**CHAPTER FIVE**

**SUMMARY AND CONCLUSION**

**5.1 Summary**

A new mobile system for an Airline reservation has been designed and implemented in this project work. The exercise was carried out based on the loopholes existing in the management of air Transport Service. Number of problems encountered in the manual service process thereby creating opportunity for Reliability in the management of customer’s document; Fast discharge of auditing responsibilities; Accuracy of computations and Provision of easy adjustment and update of customer records.

Speech enabled mobile airline reservation system has led to ease of airline ticketing, flight scheduling and also provided a means for customers to access and book flights from their homes. It has also increased the speed with which information about customers are retrieved and handled and flight scheduling is tasked. Owing to the ease and comfort of Airline flight information system, local flights which are not on the system should be encouraged to compensate the system. Secondly, the system should be made affordable so as to encourage consumers and travel agents on patronizing the system.

**5.2 Conclusion**

Nigerian Airline industry is one of the leaders in African airline industry transporting more than 2 million passengers annually. In such competitive industry increasing availability and reaching out to customers is key to succeed.

Due to the increasing number of mobile phones and their pervasiveness nature, a mobile based speech enabled app is proposed. In this project, the requirements of passengers which should be included in the mobile flight reservation have been analyzed. In designing the system, due attention is given to how data has to be communicated between mobile based clients and the airlines existing system; and how the implementation of the project will best fit with the workflow of the current system of the airlines.

The purpose of this work is to build a virile mobile speech enabled application, which a customer could use to interact with the airlines reservation through m-tickets (mobile tickets) booking in the Nigerian Aviation Industry. Based on the input specification, hardware requirement, software requirement; Java on android studio, Microsoft SQL Server 2008 and algorithm design, the application will be very flexible and dynamic in nature. Speech enabled mobile airline reservation system is the best way of airline reservation because of its ease of use and availability at when needed. Its security measures made it to be more suitable than any other way of airline reservation system. With these benefits, the introduction of the Mobile system in Nigerians’ aviation system will boost the economy and aids its process of technological advancement. The intention is to do an extensive evaluation of the application through initial pilot studies with simulated users. The outcome of the evaluation will then be used to adapt the application for commercial use by interested airlines within the Nigeria and beyond.

The design of the system is based on the software as a service approach focusing on exposing a service in the server client environment. As a result we have used services whose concept best fit the proposed solution to develop the project for the requirements identified. Particularly the project is developed using android studio which is easier and advanced in implementing services, in which mobile based clients from different platforms can access Nigerian Airlines flight reservation without concerning about the underlying technologies. Java, accomplishes this by embracing interoperability and loosely coupling that ensures accessibility from different platforms and reusability as well as easy maintenance whenever needed.

After implementing the project, testing have been conducted and the success of the developed solution in accomplishing the requirements have been evaluated. Unit testing was carried out to uncover errors and gaps in accomplishing the required functionalities, and also usability testing to evaluate the project’s success from user’s perspective.

**5.3 Recommendations**

For this research work, it is recommended that a parallel change over should be adopted by the next set of researchers in this area of study, parallel change over means a situation where by the old and the new way of working is implemented. That is, the old (manual) and the newly designed software computerized for transaction of business between the company and her customers are being run. This is to enable a sound understanding of the new system design for the effectiveness of this changeover; there are other tasks that must be accomplished. They include the following:

1. Services are the best way in software development as they eliminate many overheads that hinder flexibility in developing software. As a result large scale service usage should be considered in improving the software. Future work of this project is implementing the system at Nigeria Airlines to enable mobile based users communicate with the services provided using speech enabled mobile app developed.
2. The airline companies implementing this system should send their staff for training so that they acquire more skills and experience in operating the new designed system

**APPENDIX**

**SOURCE CODE**

class MainActivity : AppCompatActivity(){

var user : User? = null

val TAG = "MainActivity"

var navHeader : View? = null

var adapter : MainAdapter? = null

var flight\_list : ArrayList<Flight>? = null

var helper : DbHelper? = null

var lv : ListView? = null

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

setSupportActionBar(toolbar)

toolbar.title = "View Flights"

supportActionBar?.title = "Flight List"

supportActionBar!!.setDisplayHomeAsUpEnabled(true)

toolbar.setNavigationOnClickListener {

onBackPressed()

}

fab.setOnClickListener {

startActivity(Intent(this, AddActivity::class.java))

}

makeInitialization()

user = ModelConverter.GsonToClass<User>(intent.getStringExtra("session"))

Log.e(TAG, user.toString())

makeRoleCheck()

}

private fun makeInitialization() {

// navHeader = nav\_view.getHeaderView(0)

helper = DbHelper(this).open()

lv = findViewById(R.id.flight\_list)

}

override fun onResume() {

super.onResume()

adapterWork()

}

private fun adapterWork() {

flight\_list = ModelConverter.conListFlight(helper!!.getAllFlight())

flight\_list!!.reverse()

adapter = MainAdapter(this,flight\_list!!)

