INDIZA SA FLIGHT MANAGEMENT SYSTEM USER MANUAL

DEVELOPED BY CHRISTIAN VAN ZYL

Installation

IndizaSA flight management system requires three main software applications to be successfully executed:

- MySQL Server 8.0.26, Workbench 8.0.26 CE, Connector/J 8.0.27, Connector/NET 8.0.26
- Apache NetBeans IDE 12.2
- Android Studio Arctic Fox | 2020 3.1 Patch 3 using VM: OpenJDK 64-Bit Server VM by Oracle Corporation

A brief explanation of the software installation will now follow.

MySQL Software

To install the various MySQL utilities, one must firstly download MySQL Installer from the official MySQL website as shown in figure 1.

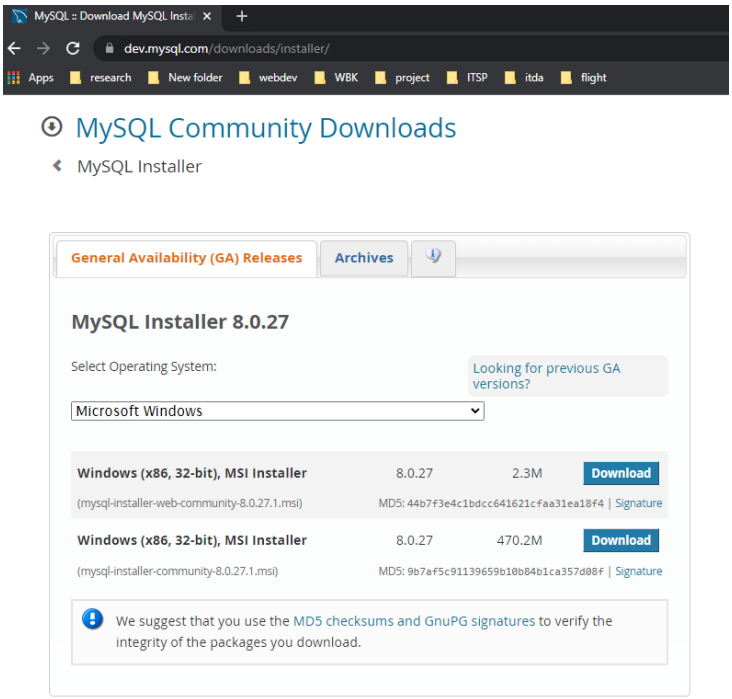


Figure 1

Upon installation, one can choose to add the necessary products to be downloaded and installed. The software can also be downloaded and installed after the installation has finished by reopening MySQL Installer as shown in figure 2.

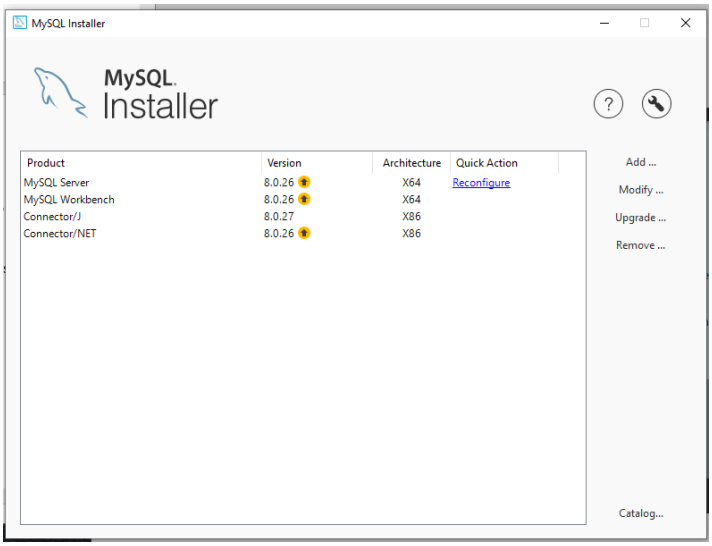


Figure 2

Clicking on the add button (as shown in figure 3) and then following the instructions on the following window (as shown in figure 4).

Figure 3

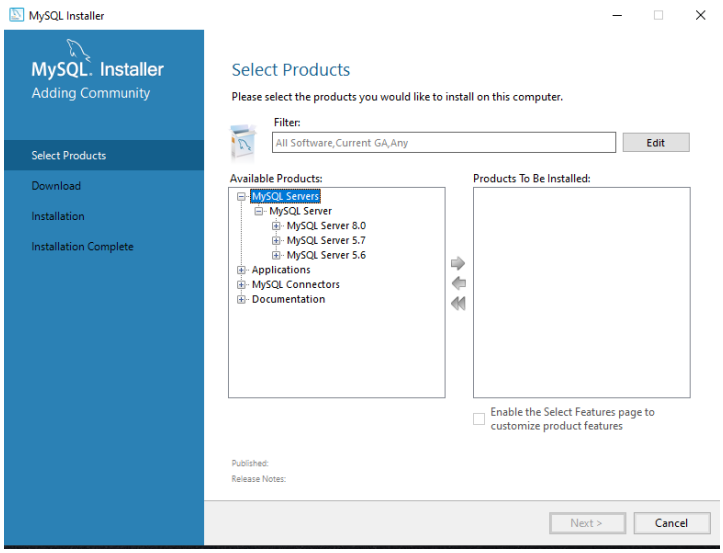


Figure 4

It is important to note that upon opening MySQL Workbench for the first time, that one can configure the MySQL Connections.

If no connection exists, one can click on the + icon as shown in figure 5. Otherwise, if a connection exists, one must click on the wrench icon to configure certain settings.

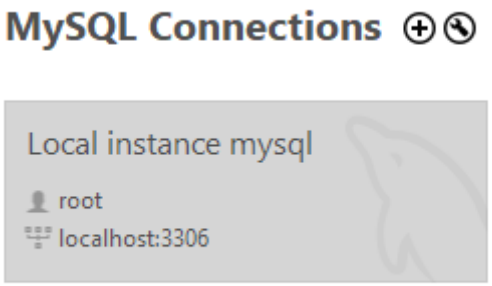


Figure 5

It is crucial that the username is set to “root” and that password of “root” is set. Furthermore, the connection should be set as Standard TCP/IP, the hostname as 127.0.0.1 and the port as 3306. This is shown in figure 6. Without these configurations the database will not be able to connect with the Java Server program launched from NetBeans.

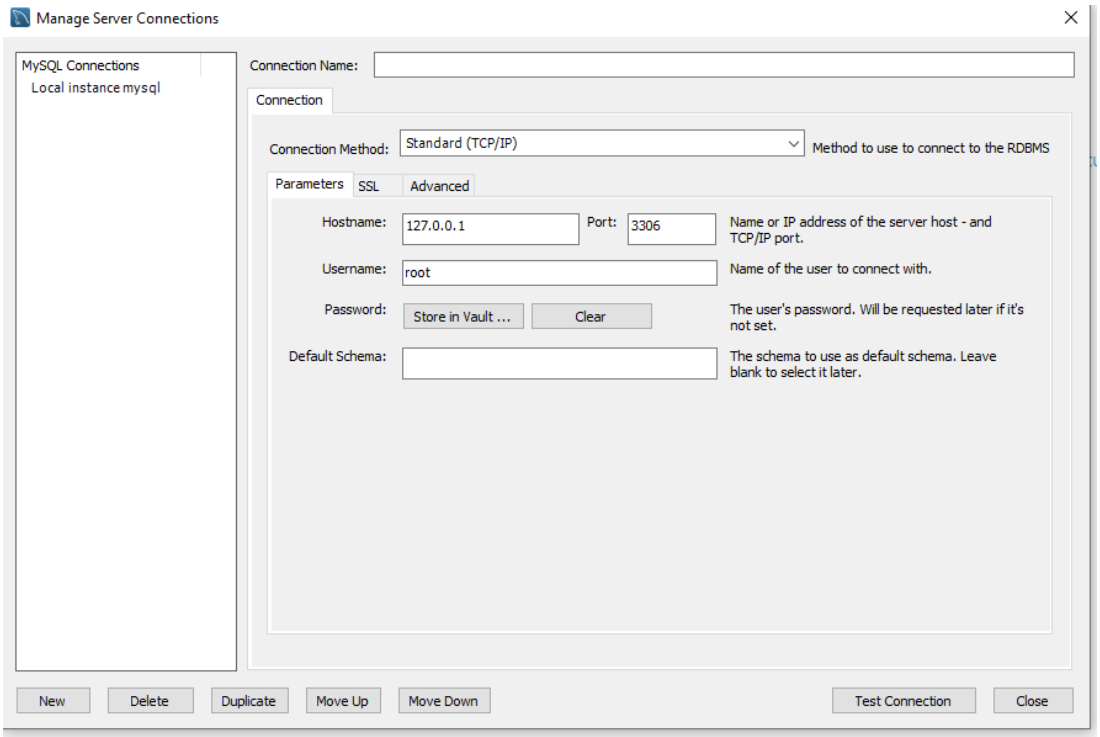


Figure 6

Click on the created/configured connection to open the interface. If the notification shown in figure 7 is present, it will be necessary to start the MySQL server manually.

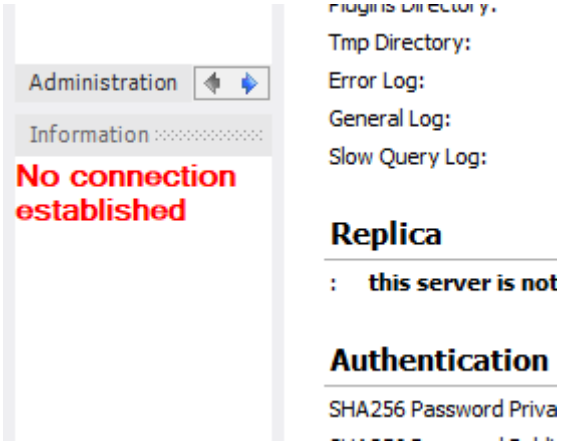


Figure 7

In order to start the MySQL server manually, one needs to search for “Services” in the Windows search bar and open the utility (as shown in figure 8).

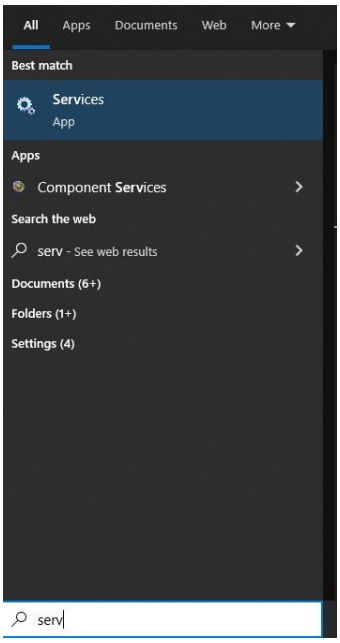


Figure 8

Select MySQL80 and click on “Start service” as shown in figure 9.

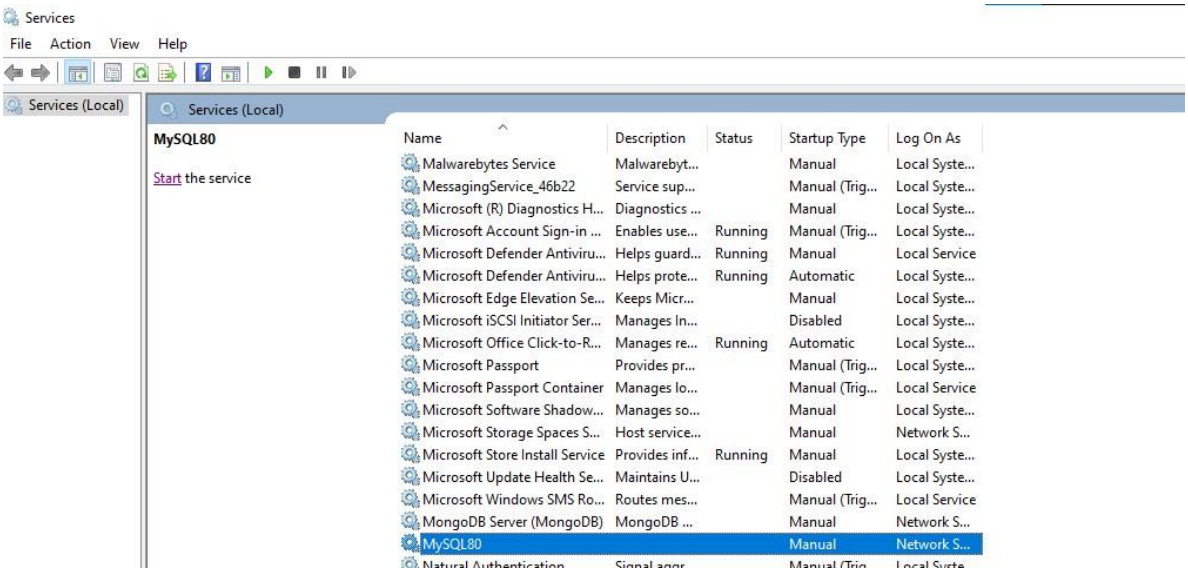


Figure 9

If the connection is still not established, one can restart MySQL Workbench or simply select “Server” from the menu tab and “Startup/Shutdown” from the dropdown (as shown in figure 10).

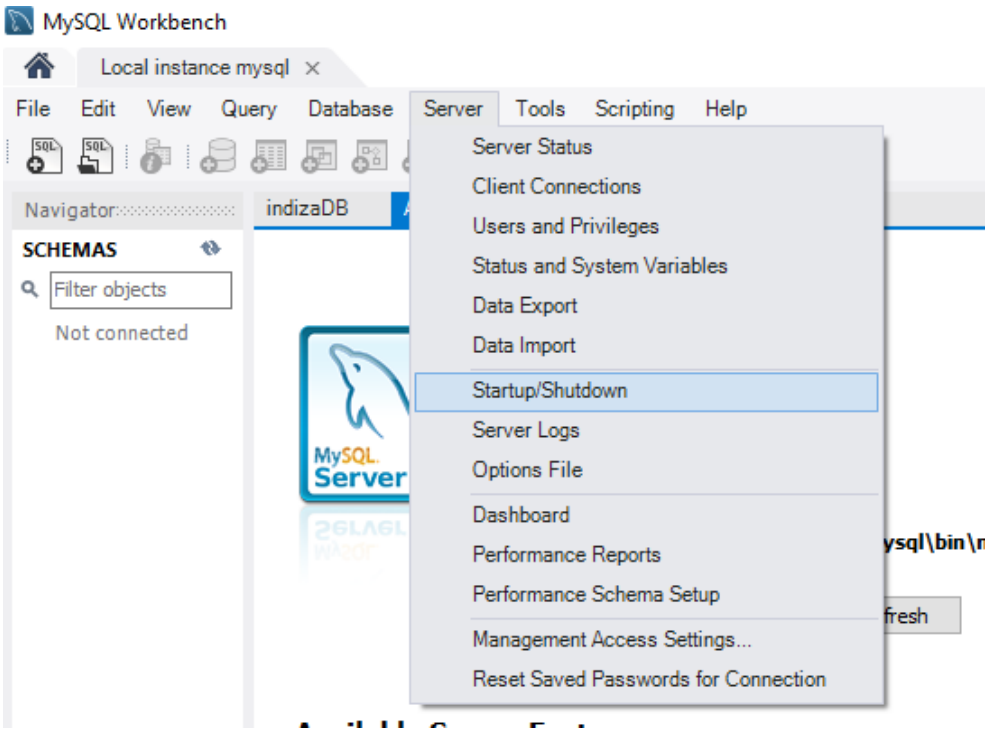


Figure 10

The database can now be imported into Workbench. Select “File” from the menu and “Open SQL Script” as shown in figure 11.

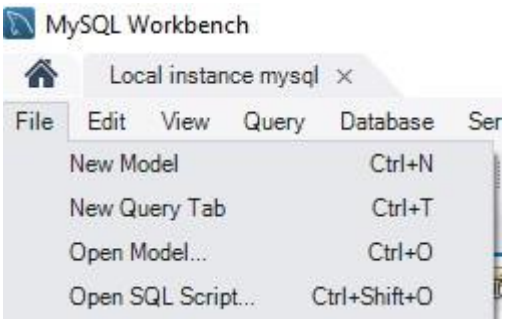


Figure 11

Browse to the DatabaseSQL folder and open the “indizaDB” file as shown in figure 12.

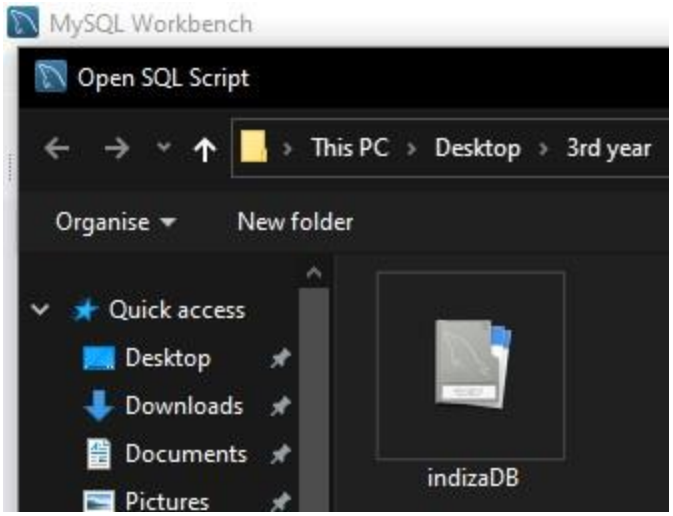


Figure 12

The opened file must now be executed for the database to be created. Select the “indizaDB” query tab and click on the yellow lightning button to execute the script, as shown in figure 13.

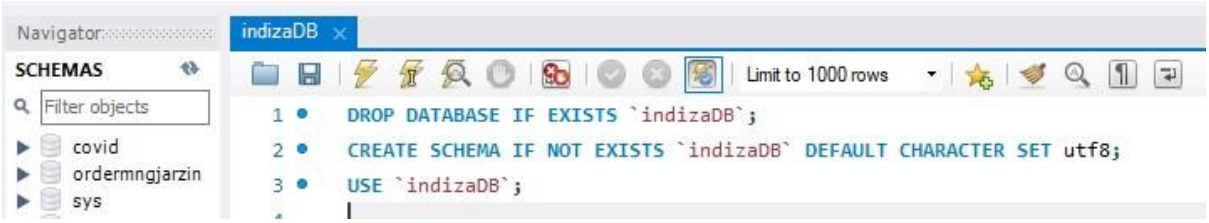


Figure 13

Clicking on the refresh icon in the Navigator window will refresh the schemas. The “indizadb” database should now appear as shown in figure 14.

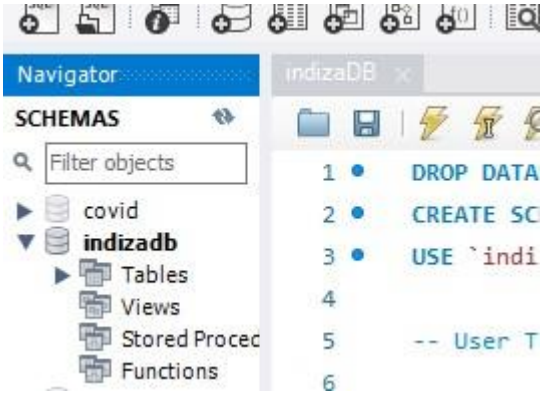


Figure 14

If the database is not printed in bold, it was not automatically selected as the default schema. Right click on the database name and from the menu select “Set as Default Schema” (shown in figure 15).

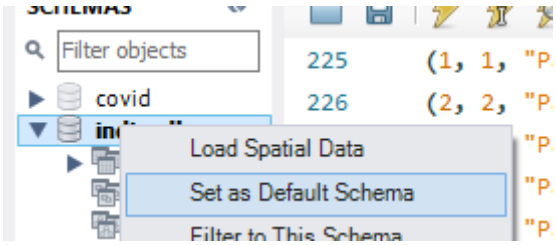


Figure 15

The database is now correctly configured.

NetBeans IDE

Navigate to the Apache NetBeans website and click on the “Download” tab in the upper right-hand corner (as shown in figure 16). Download the latest version of the IDE and follow the installation process.

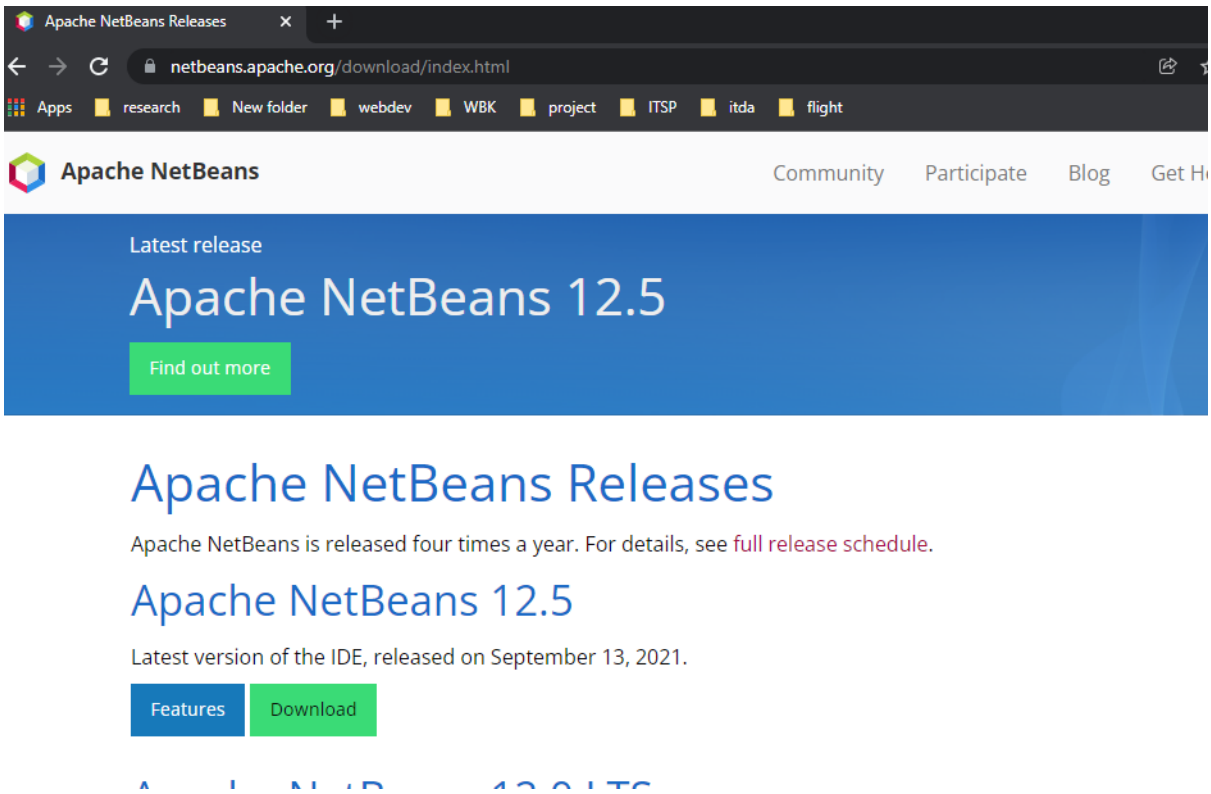


Figure 16

After the program has been installed and executed, click on the “File” tab in the menu and select “Import Project” from the dropdown menu (as shown in figure 17).

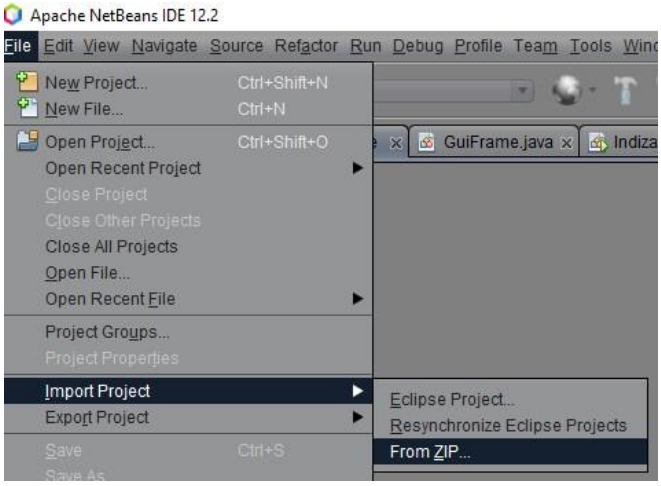


Figure 17

Click on browse and navigate to the provided NetBeans zip folder and click on import (as shown in figure 18)

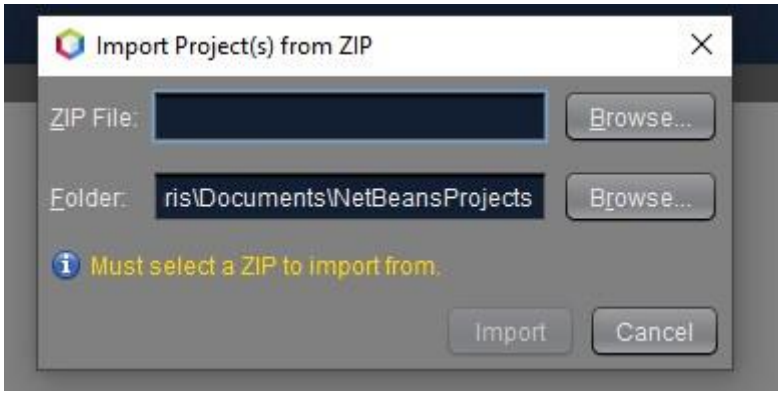


Figure 18

Click on the IndizaSAServer.zip to import the project (shown in figure 19).

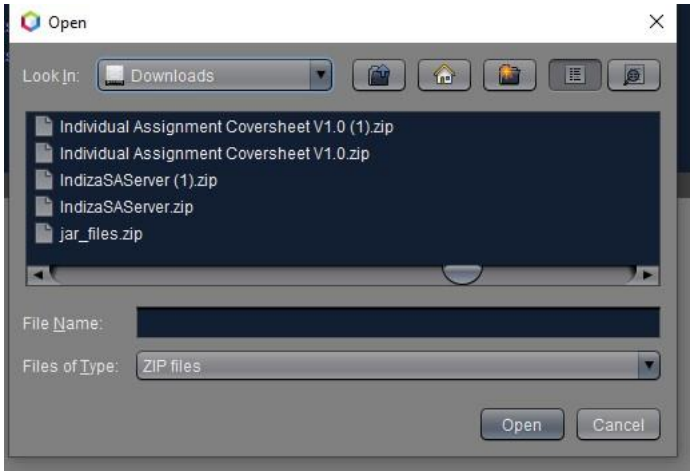


Figure 19

The Java files shown in figure 20 should now be present in the IndizaSA package.

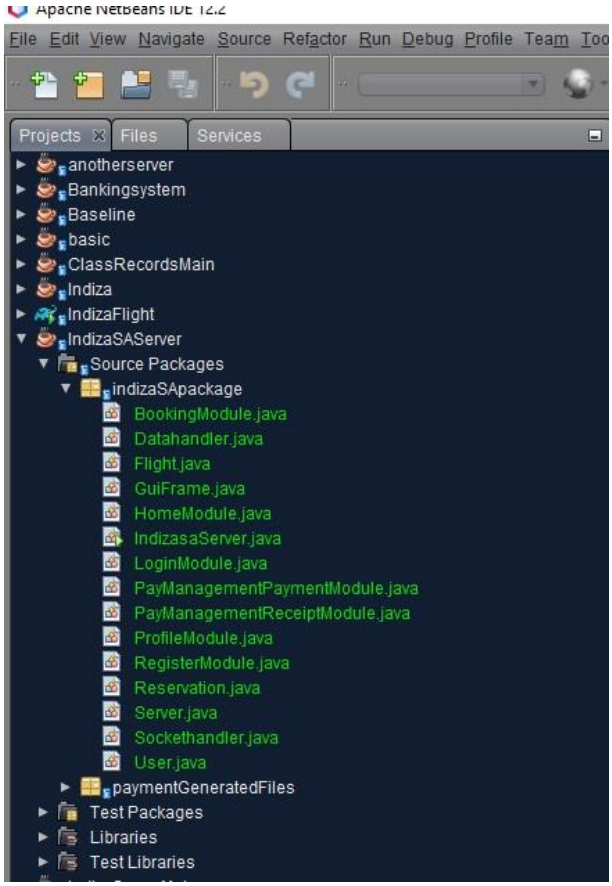


Figure 20

Click on the Libraries folder to double check that the necessary .jar files are present (as shown in figure 21). Also note that version 15 of the JDK is being used.

Figure 21

The two jar files are provided in the folder labelled “jarFiles”. If all the Java files, as well as the .jar files are present, the server is ready to be started. Select the “IndizaServer.java” file (as shown in figure 22) and click on run and build.

Figure 22

On launching, the GUI shown in figure 23 will appear on screen.

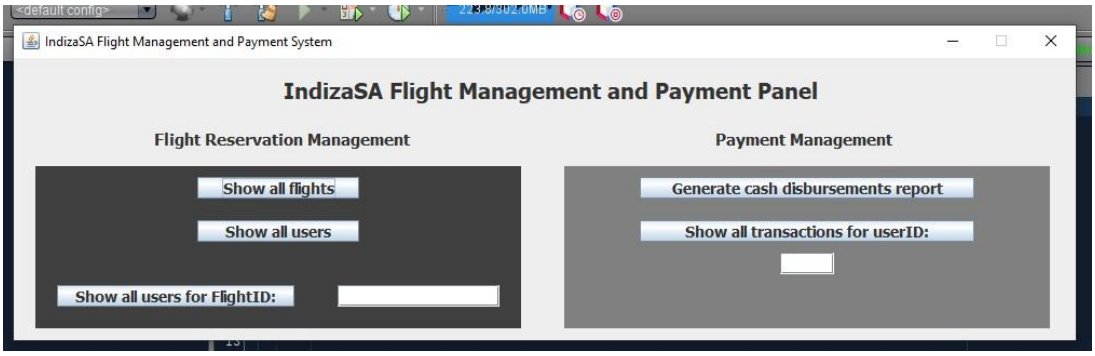


Figure 23

The output tab at the bottom of the IDE will display “Listening to connection requests” as shown in figure 24.

Figure 24

Android Studio

Navigate to the Android Studio developers' website and click on "Download Android Studio" (as shown in figure 25).

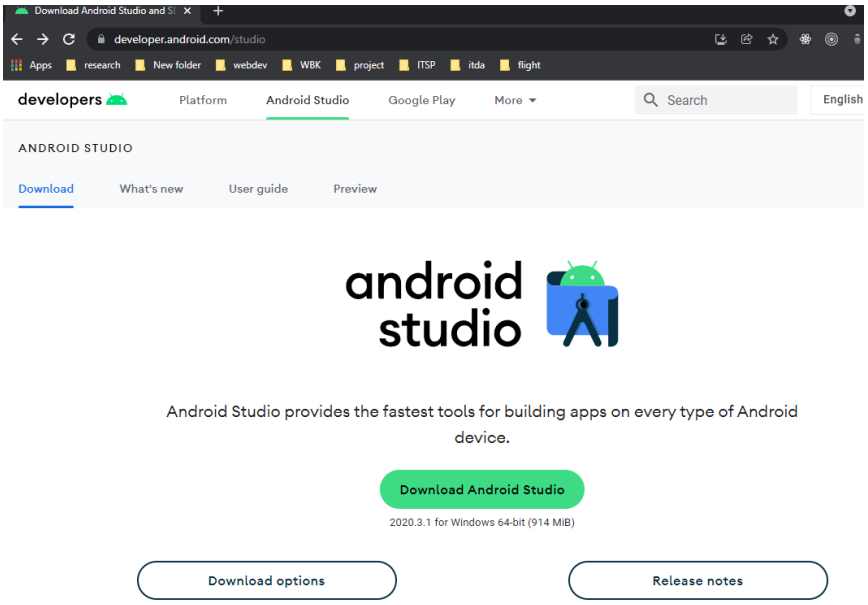


Figure 25

Make sure to install the virtual device during the installation process, as shown in figure 26.

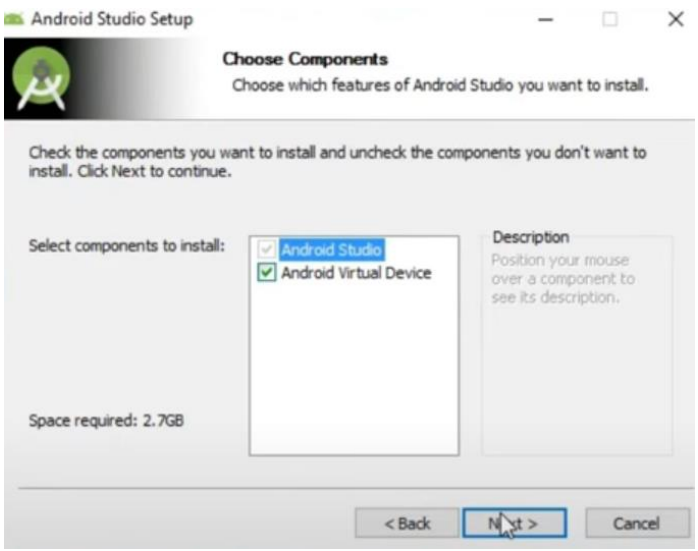


Figure 26

After the installation and opening of Android Studio, it is firstly required to install the virtual device that will run the application. Click on AVD manager to start this process, as shown in figure 27.

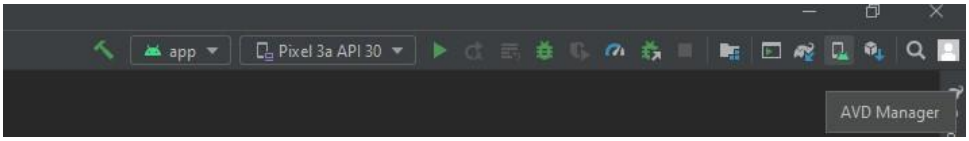


Figure 27

In the manager, click on “Create Virtual Device” in the bottom left corner of the manager, as shown in figure 28.

