HTTP request for hotels and airlines, SUDO requests for customers. If request starts with 'SUDO', it uses SUDO protocol. When we figured out the type of request protocol, we called appropriate handler.

```
// parse requests according to function
switch (sudoRequest.getFunction()){
    case "getHotelList":
        List<Hotel> hotelList = HotelService.getInstance().getAvailableHotels();
        List<String> hotelNames = new ArrayList<>();
        for (Hotel h : hotelList) hotelNames.add(h.getName());

        // create response
        sudoResponse = new SUDOResponse();
        sudoResponse.setStatus(20);
        sudoResponse.setMessage("Hotels successfully listed");
        sudoResponse.setDataList(hotelNames);
        respondSUDO(sudoResponse);
        System.out.println("Response sent!");
        System.out.println(sudoResponse);
        break;
    case "getAirlineList":
        List<Airline> airlineList = AirlineService.getInstance().getAvailableAirlines();
        List<String> airlineNames = new ArrayList<>();
        for (Airline a : airlineList) airlineNames.add(a.getName());

        // create response
        sudoResponse = new SUDOResponse();
        sudoResponse.setStatus(20);
        sudoResponse.setMessage("Airlines successfully listed");
        sudoResponse.setDataList(airlineNames);
        respondSUDO(sudoResponse);
        break;
```

2.3.6 Heartbeat Client

Heartbeat client sends an HTTP request to all hotels and airlines registered in database in every thirty seconds. So, we can figure out which hotels and airlines active.

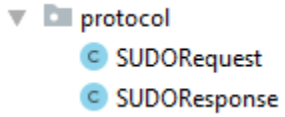
2.3.7 Socket Manager

Socket manager creates HTTP requests for communication with hotels and airlines.

```
// checks selected airline / hotel is running
public static boolean isRunning(String ip, int port){...}

// finds available hotels
public static List<String> getAvailableHotels(String startDate, String endDate, int customerCount) {...}

// finds available airlines
public static List<String> getAvailableAirlines(String startDate, String endDate, int customerCount) {...}

// is hotel / airline available in specified date
public static boolean isAvailable(String ip, int port, String startDate, String endDate, int customerCount){...}

// make hotel / airline reservation with specified properties
public static boolean makeReservation(String ip, int port, String startDate, String endDate, String customerName, int customerCount){...}
```

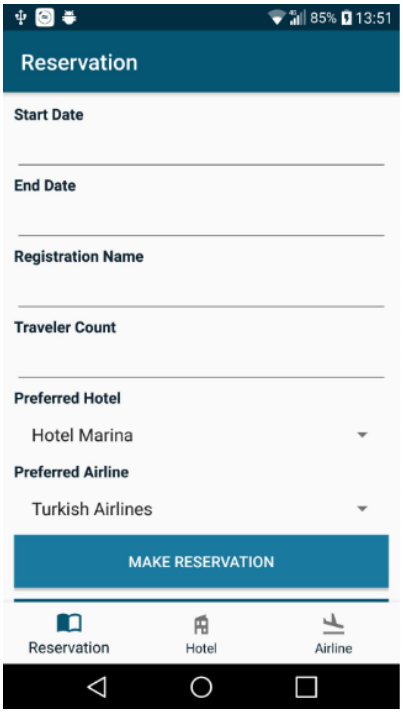
2.4 Customer GUI

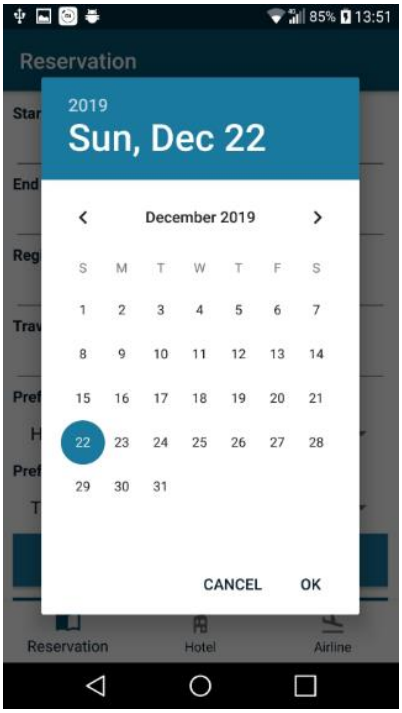
We created an Android Project for customer GUI. Customer GUI also has protocol objects to communicate Travel Agency Service. Active status of all hotels and airlines are updated in every 30 seconds in a background task.

