12/3/2018

# 

CSIS 4175 – Mobile App Development II

Instructor: Ivan Wong

Final Report

“A Matter Hotels” App

Alina Pak – 300277280

Anusha Das – 300283182

João Vitor Wilke Silva - 300278748

Volodymyr Pavlovskyi– 300283178

Contents

[Disclaimer 2](#_Toc531555293)

[Description 2](#_Toc531555294)

[Functionalities 2](#_Toc531555295)

[Features 4](#_Toc531555296)

[Additional Challenges 4](#_Toc531555297)

[System Architecture 5](#_Toc531555298)

[Incomplete Functions 5](#_Toc531555299)

[Installation Guide 6](#_Toc531555300)

[Access to admin profile within the app 6](#_Toc531555301)

[Access to firebase console 6](#_Toc531555302)

[User Manual 6](#_Toc531555303)

[Changelog Highlights 13](#_Toc531555304)

[Project Log 13](#_Toc531555305)

[Week #8 – Nov 3 – Nov 9 13](#_Toc531555306)

[Week #9 – Nov 10 – Nov 16 13](#_Toc531555307)

[Week #10 – Nov 17 – Nov 23 14](#_Toc531555308)

[Week #11 – Nov 24 - 30 14](#_Toc531555309)

[Week #12 – Dec 1st – Dec 3rd 14](#_Toc531555310)

[Issues (after Interim Report) 15](#_Toc531555311)

[References 16](#_Toc531555312)

[Appendix A – Individual Reports 17](#_Toc531555313)

[Alina Pak 17](#_Toc531555314)

[Anusha Das 18](#_Toc531555315)

[João Vitor Wilke Silva 19](#_Toc531555316)

[Volodymyr Pavlovskyi 20](#_Toc531555317)

[Appendix B – List of figures 22](#_Toc531555318)

# Disclaimer

A Matter Hotel mobile app is a project specifically designed for CSIS 4175-070 course at Douglas College for Fall 2018 semester and is not intended for sales of any sort. The material borrowed from the Internet including images, icons, descriptive text are not owned by any of the group members and are used only for educational purposes. Any code not written by authors of the application was taken from online open sources and all the links are put as references at the end of this document.

The idea, design and implementation, however, belong to the group and together are considered the property of the “A Matter App’s” developers.

# Description

Our app is designed for customers who would like to book a room in the “A Matter” hotel, in Vancouver. It allows our users to browse and book rooms and activities throughout the city, as well as to order food and drinks from our breakfast, lunch and dinner menus. It also enables users to review and rate our services and their stay at our hotel and generate an up-to-date invoice so that they can keep track of their purchases while staying with us. Furthermore, it has administrative features, which allows the hotel managers to view, modify and delete users, products, activities and bookings, so that they can keep track of the hotel’s activities and make any corrections to the information held by the hotel without much effort.

# Functionalities

* Sign Up/ Log In

Implemented with fragments. The sign-up fragment allows new users to register in the app by providing personal information such as email, password, first name, last name, and region. For already existing users, the log in fragment provides inputs for their credentials, which are their email address and password to access our app’s main features. Both use Firebase Authentication, Animation and Custom View, as well as different kinds of validation for email and password. For usability there are toast messages displays hints and any issues occurred during Sign Up/ Log In process.

* Room Booking

This feature allows users to book rooms. On the “rooms” activity, there are 3 threads which are used to create an image-changing animation every 2 seconds, as well as descriptions to every type of rooms available (Comfort, Kings, and Family Suites), which are placed in an “Expanded View”. Also, there are user inputs like a Date Picker for the date, “EditTexts” for the length of the booking (in days) and the number of guests which will be staying in the hotel. For each type of room, there is a corresponding “Book” button which triggers a Dynamic fragment on the top of the screen created to show the Booking Confirmation. Booking details are placed in Firebase under the “bookings”, which is structured based on the current user. There is also validation for the aforementioned inputs.

RULE: A user can reserve multiple rooms, but only one at a time.

* Activities Reservation

Almost the same idea as Room Booking page but separated into two different activities (“Activities” and “Booking”). For each activity, there are pictures relating to its content which constantly change through threads, as described in the “Room Booking”. There are also a description and “Book” button. Once the user clicks the button, he/she is redirected to the Booking Page, where inputs for Quantity of people and Date Picker are placed. After the booking is made, the user is taken back to the “Activities” activity and a confirmation fragment is shown on the top of the screen. Booking details are placed in Firebase under the “activities” object, which is structured based on the current user. Also, validation for the input date is present.

RULE: A user can book multiple activities, but only one at a time.

* Room Service ordering

Developed in a similar way as the “Activities” activity, but for ordering food services. Our menu is divided in the following categories: Breakfast, Lunch and Dinner, and Dessert. When a user selects one of our categories, he/she is redirected to the “room service booking” activity, where a dynamic table is implemented to enable our users to order products from that category easily. The content of this table is generated “on the fly”, and it changes according to which category the user has chosen on the previous activity. This activity also stores data about each food item ordered to Firebase under the “service” object, which is structured based on the current user.