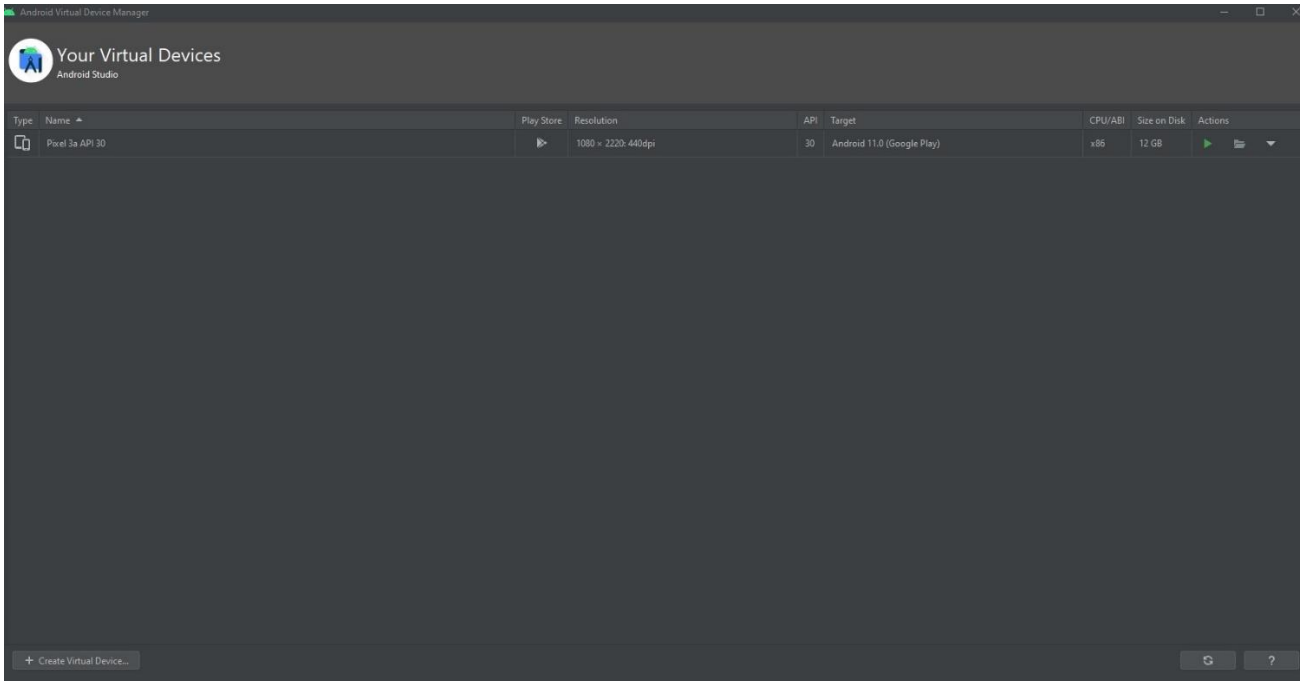


Figure 28

From the selection, choose the Pixel 3a, as shown in figure 29.

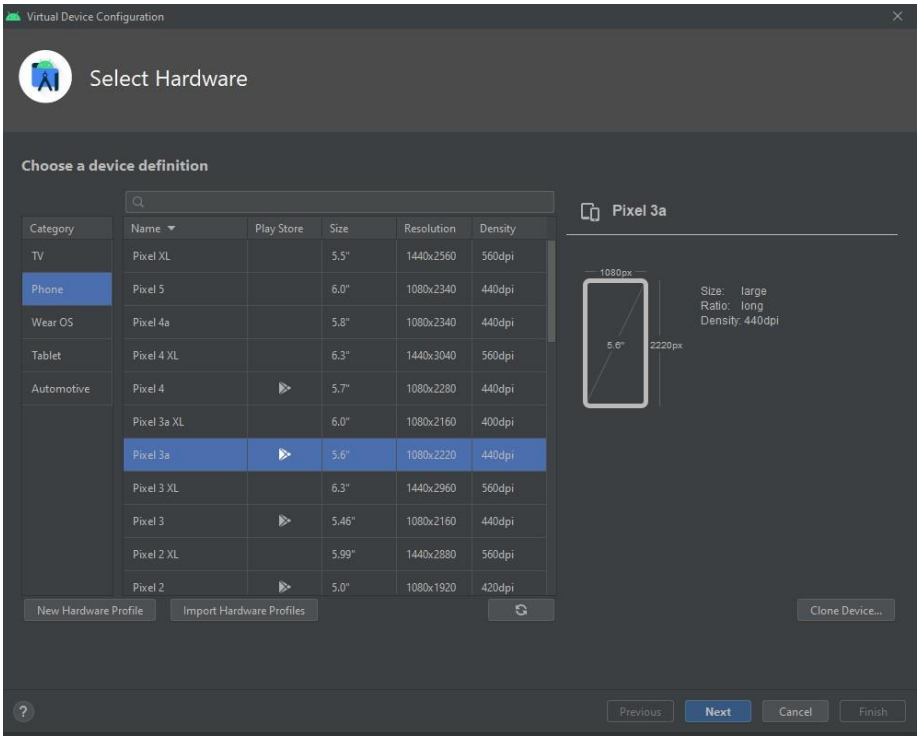


Figure 29

Select API level 30. This is crucial, as otherwise the application will not function properly (shown in figure 30).

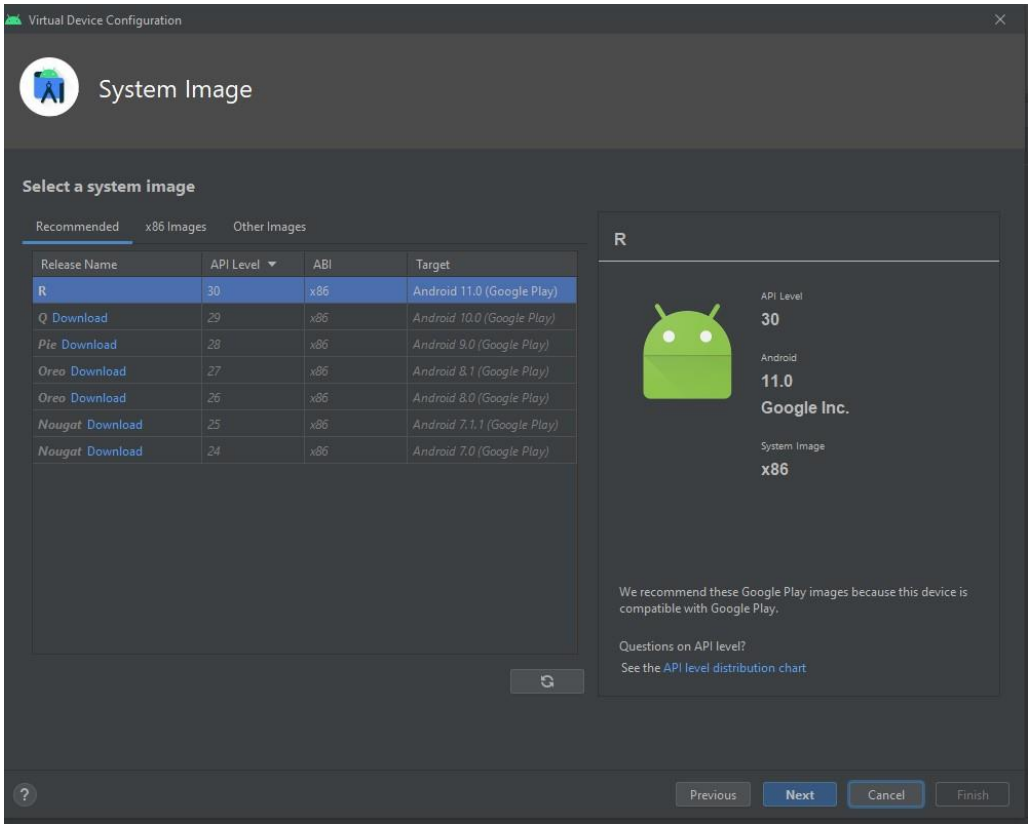


Figure 30

The VM can now be started by clicking on the “Play” button. Upon its first boot, the phone will have several downloads that will have to finish before further interaction is possible with the phone (as shown in figure 31).

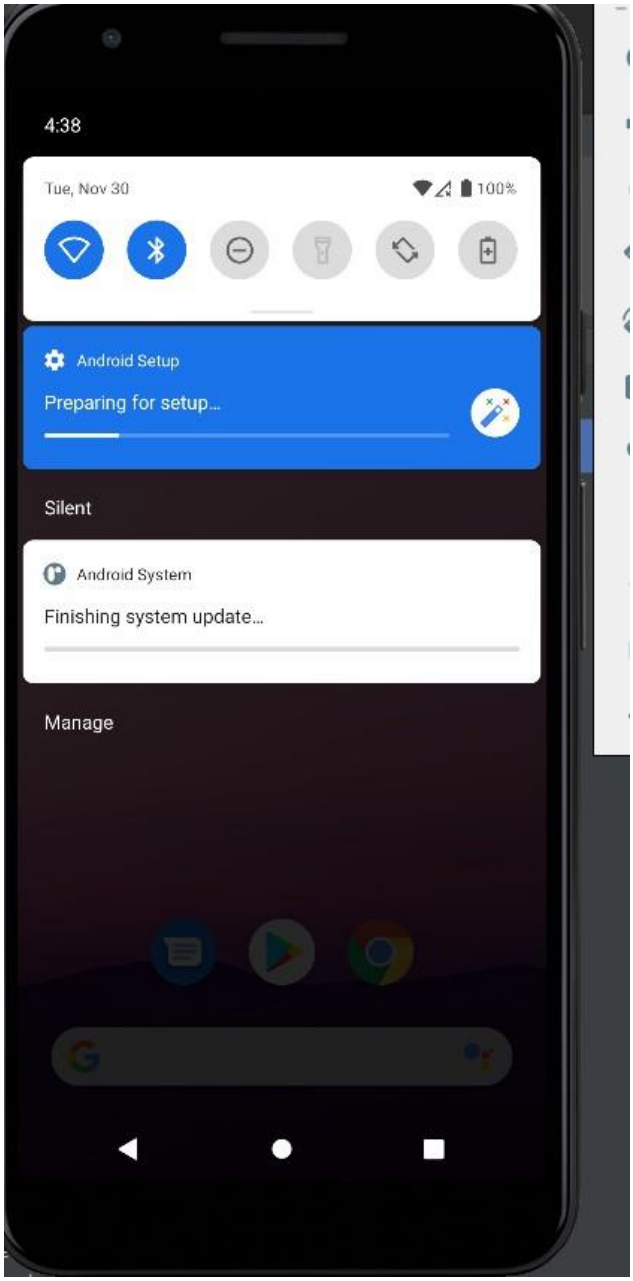


Figure 31

After the downloads, one can finish setting up the phone by clicking on “Finish setup” (as shown in figure 32).

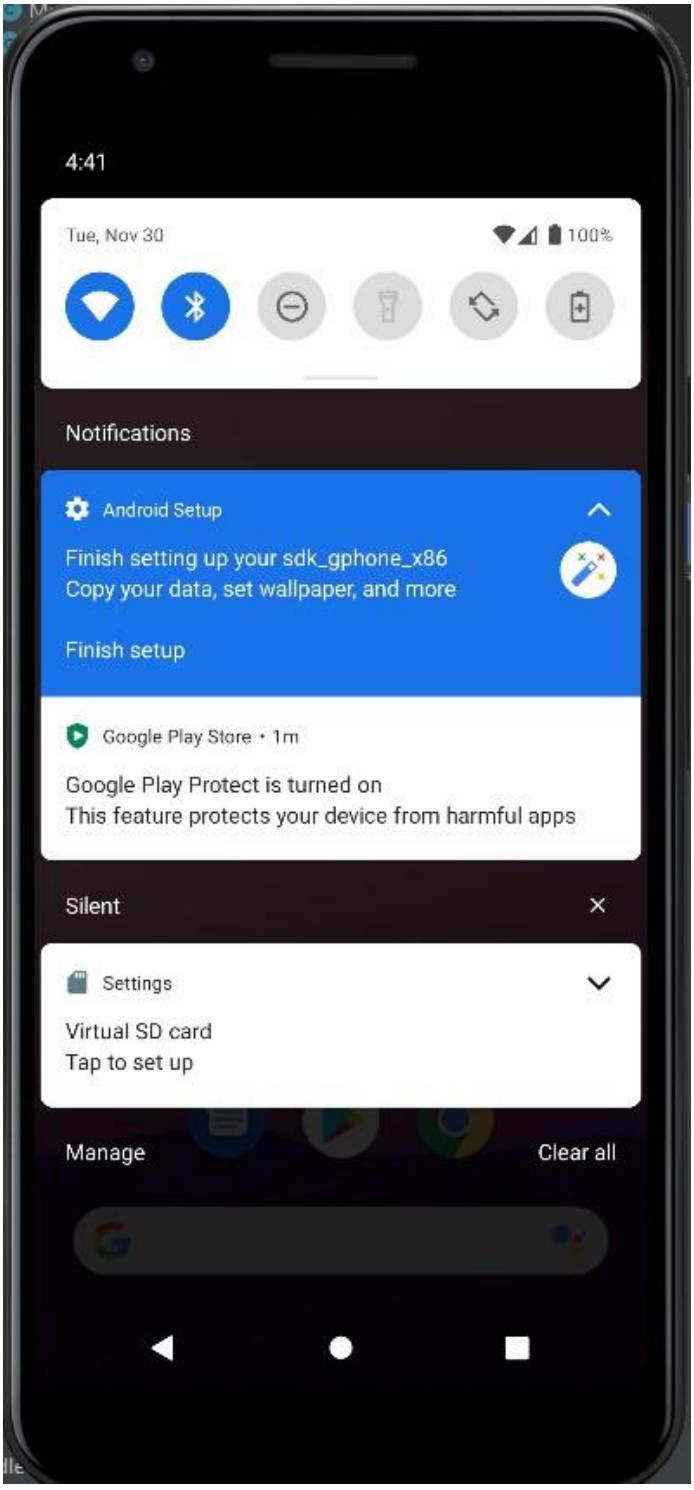


Figure 32

The VM is now ready to install the application, but the project firstly needs to be imported into Android Studio.

Select “File”, “New” and “Import project from the main menu (as shown in figure 33).

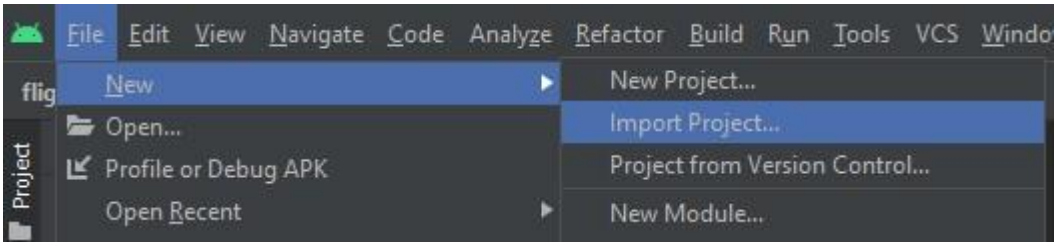


Figure 33

Navigate to the folder of the project and click on “OK” as shown in figure 34.

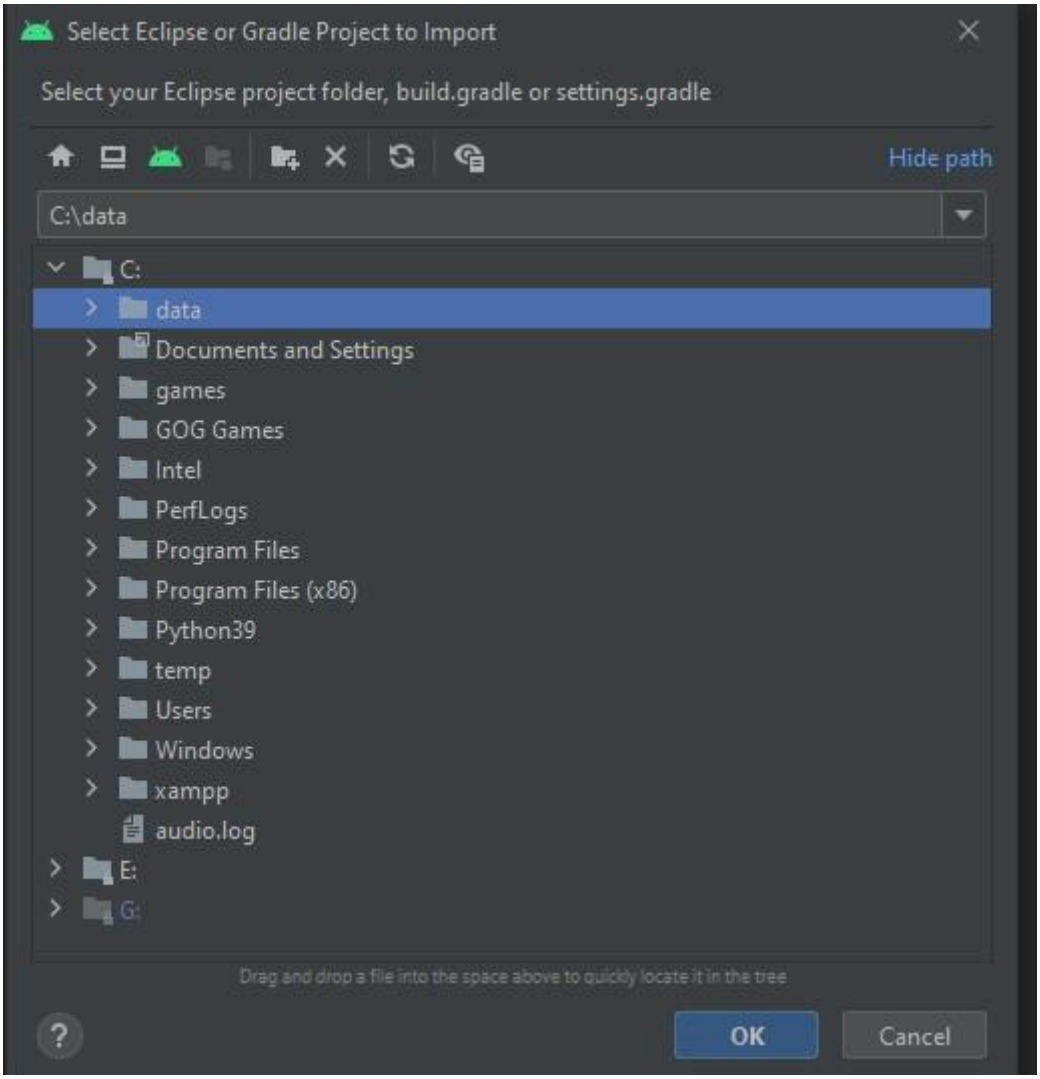


Figure 34

If “Gradle” is not installed, choose to import the project from an external model. Otherwise select “Create project from existing sources” and click on next (as shown in figure 35).

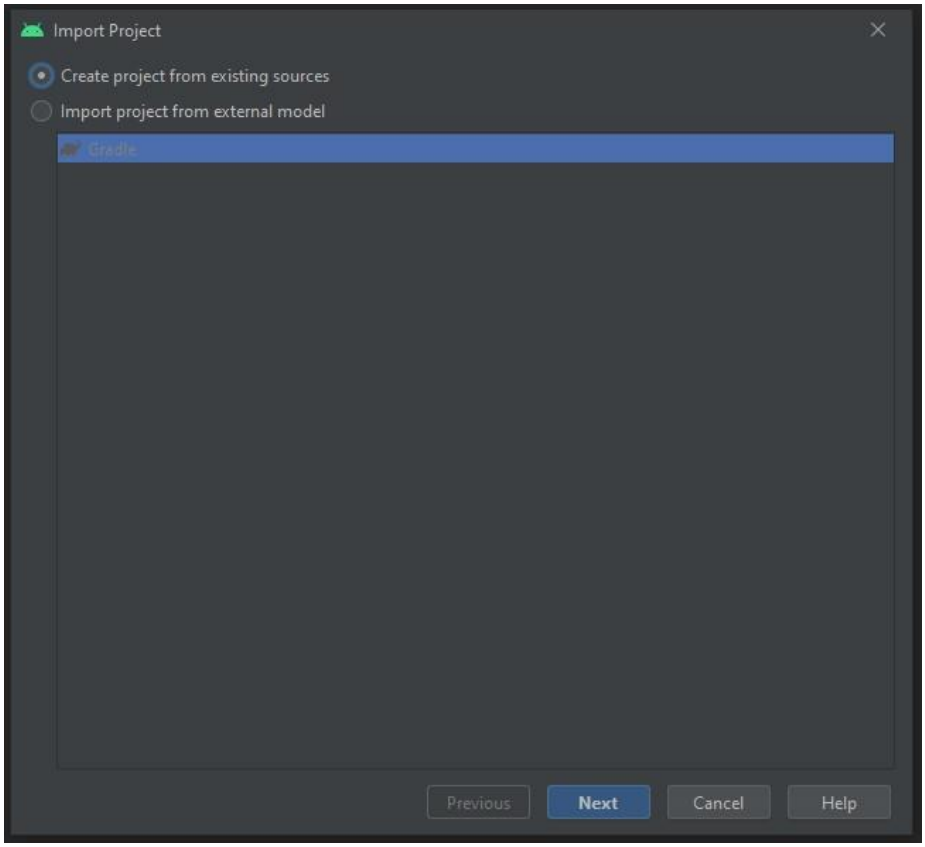


Figure 35

The project should include the following files, as shown in figure 36 and figure 37.

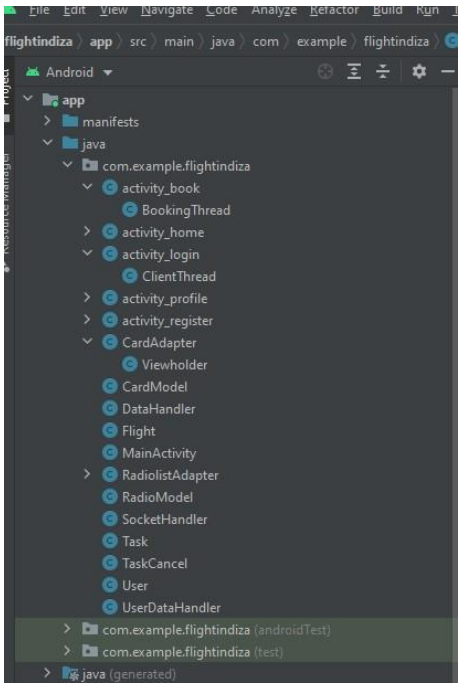


Figure 36

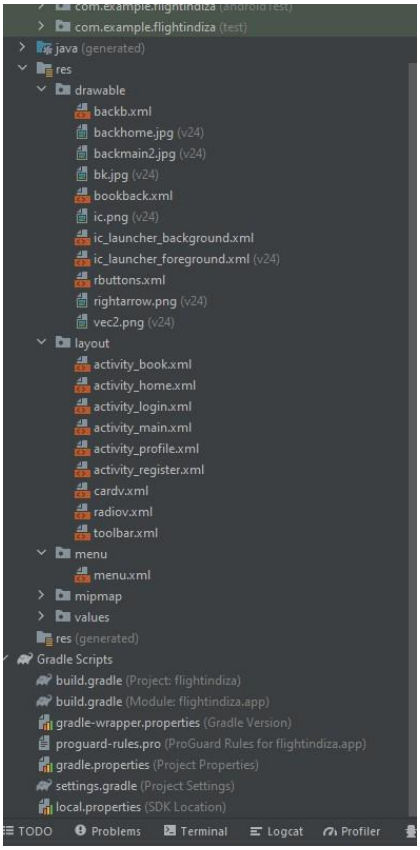


Figure 37

Before building and running the application, it is important to double check that the correct SDK is installed. Navigate to “Tools” in the main menu and select “SDK Manager” as shown in figure 38.

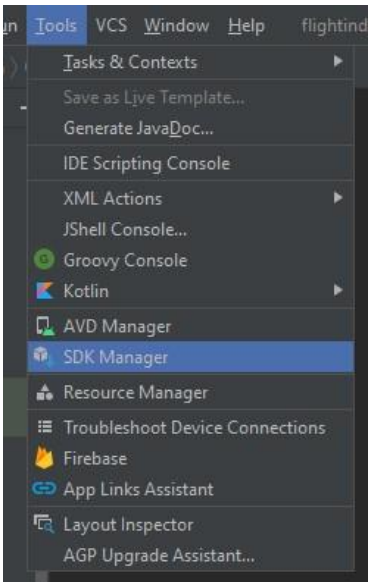


Figure 38

Click on SDK Platforms and ensure that the minimum of Android 4.4 (KitKat) is installed, as well as Android 11.0 (R), as shown in figure 39.

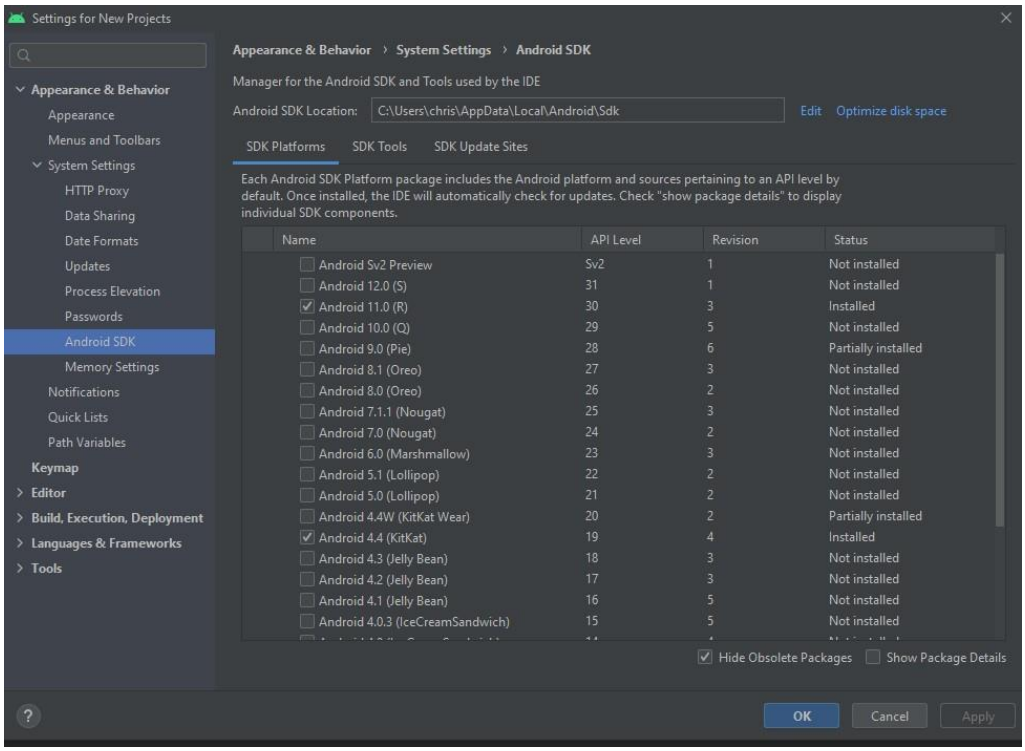


Figure 39

The project can now be “built” by selecting “Build” in the main menu and selecting “Rebuild Project” from the dropdown menu (as shown in figure 40).

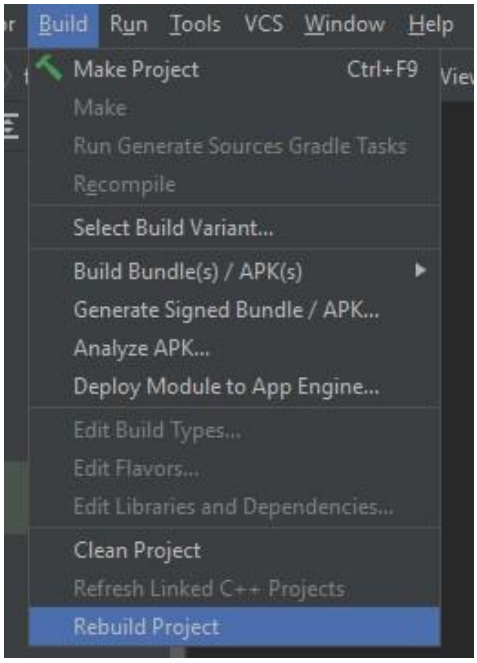


Figure 40

A pop-up will inform that the build was finished with 0 errors, as shown in figure 41.

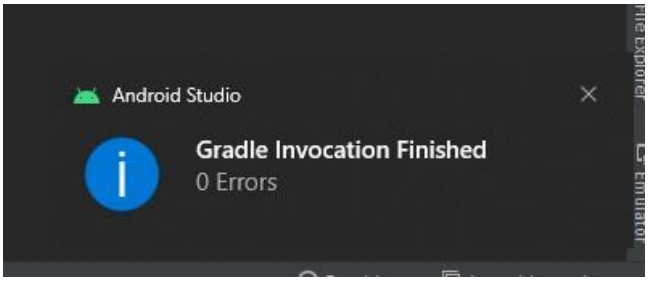


Figure 41

Recheck that:

1) The MySQL Server is running
2) The Java Server is running in NetBeans
3) The Android VM is open and on.

Click on the “Play” button. The application will install on the Android device and launch.

1.1 Registration and Login

Main:

When the application is launched, one is firstly presented with the main screen (as shown in figure 42). Two buttons, “Login” and “Register” are available to the user and will guide them to either activity.

Figure 42

On launching the server, a GUI frame will appear and the server output will print “Listening to connection requests” as shown in figure 43.

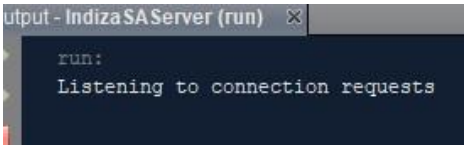


Figure 43

Registration:

Clicking on the “Register” button will guide the user to the register screen shown in figure 44.

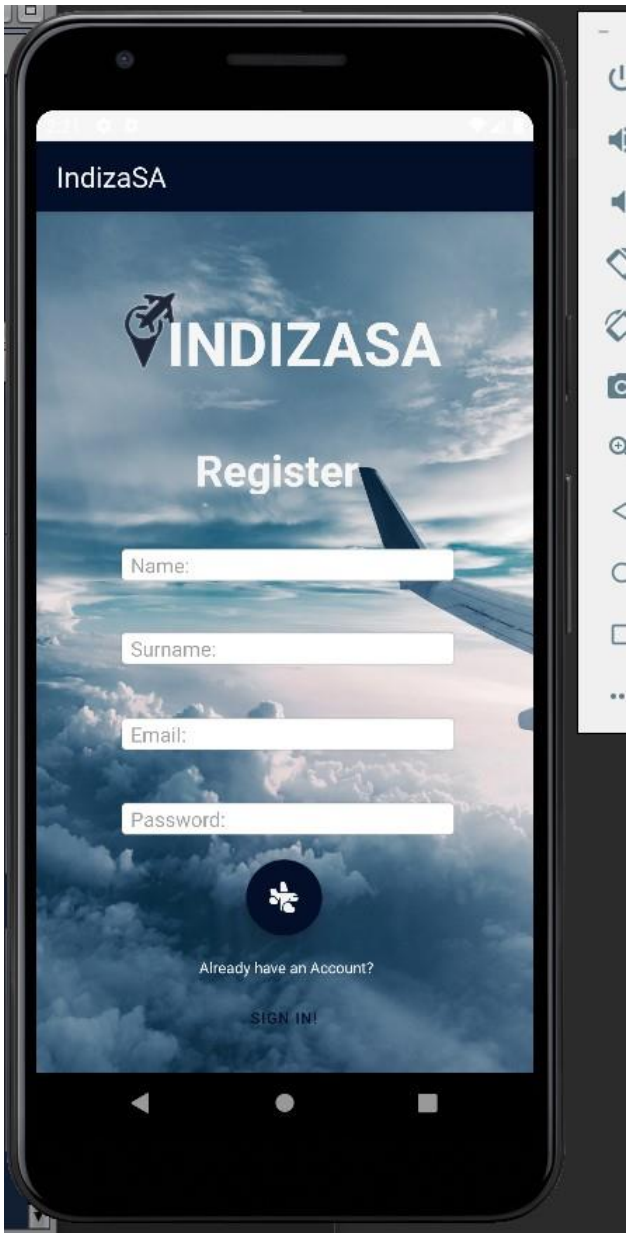


Figure 44

Four fields are presented to the user that requires input from them. Underneath the main button, the text “Already have an Account” appear, as shown in figure 45. Clicking on the “SIGN IN!” button will redirect the user to the login screen.

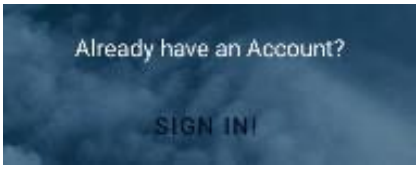


Figure 45

Server-side, one can choose to “Show all users” to see a list of registered users on the system, as shown in figure 46. Note that the last user registered had the user ID of 40.