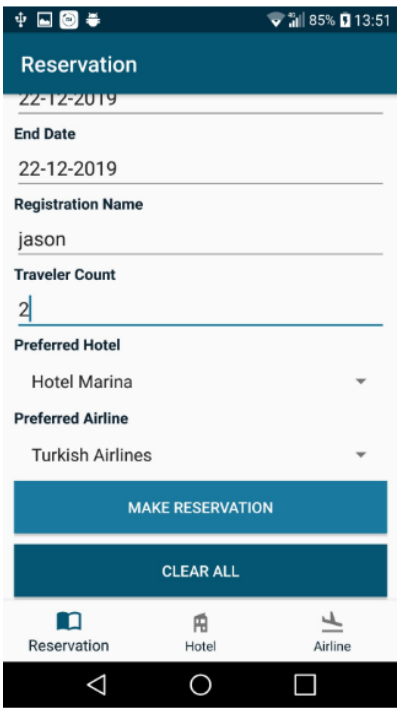


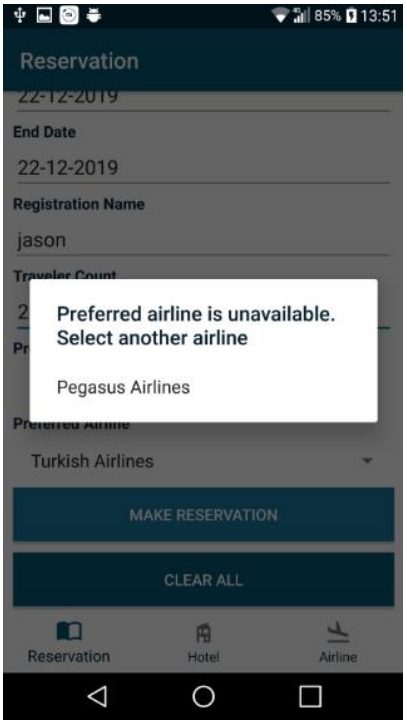
We have 3 pages. First one is for reservation operations, second one lists active hotels and the last one lists active airlines.

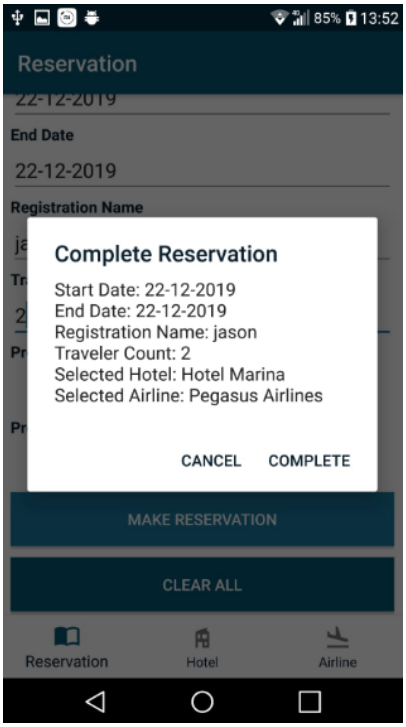
There are some screenshots available:

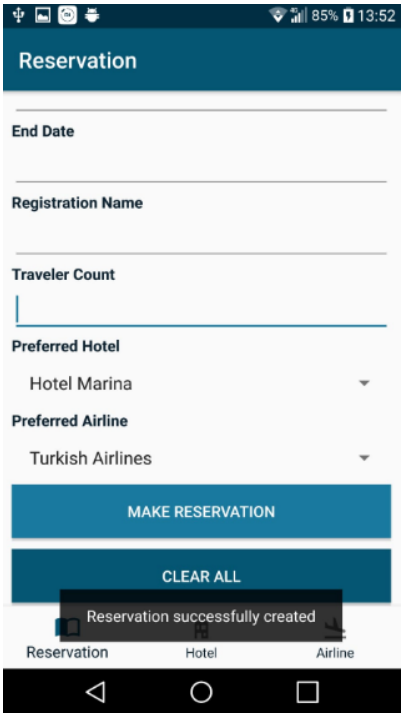

85% 13:51
Reservation
Start Date
End Date
Registration Name
Traveler Count
Preferred Hotel
Hotel Marina
Preferred Airline
Turkish Airlines
MAKE RESERVATION
Reservation
Hotel
Airline