RULE: A user can order multiple meals from one menu (e.g. Breakfast) simultaneously. If the user makes more than one order of the same meal at different times, the Invoice will show each record. This is done to differentiate separate orders.

* Rank and Review

The “Rank and Review” activity allows users to write a review about our hotel services. This activity contains an “EditText” to get the content of the user’s review, as well as a rating bar and an animation based on a “CANVAS” object implemented as an “emoji”. The emoji’s “mood” changes from sad to happy as the rate increases. The review, along with the value of the rating bar are stored in Firebase under the “review” object, , which is structured based on the current user. After the data is stored, a confirmation “Toast” is displayed.

RULE: A user can submit only one review. The database will save the most recent one, overwriting previous entries. This way we will make sure the review is fair from every user.

* Invoice

The Invoice Page displays all the user’s purchases and bookings, along with prices for each item and an updated total. This was Implemented as a “ListView “with an “Adapter” which displays the date, service(name for what was purchased or booked), and the corresponding price of each transaction made by the current user. The data used to display this table is read from the “bookings”, “activities” and “orders” objects of our Firebase database based on the current user. The data is handled and placed on the appropriate fields. All information is properly formatted to fit the “ListView”. All information is up-to-date thanks to database event listeners.

* Admin Page

The Admin Page was designed to allow the hotel manager(s) to review the hotel’s records of customers, bookings, products and activities, as well as sales (by viewing which activities have been booked or which items have been sold). It also enables the manager(s) to make changes to user’s information if needed. This activity also provides tools for updating a user’s first name, last name, and the region. Additionally, it enables the manager(s) to delete users, along with his/her entire history with the hotel. Since the methods for reading and manipulating data are placed inside database event listeners, the data displayed changes automatically with no need to execute queries again.

# Features

The following features of the “A Matter Hotel App” are based on topics covered in CSIS 4175-070 class. Their implementation heavily relied on in-class lab exercises and techniques taught in “Advanced Android Development” course of Google Developers Training (which can be found at <https://developers.google.com/training/courses/android-advanced>).

* Animation

Sliding and shaking animation were implemented in the sign-up and log in to enhance a more enjoyable and user-friendly experience. Also, animations with “Custom Views” made from scratch were incorporated in “Rank and Review” activity to improve visuals.

* Custom TextView (for passwords)

Implemented in the “Log In” fragment, to enable users to show their password so that they can make sure they typed it in correctly. It responds to on Action Down and Action Up events.

* Fragments

In our app, fragments are implemented in two ways: The “Sign Up” and “Log In” ones are designed as fragments that are used as full-working activities (they handle actual code and interaction with other pages). There are also fragments on the home screen of the app, which are used as a dynamic portion of that activity.

* Threads

Implemented to enhance the app’s visuals by providing an image-changing animation in the “Room”, “Activities” and “Room Service” activities.

* Notifications

Implemented in the “Activities” activity to notify of the successful booking and provide basic information about that booking. This was Implemented based on database listeners.

## Additional Challenges

These extra features were implemented to correspond with the application logic and are considered challenges of the project because they were either not covered in class or require a more advanced knowledge of course-related concepts.

* Restriction of orientation change

In order to prevent the app from crashing due to orientation change conflicts with threads and preventing the “reset” of these threads, the orientation was blocked for all activities throughout the app. This was achieved by firstly setting the orientation to “portrait”, which is the desired orientation for our app, and secondly, but setting the orientation to “locked”, thus preventing it to change.

* ExpandView

ExpandView allows to show and hide “big” amounts of text by clicking on the related elements on the page. For this project we choose pictures of up-facing and down-facing arrows to do that. When a user clicks on the “down” arrow, the text becomes expanded and the “down” arrow becomes an “up" arrow. By clicking the “up” arrow, most of that text becomes hidden and the “up” arrow becomes a “down” arrow. Widely used across the app for all descriptions which contain a fair amount of text.

* Dynamic Fragment

Dynamic fragments are used throughout the app for displaying booking confirmation messages to the user. When a booking is made, or an order is completed, and if it is successfully stored into our database, a dynamic fragment appears on the top of the screen to confirm and inform the user that his/her “transaction” has been completed successfully. The fragment itself is a screen-wide red rectangle on the top of the page height enough to fit confirmation text. After 2 seconds the fragment closes itself automatically. This was widely used across the app to inform the user about all successful bookings and/or orders.

* Dynamic Table

A dynamic table is used in the “Room Service Booking” which displays all the items of related to the chosen menu(Breakfast, Lunch & Dinner and Dessert). The main feature and challenge were

In ensuring the correct options were displayed according to the chosen menu, as well as ensuring that the controllers within that table were functional and independent. This was successfully implemented and allows the user to choose desired amount of each food item by clicking “+” and “-” buttons to the corresponding food item. The data from this table is stored in our database once the customer confirms his/her order.