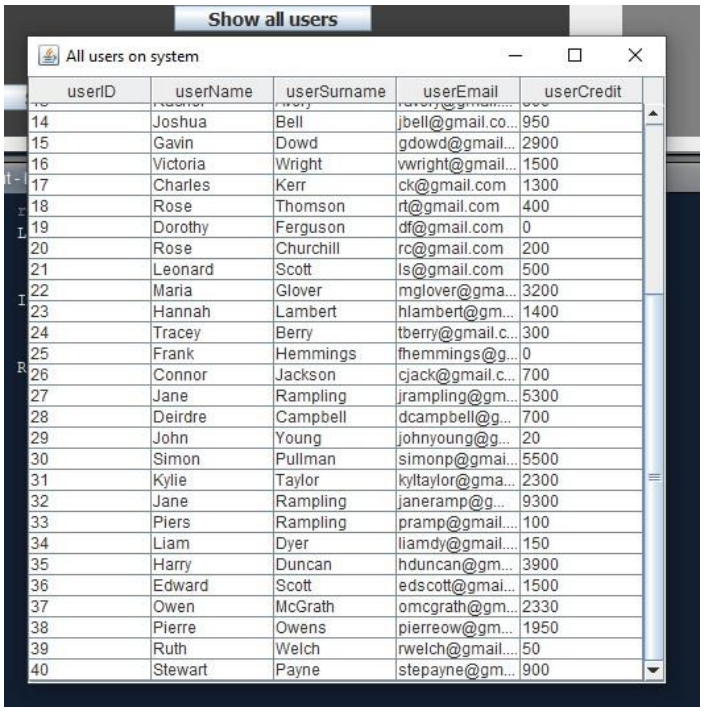


Figure 46

On clicking the registration button, the server will output “Instructions being received” and “Register” – indicating that the server acknowledges that the registration process has begun. This is shown in figure 47.

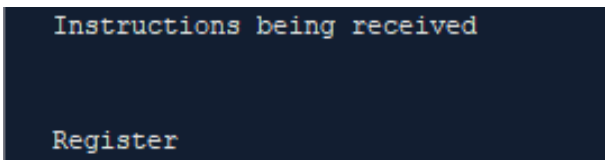


Figure 47

Validation is used for the register and login modules. If the user were to click on the register button without having filled in any fields, or just their name and surname, a pop-up message will instruct them to not leave any fields empty (as shown in figure 48).

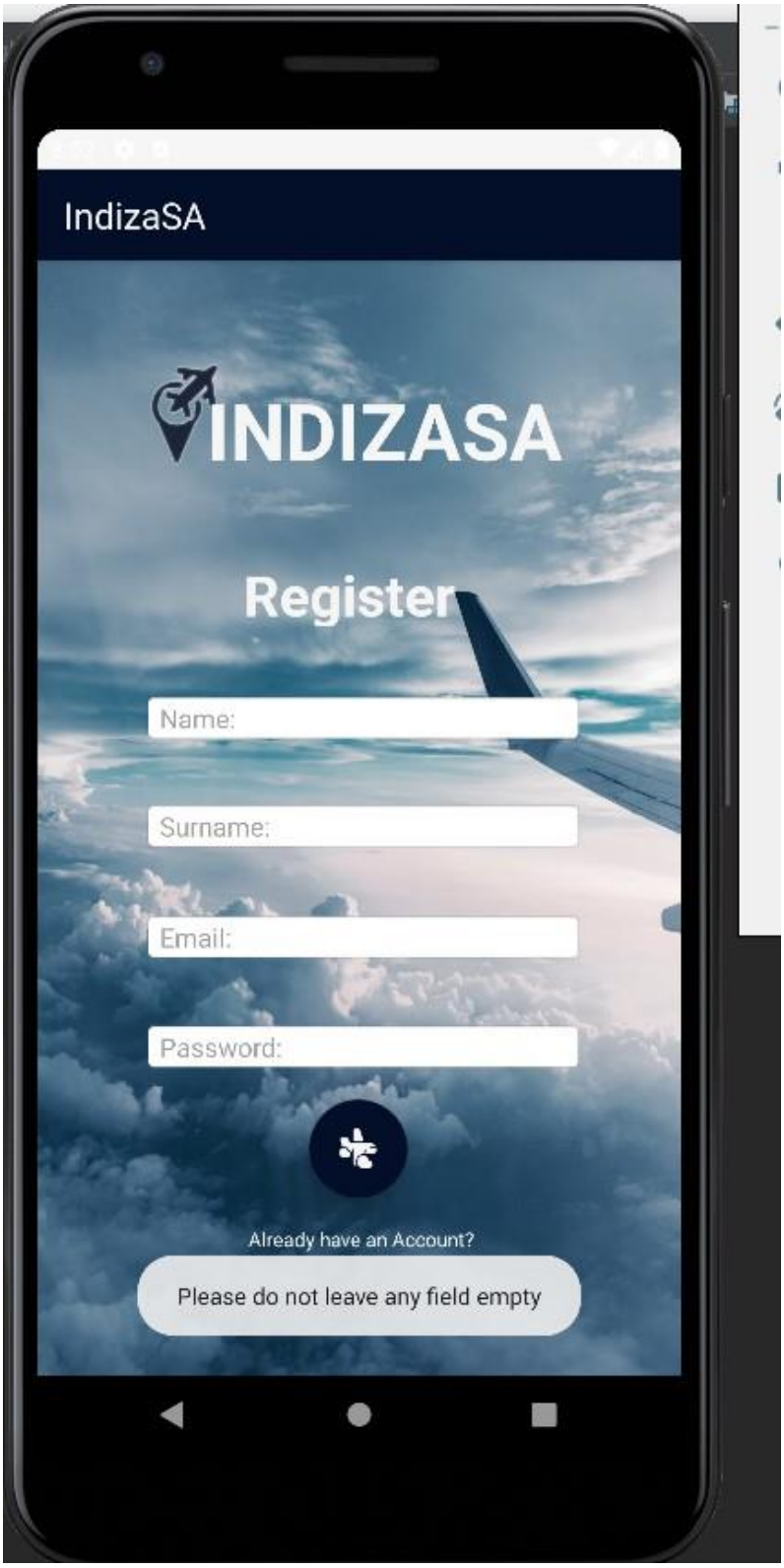


Figure 48

If the user enters an invalid email address or leaves the field empty, another pop-up message will indicate this to the user (shown in figure 49).

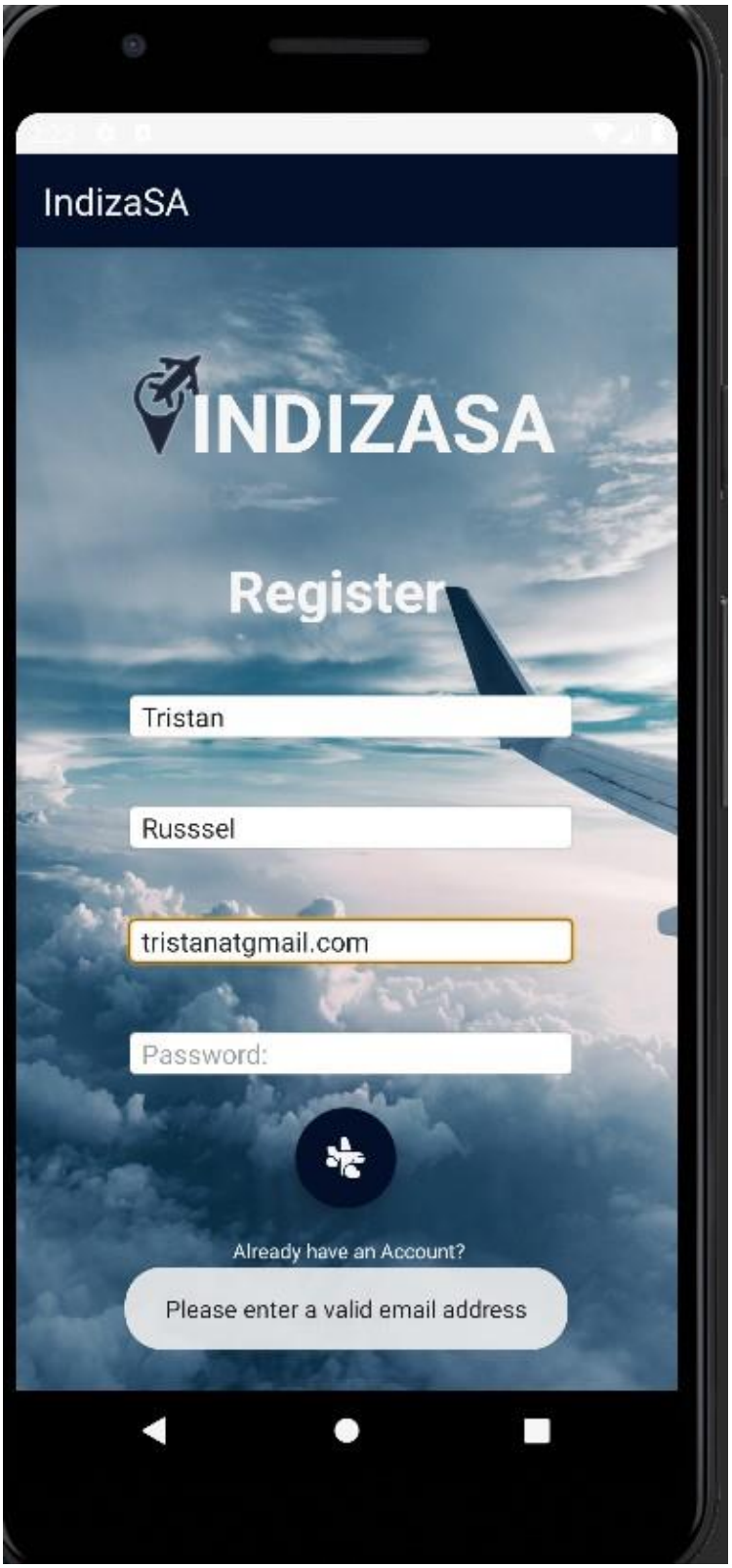


Figure 49

Finally, the password field is also validated. A message is displayed if the user leaves the field empty or does not enter a sufficiently secure password (shown in figure 50).

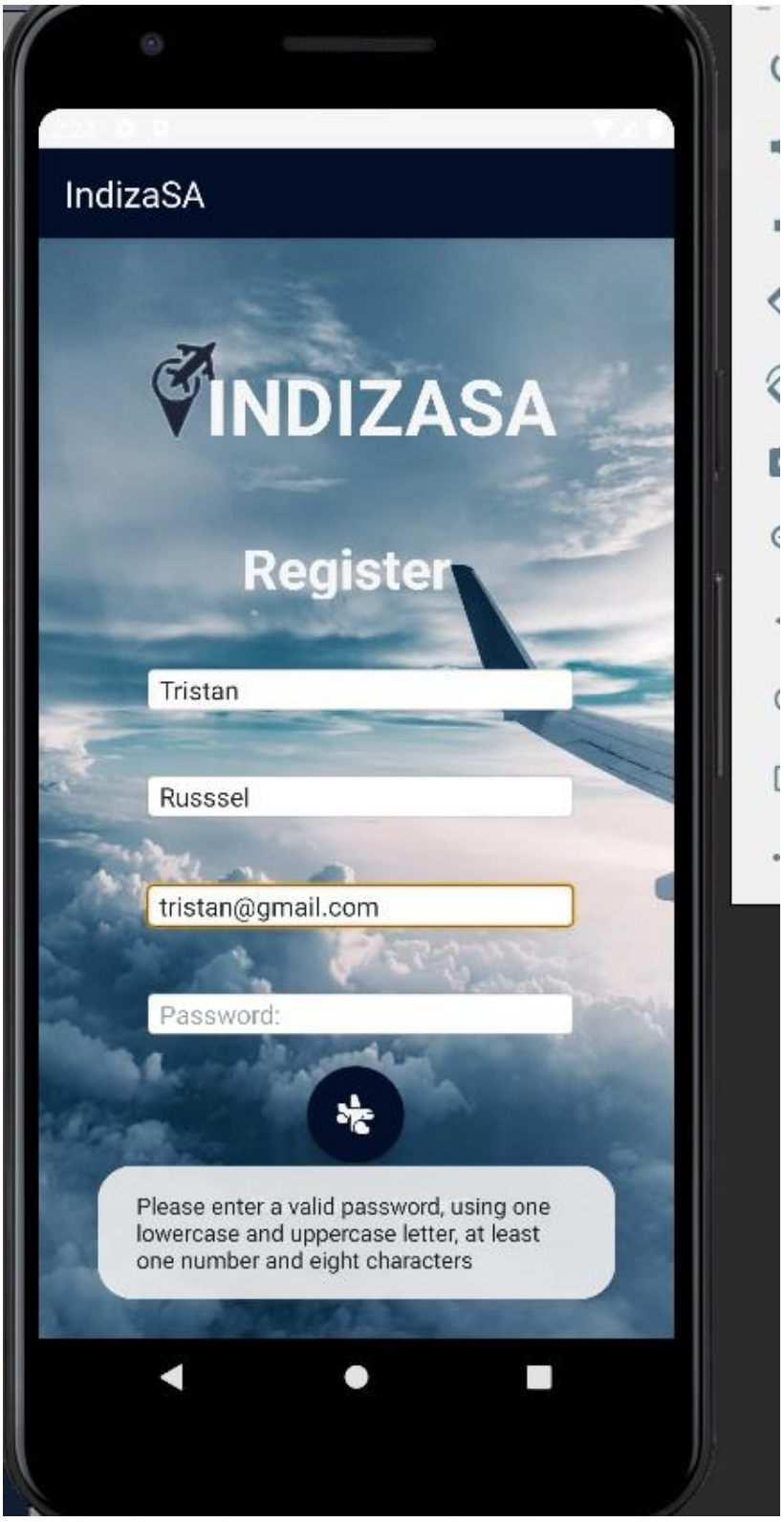
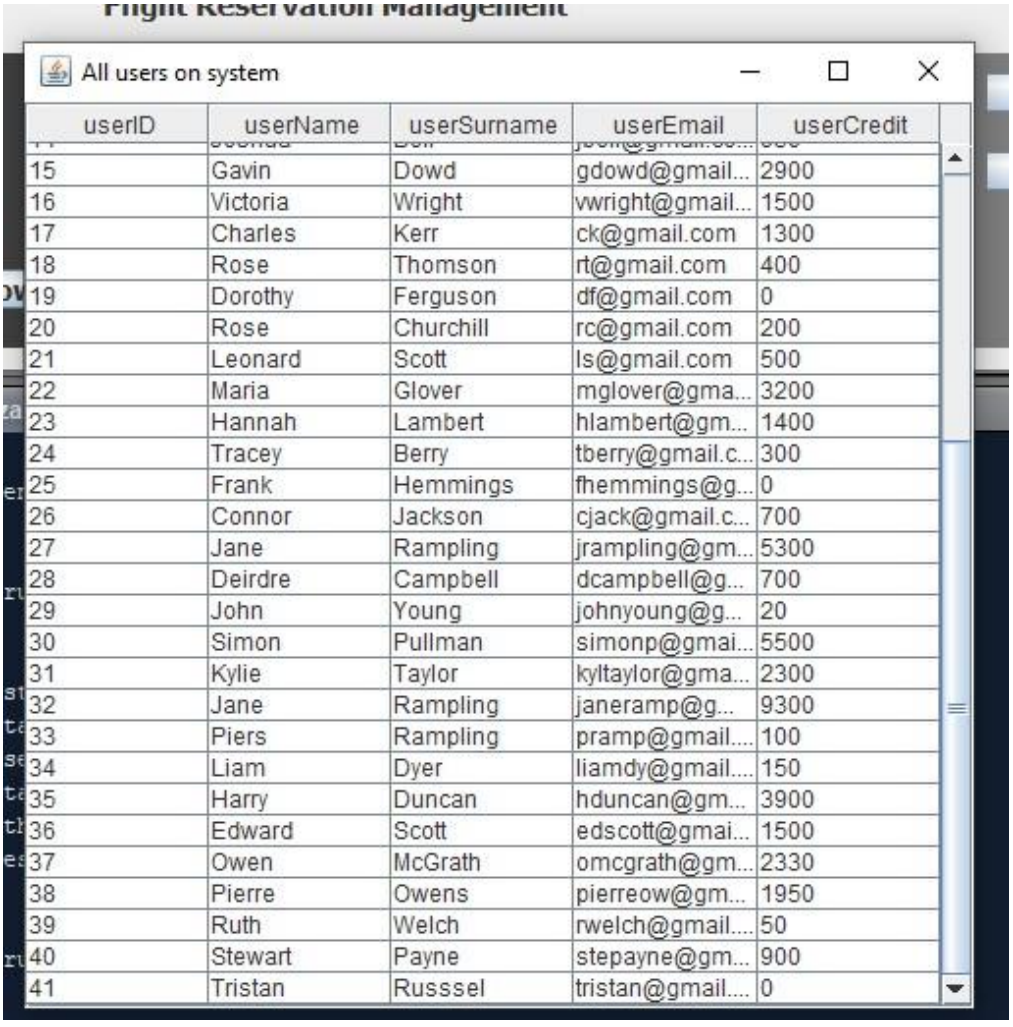


Figure 50

If all the fields are filled in correctly, the user can click on the register button, as shown in figure 51.

Figure 51

It is important to note that the users on the system are updated on the server-side. The last user to be registered had an ID of 40, whilst the new user was assigned a user ID of 41 (shown in figure 52). The user's details filled in (compare with figure 50), also appear correctly in the system (shown in figure 52).



userID	userName	userSurname	userEmail	userCredit
15	Gavin	Dowd	gdowd@gmail...	2900
16	Victoria	Wright	vwright@gmail...	1500
17	Charles	Kerr	ck@gmail.com	1300
18	Rose	Thomson	rt@gmail.com	400
19	Dorothy	Ferguson	df@gmail.com	0
20	Rose	Churchill	rc@gmail.com	200
21	Leonard	Scott	ls@gmail.com	500
22	Maria	Glover	mglover@gma...	3200
23	Hannah	Lambert	hlambert@gm...	1400
24	Tracey	Berry	tberry@gmail.c...	300
25	Frank	Hemmings	fhemmings@g...	0
26	Connor	Jackson	cjack@gmail.c...	700
27	Jane	Rampling	jrampling@gm...	5300
28	Deirdre	Campbell	dcampbell@g...	700
29	John	Young	johnyoung@g...	20
30	Simon	Pullman	simonp@gmai...	5500
31	Kylie	Taylor	kyltaylor@gma...	2300
32	Jane	Rampling	janeramp@g...	9300
33	Piers	Rampling	pramp@gmail....	100
34	Liam	Dyer	liamdy@gmail....	150
35	Harry	Duncan	hduncan@gm...	3900
36	Edward	Scott	edscott@gmai...	1500
37	Owen	McGrath	omcgrath@gm...	2330
38	Pierre	Owens	pierreow@gm...	1950
39	Ruth	Welch	rwelch@gmail....	50
40	Stewart	Payne	stepayne@gm...	900
41	Tristan	Russsel	tristan@gmail....	0

Figure 52

The server output firstly indicated that it acknowledged that the registration process had begun. It outputs the user inputs and then also displays a message “success” which is passed to the client through a socket connection to indicate that the registration was a success (shown in figure 53).

Similarly, if the registration failed, the server would send a message of “failure” to the client. Upon successful registration, the user is redirected to the login screen.

Figure 53

Login:

The login screen is reached by clicking on the “Login” button on the main screen, clicking on the “SIGN IN” button on the register page, or upon successful registration (shown in figure 54).

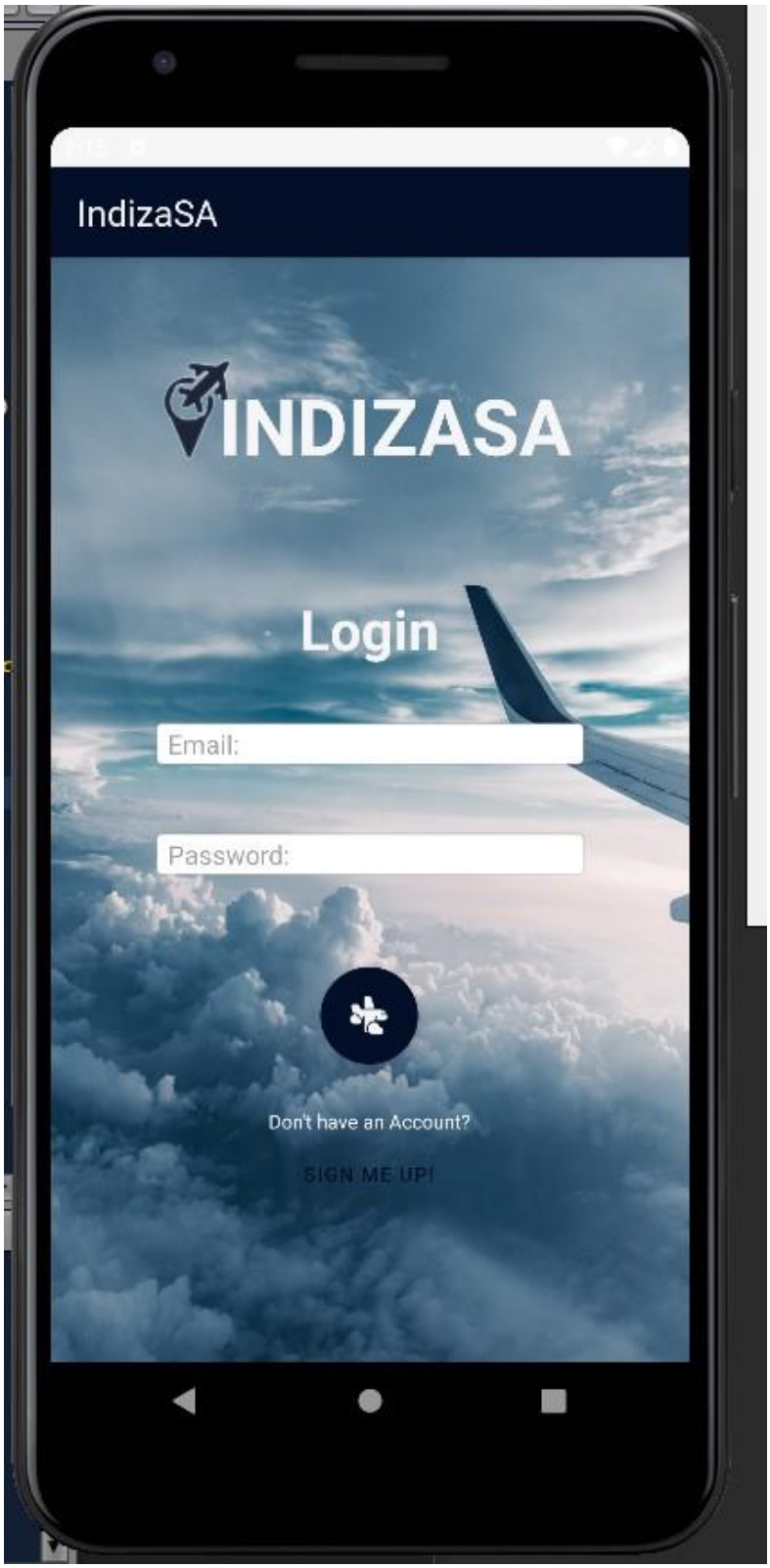


Figure 54

Validation is used to firstly ensure that the user is registered and entered the correct password and secondly to inform the user of a failed login if the entered the incorrect username or password or left and of the fields empty (shown in figure 55).

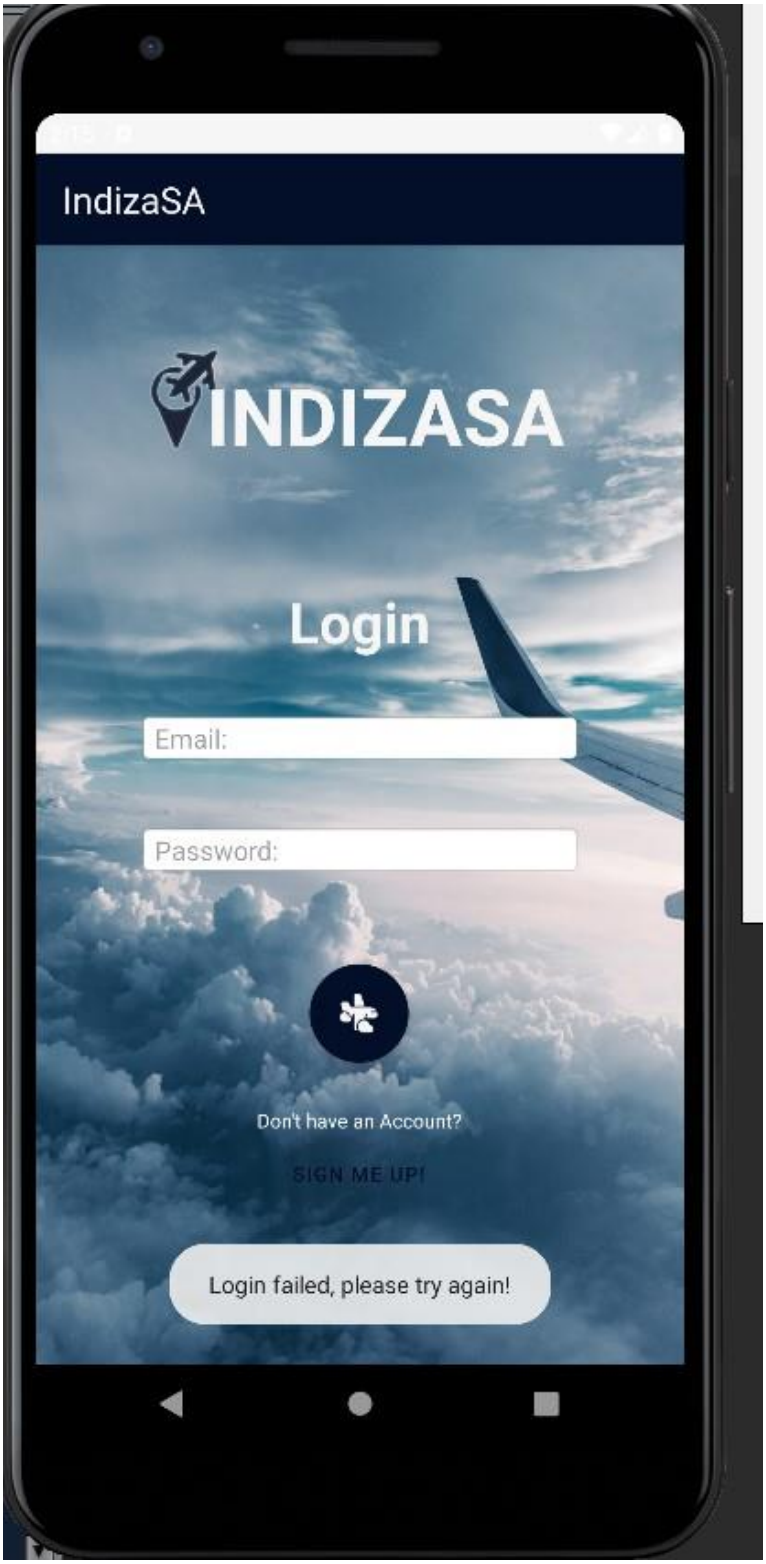


Figure 55

The server output firstly indicates that the server acknowledges that the login process has begun. It indicates the entered email and password and furthermore sends the client the result of the login process – “denied” or “success”. If the login has been denied, the server loops back to receiving instructions initially given (“Login”), as shown in figure 56.

```
Login
User email is cv
User password is d
Server sending verification result: denied


Instructions being received


Login
```

Figure 56

If the user enters the correct details (as shown in figure 57), they will be redirected to the home screen and a pop-up message reading “Welcome” will appear (as shown in figure 58)/

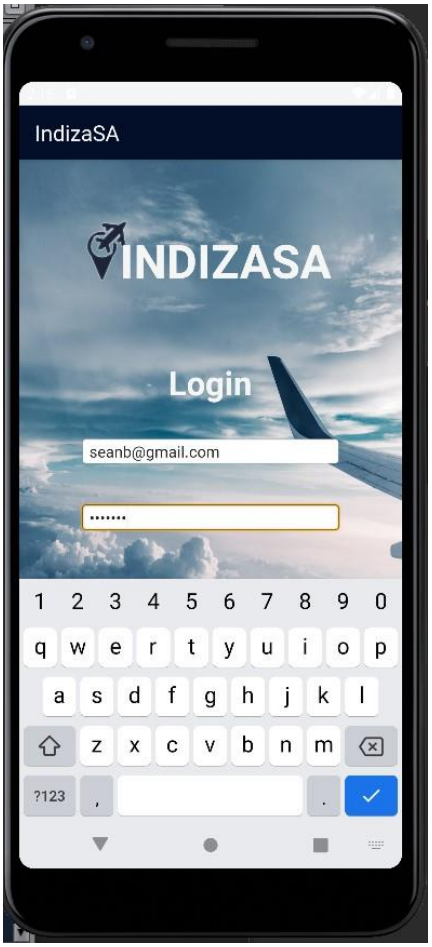


Figure 57

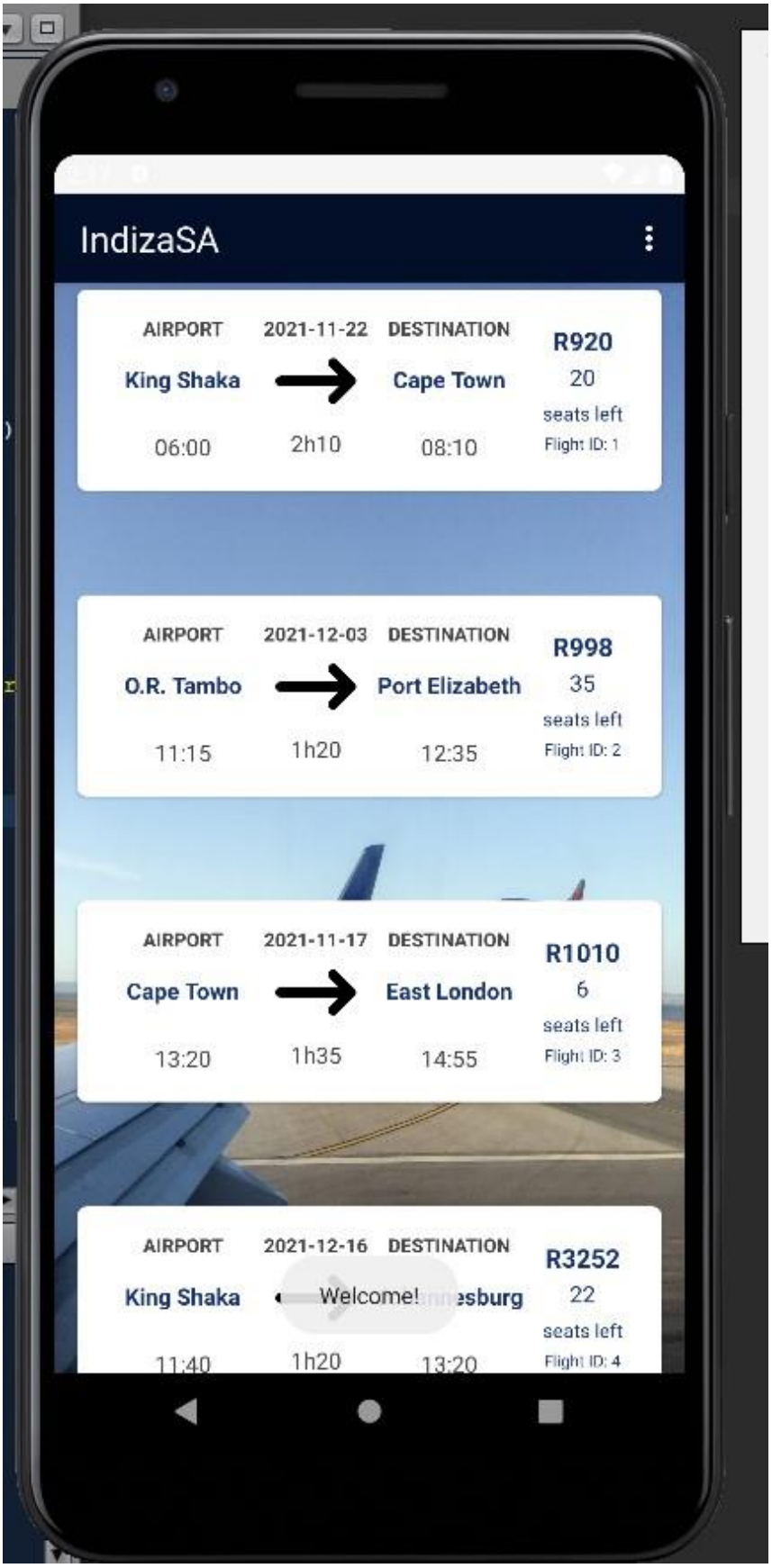

IndizaSA
AIRPORT
2021-11-22
DESTINATION
King Shaka
Cape Town
R920
20
seats left
06:00
2h10
08:10
Flight ID: 1
AIRPORT
2021-12-03
DESTINATION
O.R. Tambo
Port Elizabeth
R998
35
seats left
11:15
1h20
12:35
Flight ID: 2
AIRPORT
2021-11-17
DESTINATION
Cape Town
East London
R1010
6
seats left
13:20
1h35
14:55
Flight ID: 3
AIRPORT
2021-12-16
DESTINATION
King Shaka
Welcome!
R3252
22
seats left
11:40
1h20
13:20
Flight ID: 4

Figure 58

The server output indicates that the login was approved and that the newly received instructions from the client indicates that the “Home” process has begun (as shown in figure 59).

Figure 59

The database “user_table” is shown in figure 60. It shows that dummy data was inserted and shows the user’s email address and password. This correlates with the server output.

Figure 60

1.2 Home – List of flights

Upon successful login, the user is presented with the home screen (shown in figures 61-63). Twenty flights are listed in a card format. The card gives the essential details of the flight and can be clicked to move to the booking process.