if(flight\_list!!.size > 0){

lv!!.visibility = View.VISIBLE

no\_flight.visibility = View.GONE

lv!!.adapter = adapter

lv!!.setOnItemClickListener { parent, view, position, id ->

val flight : Flight = lv!!.getItemAtPosition(position) as Flight

flight.let{

if(flight.status == "OPEN"){

Log.e(TAG, flight.toString())

startActivity(Intent(this, DetailsActivity::class.java).apply {

putExtra("flight", Gson().toJson(flight))

putExtra("session", Gson().toJson(user))

})

Log.e(TAG, flight.toString())

}else{

Snackbar.make(drawer\_layout, "Flight Is Closed", Snackbar.LENGTH\_SHORT).show()

}

}

}

lv!!.setOnItemLongClickListener { \_, \_, position, \_ ->

val flight : Flight = lv!!.getItemAtPosition(position) as Flight

if(user!!.role == 0){

val mBottomSheet = BottomSheetDialog(this)

val sheetView = layoutInflater.inflate(R.layout.bottom\_menu, null)

mBottomSheet.setContentView(sheetView)

val data = sheetView.findViewById<LinearLayout>(R.id.delete)

data.setOnClickListener({

mBottomSheet.dismiss()

if(helper!!.deleteFlight(flight.id!!)){

flight\_list!!.remove(flight)

Snackbar.make(drawer\_layout,"Flight Record Deleted", Snackbar.LENGTH\_SHORT).show()

}

else Snackbar.make(drawer\_layout,"Unable To Delete Flight. Try Again", Snackbar.LENGTH\_SHORT).show()

adapter!!.notifyDataSetChanged()

})

mBottomSheet.show()

}

return@setOnItemLongClickListener true

}

}else{

lv!!.visibility = View.GONE

no\_flight.visibility = View.VISIBLE

}

}

fun makeRoleCheck() {

if(user?.role == 1){

fab.visibility = View.GONE

}

}

override fun onBackPressed() {

if (drawer\_layout.isDrawerOpen(GravityCompat.START)) {

drawer\_layout.closeDrawer(GravityCompat.START)

} else {

super.onBackPressed()

}

}

override fun onCreateOptionsMenu(menu: Menu): Boolean {

// Inflate the menu; this adds items to the action bar if it is present.

menuInflater.inflate(R.menu.main, menu)

return true

}

override fun onOptionsItemSelected(item: MenuItem): Boolean {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

return when (item.itemId) {

R.id.action\_search -> {

startActivity(Intent(this, SearchActivity::class.java).apply {

putExtra("session", Gson().toJson(user))

})

true

}

R.id.action\_bookings ->{

if(user!!.role == 1){

startActivity(Intent(this, BookingsActivity::class.java).apply {

putExtra("session", Gson().toJson(user))

})

}

true

}

else -> super.onOptionsItemSelected(item)

}

}

class SelectActivity : AppCompatActivity() {

var role : Int? = 0

var pb : ProgressDialog? = null

var TAG = "SelectActivity"

var helper : DbHelper? = null

var user : User? = null

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_select)

helper = DbHelper(this)

helper!!.open()

pb = ProgressDialog(this)

admin\_sel.setOnClickListener {

role = 0

select\_layout.visibility = View.GONE

form\_layout.visibility = View.VISIBLE

}

user\_sel.setOnClickListener {

role = 1

select\_layout.visibility = View.GONE

form\_layout.visibility = View.VISIBLE

}

login\_btn.setOnClickListener {

doLogin()

}

}

private fun doLogin(){

val email = input\_email.text.toString()

val password = input\_password.text.toString()

if(email.isNotEmpty() && password.isNotEmpty()){

pb?.setMessage("Please Wait...$role")

pb?.show()

Thread(Runnable {

Log.e(TAG,"New Tag")

//Thread.sleep(TimeUnit.SECONDS.toMillis(3))

user = ModelConverter.conUser(helper!!.getSpecificUser(email,password))

Log.e(TAG,user.toString())

user.let {

Log.e(TAG, "Not Null")

}

if(user != null || user != User()){

startActivity(Intent(this, MainActivity::class.java).apply {

putExtra("session",Gson().toJson(user))

})

}else{

this.runOnUiThread({

pb!!.dismiss()

input\_email.error = "Invalid Credentials"

input\_password.error = "Invalid Credential"

})

}

}).start()

}else{

pb!!.dismiss()

if(email.isEmpty()) input\_email.error = "Cannot be empty"

if(password.isEmpty()) input\_password.error = "Cannot be empty"

}

}

override fun onBackPressed() {

if(form\_layout.visibility == View.VISIBLE){

select\_layout.visibility = View.VISIBLE

form\_layout.visibility = View.GONE

}else super.onBackPressed()

}

class ExecuteLogin : AsyncTask<String, Void, String>() {

override fun doInBackground(vararg params: String?): String {

return "true"

}

}

}