* Rating Bar

This was implemented in the “Rank and Review” activity. The main feature and challenge were connecting the Rating Bar to the CANVAS emoji object, which makes the emoji’s expression to change as the user increases or decreases his/her review’s rating. The emoji’s expression changes from sad to happy as the rating changes. This was achieved by redrawing the emoji’s face every time the rating bar’s score changes.

* ListView with adapter

Developed to display a user’s invoice which gets the current user’s transaction history from the database. All of this data is structured and added to a List object, that was added to a List of the type Invoice Handler (a class that provides methods to populate the layout for Invoice using the Adapter class).

## System Architecture

The core of the system architecture in the “A Matter Hotel app” is Firebase. Platform usage includes:

* Firebase authentication
* CRUD operations on a real time database(Firebase):
  + Storing data properly into the database from the app
  + Reading data from the database from the app
  + Updating “users” objects in the database from the app
  + Deleting objects from the database through the app

## Incomplete Functions

Our team was able to implement the functions we had originally planned, however, we wished we had been able to have implemented more thorough constraints and/or validation on some parts of our app. For example, we originally wanted the app not to allow multiple bookings from one account during a period that already has a booking. However, since we found Firebase to be a bit “tricky”, we decided that for now, we would only ensure that the app was fully operational, and we might work further in this app in the future as to improve it further. Also, there was an idea of implementing service which would generate and send an e-mail to User`s address when any related data changes. The e-mail would be sent from a.matterhotel@gmail.com account and should contain all bookings, reservations, and orders made by the user. After closer acquaintance with Firebase, we decided to get rid of it because couldn`t make it work properly (was significantly slowing down the app).

# Installation Guide

**DISCLAIMER:**  For unknown reasons, emulators run on Douglas College’s computers occasionally “die” when running our app. We have debugged this app thoroughly and have not been able to identify the underlying reason for this. We have also debugged our app in 4 different computers and 2 cellphones and this error has never occurred on any of these devices. We ask that when you test our app, do so in a computer other than the ones at Douglas College.

Since the current version of the app is a “beta version” there is no certainty that it will run on every Android device. Therefore, to ensure the best experience with “A Matter” consider the following:

* The app was tested on Nexus 5x emulator, Samsung Galaxy Note 8 and Samsung Galaxy S7 phones.
* For a faster performance on the emulator a user must close the app completely after Signing Out if he wants to Log In again in the same session.

## Access to admin profile within the app

In order to see the Admin Page, the following profile should be used:

Email: [a.matterhotel@gmail.com](mailto:a.matterhotel@gmail.com)

Password: android1234

## Access to firebase console

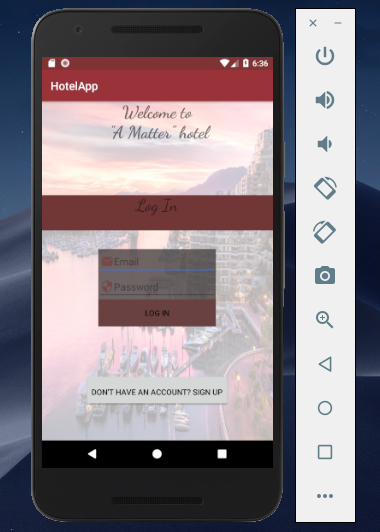
Email: [a.matterhotel@gmail.com](mailto:a.matterhotel@gmail.com)

Password: android4175070

# User Manual

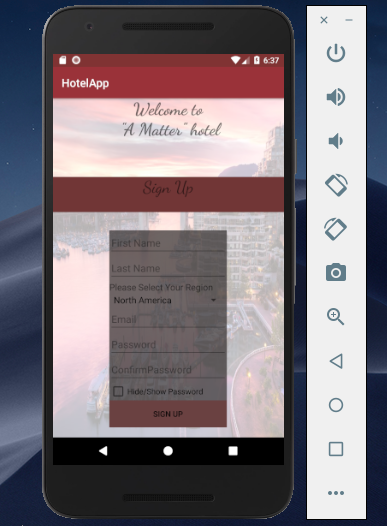
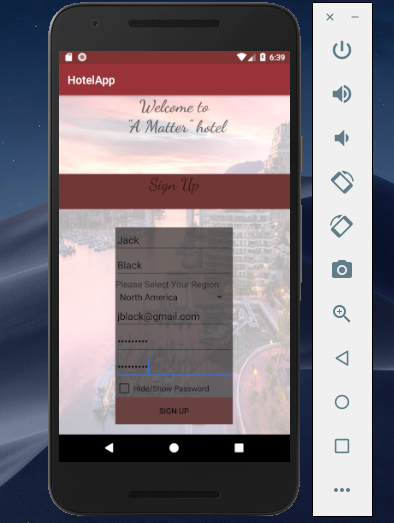
The following section provides a more detailed user manual comparing to the video in the presentation (can be found at the last slide of the A Matter.pptx)