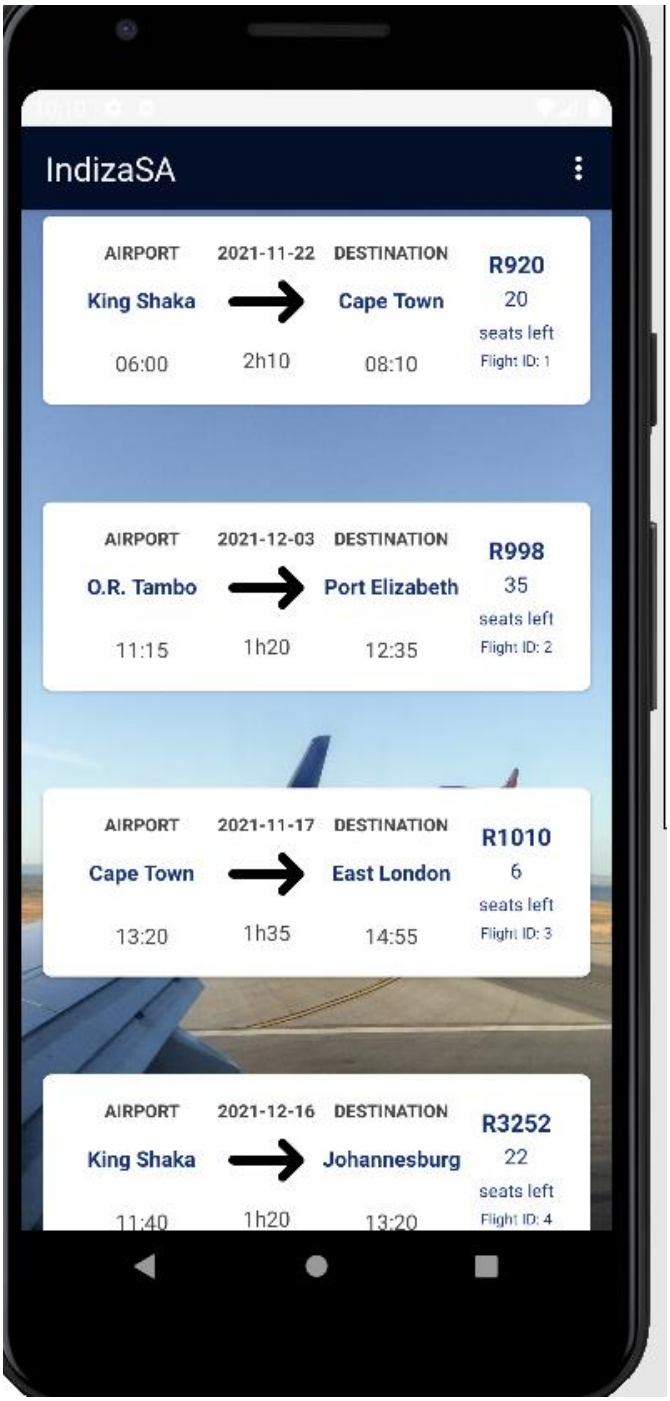


Figure 61

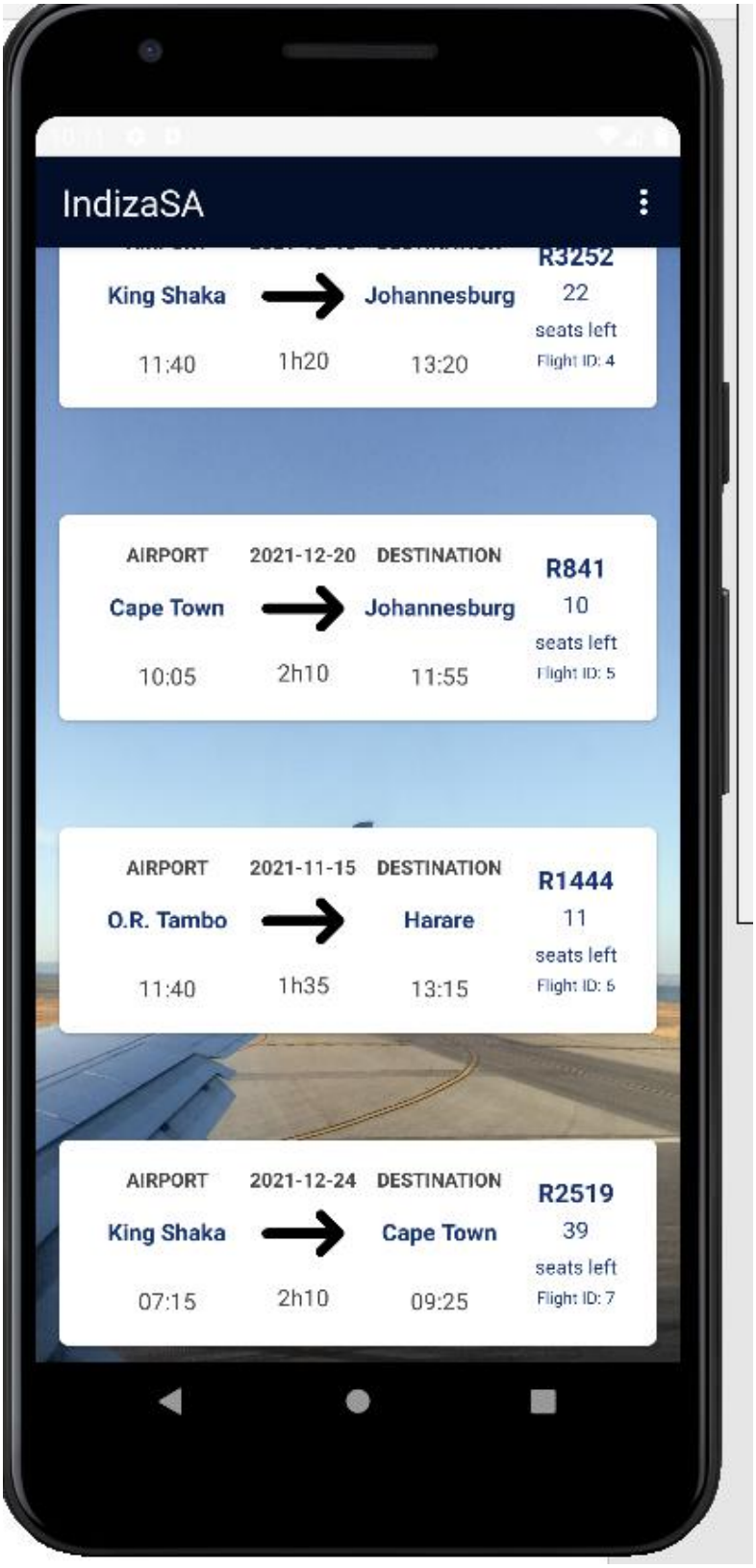

IndizaSA
King Shaka
Johannesburg
R3252
22
seats left
11:40
1h20
13:20
Flight ID: 4
AIRPORT
2021-12-20
DESTINATION
Cape Town
Johannesburg
R841
10
seats left
10:05
2h10
11:55
Flight ID: 5
AIRPORT
2021-11-15
DESTINATION
O.R. Tambo
Harare
R1444
11
seats left
11:40
1h35
13:15
Flight ID: 6
AIRPORT
2021-12-24
DESTINATION
King Shaka
Cape Town
R2519
39
seats left
07:15
2h10
09:25
Flight ID: 7

Figure 62

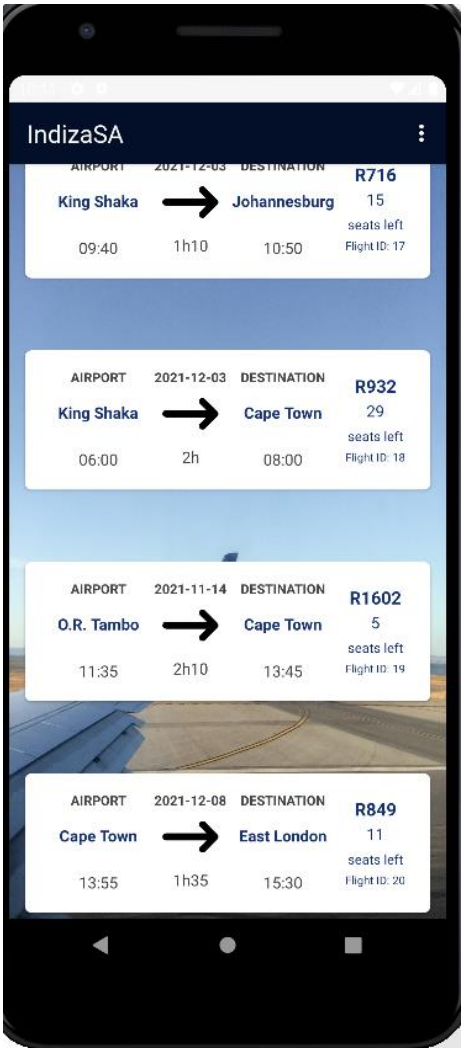


Figure 63

Clicking on a flight that has zero seats left, as shown in figure 64, will result in a popup message indicating to the user that the flight has been fully booked (as shown in figure 65)

Figure 64

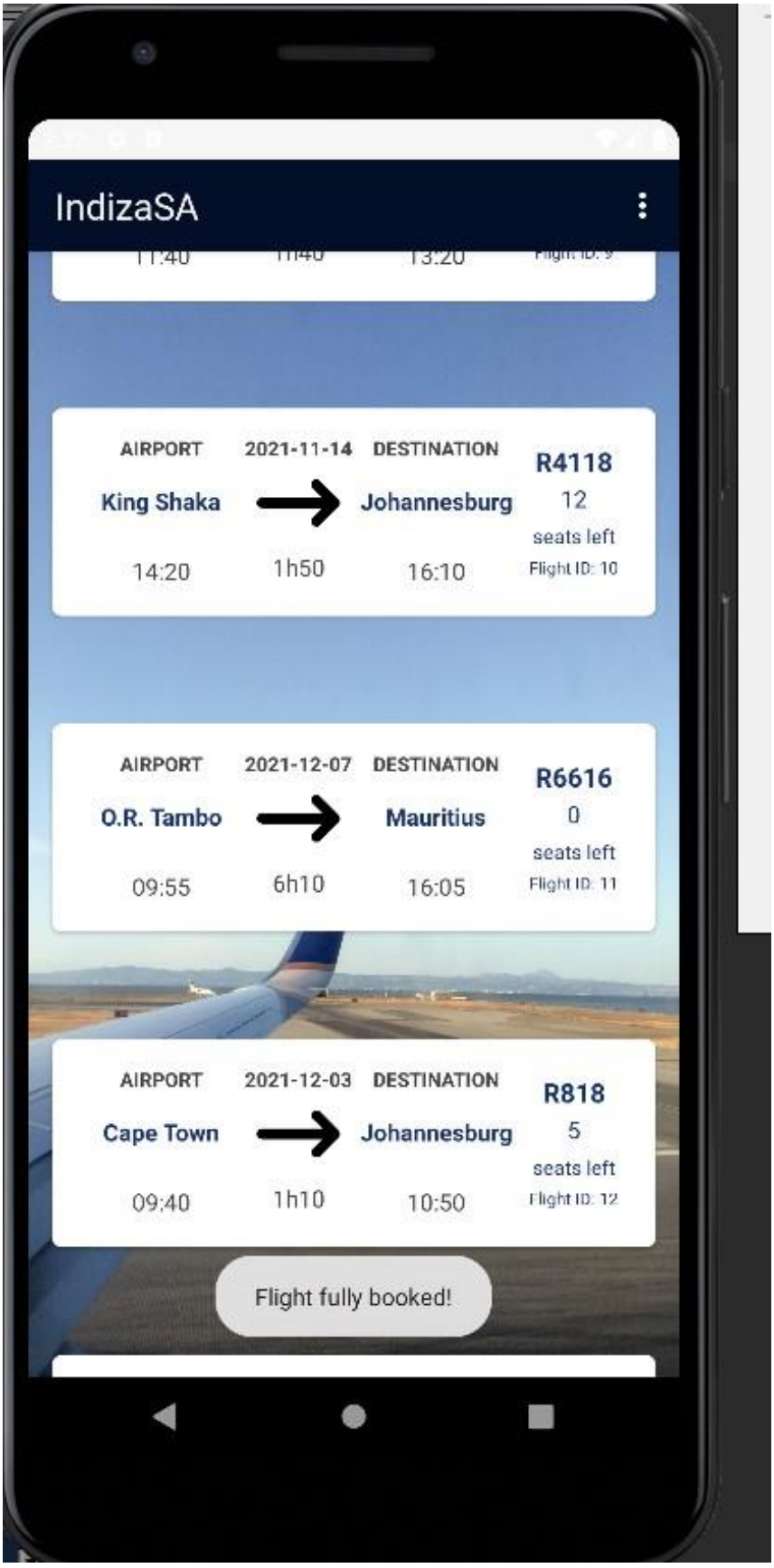

IndizaSA
AIRPORT
2021-11-14
DESTINATION
King Shaka
Johannesburg
R4118
12
seats left
14:20
1h50
16:10
Flight ID: 10
AIRPORT
2021-12-07
DESTINATION
O.R. Tambo
Mauritius
R6616
0
seats left
09:55
6h10
16:05
Flight ID: 11
AIRPORT
2021-12-03
DESTINATION
Cape Town
Johannesburg
R818
5
seats left
09:40
1h10
10:50
Flight ID: 12
Flight fully booked!

Figure 65

The database “flight_table” consists of twenty entered flights. The flight with ID 11 has zero “numSeatsLeft” and thus correlates with the client-side card (as shown in figure 66).

```
INSERT INTO `flight_table` (`flightID`, `destination`, `price`, `dateOf`, `deptTime`, `duration`, `arrivalTime`, `numSeatsLeft`, `airportID`) VALUES
(1, "Cape Town", 920, "2021-11-22", "06:00", "2h10", "08:10", 20, 2),
(2, "Port Elizabeth",  998, "2021-12-03", "11:15", "1h20", "12:35", 35, 1),
(3, "East London",  1010, "2021-11-17", "13:20", "1h35", "14:55", 6, 3),
(4, "Johannesburg",  3252, "2021-12-16", "11:40", "1h20", "13:20", 22, 2),
(5, "Johannesburg",  841, "2021-12-20", "10:05", "2h10", "11:55", 10, 3),
(6, "Harare", 1444, "2021-11-15", "11:40", "1h35", "13:15", 11, 1),
(7, "Cape Town", 2519, "2021-12-24", "07:15","2h10", "09:25",  39, 2),
(8, "Durban", 1804, "2021-11-24","06:40", "2h", "08:40", 11, 1),
(9, "Johannesburg", 3152, "2021-12-09", "11:40", "1h40", "13:20", 15, 3),
(10, "Johannesburg", 4118, "2021-11-14", "14:20","1h50", "16:10", 12, 2),
(11, "Mauritius", 6616, "2021-12-07", "09:55", "6h10", "16:05",  0, 1),
(12, "Johannesburg", 818, "2021-12-03", "09:40", "1h10", "10:50",  5, 3),
(13, "Cape Town", 1117, "2021-11-17", "14:30", "2h", "16:30",  25, 1),
(14, "Mauritius", 4639, "2021-12-02", "17:10", "2h30", "19:40",  35, 1),
(15, "Victoria Falls", 2232, "2021-11-27", "13:45", "1h35", "15:20", 2, 1),
(16, "Victoria Falls", 2014, "2021-11-20", "11:30", "1h37", "13:07",  3, 1),
(17, "Johannesburg", 716, "2021-12-03", "09:40", "1h10", "10:50", 15, 2),
(18, "Cape Town", 932, "2021-12-03", "06:00", "2h", "08:00", 30, 2),
(19, "Cape Town", 1602, "2021-11-14", "11:35", "2h10", "13:45", 5, 1),
(20, "East London", 849, "2021-12-08", "13:55", "1h35", "15:30",  11, 3);
```

Figure 66

1.3 Booking and Cancellation

Booking:

Clicking on a flight card in the home screen will redirect the user to the profile screen (as shown in figure 67). The user is presented with all the flight details in table form. The user can click on the “Back” button to be redirected to the home screen, or click on the “Book” button to book the flight.

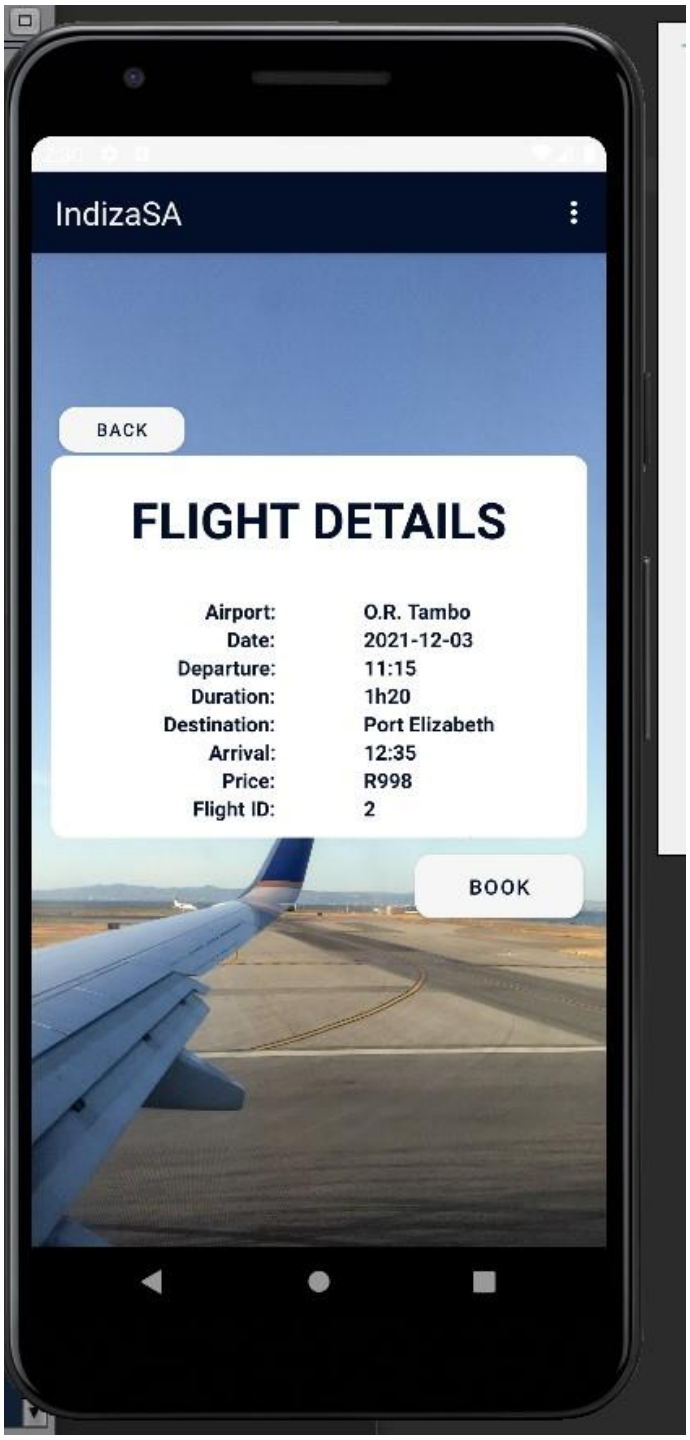


Figure 67

The user will be informed if the booking has failed per pop-up notification. However, this is covered in 2.4, as it links with the payment module and needs detailed coverage. Upon successful booking, the user will be presented with a notification indicating that their booking was successful (shown in figure 68).

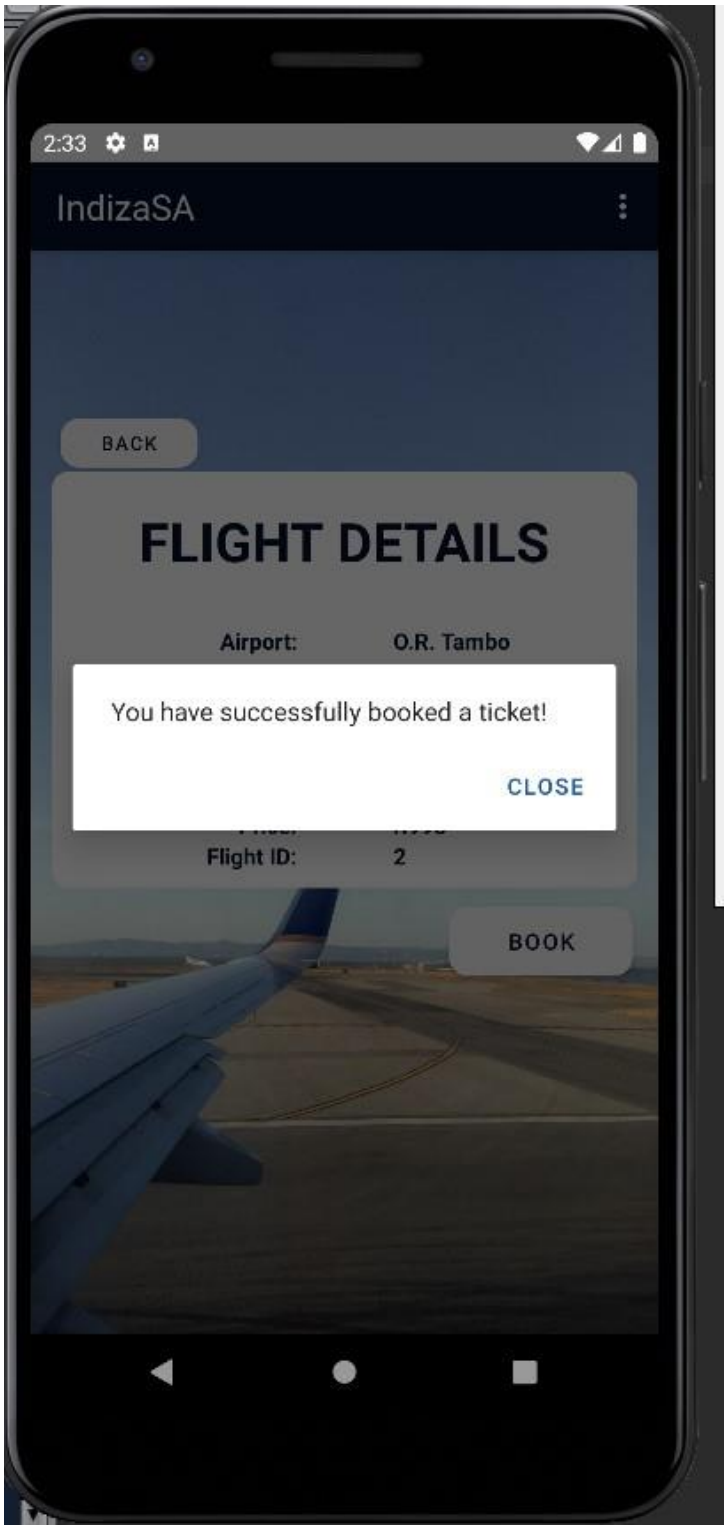


Figure 68

Upon a successful booking, the user will be redirected to the profile screen. The booked flight will be displayed in card format under the heading “Bookings made” (shown in figure 69).

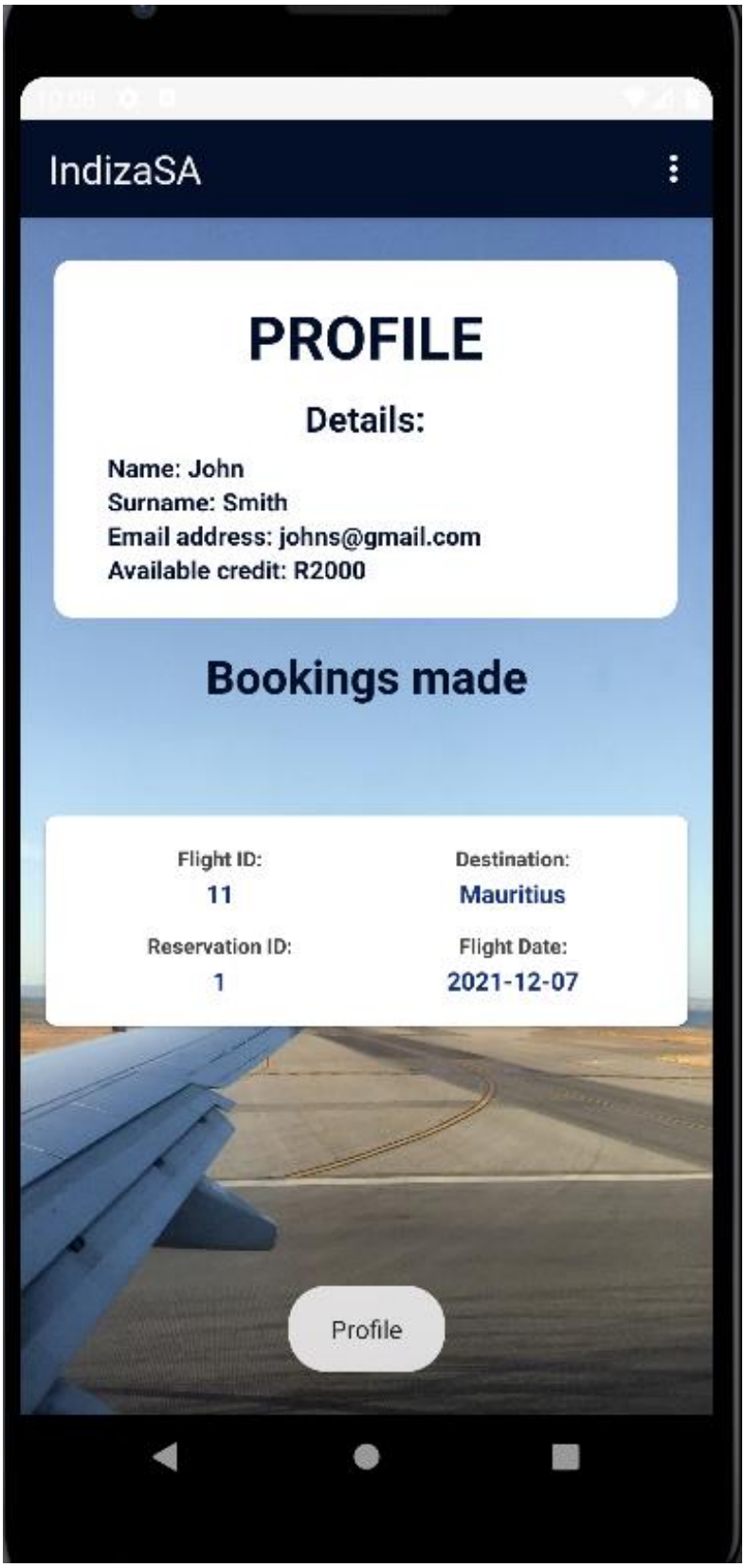


Figure 69

Additional flights can be booked and will stack underneath the previous booked flights (shown in figure 70).

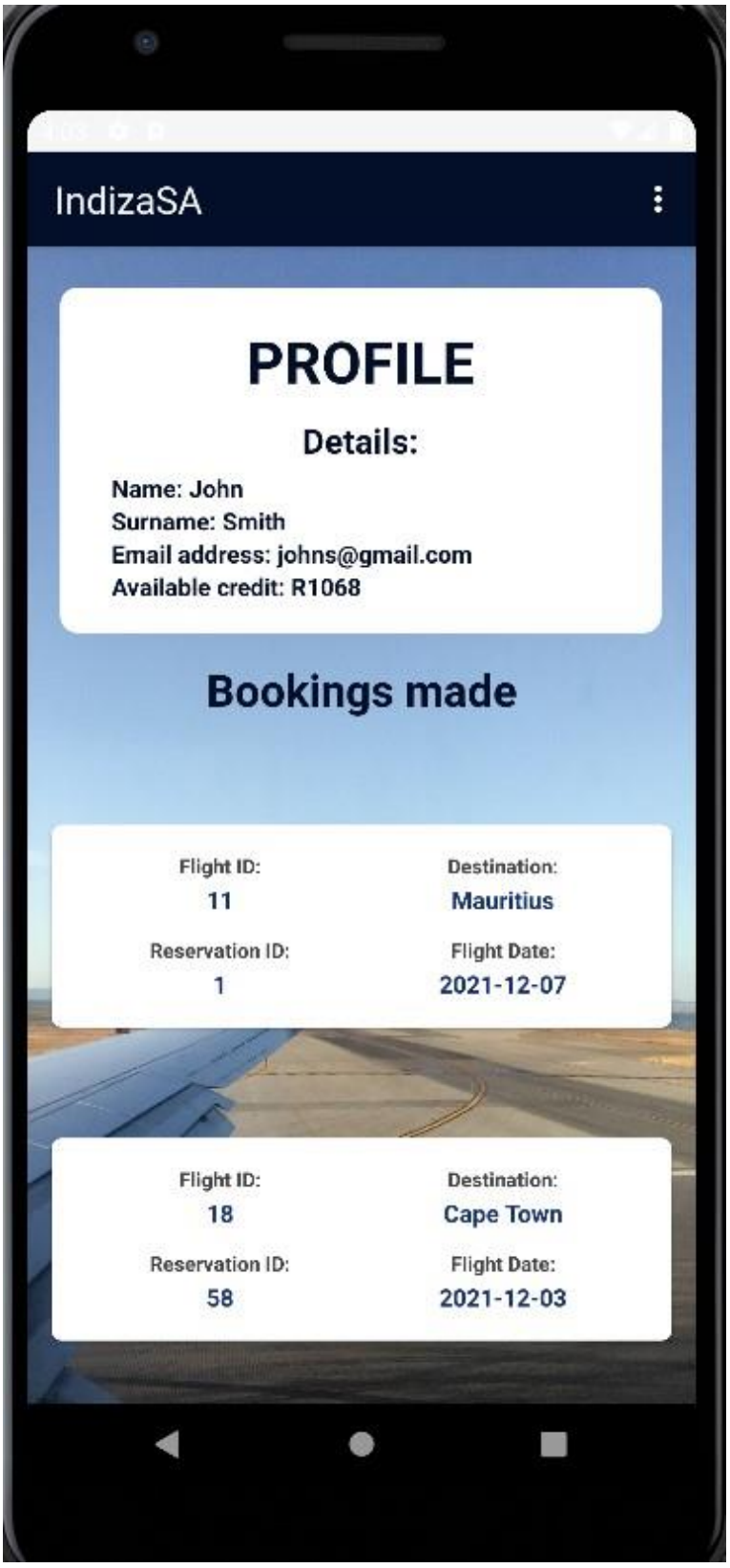


Figure 70

The server output is detailed. Firstly, it indicates that instructions were received from the client that the booking process has begun. The server then checks the credit status of the user and either sends a result of “approved” or “denied” back to the client (“denied” requests is covered in 2.4).

An approved credit rating is indicated in the server output and confirmation follows that the reservation was successfully inserted into the database. Furthermore, it is indicated that the user’s credit balance was adjusted (also covered in 2.4) and that the payment table was also updated.

The server also shows that a data report was written and stored in the database, pertaining to the user’s transaction (covered in 2.4). Lastly, the server indicates that the number of seats left on the flight has also been adjusted.

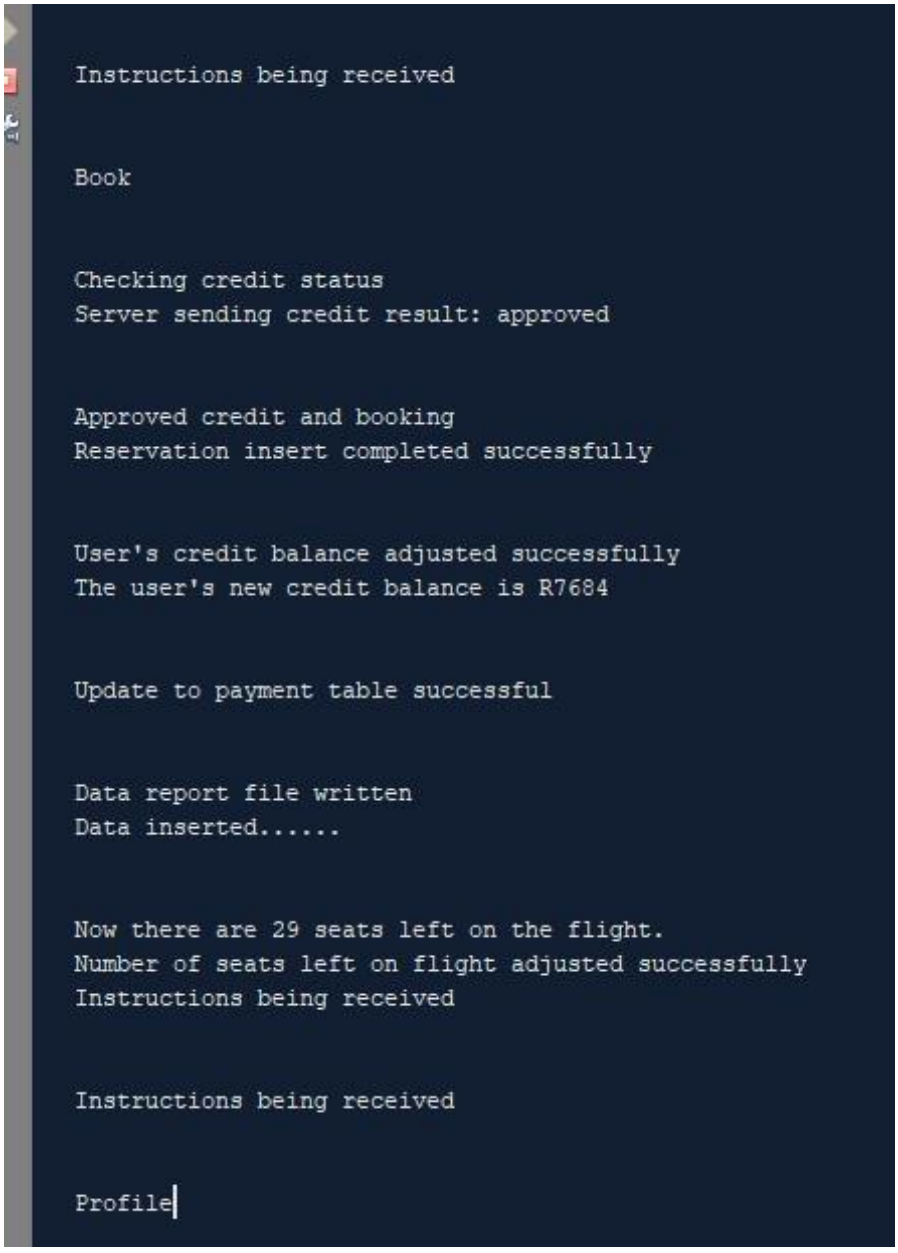


Figure 71

Cancellation:

A user can choose to cancel a booked flight by navigating to their profile (using the navigation bar, covered in 1.4). Clicking on one of the booked flights will result in a pop-up notification where a user can choose to either cancel or confirm the cancellation of the flight (shown in figure 72).