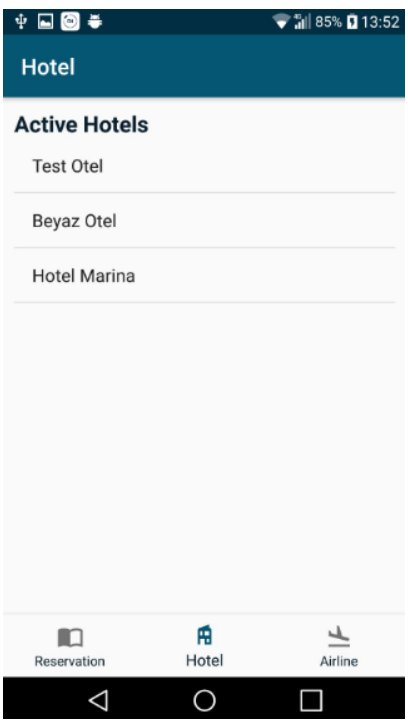

85% 13:51
Reservation
2019
Sun, Dec 22
December 2019
S M T W T F S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
CANCEL
OK
Reservation
Hotel
Airline

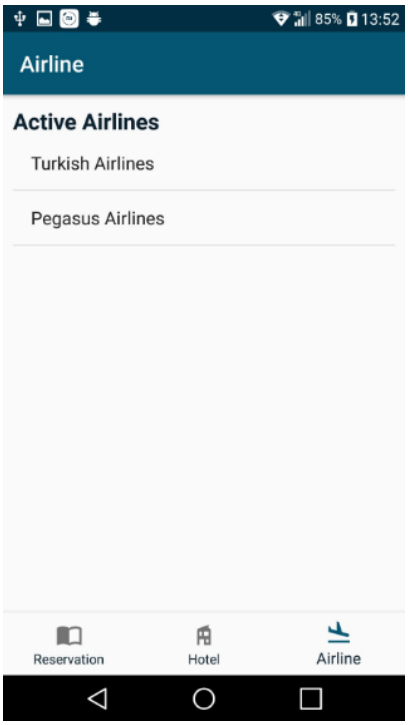

85% 13:51
Reservation
End Date
22-12-2019
Registration Name
jason
Traveler Count
2
Preferred Hotel
Hotel Marina
Preferred Airline
Turkish Airlines
MAKE RESERVATION
CLEAR ALL
Reservation
Hotel
Airline

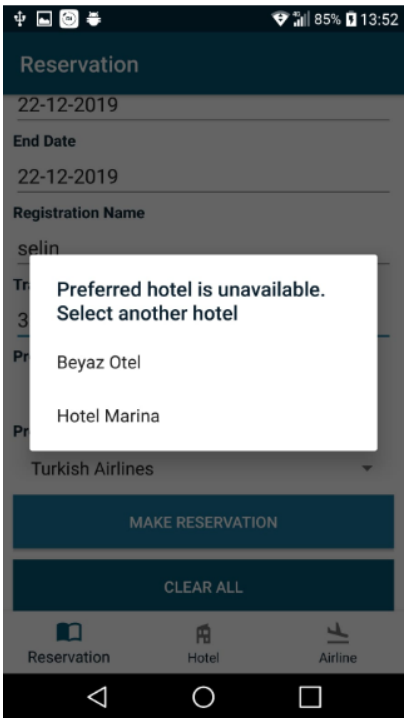

85% 13:51
Reservation
End Date
22-12-2019
Registration Name
jason
Preferred airline is unavailable.
Select another airline
Pegasus Airlines
Turkish Airlines
MAKE RESERVATION
CLEAR ALL
Reservation
Hotel
Airline


85% 13:52
Reservation
End Date
22-12-2019
Registration Name
Complete Reservation
Start Date: 22-12-2019
End Date: 22-12-2019
Registration Name: jason
Traveler Count: 2
Selected Hotel: Hotel Marina
Selected Airline: Pegasus Airlines
CANCEL
COMPLETE
MAKE RESERVATION
CLEAR ALL
Reservation
Hotel
Airline


85% 13:52
Reservation
End Date
Registration Name
Traveler Count
Preferred Hotel
Hotel Marina
Preferred Airline
Turkish Airlines
MAKE RESERVATION
CLEAR ALL
Reservation successfully created
Reservation
Hotel
Airline







3 User Manuel

- Update config files if necessary (IP and port configuration).
- Start ‘run-travel-agency.bat’. It will run travel agency jar file (You can manually run jar file also).
- Start ‘run-hotel.bat’ for each hotel service.
- Start ‘run-airline.bat’ for each airline service.
- Update IP for Customer GUI application. It defined in Trip Planner Service in Android project. Default is ‘127.0.0.1’

```
public class TripPlannerService {

    private static final String TRAVEL_AGENCY_IP = "192.168.1.37";
    private static final int TRAVEL_AGENCY_PORT = 8080;
```

- Build apk or run project on the phone or emulator.