**Step 1**: The first screen that shows up when you run this app is the login screen, as shown below:



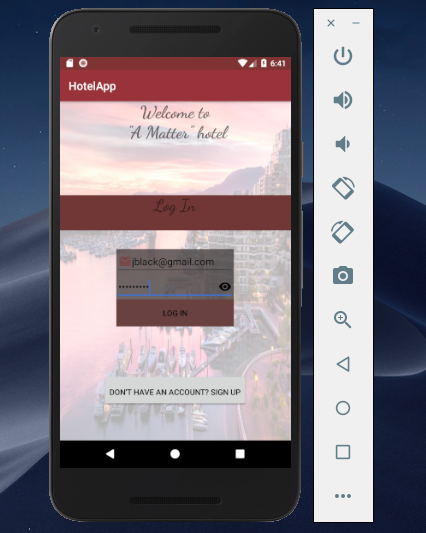
For first-time users, a new profile must be created. To do so, the button placed below the login panel, labelled “Don’t have an account? Sign Up” must be clicked. This takes the user to the sign-up fragment.

**Step 2**: The signup screen asks the user to enter their first and last names, an email id, choose the region they belong to, and create a password. This password couple with the email will be used by the user to log in. The sign-up screen looks like this:

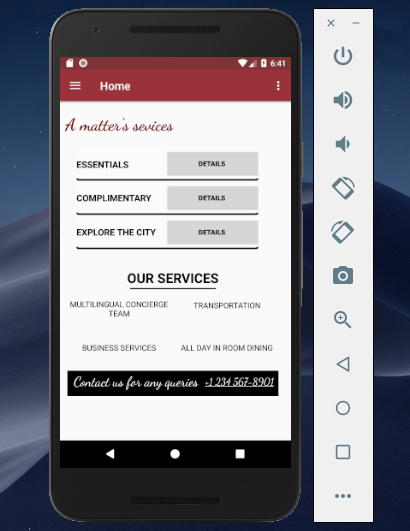
 

After filling out the required fields, and you press the sign up button.

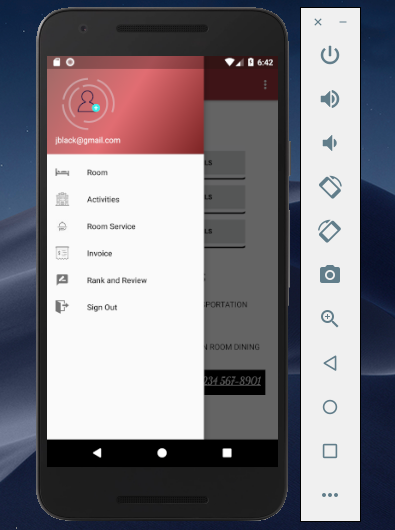
**Step 3**: This takes you back to the login screen where you enter the email and password you used while signing up. Once you’ve entered these correctly, you press the log in button. If you weren’t a first-time user, you could simply enter your details here and click this button.



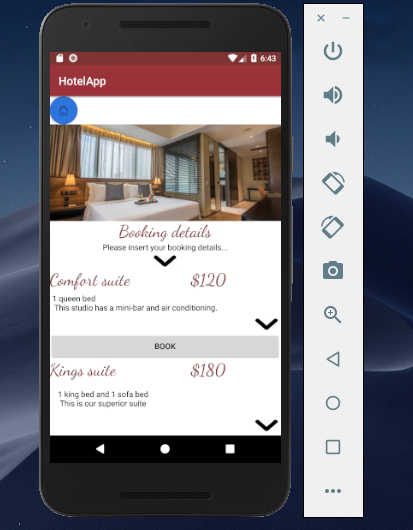
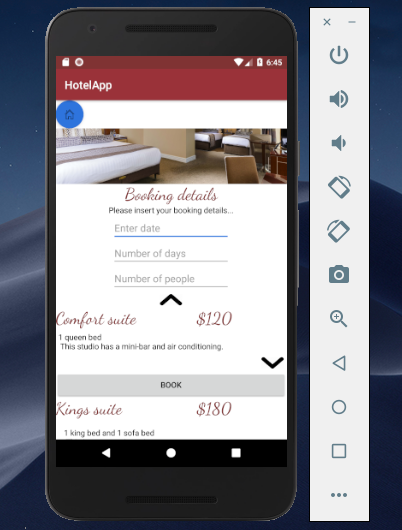
**Step 4**: This takes you to our home page. It gives you a detailed description of the services provided by A Matter Hotel. This includes essentials, complimentary services, and services provided by us to explore Vancouver. Clicking on ‘details’ gives you a description of each service.