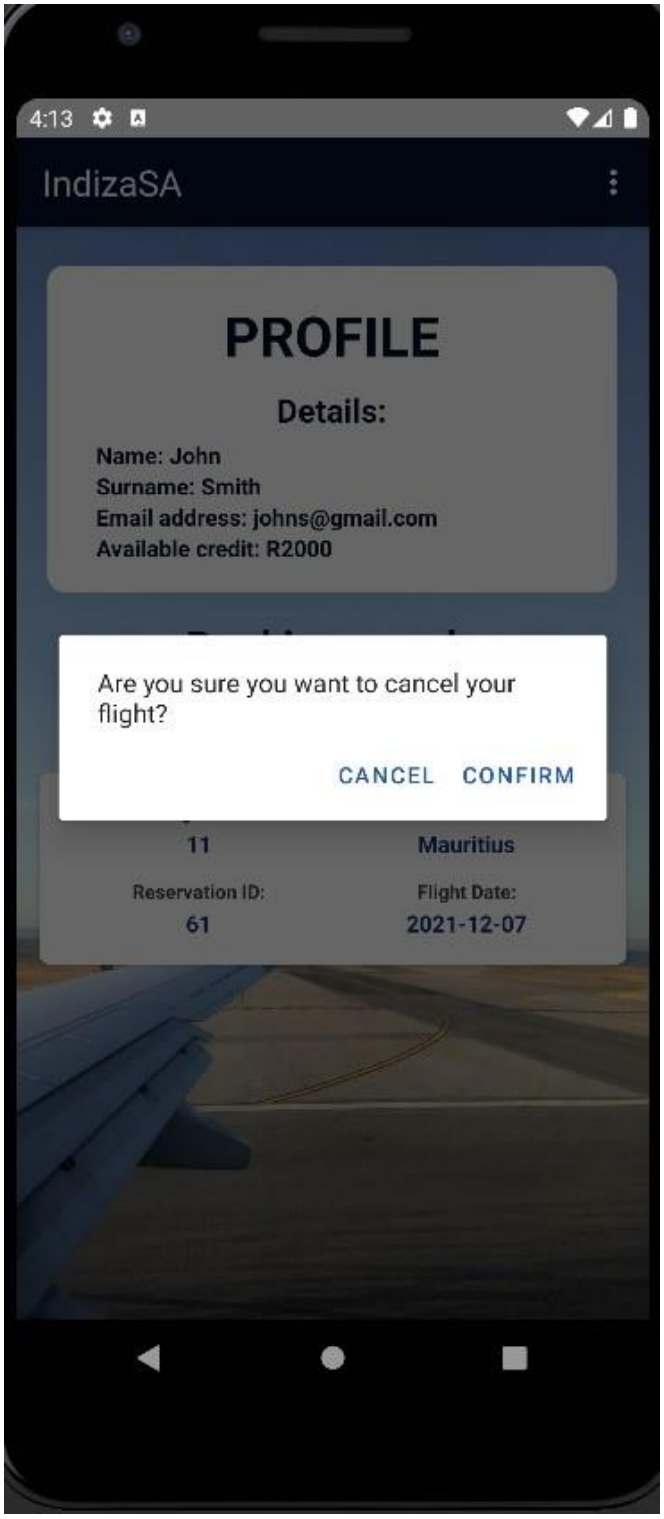


Figure 72

The user's available credit adjusts (compare figure 72 and 73 – R2000 to R8616) and the booking disappears from their listed bookings. Credit adjustments and more in-depth coverage of cancellations will follow in 2.4.

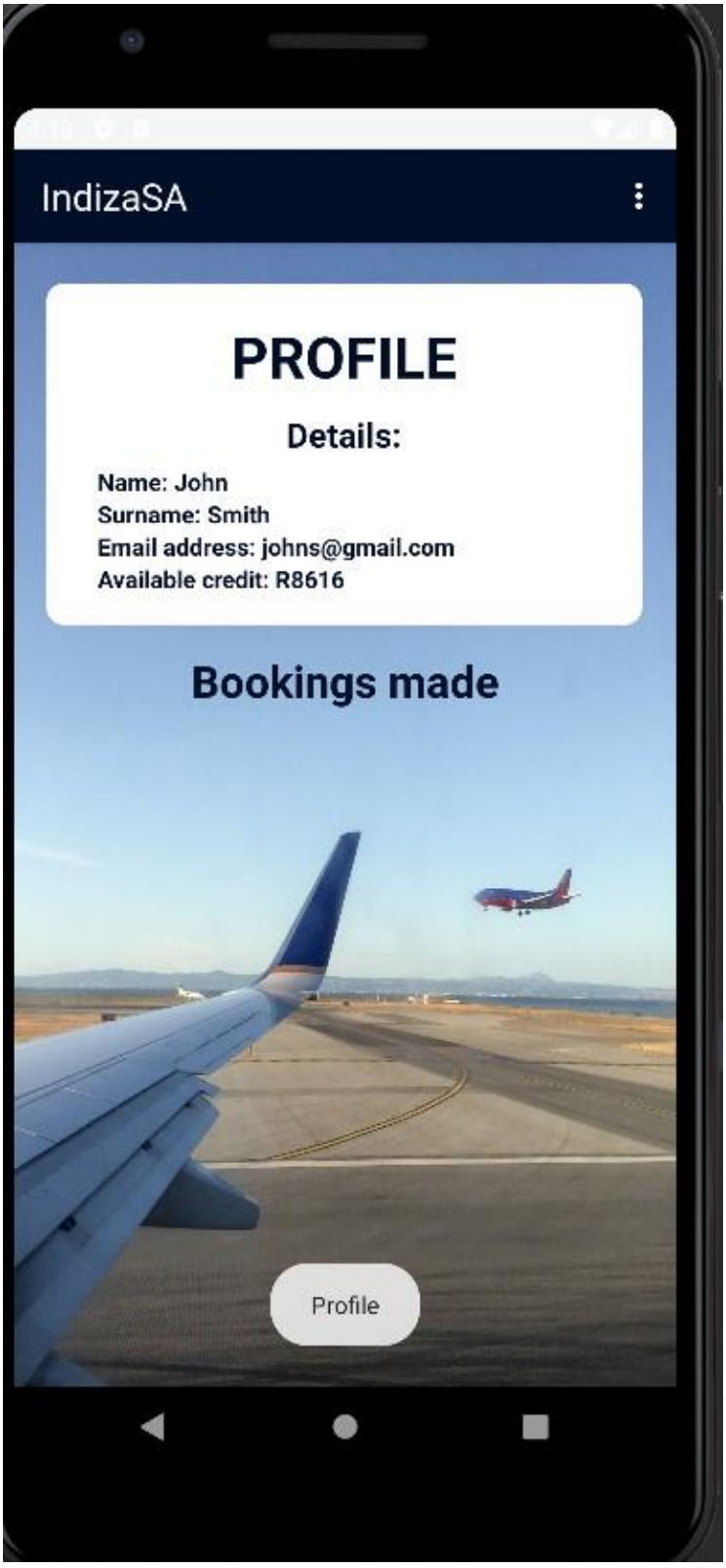


Figure 73

The server output indicates that instructions were received from the client that the cancellation process has started (shown in figure 74). The server prints out the reservation ID – 61 (compare with figure 72), adjusted user credit balance and that data for cancellation has been passed to the payment module.

Furthermore, it indicates that the receipt module received the data, as a file was written and the data was inserted into the database.

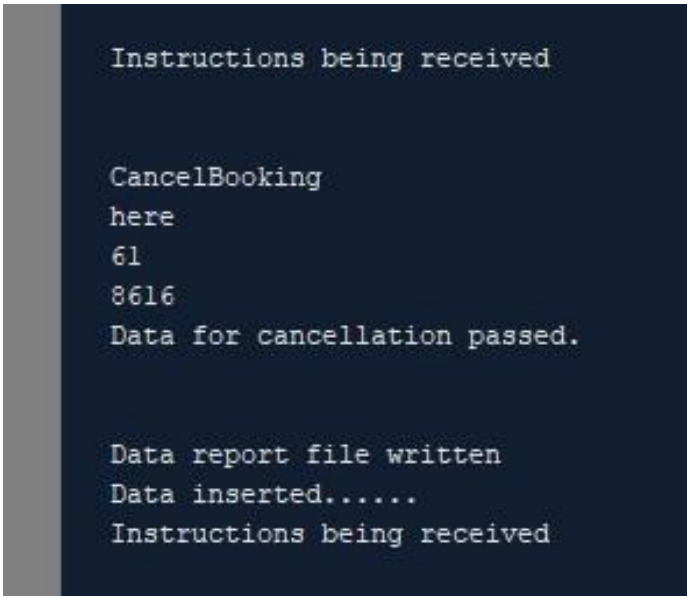


Figure 74

1.4 List users for specfic flight and other features

The server GUI has two main panels, the flight reservation management panel, and the payment management panel (as shown in figure 75). A user can choose to click on the “Show all flights” button to view a list of all the flights in the database. Clicking on the “Show all users” button lists all the users registered on the database. Clicking on “Show all users for FlightID” requires a user to input a flight ID.

Figure 75 shows an input of flight ID 14. Figure 76 shows the resulting pop-up listing all the users registered to the specific flight.

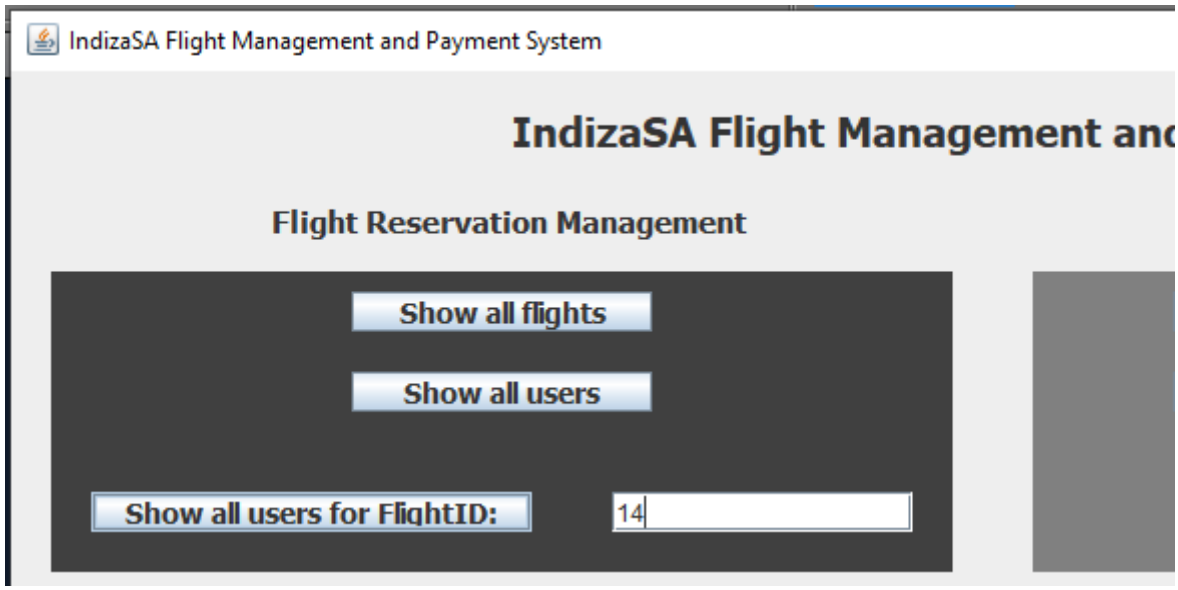
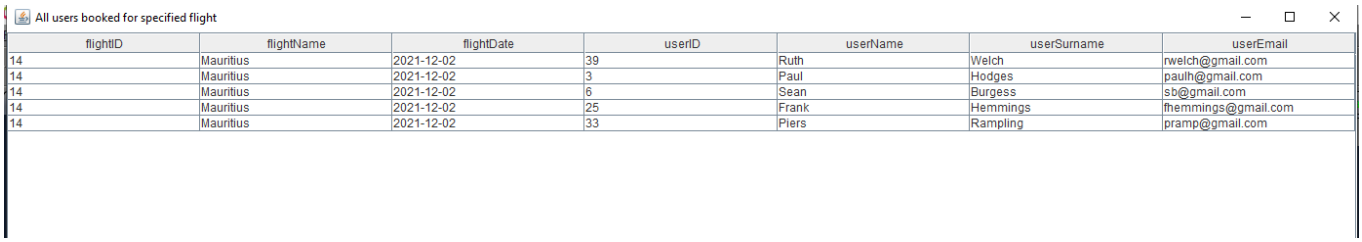


Figure 75



flightID	flightName	flightDate	userID	userName	userSurname	userEmail
14	Mauritius	2021-12-02	39	Ruth	Welch	rwelch@gmail.com
14	Mauritius	2021-12-02	3	Paul	Hodges	paulh@gmail.com
14	Mauritius	2021-12-02	6	Sean	Burgess	sb@gmail.com
14	Mauritius	2021-12-02	25	Frank	Hemmings	fhemmings@gmail.com
14	Mauritius	2021-12-02	33	Piers	Rampling	pramp@gmail.com

Figure 76

The application also includes a navigation bar in the top right corner of the screen in the form of a hamburger menu (shown in figure 77). The navigation bar only appears on a successful login and allows a user to navigate to their profile and back to the home screen. Lastly, it allows a user to log out of the application, resulting in the application shutting down.

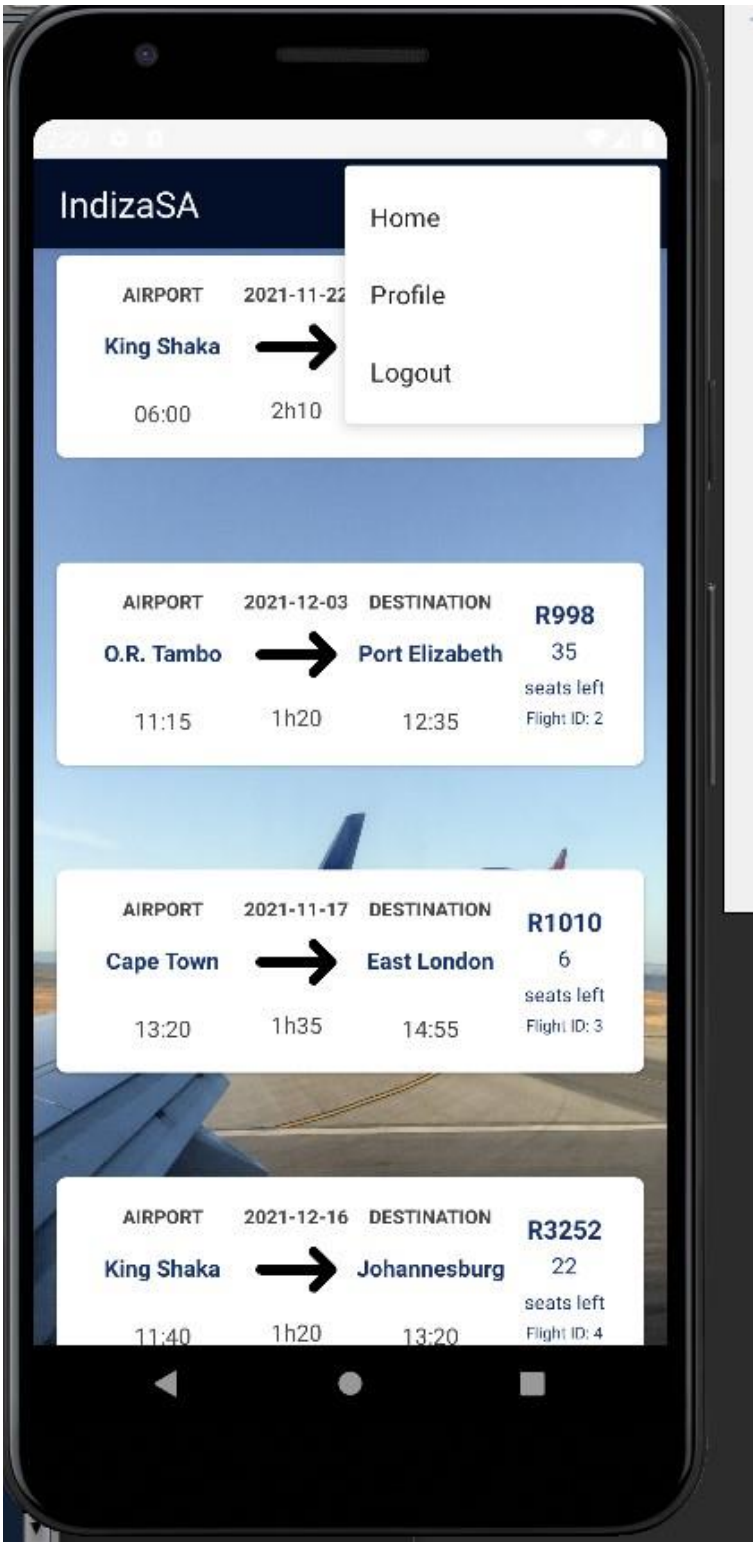


Figure 77

2.1 Software architecture model and database design

In its simplest form, the application makes use of a client-server model using a TCP/IP connection, where the client and server communicate through the TCP/IP connection (shown in figure 78.1)

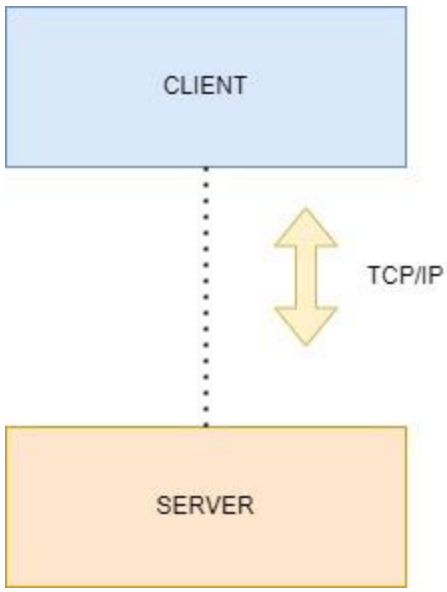


Figure 78.1

A more complicated diagram (shown in figure 78.2) shows that the architecture is based on a broker pattern structure, often used in a distributed system. Java's RMI (Remote Method Invocation) would have been ideal for the purpose of the created distributed system, but unfortunately the library is incompatible with Android Studio.

Another broker was thus used, namely Gson. It is an open-source library and is used to serialize and deserialize created Java objects to a JSON (JavaScript Object Notation) data format on both the client- and server-side.

This allows the Java server to make use of a model, such as a User model, to create an object, deserialize it using Gson and send the data it via a socket connection to the client.

Upon receiving the object, the client can serialize the JSON data back to a User object. Having the exact same Java User model created client-side allows the client to serialize the JSON data to essentially the same User object. This process is used to pass objects containing data (such as user data and flight data) between the server and client.

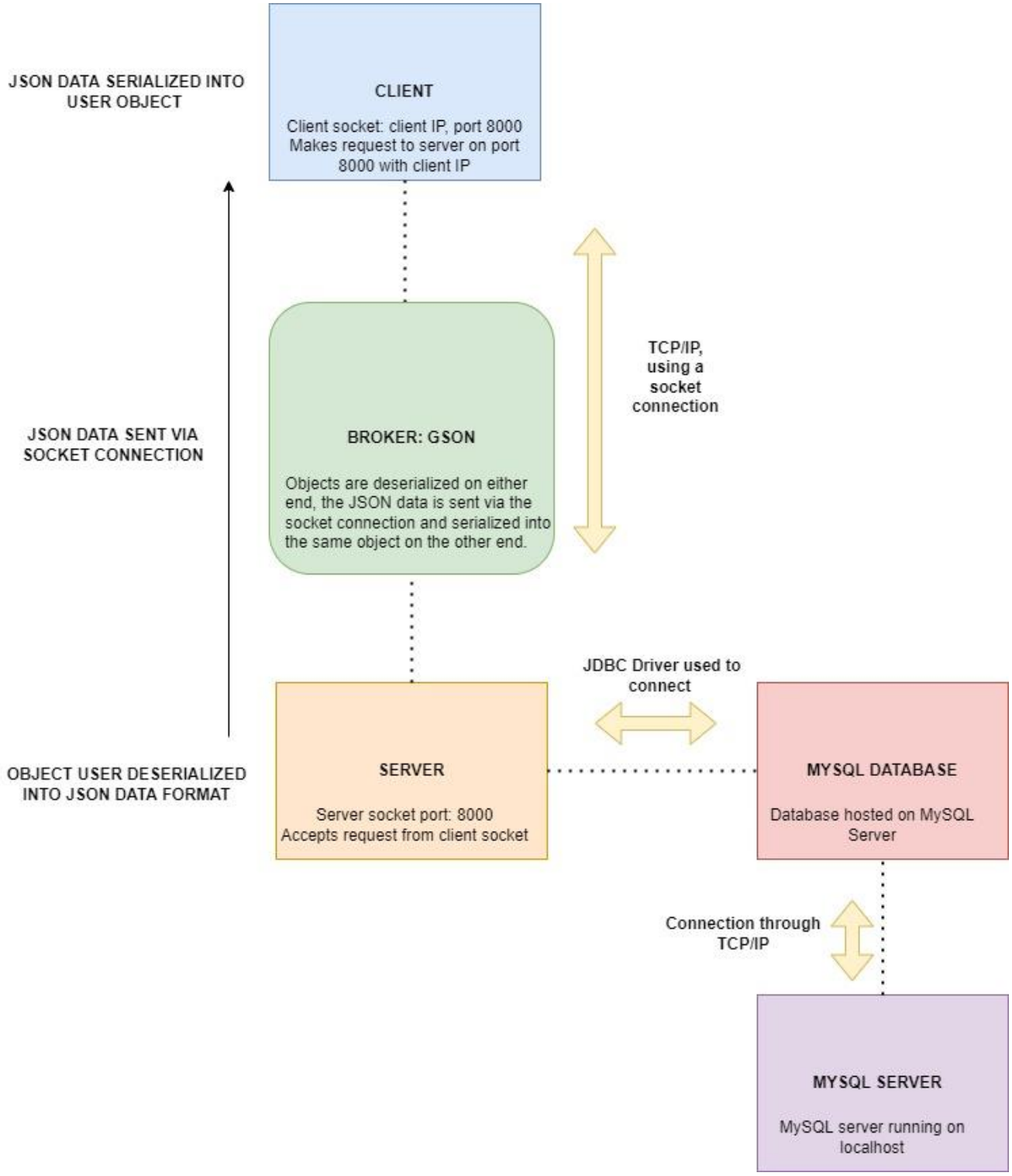


Figure 78.2

A UML Deployment model is also displayed in figure 78.3 to provide further clarity on the distributed system.

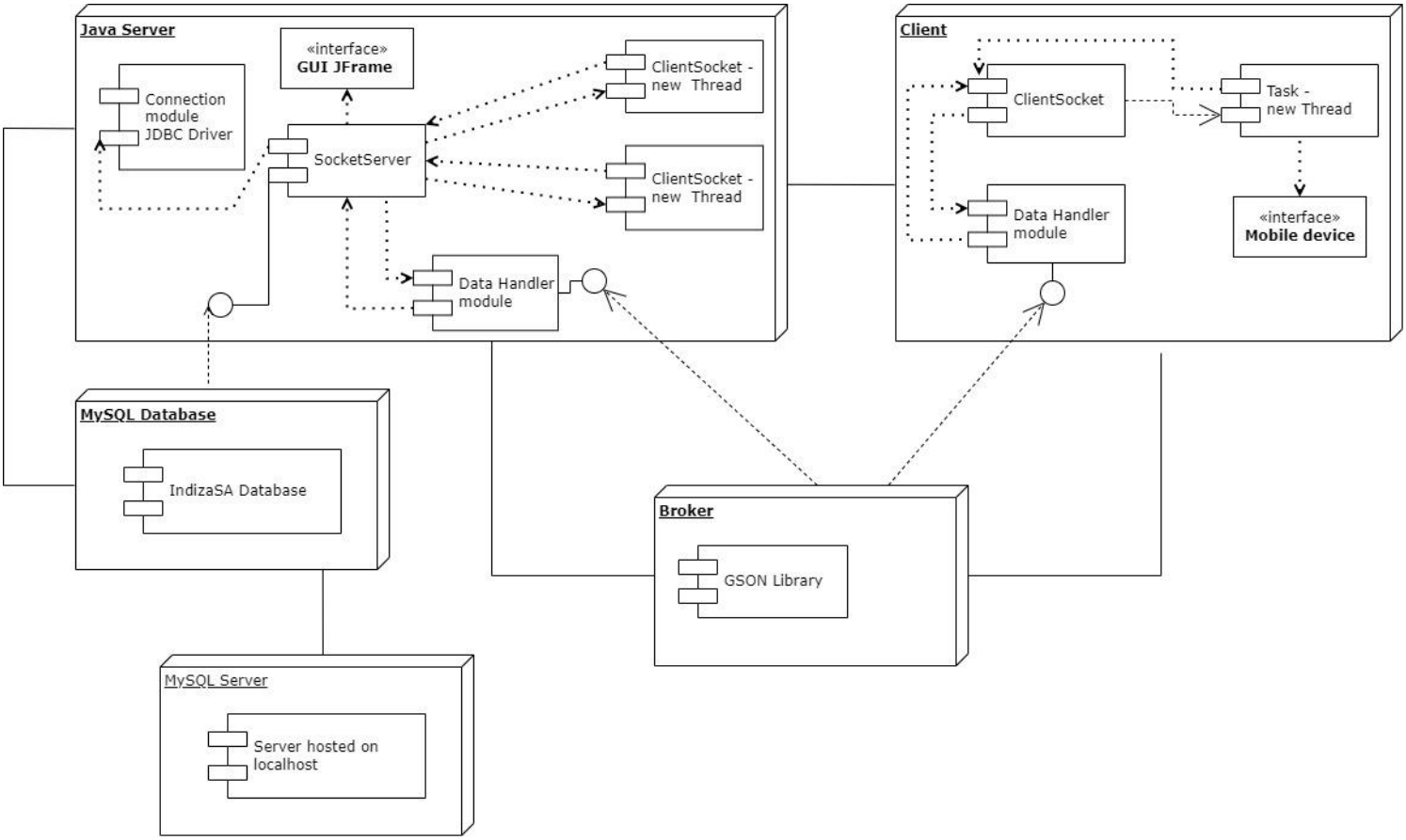


Figure 78.3

The MySQL database architecture, in the form of an Enhanced Entity-Relationship (EER) diagram, is shown in figure 79. The MySQL tables and dummy data is shown in figure 80 to 87.

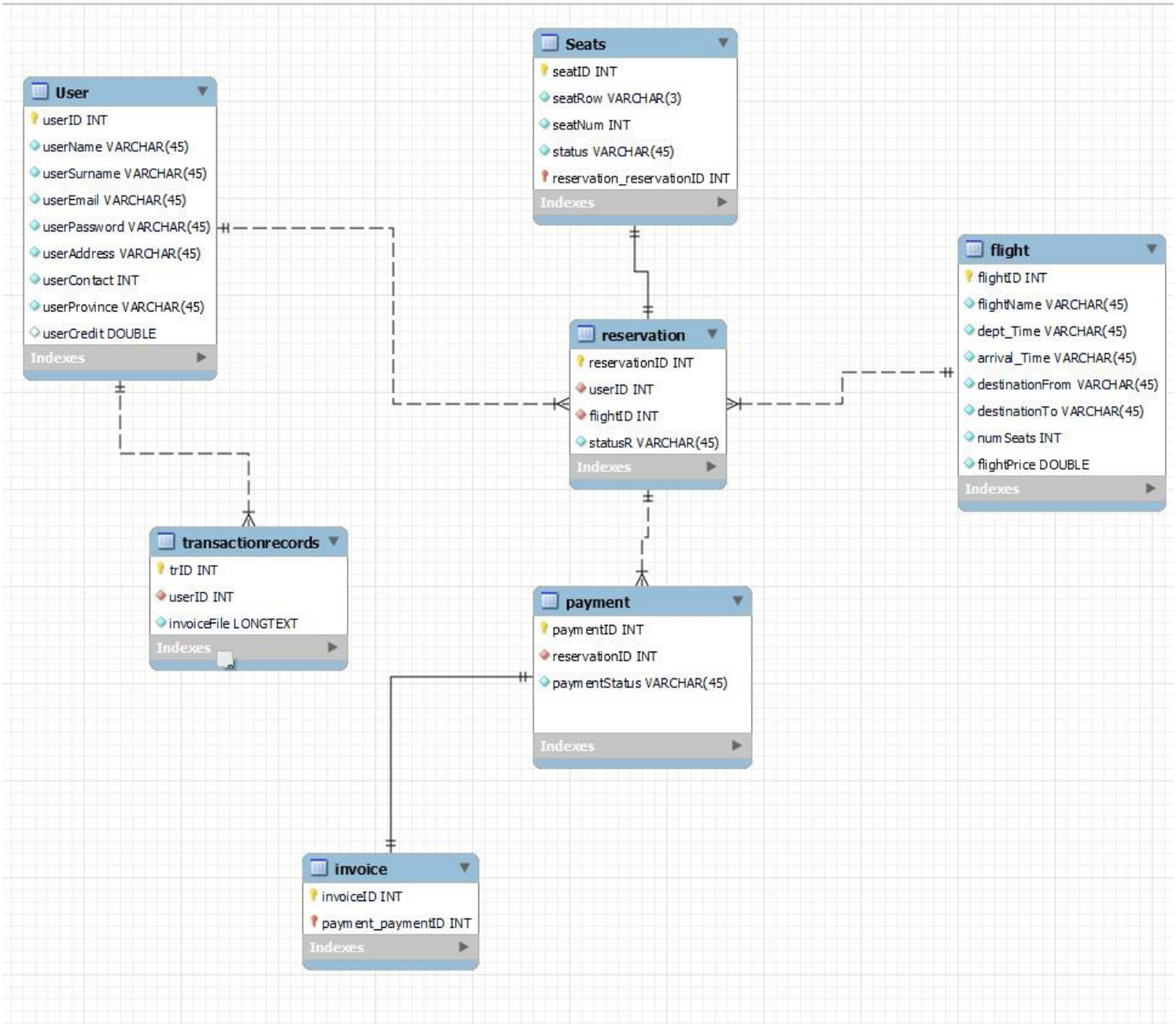
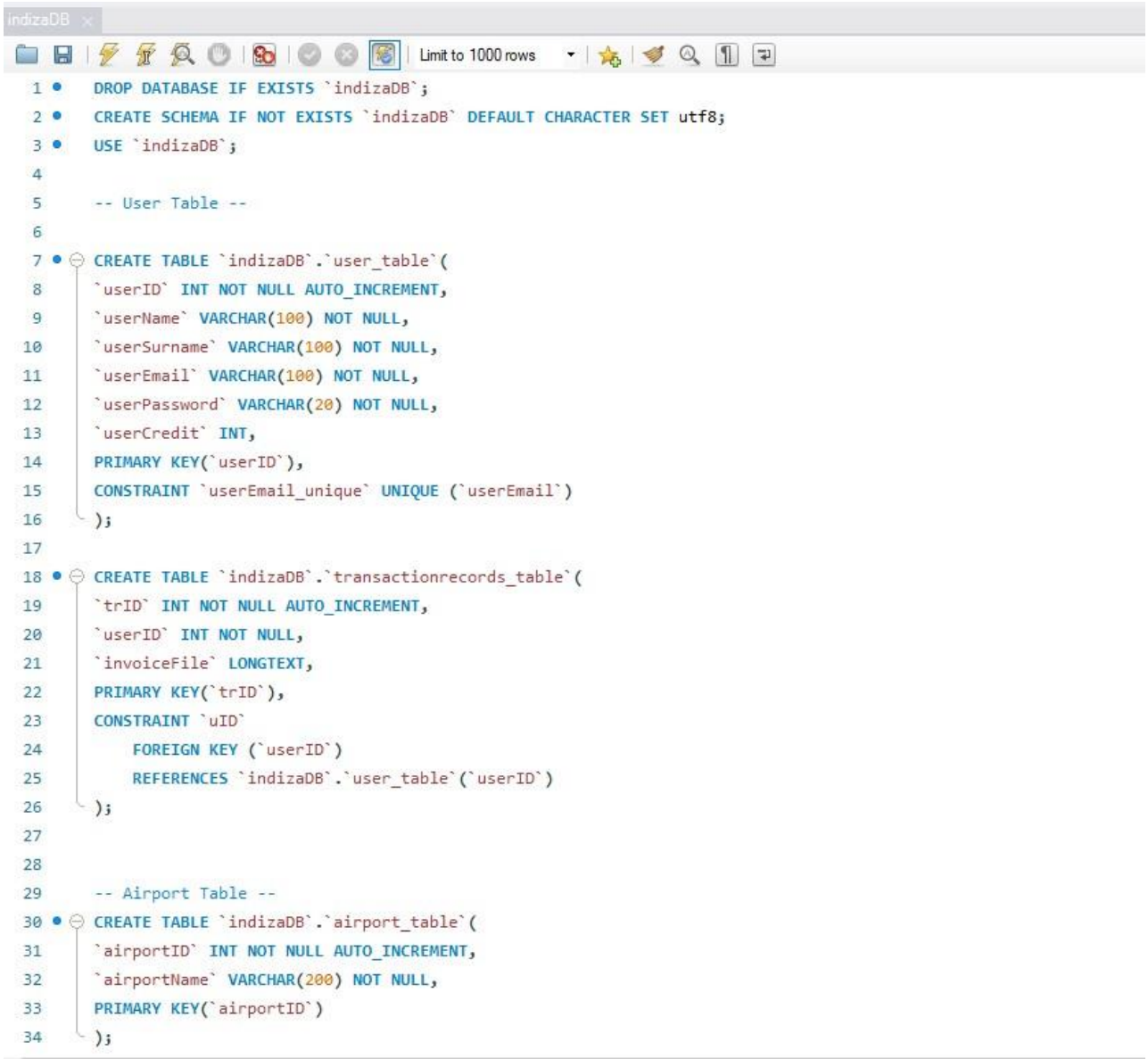

Seats
seatID INT
seatRow VARCHAR(3)
seatNum INT
status VARCHAR(45)
reservation_reservationID INT
Indexes
User
userID INT
userName VARCHAR(45)
userSurname VARCHAR(45)
userEmail VARCHAR(45)
userPassword VARCHAR(45)
userAddress VARCHAR(45)
userContact INT
userProvince VARCHAR(45)
userCredit DOUBLE
Indexes
reservation
reservationID INT
userID INT
flightID INT
statusR VARCHAR(45)
Indexes
flight
flightID INT
flightName VARCHAR(45)
dept_Time VARCHAR(45)
arrival_Time VARCHAR(45)
destinationFrom VARCHAR(45)
destinationTo VARCHAR(45)
numSeats INT
flightPrice DOUBLE
Indexes
transactionrecords
trID INT
userID INT
invoiceFile LONGTEXT
Indexes
payment
paymentID INT
reservationID INT
paymentStatus VARCHAR(45)
Indexes
invoice
invoiceID INT
payment_paymentID INT
Indexes

Figure 79



```
DROP DATABASE IF EXISTS `indizaDB`;
CREATE SCHEMA IF NOT EXISTS `indizaDB` DEFAULT CHARACTER SET utf8;
USE `indizaDB`;

-- User Table --

CREATE TABLE `indizaDB`.`user_table`(
`userID` INT NOT NULL AUTO_INCREMENT,
`userName` VARCHAR(100) NOT NULL,
`userSurname` VARCHAR(100) NOT NULL,
`userEmail` VARCHAR(100) NOT NULL,
`userPassword` VARCHAR(20) NOT NULL,
`userCredit` INT,
PRIMARY KEY(`userID`),
CONSTRAINT `userEmail_unique` UNIQUE (`userEmail`)
);

CREATE TABLE `indizaDB`.`transactionrecords_table`(
`trID` INT NOT NULL AUTO_INCREMENT,
`userID` INT NOT NULL,
`invoiceFile` LONGTEXT,
PRIMARY KEY(`trID`),
CONSTRAINT `uID`
    FOREIGN KEY (`userID`)
    REFERENCES `indizaDB`.`user_table`(`userID`)
);


-- Airport Table --
CREATE TABLE `indizaDB`.`airport_table`(
`airportID` INT NOT NULL AUTO_INCREMENT,
`airportName` VARCHAR(200) NOT NULL,
PRIMARY KEY(`airportID`)
);
```

Figure 80

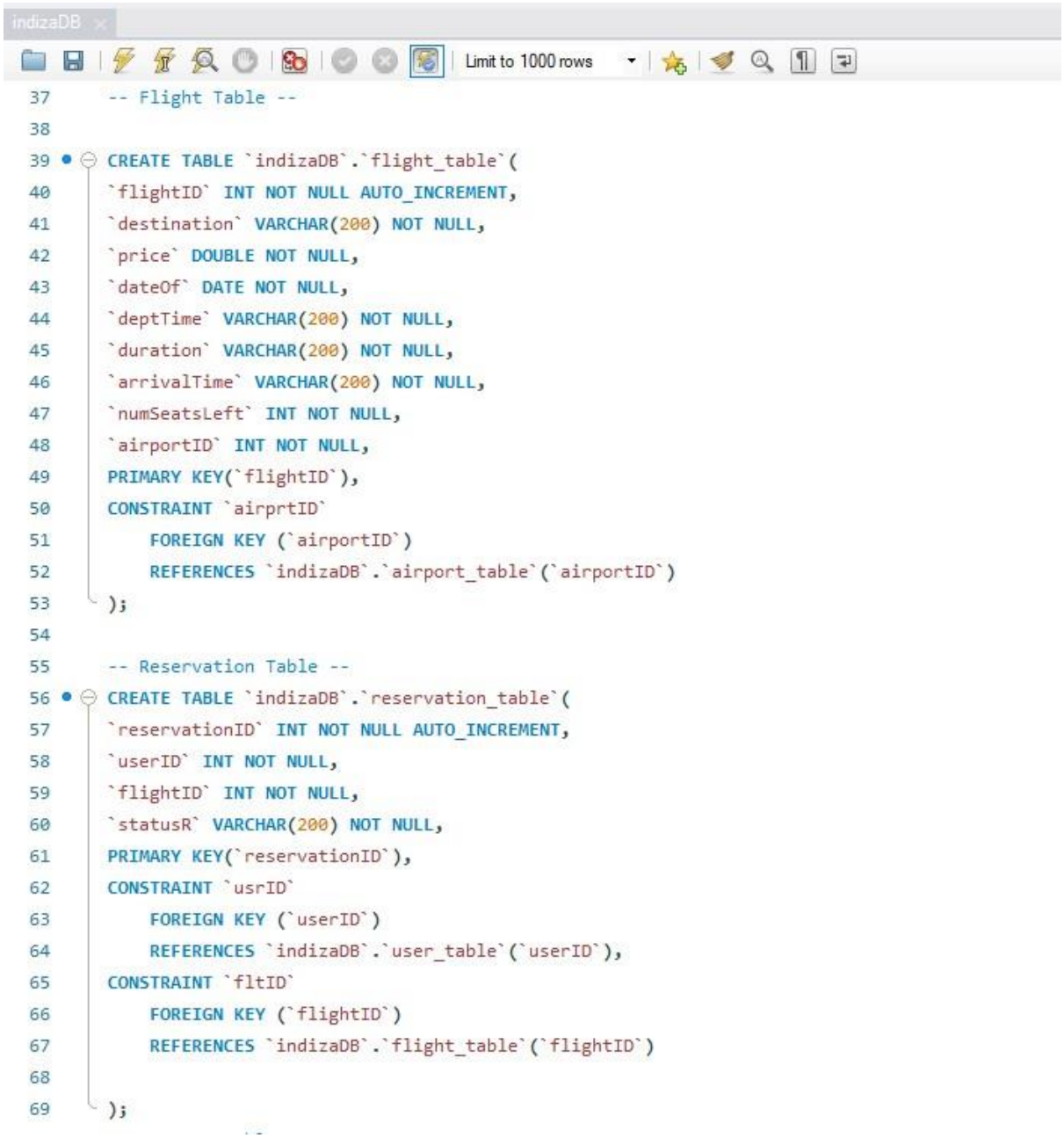
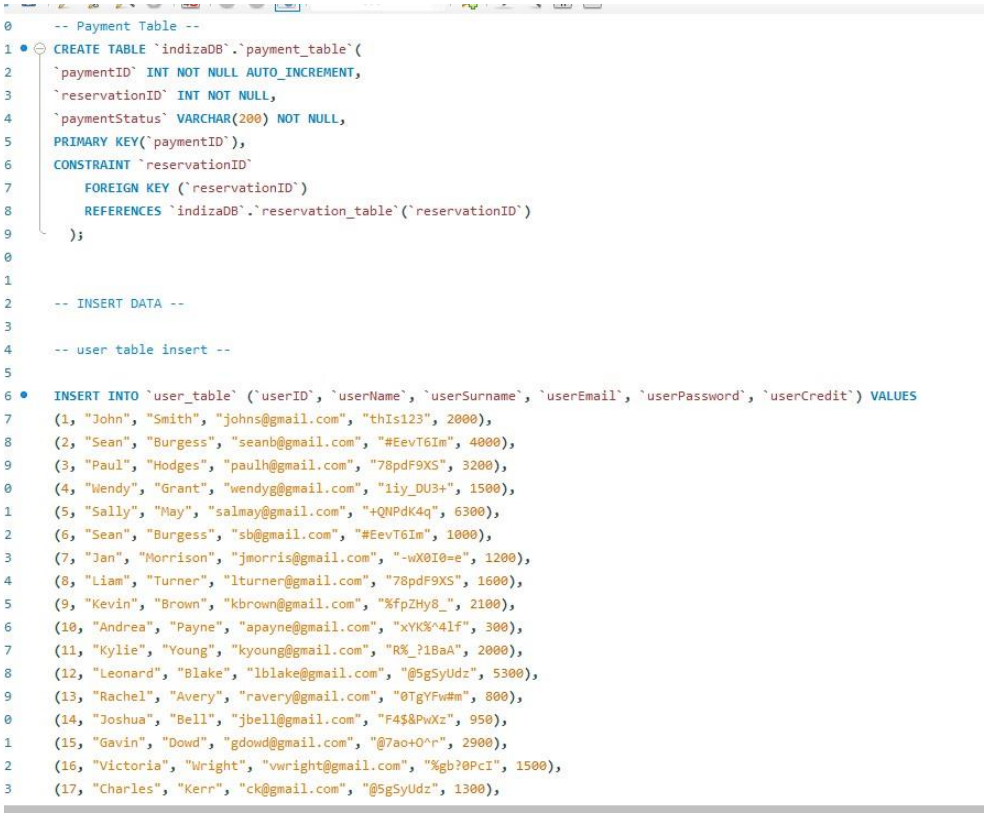

indizaDB
Limit to 1000 rows
-- Flight Table --
CREATE TABLE `indizaDB`.`flight_table`(
`flightID` INT NOT NULL AUTO_INCREMENT,
`destination` VARCHAR(200) NOT NULL,
`price` DOUBLE NOT NULL,
`dateOf` DATE NOT NULL,
`deptTime` VARCHAR(200) NOT NULL,
`duration` VARCHAR(200) NOT NULL,
`arrivalTime` VARCHAR(200) NOT NULL,
`numSeatsLeft` INT NOT NULL,
`airportID` INT NOT NULL,
PRIMARY KEY(`flightID`),
CONSTRAINT `airprtID`
FOREIGN KEY (`airportID`)
REFERENCES `indizaDB`.`airport_table`(`airportID`)
);
-- Reservation Table --
CREATE TABLE `indizaDB`.`reservation_table`(
`reservationID` INT NOT NULL AUTO_INCREMENT,
`userID` INT NOT NULL,
`flightID` INT NOT NULL,
`statusR` VARCHAR(200) NOT NULL,
PRIMARY KEY(`reservationID`),
CONSTRAINT `usrID`
FOREIGN KEY (`userID`)
REFERENCES `indizaDB`.`user_table`(`userID`),
CONSTRAINT `fltID`
FOREIGN KEY (`flightID`)
REFERENCES `indizaDB`.`flight_table`(`flightID`)
);

Figure 81



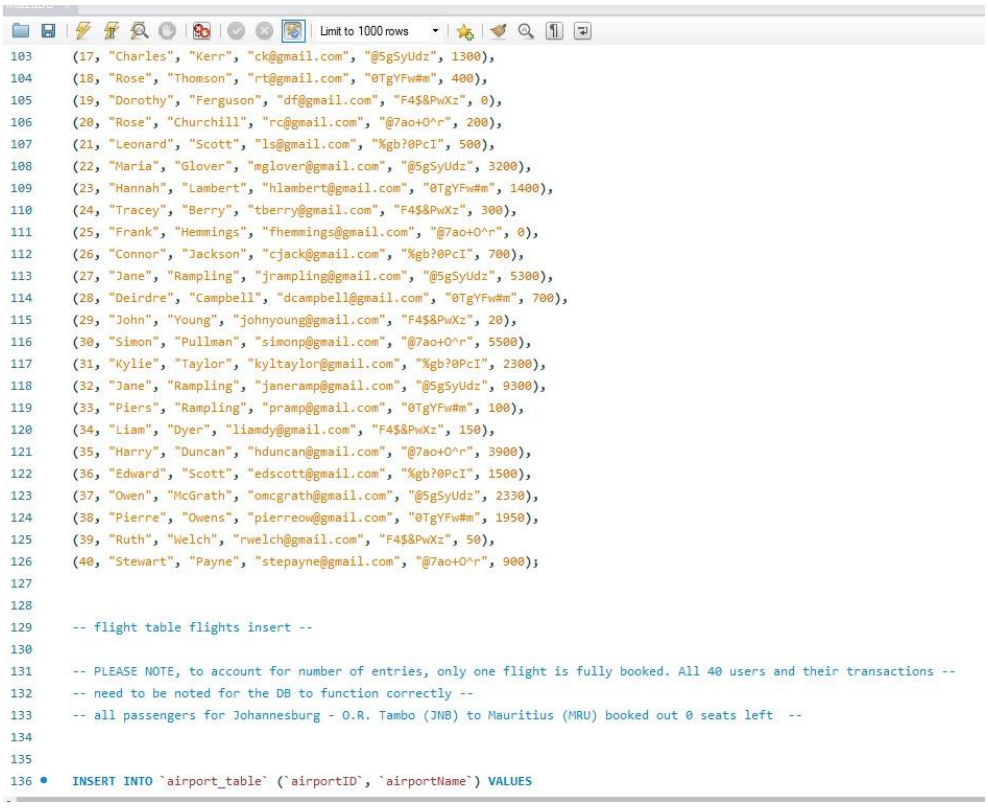
```
-- Payment Table --
CREATE TABLE `indizaDB`.`payment_table`(
`paymentID` INT NOT NULL AUTO_INCREMENT,
`reservationID` INT NOT NULL,
`paymentStatus` VARCHAR(200) NOT NULL,
PRIMARY KEY(`paymentID`),
CONSTRAINT `reservationID`
    FOREIGN KEY (`reservationID`)
    REFERENCES `indizaDB`.`reservation_table`(`reservationID`)
  );


-- INSERT DATA --

-- user table insert --

INSERT INTO `user_table` (`userID`, `userName`, `userSurname`, `userEmail`, `userPassword`, `userCredit`) VALUES
(1, "John", "Smith", "johns@gmail.com", "thIs123", 2000),
(2, "Sean", "Burgess", "seanb@gmail.com", "#EevT6Im", 4000),
(3, "Paul", "Hodges", "paulh@gmail.com", "78pdF9XS", 3200),
(4, "Wendy", "Grant", "wendyg@gmail.com", "1iy_DU3+", 1500),
(5, "Sally", "May", "salmay@gmail.com", "+QNPdK4q", 6300),
(6, "Sean", "Burgess", "sb@gmail.com", "#EevT6Im", 1000),
(7, "Jan", "Morrison", "jmorris@gmail.com", "-wX0I0=e", 1200),
(8, "Liam", "Turner", "lturner@gmail.com", "78pdF9XS", 1600),
(9, "Kevin", "Brown", "kbrown@gmail.com", "%fpZHy8_", 2100),
(10, "Andrea", "Payne", "apayne@gmail.com", "xYK%^41f", 300),
(11, "Kylie", "Young", "kyoung@gmail.com", "R%_?1BaA", 2000),
(12, "Leonard", "Blake", "lblake@gmail.com", "@5gSyUdz", 5300),
(13, "Rachel", "Avery", "ravery@gmail.com", "0TgYFw#m", 800),
(14, "Joshua", "Bell", "jbell@gmail.com", "F4$&PwXz", 950),
(15, "Gavin", "Dowd", "gdowd@gmail.com", "@7ao+O^r", 2900),
(16, "Victoria", "Wright", "vwright@gmail.com", "%gb?0PcI", 1500),
(17, "Charles", "Kerr", "ck@gmail.com", "@5gSyUdz", 1300),
```

Figure 82



```
Limit to 1000 rows
(17, "Charles", "Kerr", "ck@gmail.com", "@5gSyUdz", 1300),
(18, "Rose", "Thomson", "rt@gmail.com", "0TgYFw#m", 400),
(19, "Dorothy", "Ferguson", "df@gmail.com", "F4$&PwXz", 0),
(20, "Rose", "Churchill", "rc@gmail.com", "@7ao+O^r", 200),
(21, "Leonard", "Scott", "ls@gmail.com", "%gb?0PcI", 500),
(22, "Maria", "Glover", "mglover@gmail.com", "@5gSyUdz", 3200),
(23, "Hannah", "Lambert", "hlambert@gmail.com", "0TgYFw#m", 1400),
(24, "Tracey", "Berry", "tberry@gmail.com", "F4$&PwXz", 300),
(25, "Frank", "Hemmings", "fhemmings@gmail.com", "@7ao+O^r", 0),
(26, "Connor", "Jackson", "cjack@gmail.com", "%gb?0PcI", 700),
(27, "Jane", "Rampling", "jrampling@gmail.com", "@5gSyUdz", 5300),
(28, "Deirdre", "Campbell", "dcampbell@gmail.com", "0TgYFw#m", 700),
(29, "John", "Young", "johnyoung@gmail.com", "F4$&PwXz", 20),
(30, "Simon", "Pullman", "simonp@gmail.com", "@7ao+O^r", 5500),
(31, "Kylie", "Taylor", "kyltaylor@gmail.com", "%gb?0PcI", 2300),
(32, "Jane", "Rampling", "janeramp@gmail.com", "@5gSyUdz", 9300),
(33, "Piers", "Rampling", "pramp@gmail.com", "0TgYFw#m", 100),
(34, "Liam", "Dyer", "liamdy@gmail.com", "F4$&PwXz", 150),
(35, "Harry", "Duncan", "hduncan@gmail.com", "@7ao+O^r", 3900),
(36, "Edward", "Scott", "edscott@gmail.com", "%gb?0PcI", 1500),
(37, "Owen", "McGrath", "omcgrath@gmail.com", "@5gSyUdz", 2330),
(38, "Pierre", "Owens", "pierreow@gmail.com", "0TgYFw#m", 1950),
(39, "Ruth", "Welch", "rwelch@gmail.com", "F4$&PwXz", 50),
(40, "Stewart", "Payne", "stepayne@gmail.com", "@7ao+O^r", 900);


-- flight table flights insert --

-- PLEASE NOTE, to account for number of entries, only one flight is fully booked. All 40 users and their transactions --
-- need to be noted for the DB to function correctly --
-- all passengers for Johannesburg - O.R. Tambo (JNB) to Mauritius (MRU) booked out 0 seats left  --


INSERT INTO `airport_table` (`airportID`, `airportName`) VALUES
```

Figure 83

```
136 • INSERT INTO `airport_table` (`airportID`, `airportName`) VALUES
137   (1, "O.R. Tambo"),
138   (2, "King Shaka"),
139   (3, "Cape Town");
140
141 • INSERT INTO `flight_table` (`flightID`, `destination`, `price`, `dateOf`, `deptTime`, `duration`, `arrivalTime`, `numSeatsLeft`, `airportID`) VALUES
142   (1, "Cape Town", 920, "2021-11-22", "06:00", "2h10", "08:10", 20, 2),
143   (2, "Port Elizabeth",  998, "2021-12-03", "11:15", "1h20", "12:35", 35, 1),
144   (3, "East London",  1010, "2021-11-17", "13:20", "1h35", "14:55", 6, 3),
145   (4, "Johannesburg",  3252, "2021-12-16", "11:40", "1h20", "13:20", 22, 2),
146   (5, "Johannesburg",  841, "2021-12-20", "10:05", "2h10", "11:55", 10, 3),
147   (6, "Harare", 1444, "2021-11-15", "11:40", "1h35", "13:15", 11, 1),
148   (7, "Cape Town", 2519, "2021-12-24", "07:15","2h10", "09:25",  39, 2),
149   (8, "Durban", 1804, "2021-11-24","06:40", "2h", "08:40", 11, 1),
150   (9, "Johannesburg", 3152, "2021-12-09", "11:40", "1h40", "13:20", 15, 3),
151   (10, "Johannesburg", 4118, "2021-11-14", "14:20","1h50", "16:10", 12, 2),
152   (11, "Mauritius", 6616, "2021-12-07", "09:55", "6h10", "16:05",  0, 1),
153   (12, "Johannesburg", 818, "2021-12-03", "09:40", "1h10", "10:50",  5, 3),
154   (13, "Cape Town", 1117, "2021-11-17", "14:30", "2h", "16:30",  25, 1),
155   (14, "Mauritius", 4639, "2021-12-02", "17:10", "2h30", "19:40",  35, 1),
156   (15, "Victoria Falls", 2232, "2021-11-27", "13:45", "1h35", "15:20", 2, 1),
157   (16, "Victoria Falls", 2014, "2021-11-20", "11:30", "1h37", "13:07",  3, 1),
158   (17, "Johannesburg", 716, "2021-12-03", "09:40", "1h10", "10:50", 15, 2),
159   (18, "Cape Town", 932, "2021-12-03", "06:00", "2h", "08:00", 30, 2),
160   (19, "Cape Town", 1602, "2021-11-14", "11:35", "2h10", "13:45", 5, 1),
161   (20, "East London", 849, "2021-12-08", "13:55", "1h35", "15:30",  11, 3);
162
163
164   -- flight table flights insert --
165
166 • INSERT INTO `reservation_table` (`reservationID`, `userID`, `flightID`, `statusR`) VALUES
167   (1, 1, 11, "active"),
168   (2, 2, 11, "active"),
169   (3, 3, 11, "active"),
```

Figure 84

```
170     (4, 4, 11, "active"),
171     (5, 5, 11, "active"),
172     (6, 6, 11, "active"),
173     (7, 7, 11, "active"),
174     (8, 8, 11, "active"),
175     (9, 9, 11, "active"),
176     (10, 10, 11, "active"),
177     (11, 11, 11, "active"),
178     (12, 12, 11, "active"),
179     (13, 13, 11, "active"),
180     (14, 14, 11, "active"),
181     (15, 15, 11, "active"),
182     (16, 16, 11, "active"),
183     (17, 17, 11, "active"),
184     (18, 18, 11, "active"),
185     (19, 19, 11, "active"),
186     (20, 20, 11, "active"),
187     (21, 21, 11, "active"),
188     (22, 22, 11, "active"),
189     (23, 23, 11, "active"),
190     (24, 24, 11, "active"),
191     (25, 25, 11, "active"),
192     (26, 26, 11, "active"),
193     (27, 27, 11, "active"),
194     (28, 28, 11, "active"),
195     (29, 29, 11, "active"),
196     (30, 30, 11, "active"),
197     (31, 31, 11, "active"),
198     (32, 32, 11, "active"),
199     (33, 33, 11, "active"),
200     (34, 34, 11, "active"),
201     (35, 35, 11, "active"),
202     (36, 36, 11, "active"),
```

Figure 85

```
(37, 37, 11, "active"),
(38, 38, 11, "active"),
(39, 39, 11, "active"),
(40, 40, 11, "active"),
(41, 39, 14, "active"),
(42, 3, 14, "active"),
(43, 6, 14, "active"),
(44, 25, 14, "active"),
(45, 33, 14, "active"),
(46, 5, 18, "active"),
(47, 7, 18, "active"),
(48, 9, 18, "active"),
(49, 28, 18, "active"),
(50, 37, 18, "active"),
(51, 4, 18, "active"),
(52, 12, 18, "active"),
(53, 13, 18, "active"),
(54, 21, 18, "active"),
(55, 34, 18, "active");


INSERT INTO `payment_table` (`paymentID`, `reservationID`, `paymentStatus`) VALUES
(1, 1, "Paid"),
(2, 2, "Paid"),
(3, 3, "Paid"),
(4, 4, "Paid"),
(5, 5, "Paid"),
(6, 6, "Paid"),
(7, 7, "Paid"),
(8, 8, "Paid"),
(9, 9, "Paid"),
(10, 10, "Paid"),
(11, 11, "Paid"),
```

Figure 86

```
(11, 11, "Paid"),
(12, 12, "Paid"),
(13, 13, "Paid"),
(14, 14, "Paid"),
(15, 15, "Paid"),
(16, 16, "Paid"),
(17, 17, "Paid"),
(18, 18, "Paid"),
(19, 19, "Paid"),
(20, 20, "Paid"),
(21, 21, "Paid"),
(22, 22, "Paid"),
(23, 23, "Paid"),
(24, 24, "Paid"),
(25, 25, "Paid"),
(26, 26, "Paid"),
(27, 27, "Paid"),
(28, 28, "Paid"),
(29, 29, "Paid"),
(30, 30, "Paid"),
(31, 31, "Paid"),
(32, 32, "Paid"),
(33, 33, "Paid"),
(34, 34, "Paid"),
(35, 35, "Paid"),
(36, 36, "Paid"),
(37, 37, "Paid"),
(38, 38, "Paid"),
(39, 39, "Paid"),
(40, 40, "Paid"),
(41, 41, "Paid"),
(42, 42, "Paid"),
(43, 43, "Paid"),
(44, 44, "Paid"),
```

Figure 87

2.2 Receipt and related methods module

The Receipt module is displayed on the right panel of the server GUI. The user can choose to generate a cash disbursements report or to show all transactions for a particular user (as shown in figure 88).

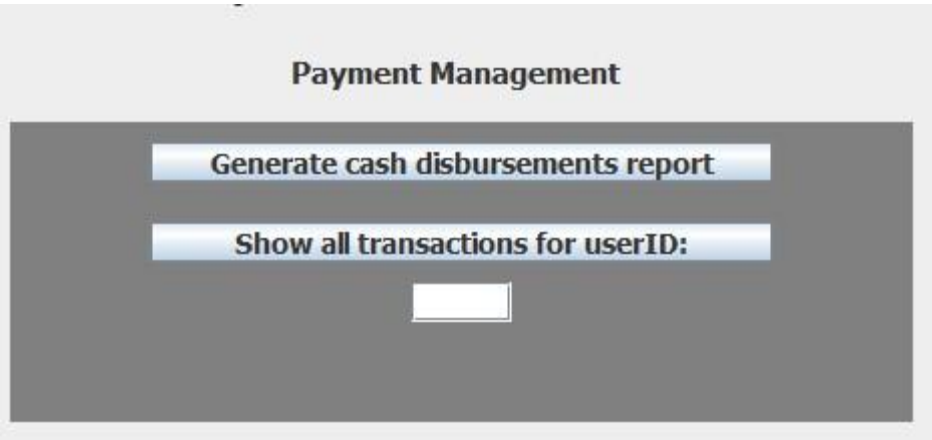


Figure 88

If the user clicks on the “Generate” button, a pop-up notification will indicate that the report was created and is available for viewing in the folder labelled “paymentGeneratedFiles” (as shown in figure 89). The folder can be found within the IndizaServer folder (as shown in figure 90).

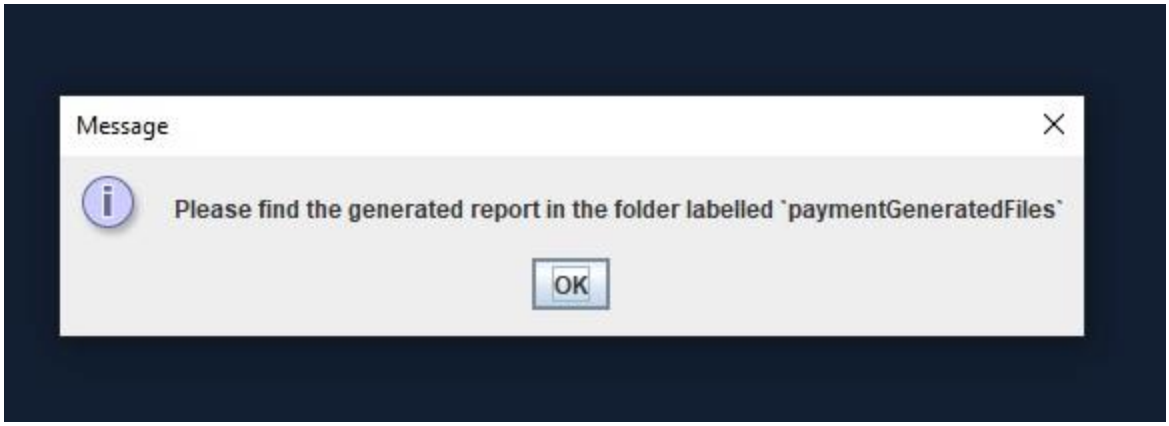


Figure 89

Figure 89

Alternatively, one can navigate to the NetBeans project folder and to the IndizaSAServer (as shown in figure 90 and 91).

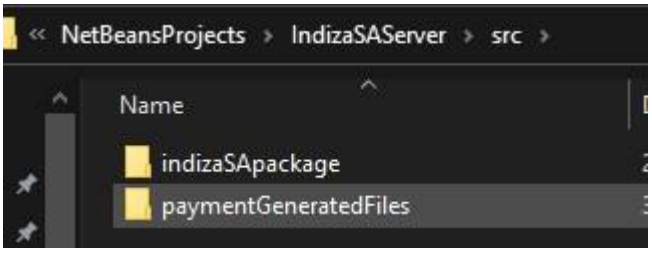


Figure 90

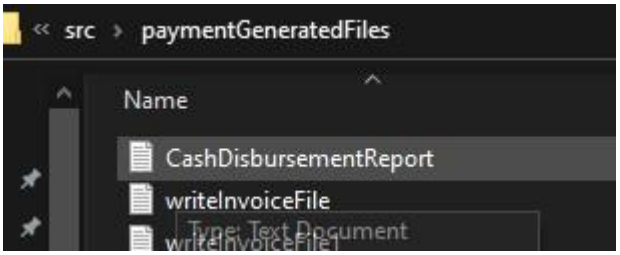
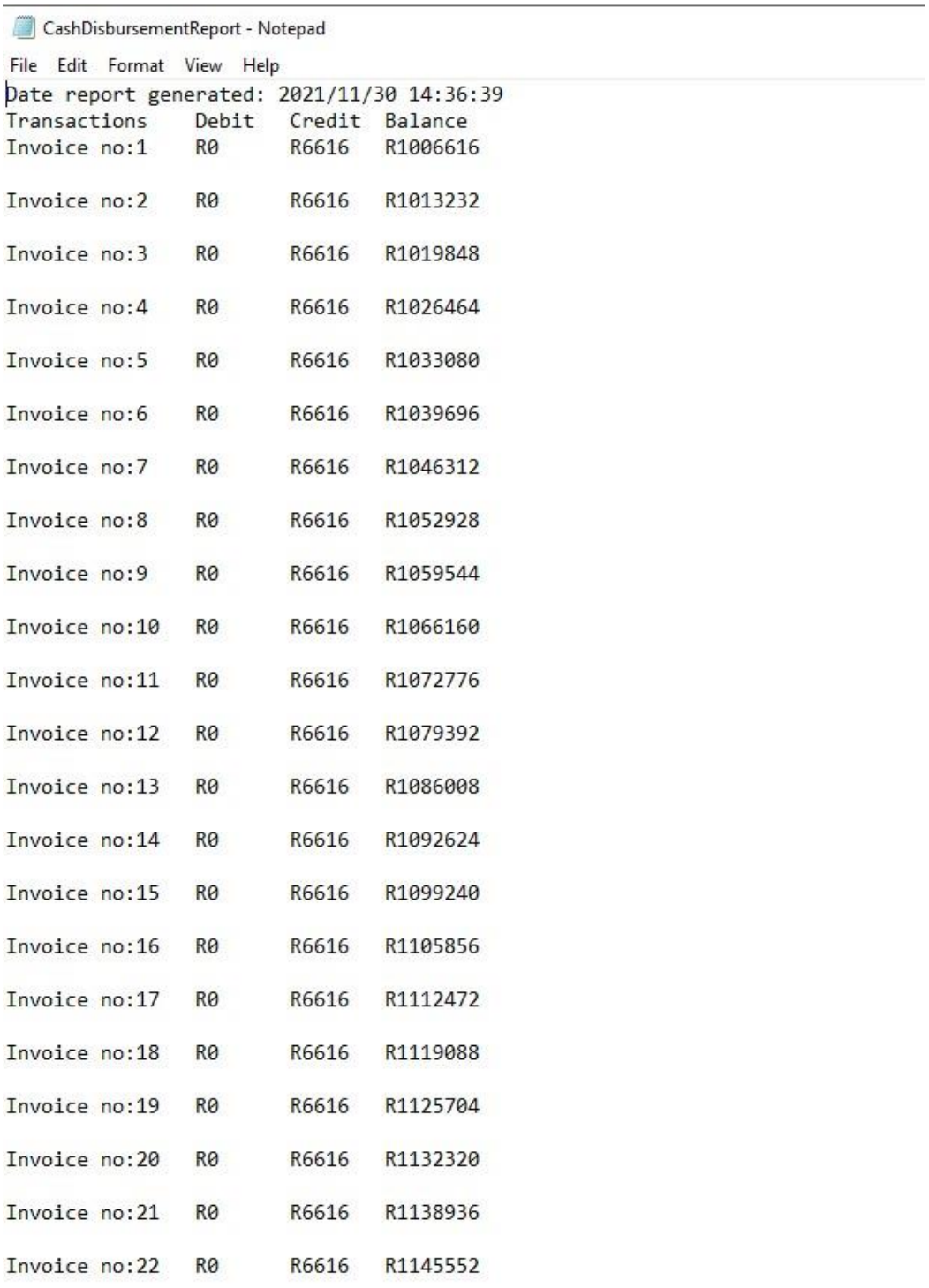


Figure 91

The report will display as shown in figure 92.