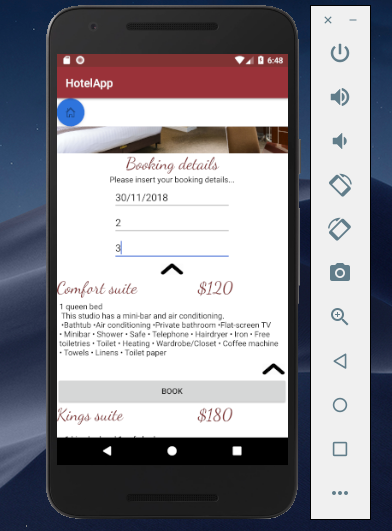
**Step 5**: To explore the various features of the app you press on the navigation bar located on the upper left corner of the screen. It lists out the following features (Room, Activities, Room Service, Invoice, Rank and Review, Logout) :



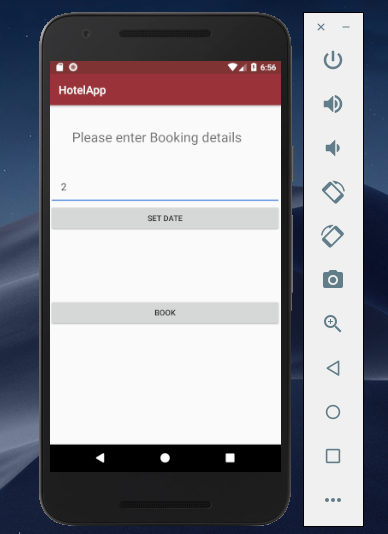
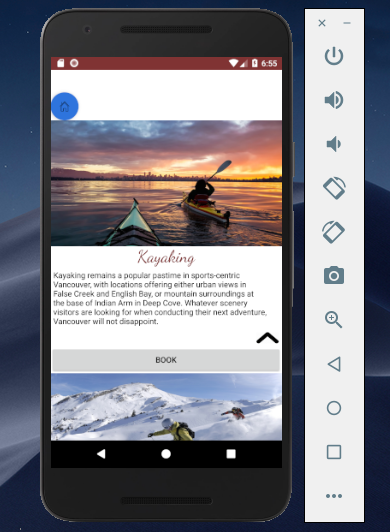
**Step 6**: The first item on the Navigation Bar menu is Room. It allows the user to look through the type of rooms available at A Matter Hotel, and book whichever room they want after entering the booking details: date, no of persons, no of days.

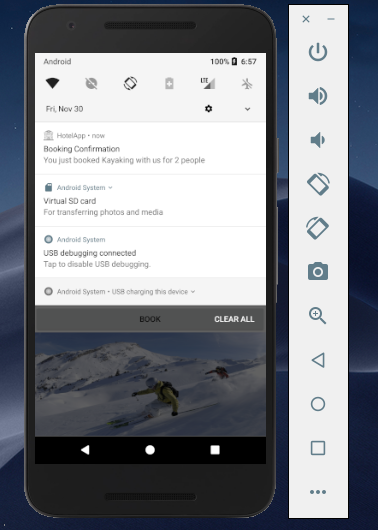
**Step 7**: The date field is filled out using a calendar, and the other two fields take integer inputs.



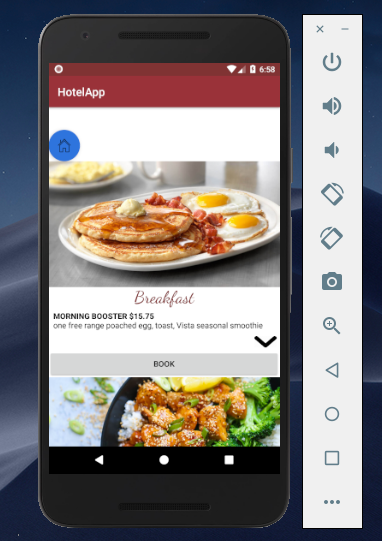
**Step 8**: The second item on the menu is Activities. This lists out the various activities a user can book with A matter hotel. Once the user has chosen an activity they want to book, they press the book button. This takes them to a new screen where they are required to enter details like: Number of people they’re booking for and the date on which they’d like to do the activity.



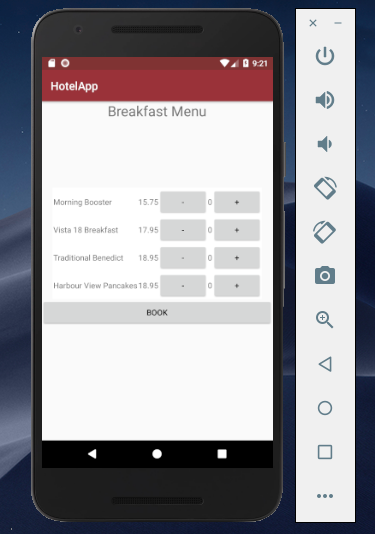
On finally pressing the book button, a confirmation appears at the top on the screen (red background).A notification is also sent containing the booking details(activity name and qty of people)

 (in the image, notification is at the top of the drop-down list)

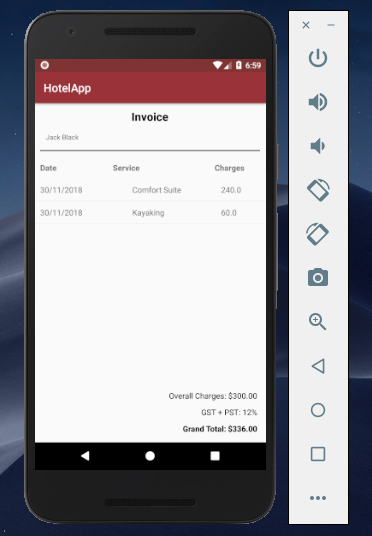
**Step 9**: The final thing a user can book is room service. This includes Breakfast, Lunch and Dinner, and Desserts. The screen looks like so:



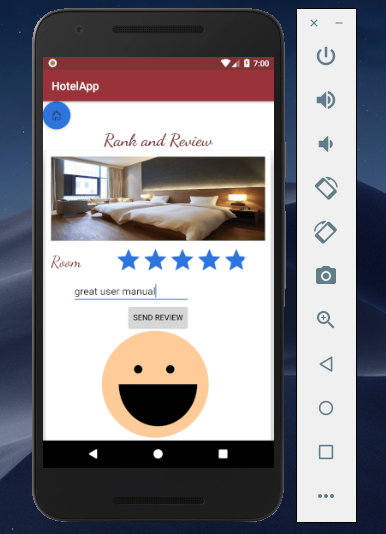
Clicking on book takes the user to a list of items we have for that meal. You have the option to enter the qty you want for each dish. The table looks like so:



**Step 10**: The Invoice option on the menu gives the user a list of their purchases. It lists the date of purchase/activity, the name of the activity and the exact price as shown on that activity’s page. It totals the prices and adds 12% taxes to give the final price. The app does not have an option for payment.



**Step 11**: The user can leave a rank and review too. Pressing this option on the menu takes you to this screen:



**Step 12**: Finally you press the sign out button, which takes you back to the log in page. Or, you could go back to make more reservations.

# Changelog Highlights

There’s been a change in a structure of the app since the Interim Report. Some of the features that were not initially considered were added to make the app more complex. Moreover, one of the functionalities of “A matter” that was put in a Proposal but removed from Interim Report was revived.

* Added Custom Text View feature
* Implemented a List View with an Adapter for the dynamic Invoice
* Made use of Canvas object
* Revived “Rank and Review” page

# Project Log

## Week #8 – Nov 3 – Nov 9

Alina implemented the firebase authentication and writing a “user” object in the real time database. She also added the “Sign Out” button. (4 hours)

## Week #9 – Nov 10 – Nov 16

João implemented the writing to the database using a 2-fragment version for “Log In” and “Sign Up” of the project (Alina’s version contained 3 fragments). He also created the “Custom View” for passwords and developed the code to store data into the database on the “Rooms” activity. (5 hours)

Alina fixed a bug on the “Custom View” developed by João and further developed on the database-related codes in the “Rooms” activity. (2 hours)

Anusha designed the invoice. (1 hour)

Volodymyr changed appearance and some input elements of Admin page in order to display information from the database properly. (2 hours)

## Week #10 – Nov 17 – Nov 23

Alina made the presentation draft with user manual video. (3 hours)

João developed the code for storing data from our “Bookings” and “Room Service Booking” into our database. He also looked over Alina’s draft of the presentation and provided some suggestions to improve content. ( 4 hours)

Anusha read data from DB to be used by Notifications, and for Invoice (4 hours)

Volodymyr partly used improved version of Anusha`s way to read the data from the database. Also implemented user search with all related information, user`s personal information update, and user`s delete. Made possible to review all data in the database sorted by categories. (6 hours)

## Week #11 – Nov 24 - 30

Anusha fixed the price of Room Service being written to DB and finished working of Invoice. (1 hour)

Alina fixed date fields in the database and formatted the prices in the Invoice. Completed the presentation, implemented the “Rank and Review” page with writing a “room\_review” object in the db and Custom View with animation and CANVAS object for smile. (6 hours)

Volodymyr made notifications being sent every time booking information in the database changes. (30 min)

João wrote his individual report and made a few changes to the presentation slides. (2 hours)

## Week #12 – Dec 1st – Dec 3rd

Alina wrote individual report and completed her part in final report writing. (4 hours)

Anusha wrote her individual report and added her parts to the final report. (2 hours)

Volodymyr wrote his individual report and added his parts to the final report. (4 hours)

João proofread the final report, correcting and/or modifying as it was required. (3 hours)