CashDisbursementReport - Notepad

File Edit Format View Help

Date report generated: 2021/11/30 14:36:39

Transactions	Debit	Credit	Balance
Invoice no:1	R0	R6616	R1006616
Invoice no:2	R0	R6616	R1013232
Invoice no:3	R0	R6616	R1019848
Invoice no:4	R0	R6616	R1026464
Invoice no:5	R0	R6616	R1033080
Invoice no:6	R0	R6616	R1039696
Invoice no:7	R0	R6616	R1046312
Invoice no:8	R0	R6616	R1052928
Invoice no:9	R0	R6616	R1059544
Invoice no:10	R0	R6616	R1066160
Invoice no:11	R0	R6616	R1072776
Invoice no:12	R0	R6616	R1079392
Invoice no:13	R0	R6616	R1086008
Invoice no:14	R0	R6616	R1092624
Invoice no:15	R0	R6616	R1099240
Invoice no:16	R0	R6616	R1105856
Invoice no:17	R0	R6616	R1112472
Invoice no:18	R0	R6616	R1119088
Invoice no:19	R0	R6616	R1125704
Invoice no:20	R0	R6616	R1132320
Invoice no:21	R0	R6616	R1138936
Invoice no:22	R0	R6616	R1145552

Figure 92

The second option available to the user, is to show all the transactions made by a particular user (as shown in figure 93). Every transaction of a user is written to a file and then extracted into LONG format to be stored in the MySQL database (specifically the transactionrecords_table).

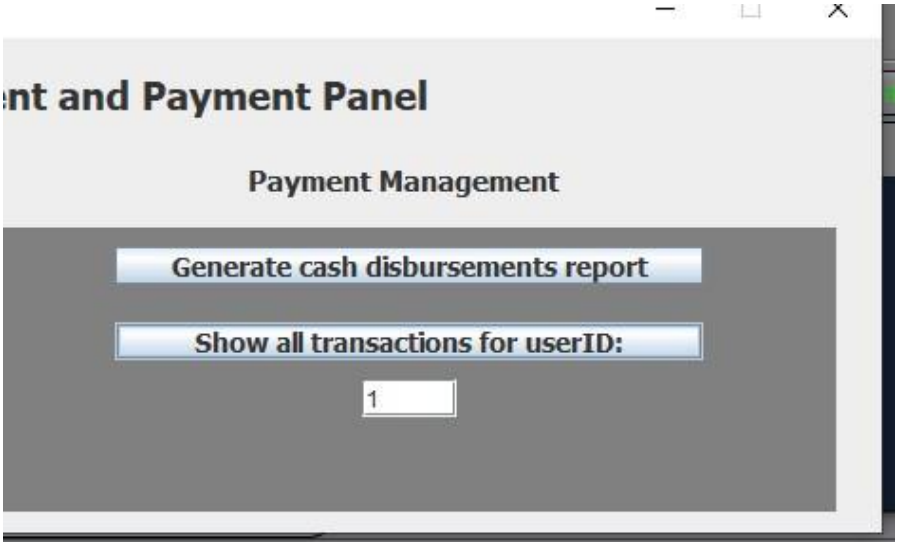


Figure 93

The second option requires user input in the form of the user ID of the registered user to be viewed. Upon clicking on the button, a pop-up notification will inform the user that the report has been generated in the ProgramData/MySQL server folder (as shown in figure 94).

This folder had to be used, as MySQL does not allow written data files to be downloaded to another location - due to security reasons. This can be changed, but requires tinkering in the .ini files and changing privilege settings – a laborious and complicated process.

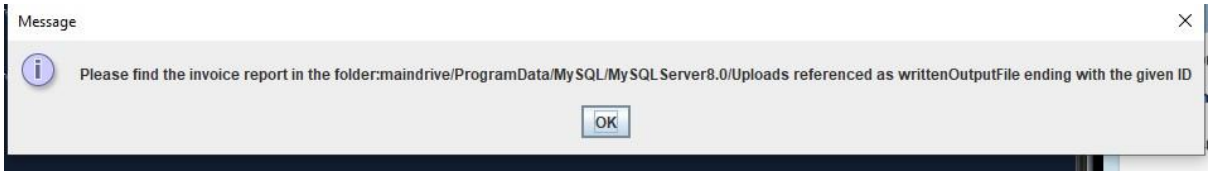


Figure 94

The file will appear in the Uploads folder of the MySQL Server folder (as shown in figure 95). The naming convention followed is “QueryOutput”, followed by the user ID (in this case ID number 1), followed by a report number that is generated by a random number generator. This ensures that individual reports can be created and saved to the hard drive, instead of the file being overwritten.

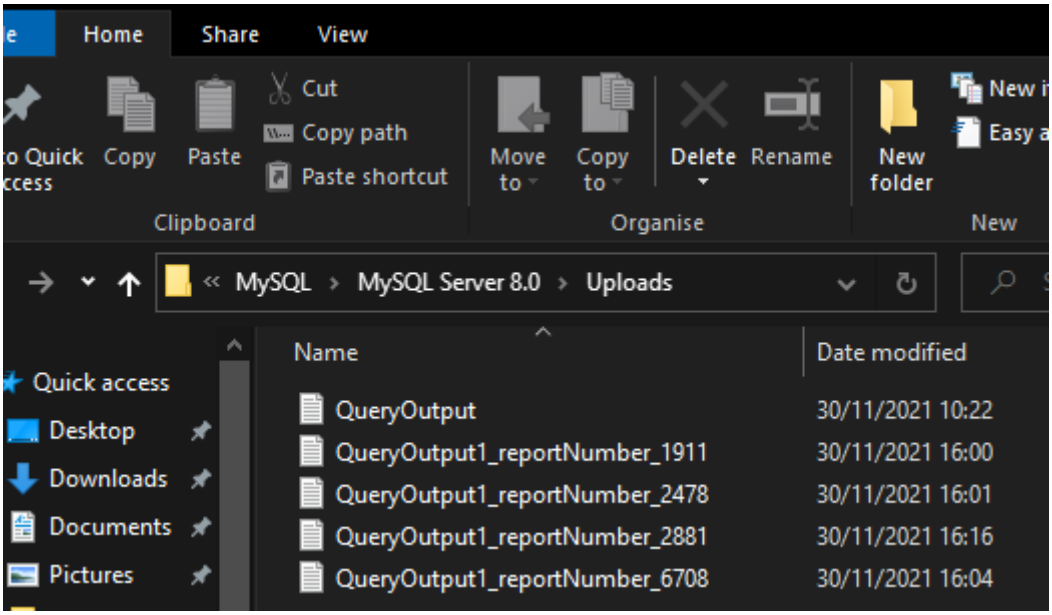


Figure 95

2.3 Payment and related methods module

The payment module is covered extensively in 2.4. It is heavily linked to the integration process and to separate the two would lead to redundant information in this report.

2.4 Integration of payment management system with flight system and database

The integration of the payment module and receipt module with the application will now be extensively covered. Three use cases will be used to show how the distributed system works.

Use case: Flight booking failure - User ID 1

Client-side:

The booking table is displayed in figure 96. The price of the flight is R998.

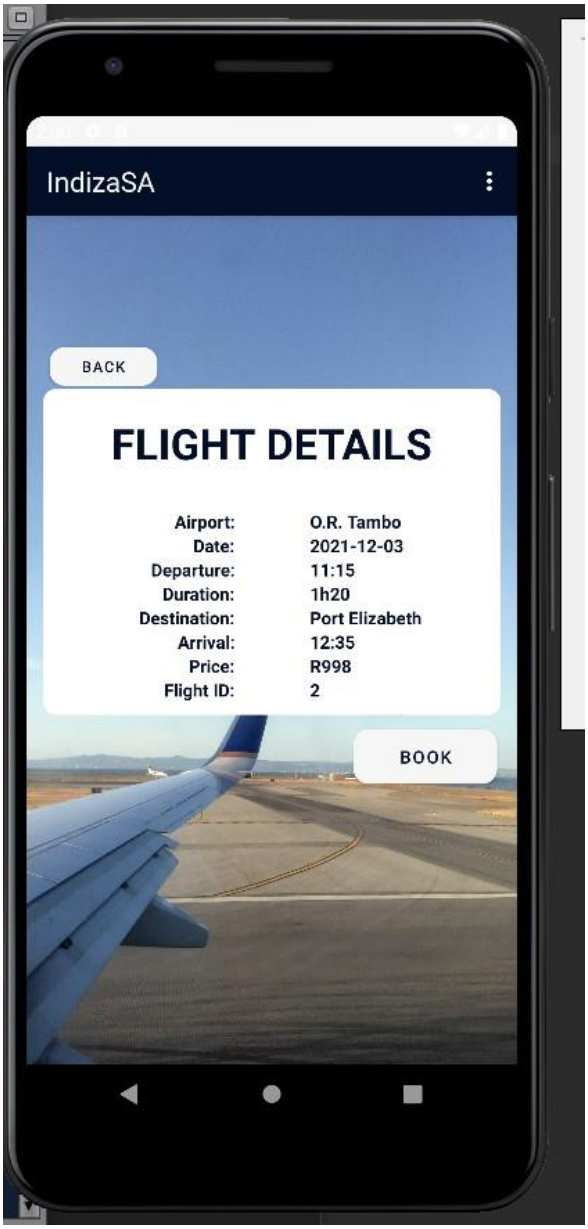


Figure 96

The user's profile has an available credit of R4, as shown in figure 97.

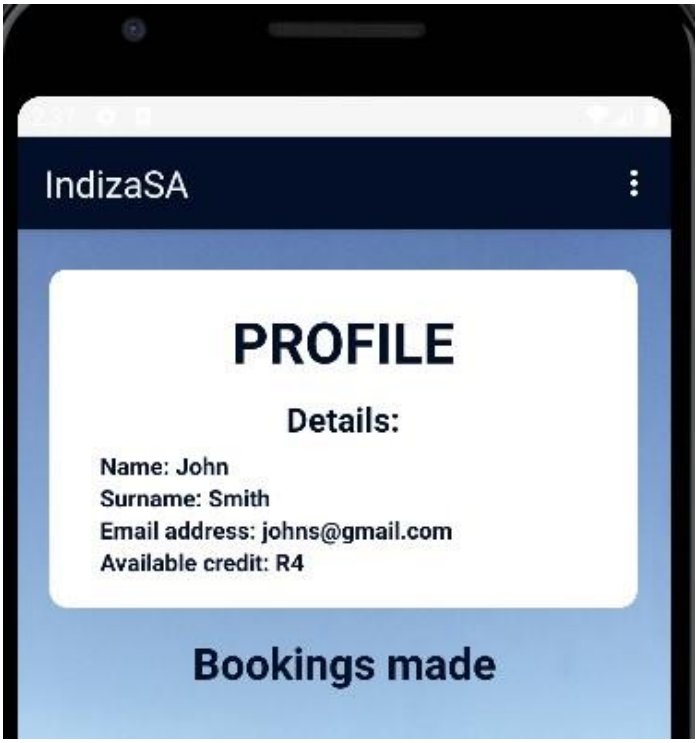


Figure 97

If the user navigates to the booking screen and tries to book the flight by clicking on the "Book" button, a pop-up notification will inform the user that he has insufficient funds to complete the booking (shown in figure 98).

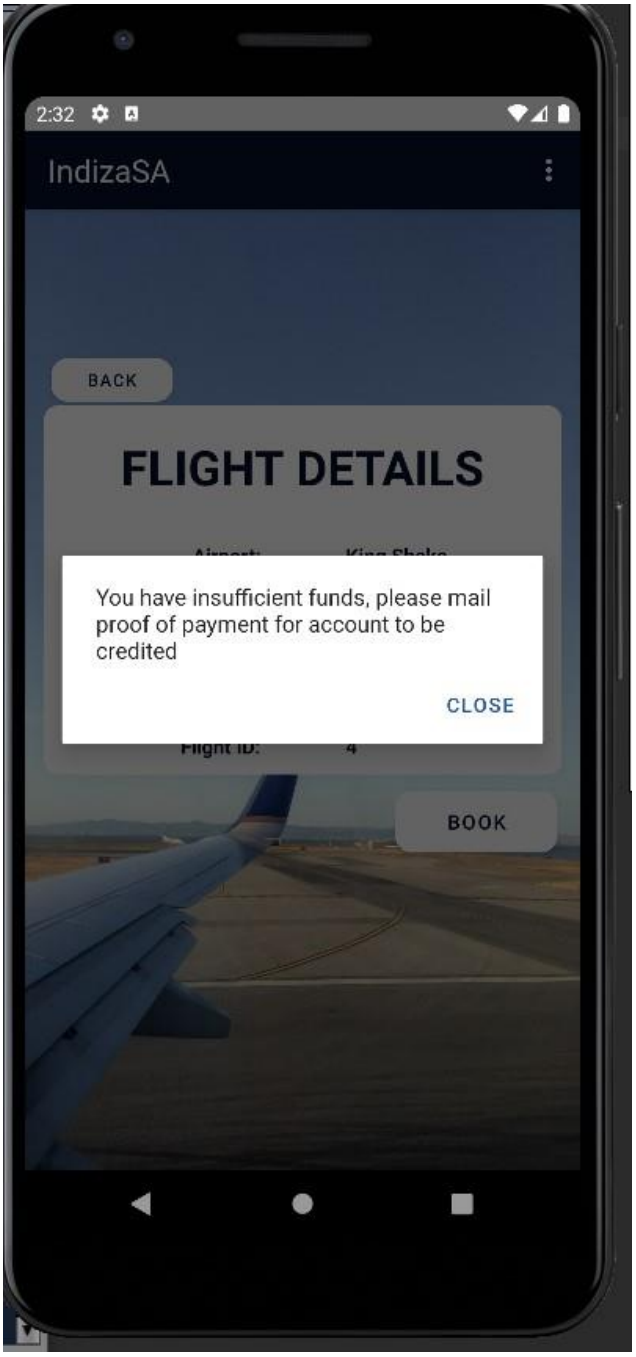


Figure 98

Server-side:

The server output indicates that the user's credit status was checked by the payment module and denied due to insufficient funds (shown in figure 99). The server exits the booking activity and awaits further instructions from the client application.

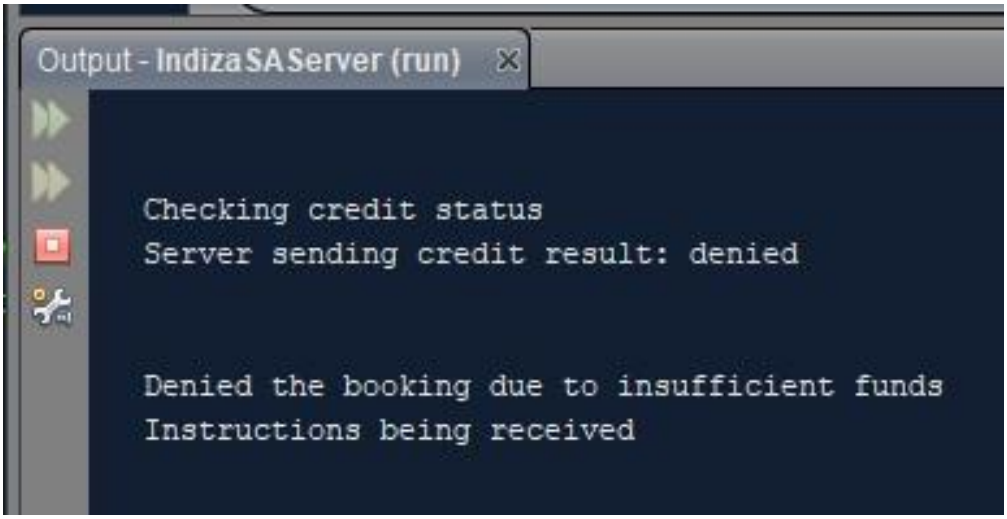


Figure 99

Use case: Booking success - Flight ID 18, User ID 1

Client-side showing listed flight before booking:

The flight is listed in the home activity and has 30 seats left (as shown in figure 100)

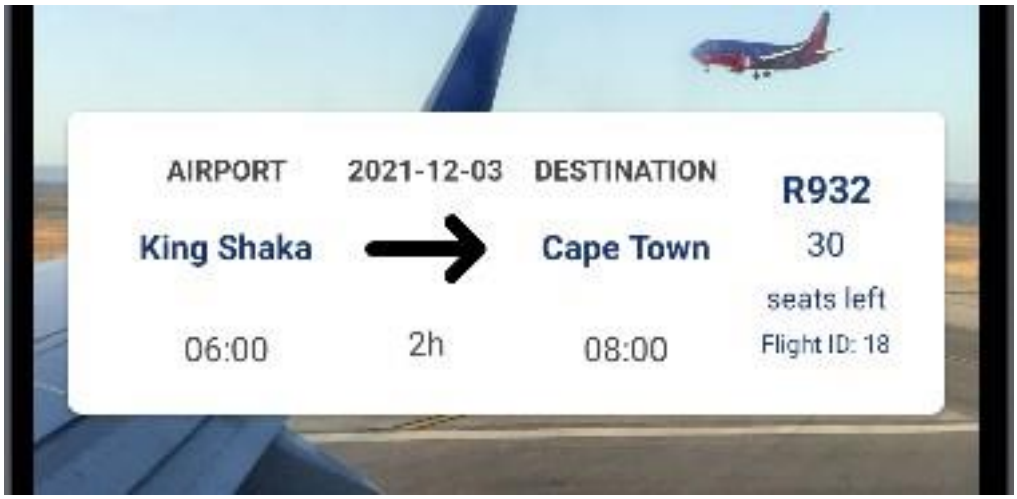


Figure 100

Server-side showing listed flight before booking:

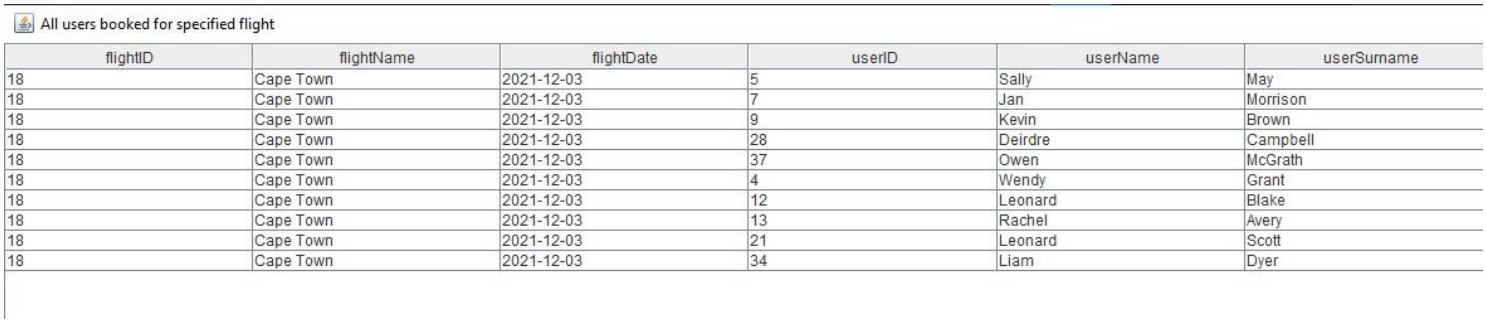
Using the server GUI, one can view all the flights on the database. Flight ID 18, listed last, shows to have 30 seats left for booking (as shown in figure 101).

flightID	flightName	flightPrice	flightDate	deptTime	duration	arrivalTime	numSeatsLeft	airport
4	Johannesburg	3252	2021-12-16	11:40	1h20	13:20	22	King Shaka
5	Johannesburg	841	2021-12-20	10:05	2h10	11:55	10	Cape Town
6	Harare	1444	2021-11-15	11:40	1h35	13:15	11	O.R. Tambo
7	Cape Town	2519	2021-12-24	07:15	2h10	09:25	39	King Shaka
8	Durban	1804	2021-11-24	06:40	2h	08:40	11	O.R. Tambo
9	Johannesburg	3152	2021-12-09	11:40	1h40	13:20	15	Cape Town
10	Johannesburg	4118	2021-11-14	14:20	1h50	16:10	12	King Shaka
11	Mauritius	6616	2021-12-07	09:55	6h10	16:05	0	O.R. Tambo
12	Johannesburg	818	2021-12-03	09:40	1h10	10:50	5	Cape Town
13	Cape Town	1117	2021-11-17	14:30	2h	16:30	25	O.R. Tambo
14	Mauritius	4639	2021-12-02	17:10	2h30	19:40	35	O.R. Tambo
15	Victoria Falls	2232	2021-11-27	13:45	1h35	15:20	2	O.R. Tambo
16	Victoria Falls	2014	2021-11-20	11:30	1h37	13:07	3	O.R. Tambo
17	Johannesburg	716	2021-12-03	09:40	1h10	10:50	15	King Shaka
18	Cape Town	932	2021-12-03	06:00	2h	08:00	30	King Shaka

Figure 101

Server-side showing listed users for flight ID 18 before booking:

The flight list shows that the last user booked for the flight is user ID 34 as shown in figure 102.



flightID	flightName	flightDate	userID	userName	userSurname
18	Cape Town	2021-12-03	5	Sally	May
18	Cape Town	2021-12-03	7	Jan	Morrison
18	Cape Town	2021-12-03	9	Kevin	Brown
18	Cape Town	2021-12-03	28	Deirdre	Campbell
18	Cape Town	2021-12-03	37	Owen	McGrath
18	Cape Town	2021-12-03	4	Wendy	Grant
18	Cape Town	2021-12-03	12	Leonard	Blake
18	Cape Town	2021-12-03	13	Rachel	Avery
18	Cape Town	2021-12-03	21	Leonard	Scott
18	Cape Town	2021-12-03	34	Liam	Dyer

Figure 102

Client-side showing user account before booking:

The user account indicates that the user has R2000 credit available (as shown in figure 103). It also shows that the user only has one current flight booked – flight ID 11.

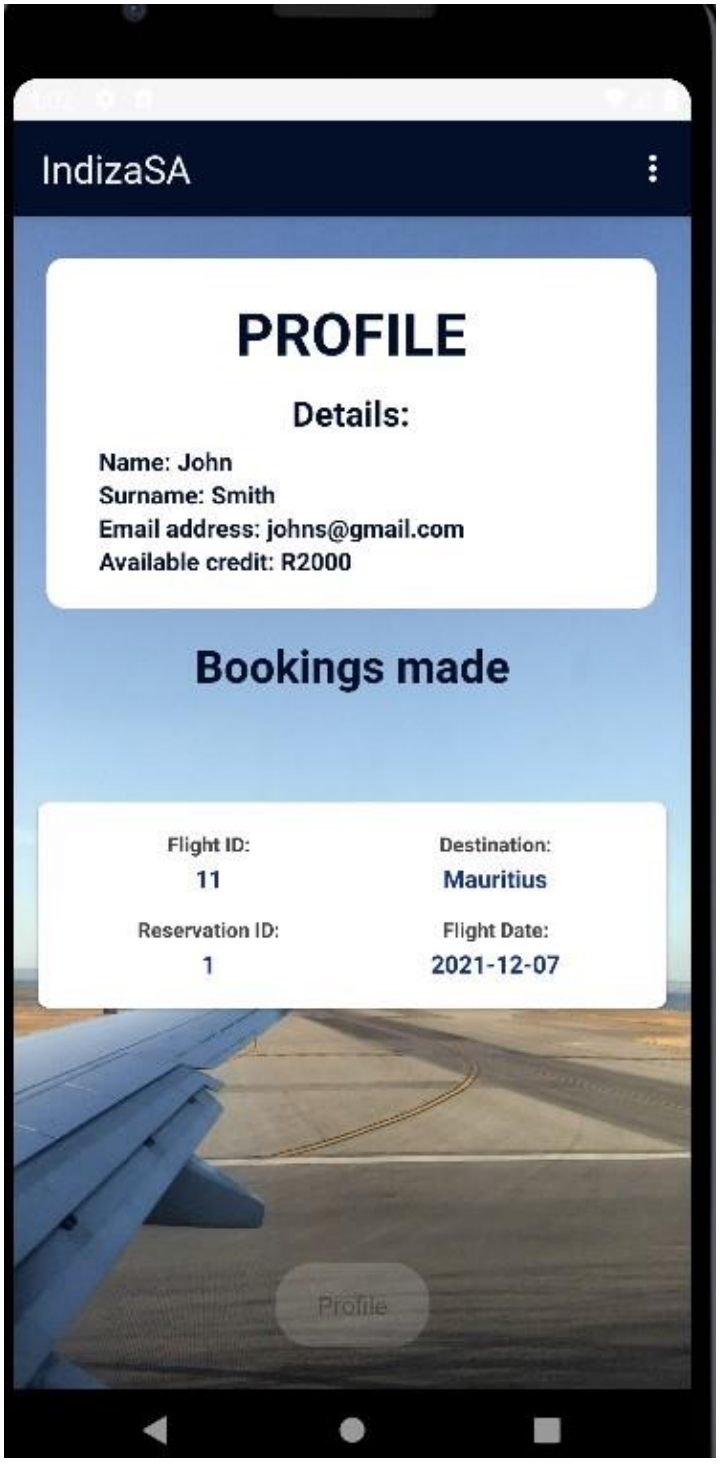


Figure 103

Server-side showing user account before booking:

Clicking on the “Show all users” button displays all registered users on the database. The data of the user with ID 1 is shown to correlate with the user data displayed on the client-side (shown in figure 104).

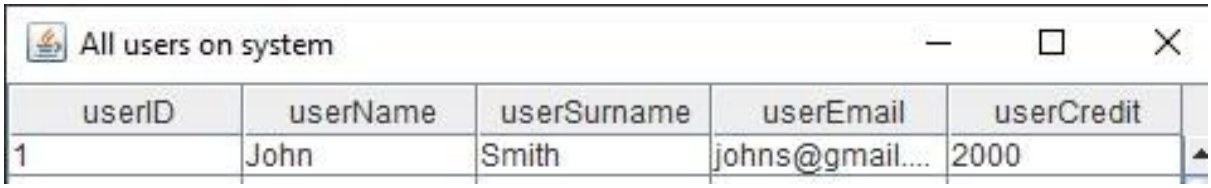


Figure 104

Client-side showing listed flight after booking:

After a successful booking, the flight details show that 29 seats remain (shown in figure 105) – thus the booking led to a decrease in seats compared to figure 100.

Figure 105

Server-side showing listed flight after booking:

This deduction correlates on the server-side, as shown in figure 106. Flight 18, appearing at the bottom of figure 106, shows that the number of seats left have decreased to 29.

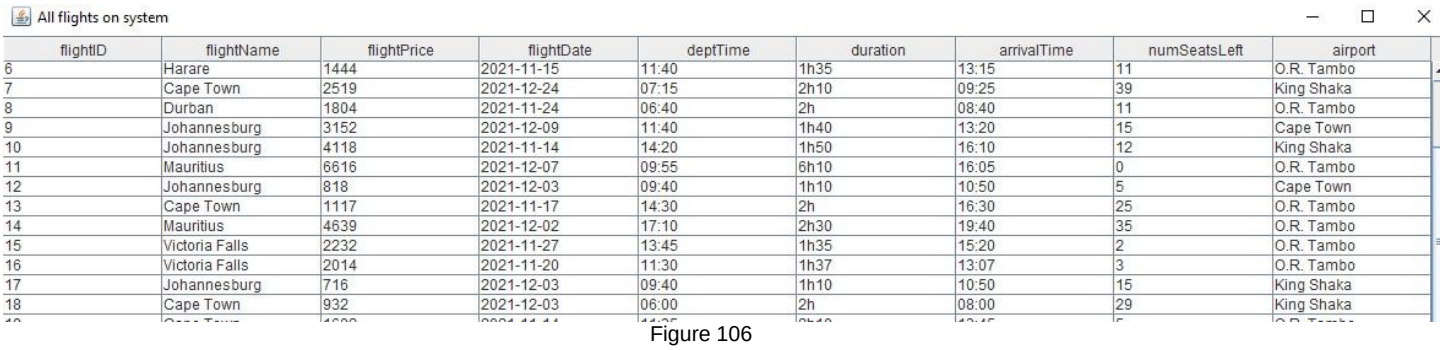


Figure 106

Server-side showing listed users for flight ID 18 before booking:

By clicking on show all users for a specific flight, in this case flight ID 18, it is shown that the latest booking was made by the user with ID 1 (shown in figure 107 and 108). This correlates with the previously displayed flight list (figure 102).

Figure 107

All users booked for specified flight

flightID	flightName	flightDate	userID	userName	userSurname	userEmail
18	Cape Town	2021-12-03	5	Sally	May	salmay@gmail.com
18	Cape Town	2021-12-03	7	Jan	Morrison	jmorris@gmail.com
18	Cape Town	2021-12-03	9	Kevin	Brown	kbrown@gmail.com
18	Cape Town	2021-12-03	28	Deirdre	Campbell	dcampbell@gmail.com
18	Cape Town	2021-12-03	37	Owen	McGrath	omcgrath@gmail.com
18	Cape Town	2021-12-03	4	Wendy	Grant	wendyg@gmail.com
18	Cape Town	2021-12-03	12	Leonard	Blake	lblake@gmail.com
18	Cape Town	2021-12-03	13	Rachel	Avery	ravery@gmail.com
18	Cape Town	2021-12-03	21	Leonard	Scott	ls@gmail.com
18	Cape Town	2021-12-03	34	Liam	Dyer	liamdy@gmail.com
18	Cape Town	2021-12-03	1	John	Smith	johns@gmail.com

Figure 108

Client-side showing user account after booking:

The user now has two flights listed under the bookings heading, shown in figure 109. Also note that the user now has R1068 left in his account.

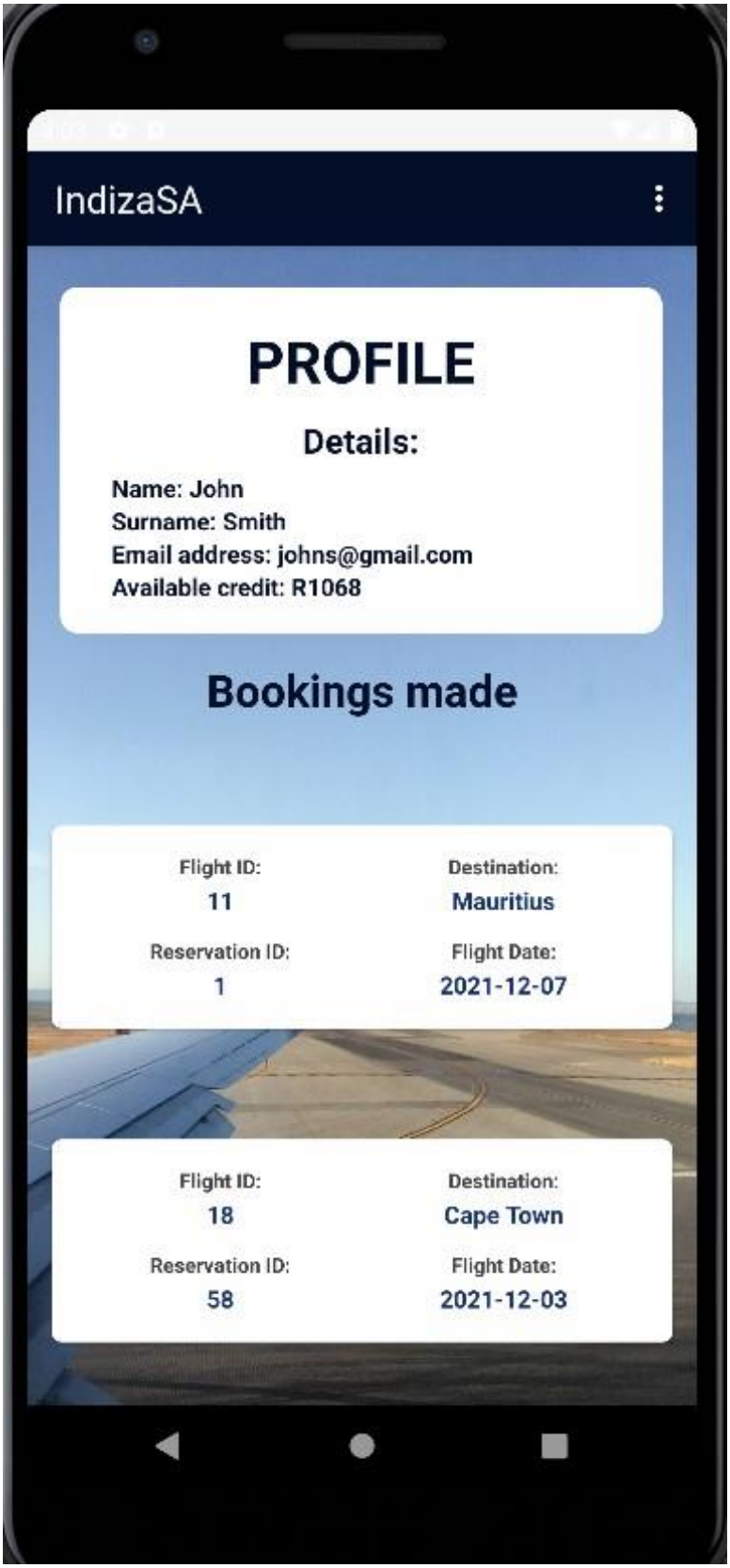

IndizaSA
PROFILE
Details:
Name: John
Surname: Smith
Email address: johns@gmail.com
Available credit: R1068
Bookings made
Flight ID:
11
Destination:
Mauritius
Reservation ID:
1
Flight Date:
2021-12-07
Flight ID:
18
Destination:
Cape Town
Reservation ID:
58
Flight Date:
2021-12-03

Figure 109

Server-side showing user after booking:

The data relating to user credit correlates after the booking, showing that the user has R1068 credit left in his account.

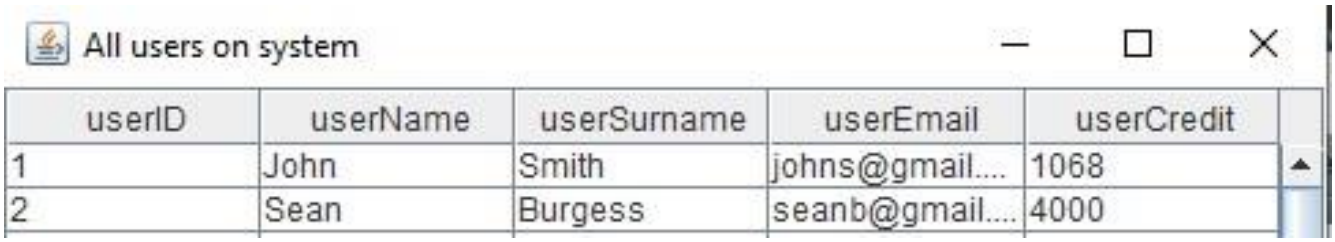


Figure 110

The receipt modules

User transactions

By entering ID 1 and clicking on the “Show user transactions” on the server GUI, a file is generated confirming the transaction that took place. The last line shows that the user with ID 1 booked a flight, number 18. The flight price was R932 and the balance after deducting the amount from his account is R1068. This is displayed in figure 111.

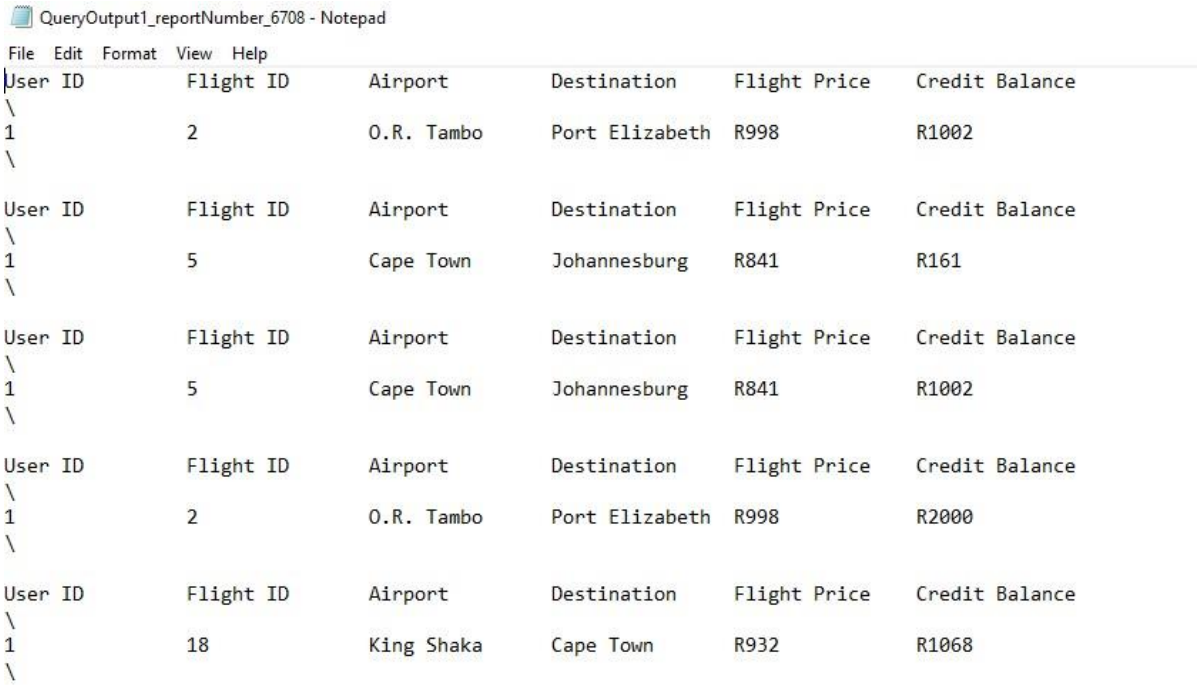


Figure 111

Cash disbursements report

By clicking on the “Generate report” button in the server GUI, a cash disbursement report is generated – showing all the transactions that have taken place for IndizaSA (shown in figure 112).

Invoice no.56, the last one generated invoice, displays that IndizaSA’s account was credited with R932 – the cost of a ticket for flight ID 18.

CashDisbursementReport - Notepad

File Edit Format View Help

```
Invoice no:31   R0      R6616   R1205096

Invoice no:32   R0      R6616   R1211712

Invoice no:33   R0      R6616   R1218328

Invoice no:34   R0      R6616   R1224944

Invoice no:35   R0      R6616   R1231560

Invoice no:36   R0      R6616   R1238176

Invoice no:37   R0      R6616   R1244792

Invoice no:38   R0      R6616   R1251408

Invoice no:39   R0      R6616   R1258024

Invoice no:40   R0      R6616   R1264640

Invoice no:41   R0      R4639   R1269279

Invoice no:42   R0      R4639   R1273918

Invoice no:43   R0      R4639   R1278557

Invoice no:44   R0      R4639   R1283196

Invoice no:45   R0      R4639   R1287835

Invoice no:46   R0      R932    R1288767

Invoice no:47   R0      R932    R1289699

Invoice no:48   R0      R932    R1290631

Invoice no:49   R0      R932    R1291563

Invoice no:50   R0      R932    R1292495

Invoice no:51   R0      R932    R1293427

Invoice no:52   R0      R932    R1294359

Invoice no:53   R0      R932    R1295291

Invoice no:54   R0      R932    R1296223

Invoice no:55   R0      R932    R1297155

Invoice no:56   R0      R932    R1298087

......  ......  Final Balance: R1298087
```

Figure 112

Server output

The server output gives a clear view on what events transpired (shown in figure 113).

The user clicked on the card of flight ID 18. By doing this, the client sent the server instructions indicating that the booking process would start.

Upon clicking on the “Book” button, the server was sent the user’s ID for credit checking and approval.

The credit check succeeded and triggers the reservation module on the server side.

After inserting the data into the database, the payment module adjusts the user’s credit balance to reflect correctly.

The payment table is also updated with the invoice number and reservation ID.

The data report for the user’s transaction is written by the receipt module and inserted into the database.

The booking module on the server-side adjusts the number of seats left on flight ID 18 by updating the MySQL flight table.

Lastly, the server listens for the next instruction from the client.

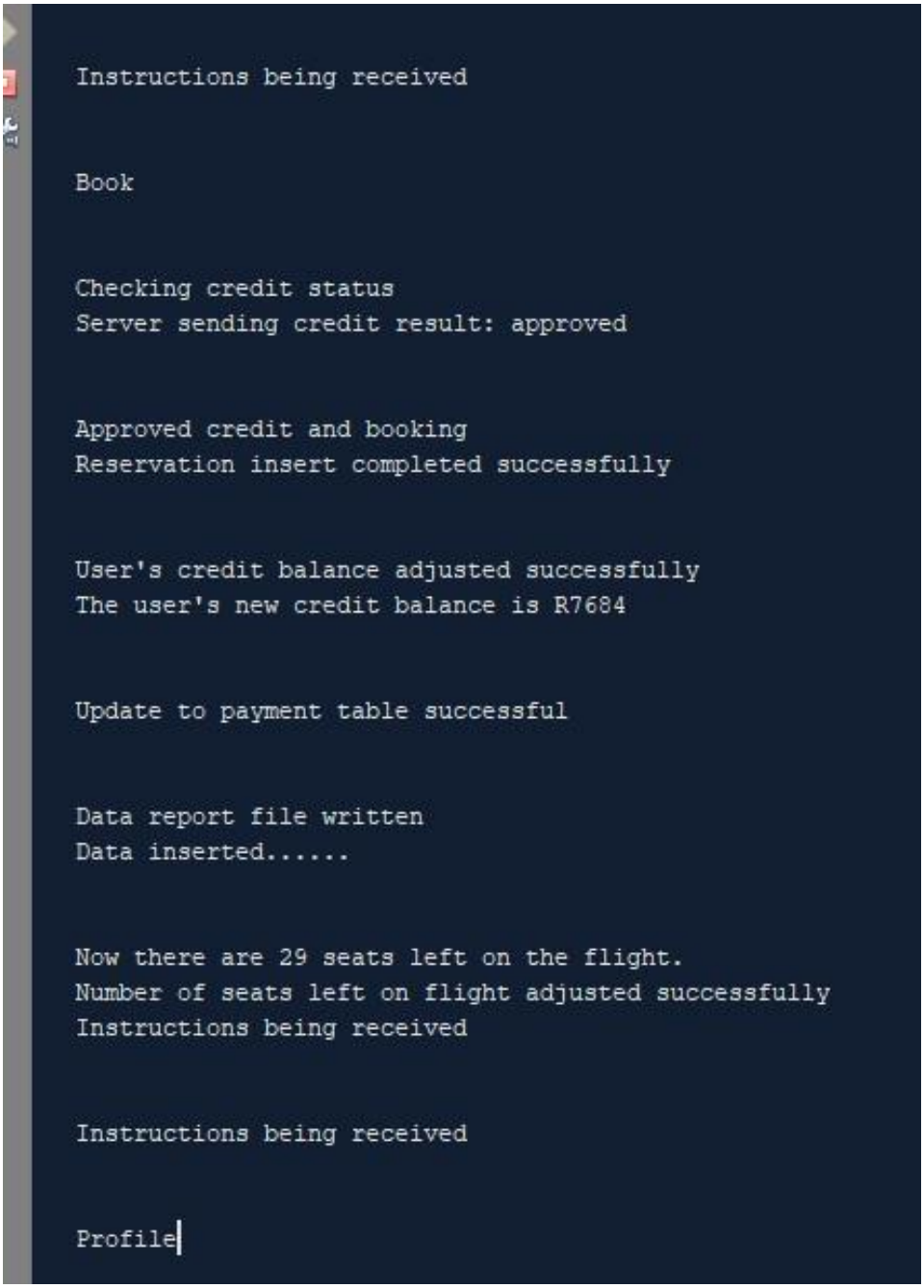

Instructions being received
Book
Checking credit status
Server sending credit result: approved
Approved credit and booking
Reservation insert completed successfully
User's credit balance adjusted successfully
The user's new credit balance is R7684
Update to payment table successful
Data report file written
Data inserted.....
Now there are 29 seats left on the flight.
Number of seats left on flight adjusted successfully
Instructions being received
Instructions being received
Profile

Figure 113

Use case: Booking cancellation – Flight ID 11, User ID 1

Client-side Profile screen

By clicking on a listed flight in the profile screen, a pop-up notification is used to confirm or cancel a user's wish to cancel a booking made (shown in figure 114).

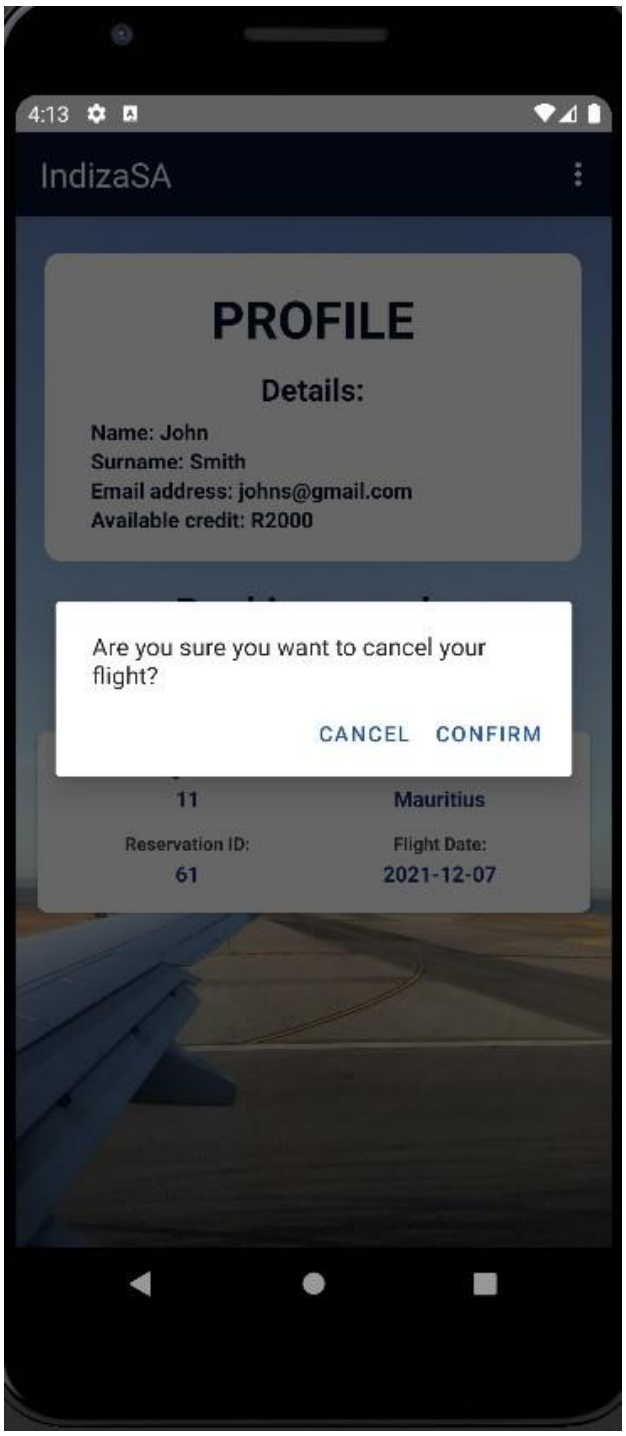


Figure 114

Server-side flight listed before cancellation

By clicking on “Show all flights” using the server GUI, one can see that the flight at the bottom of figure 115, flight ID 11, has 0 seats left.

All flights on system

flightID	flightName	flightPrice	flightDate	deptTime	duration	arrivalTime	numSeatsLeft	airport
6	Harare	1444	2021-11-15	11:40	1h35	13:15	11	O.R. Tambo
7	Cape Town	2519	2021-12-24	07:15	2h10	09:25	39	King Shaka
8	Durban	1804	2021-11-24	06:40	2h	08:40	11	O.R. Tambo
9	Johannesburg	3152	2021-12-09	11:40	1h40	13:20	15	Cape Town
10	Johannesburg	4118	2021-11-14	14:20	1h50	16:10	12	King Shaka
11	Mauritius	6616	2021-12-07	09:55	6h10	16:05	0	O.R. Tambo

Figure 115

Client-side flight listed before cancellation

This is also shown to be the case on the client side, as can be seen in figure 116.

Figure 116

Server-side users listed on flight 11 before cancellation

Figure 117 shows that the first reservation made was by the user with ID 1. He is currently listed on the booked flight list.

All users booked for specified flight

flightID	flightName	flightDate	userID	userName	userSurname	userEmail
11	Mauritius	2021-12-07	1	John	Smith	johns@gmail.com
11	Mauritius	2021-12-07	2	Sean	Burgess	seanb@gmail.com
11	Mauritius	2021-12-07	3	Paul	Hodges	paulh@gmail.com

Figure 117

Server-side user account before cancellation

Figure 118 shows that the user with ID 1 has a credit balance of R2000.

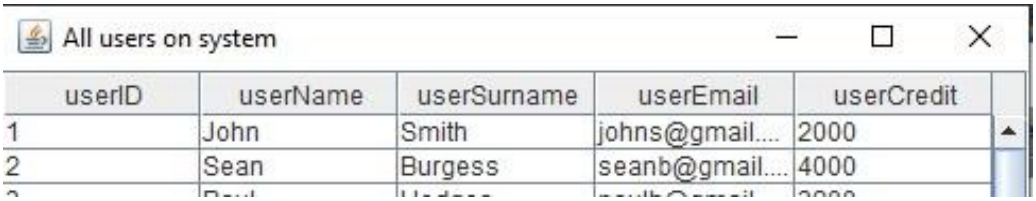


Figure 118

Client-side user account before cancellation

Figure 119 shows that the user currently has R2000 credit available and one flight listed, with ID 11, under his bookings.

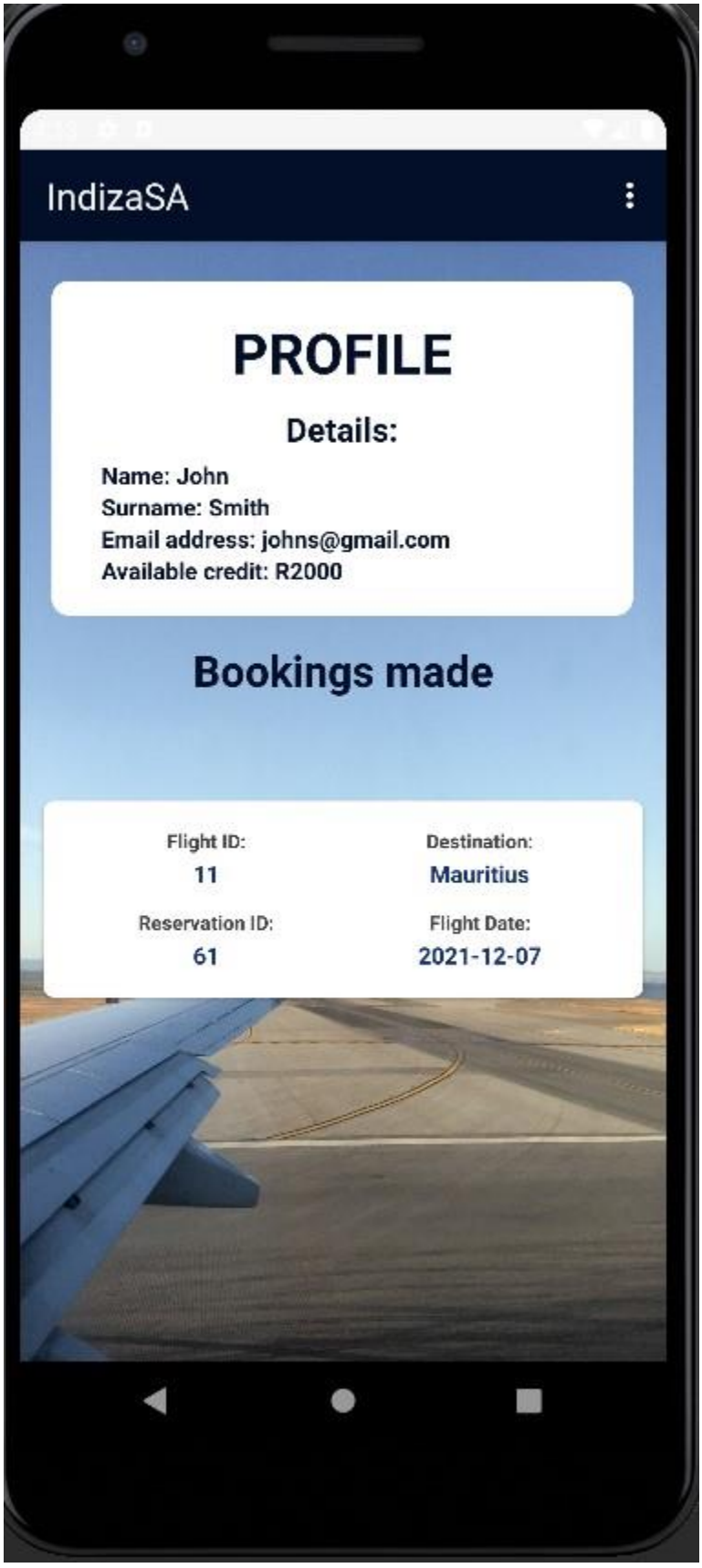

IndizaSA
PROFILE
Details:
Name: John
Surname: Smith
Email address: johns@gmail.com
Available credit: R2000
Bookings made
Flight ID:
11
Destination:
Mauritius
Reservation ID:
61
Flight Date:
2021-12-07

Figure 119

Server-side flight listed after cancellation

Viewing the flights available after the cancellation will show that the number of seats left for flight 11 has now increased with 1 (as shown in figure 120)

All flights on system

flightID	flightName	flightPrice	flightDate	deptTime	duration	arrivalTime	numSeatsLeft	airport
6	Harare	1444	2021-11-15	11:40	1h35	13:15	11	O.R. Tambo
7	Cape Town	2519	2021-12-24	07:15	2h10	09:25	39	King Shaka
8	Durban	1804	2021-11-24	06:40	2h	08:40	11	O.R. Tambo
9	Johannesburg	3152	2021-12-09	11:40	1h40	13:20	15	Cape Town
10	Johannesburg	4118	2021-11-14	14:20	1h50	16:10	12	King Shaka
11	Mauritius	6616	2021-12-07	09:55	6h10	16:05	1	O.R. Tambo

Figure 120

Client-side flight listed after cancellation

Returning to the Home screen, the flight with ID 11 shows to now have 1 available seat after the booking was cancelled, correlating with the server-side data (as shown in figure 121).

Figure 121

Server-side users listed after cancellation

Viewing flight with ID 11 through use of the GUI, will reveal that user ID 1 has disappear from the booking list (as shown in figure 122).

All users booked for specified flight

flightID	flightName	flightDate	userID	userName	userSurname	userEmail
11	Mauritius	2021-12-07	2	Sean	Burgess	seanb@gmail.com
11	Mauritius	2021-12-07	3	Paul	Hodges	paulh@gmail.com
11	Mauritius	2021-12-07	4	Wendy	Grant	wendyg@gmail.com
11	Mauritius	2021-12-07	5	Sally	May	salmay@gmail.com
11	Mauritius	2021-12-07	6	Sean	Burgess	sb@gmail.com
11	Mauritius	2021-12-07	7	Jan	Morrison	jmorris@gmail.com
11	Mauritius	2021-12-07	8	Liam	Turner	lturner@gmail.com
11	Mauritius	2021-12-07	9	Kevin	Brown	kbrown@gmail.com
11	Mauritius	2021-12-07	10	Andrea	Payne	apayne@gmail.com
11	Mauritius	2021-12-07	11	Kylie	Young	kyoung@gmail.com
11	Mauritius	2021-12-07	12	Leonard	Blake	lblake@gmail.com
11	Mauritius	2021-12-07	13	Rachel	Avery	ravery@gmail.com

Figure 122

Server-side user account after cancellation

Viewing a list of all the registered users will show that the user with ID 1 now has R8616 credit in his account, whereas before the cancellation he only had R2000 in his account (shown in figure 123)

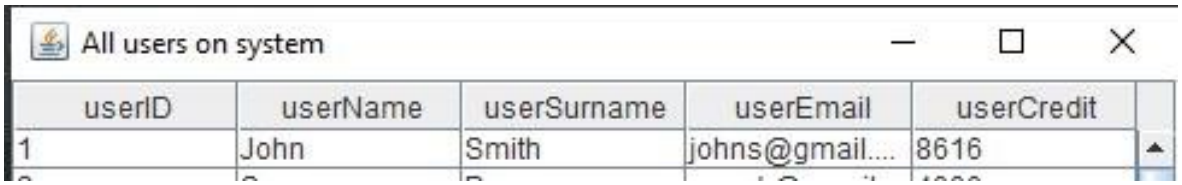


Figure 123

Client-side user account after cancellation

The user's profile correlates the data that the server shows. The user now has R8616 credit available and no flights listed under bookings (as shown in figure 124).

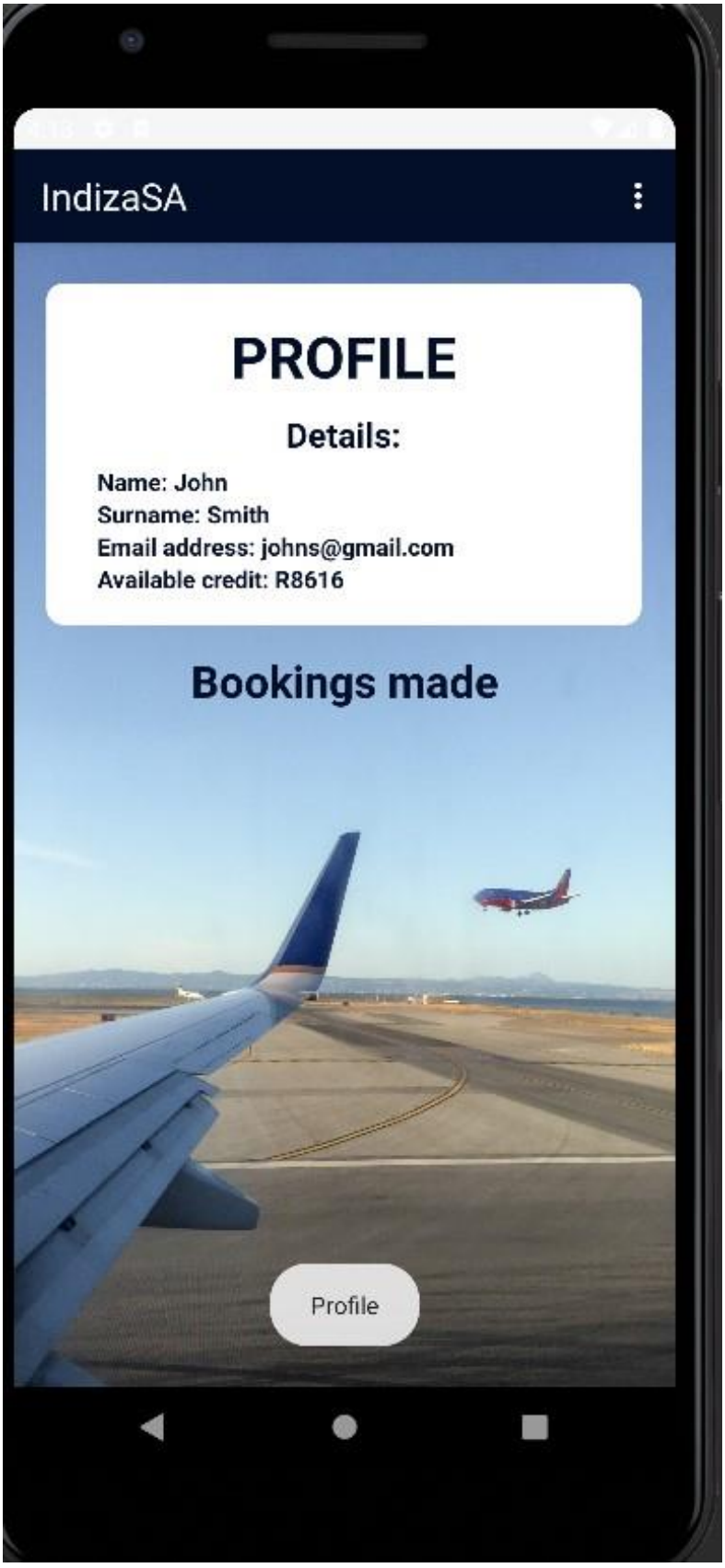

IndizaSA
PROFILE
Details:
Name: John
Surname: Smith
Email address: johns@gmail.com
Available credit: R8616
Bookings made
Profile

Figure 124

The receipt modules

User transactions

The user’s transaction report indicates that his last transaction resulted was related to flight with ID 11 (shown in figure 125). It further shows that his account was credited with R6616, as he now has a balance of R8616. This data correlates with the data on the server and what is shown on the client-side.

QueryOutput1_reportNumber_2881 - Notepad

File Edit Format View Help

```
\
1          5          Cape Town     Johannesburg     R841           R1002
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          2          O.R. Tambo    Port Elizabeth   R998           R2000
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          18         King Shaka    Cape Town        R932           R1068
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          18         King Shaka    Cape Town        R932           R2000
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          18         King Shaka    Cape Town        R932           R1068
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          18         King Shaka    Cape Town        R932           R2000
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          18         King Shaka    Cape Town        R932           R1068
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          18         King Shaka    Cape Town        R932           R2000
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          11         O.R. Tambo    Mauritius        R6616          R8616
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          11         O.R. Tambo    Mauritius        R6616          R2000
\

User ID    Flight ID  Airport       Destination      Flight Price   Credit Balance
\
1          11         O.R. Tambo    Mauritius        R6616          R8616
\
```

Figure 125

Cash Disbursement Report

The report indicates that the last transaction, invoice no. 61, had R6616 listed in the debit column. IndizaSA thus had to reimburse the user after the booked flight's cancellation. The balance of the company is negatively affected and lowered as shown in figure 126.

CashDisbursementReport - Notepad

File Edit Format View Help

```
Invoice no:37   R0      R6616   R1238176

Invoice no:38   R0      R6616   R1244792

Invoice no:39   R0      R6616   R1251408

Invoice no:40   R0      R6616   R1258024

Invoice no:41   R0      R4639   R1262663

Invoice no:42   R0      R4639   R1267302

Invoice no:43   R0      R4639   R1271941

Invoice no:44   R0      R4639   R1276580

Invoice no:45   R0      R4639   R1281219

Invoice no:46   R0      R932    R1282151

Invoice no:47   R0      R932    R1283083

Invoice no:48   R0      R932    R1284015

Invoice no:49   R0      R932    R1284947

Invoice no:50   R0      R932    R1285879

Invoice no:51   R0      R932    R1286811

Invoice no:52   R0      R932    R1287743

Invoice no:53   R0      R932    R1288675

Invoice no:54   R0      R932    R1289607

Invoice no:55   R0      R932    R1290539

Invoice no:1    R6616   R0      R1283923

Invoice no:56   R998    R0      R1282925

Invoice no:57   R841    R0      R1282084

Invoice no:58   R932    R0      R1281152

Invoice no:59   R932    R0      R1280220

Invoice no:60   R932    R0      R1279288

Invoice no:61   R6616   R0      R1272672

......  ......  Final Balance: R1272672
```

Figure 126

Server output

The server output shows that the client sent the server instructions, indicating that a cancellation of a booking was to take place. The client also sent the server the required data, such as the invoice ID that is printed (number 61). The server further indicates that the payment module credited the user's account positively and that the data related to the transaction was passed to the receipt module.

Lastly, it indicates that the data report was written and inserted into the database and that the server is again waiting on instructions from the client (as shown in figure 127).

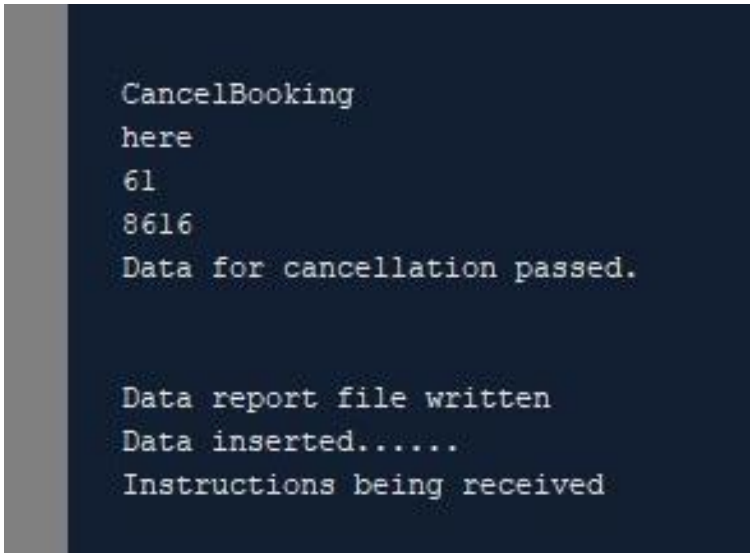


Figure 127

Referencing

Android. 2021. ServerSocket | Android Developers. *Android*, 18 February 2021. [Online] Available at: https://developer.android.com/reference/java/net/ServerSocket [Accessed: 13 November, 2021].

Behl R. & Dagar, R. 2012. Analysis of Effectiveness of Concurrency Control Techniques in Databases. *International Journal of Engineering Research & Technology (IJERT)*¸ 1(5), pp.1-5.

Bennett, S., McRobb, S. and Farmer, R. 2016. *Object-Oriented Systems Analysis and Design Using UML.* 4th Edition. Berkshire: McGraw Hill Higher Education.

Connolly, T. & Begg, C. 2015. *Database Systems: A Practical Approach to Design, Implementation, and Management.* 6th Edition. Harlow: Pearson Education Limited.

Drake, P. 2013. *Data Structures and Algorithms in Java.* 1st Edition. Harlow: Pearson Education Limited.

Friesen, J. 2019. Inheritance in Java, Part 1: The extends keyword. *Info World*, 7 May 2019. [Online] Available at: https://www.javaworld.com/article/2987426/java-101-inheritance-in-java-part-1.html [Accessed: 1 November, 2021].

GeeksForGeeks. 2020. *Difference between Natural Join and Inner Join in SQL*. Binary Terms. *GeeksForGeeks*, 31 August 2021. [Online] Available at: https://www.geeksforgeeks.org/difference-between-natural-join-and-inner-join-in-sql/ [Accessed: 20 November 2021].

Gupta, M.K., Arora, R.K. & Bhati, B.S. 2018. Study of Concurrency Control Techniques in Distributed DBMS. *International Journal of Maching Learning and Networked Collaborative Engineering*¸ 2(4), pp.180-187.

Hande, S. 2019. Manage Gradle Dependencies in Android Studio. *Medium,* 1 August 2019. [Online] Available at: https://medium.com/@sushanthande1/dependency-management-in-android-studio-cc4352122dcf [Accessed: 12 November, 2021].

Lathiya, A. 2019. Java Encapsulation Tutorial with Example. *AppDividend*, 6 January 2020. [Online] Available at: https://appdividend.com/2019/05/14/java-encapsulation-tutorial-with-example-encapsulation-in-java/ [Accessed: 1 November, 2021].

Marques, G. 2012. How to connect to an Android server socket in the emulator. *Bytes Lounge*, 22 December 2012. [Online] Available at: https://www.byteslounge.com/tutorials/how-to-connect-to-an-android-server-socket-in-the-emulator [Accessed: 11 November, 2021].

Mohan, P. 2013. *Fundamentals of Object-Oriented Programming in Java*. America: Bell & Bain CreateSpace Independent Publishing Platform.

Podeswa, H. 2010. *UML for the IT Business Analyst: A Practical guide to Requirements Gathering Using the Unified Modelling Language.* 2nd Edition. Boston: Cengage Learning.

TutorialsPoint. n.d. Java RMI - Introduction. *TutorialsPoint.* [Online] Available at: https://www.tutorialspoint.com/java_rmi/java_rmi_introduction.htm [Accessed: 10 November, 2021].

Yilmaz, O. 2020. Android Device Matching with Socket Programming. *DZone*, 26 June 2020. [Online] Available at: https://dzone.com/articles/android-device-matching-with-socket-programming [Accessed: 8 November, 2021].