# Issues (after Interim Report)

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue #** | **Description** | **Fixed?** | **How to Fix / Possible Cause** |
| 1 | Could not connect the app to Firebase authentication | Yes | The problem was in the version of Android Studio as it tried to connect to firebase-auth-15.0.0, but firebase required version 11.6.0. Solved by using an older version of Android Studio. |
| 2 | Could not connect the app to Firebase Real-Time database | Yes | New version of Android Studio included dependencies for 16.0.1:15.0.0 version. Needed version 11.6.0. Solved by using an older version of Android Studio. |
| 3 | Authentication implementation did not work inside the fragment | Yes | Solved by referencing the fragmnet by getContext() method instead of SignUpFragment.this() |
| 4 | “date” field in the “orders” object was not stored correctly. Also, all date fields in db were getting selected month – 1 (e.g. October instead of November) | Yes | Solved by converting date data type to a string with proper formatting |
| 5 | Prices in the Invoice were not correctly formatted | Yes | Solved by using getCurrency() method on overall and total price. The output of surcharges formatted by converting double to BigDecimal and specifying number of decimal places |
| 6 | Even after successfully writing to “booking” object, the app showed toast error message | Yes | Solved by using try … catch() instead of if()…else |
| 7 | The table in Room Service Booking page was cropped and not fully displayed. | Yes | Solved by changing layout width and gravity properties |
| 8 | Couldn`t read the information which stored under object which stored under another object | Yes | Solved by getting and parsing several snapshots within the foreach loops placed one inside of another |
| 9 | Couldn`t search all user`s related data in database | Yes | Solved by storing unique key of user`s object in separate variable and search for every record which got the key in header or as a property |
| 10 | Displayed data in Admin Page wasn`t updating automatically | Yes | Solved by adding database event listeners across the page |
| 11 | Database getGoogleApiForMethod() returned Gms and didn`t allow to do anything related to database | Yes | Solved by downgrading Firebase 11.6.0 to 11.4.2 version |

# References

<https://www.fourseasons.com/beverlywilshire/services-and-amenities/>. Hotel in Beverly Hills, 2018

<https://www.grecotel.com/luxury-lifestyle/hotel-activities.html>. Grecotel Hotels & Resorts, 2018

<https://www.rosewoodhotels.com/en/hotel-georgia-vancouver/interests/adventure-activities>. Rosewood Hotels and Resorts. *Adventure Activities,* 2018

[*https://www.chateauvictoria.com/dining-/room-service/breakfast/*](https://www.chateauvictoria.com/dining-/room-service/breakfast/)*.* Chateau Victoria Hotel and Suites. Room Service, 2014

<https://www.geeksforgeeks.org/check-email-address-valid-not-java/>. GeeksforGeeks. *Check if email address valid or not in Java.*

[https://stackoverflow.com/questions/45686276/how-to-use-firebase-email-login-with-a-fragment#](https://stackoverflow.com/questions/45686276/how-to-use-firebase-email-login-with-a-fragment). Stack Overflow. *How to use Firebase Email login with a fragment?,* 2017

<https://developer.android.com/guide/topics/ui/notifiers/notifications> Notifications Overview

<https://stackoverflow.com/questions/40000899/how-to-link-between-authenticated-users-and-database-in-firebase>. *How to link between Authenticated users and Database in Firebase?,* 2016

<https://www.androidhive.info/2016/10/android-working-with-firebase-realtime-database/> Android working with Firebase, 2017

<https://androidexample365.com/a-custom-android-rating-view-with-interactive-smiles/>. YuganshT79. *A custom Android Rating view with Interactive Smiles,* 2018

<http://www.androhub.com/login-signup-and-forgot-password-screen-design-android/>. Dr Droid. *Login, SignUp and Forgot Password Screen Design Android,* 2015

<https://stackoverflow.com/questions/2808535/round-a-double-to-2-decimal-places>. Stack Overflow. *Round a double to 2 decimal places [duplicate],* 2011

<https://www.youtube.com/watch?v=sKFLI5FOOHs&list=PLl-K7zZEsYLlP-k-RKFa7RyNPa9_wCH2s&index=4> YouTube. *Common SQL Queries converted for the Firebase Database - The Firebase Database For SQL Developers #4, 2016*

<https://stackoverflow.com/questions/47726233/w-bichannelgoogleapi-firebaseauth-getgoogleapiformethod-returned-gms> Stack Overflow. *W/BiChannelGoogleApi: [FirebaseAuth: ] getGoogleApiForMethod() returned Gms, 2017*

<https://stackoverflow.com/questions/38232140/how-to-get-the-key-from-the-value-in-firebase> Stack Overflow. *How to get the key from the value in firebase, 2016*

<https://stackoverflow.com/questions/38583278/firebasecloudmessaging-firebaseinstanceid-background-sync-failed-service-not> Stack Overflow. FirebaseCloudMessaging : FirebaseInstanceId background sync failed - SERVICE\_NOT\_AVAILABLE, 2016

<https://github.com/firebase/quickstart-android/blob/564601cb1da6435e92e53ff281f883d89d815e8e/database/app/src/main/java/com/google/firebase/quickstart/database/java/PostDetailActivity.java#L85-L107> GitHub. *Sample of Firebase CRUD operations, 2018*

# Appendix A – Individual Reports

## Alina Pak

**Implementations:**

Sign Up/ Log In

Was in charge of Sign Up and Log In Pages which allow users to create and access their accounts.

Features:

* Fragments - both Log In and Sign Up pages are fragments residing on MainActivity
* Animation.
  + Sliding - to slide the pages from Sign Up to Log In and vice versa.
  + Shaking – related to user input validation. The whole user input fields’ form (LinearLayout) is shaking when a user press LogIn/SignUp buttons without entering any data or when authentication fails. Only a certain field is shaking (e.g. email, password…) when the input is incorrect.
* User input validation.
  + All fields are required
  + Requires valid email addresses (checked for correct pattern using regular expression)
  + Password should contain at least eight (8) characters
  + Password and Confirm Password fields should match with each other
* Custom TextView.
  + Comes with Hide/Show Password feature
  + In Log In fragment the icon to show password appears when a user starts typing the password. If he will press and hold the icon, the user will make the password visible. If he will un-press the icon – the password will become hidden again. (It was **Joao’s** idea to implement it)
* Firebase
  + User Authentication.
  + Creating User object in the Realtime database. To implement this feature, created another fragment (Set Up the Profile) which was only to write to the database the current user (the one already authenticated). That means that user authentication was separate from database writing and these features were implemented on separate pages. It worked fine, but **Joao** proposed to combine both, so in our final version we used just one Sign Up fragment for firebase auth and db.

Room Page and Room Service Page

Used **Waldi’s** template for ExpandView and put the content (text and images) for Room, Activities and Room Service Pages.

Features:

* Multi-Threading. Reused **Joao’s** idea to implement AsyncTasks. Since AsyncTasks were slowing the app, changed it to multi-threading technique where three changing Images are three different threads.
* Home Button. Implemented the Home Button, but when we started to display current user’s email address on top of the NavBar, it was crashing because it was not sending the string. **Waldi’s** fix.

Rank and Review

Designed Rank and Review Page to allow users to give hotel managers some feedback on room and the hotel in general.

Features:

* Writing to the db. Created a “room\_review” object in the real time db. Rule: object can hold multiple reviews, but only one review per user. This is done to prevent users from giving us many reviews with totally different marks.
* RatingBar
* CardView
* CustomView from scratch (Smile)
* Canvas object with animation (Smile)

**Additions:**

* Was in charge of the app’s design (colors, icons, NavBar)
* Put separate activities of each group member into one project - final draft
* Implemented Sign Out firebase’s feature
* Formatted the layout of Room Service Page: adjusted the width property of the table and added a “Book” button
* Fixed the writing of “bookings” object so the program would not crash when a user will press on the second or third book button.
* Formatted the prices in the Invoice
* Fixed date fields in the database

**Final Documentation:**

* Presentation with user manual video. Used PowerPoint, DU Recorder, powtoon.com
* The draft of FinalReport with complete structure plus the disclaimer, features, system architecture, installation guide, changelog highlights parts

## Anusha Das

Home Page

Used a template from Hotel at Beverly Hills’ website. Made fragments for each service provided. Imitated the design of this website to decorate the text on the fragment. Used buttons to navigate between fragments. After researching various ways of working with fragments, chose to go with Dynamic Fragments. Used the replace function to do so. Also included an option to allow users to directly call for help from the fragment by linking a phone number. These fragments were intended for room service page, but ultimately we decided to use it on the home page.

Notifications

Included notifications that were triggered whenever a button for booking an activity was pressed. This notification initially just showed up with a default text whenever the button was pressed, but after including the database, it retrieved the activity’s name and the number of people booking that activity and displayed it. Searched for a logo to accompany this notification. The first selection did not show up but found a transparent one that worked.

Invoice

Created a layout for Invoice using ListView. Used an Invoice Adapter and Handler to help out with populating the Invoice. Once the layout and the basic calculations for Invoice were in place, started to do research for reading data from Firebase.

First, tried to include the current user’s full name at the top of the invoice. Tried several times but to no avail. Finally found the code required to do this,

After figuring out the basics of reading data from Firebase, linked it to Notifications.

After a few additions to the code used for the above, was able to retrieve activities’ information, room booking information and room service orders.

Added all this information to a List object, that was added to a List of the type Invoice Handler (a class that provides methods to populate the layout for Invoice using the Adapter class)

Retrieved charges from Invoice List and calculated overall charges. Added taxes, a total of 12%, and calculated final charges.

Miscellaneous

Fixed the room service charges being written to DB

Fixed date that was shown as one month lesser than what was selected on the calendar

Tried doing the Custom View from scratch for Rank and Review but couldn’t since I missed that lecture.

Removed settings option from the menu, since it had no function.

## João Vitor Wilke Silva

* João developed the basic design/layout for the original “hotels” activity and the *Asynchronous Tasks* for dynamically changing the pictures of the hotels on the first version of our app.
* When the group decided to test *Threads* as an alternative to *Asynchronous Tasks* in an attempt to improve performance, he helped in the structuring and debugging of these threads, along with Alina.
* He was also in charge of blocking the orientation change in our app, as to prevent issues with our *Asynchronous Tasks* resetting, along with other design disruptions it would cause by using the following codes:

// Blocks orientation change  
setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_PORTRAIT*);  
setRequestedOrientation(ActivityInfo.*SCREEN\_ORIENTATION\_LOCKED*);  
setContentView(R.layout.*activity\_activities*);

* He came up with the design and code for the dynamic table in the “roomservice\_booking” activity, which he later modified and adapted as per the group’s needs. **Important remark: he asked for Stephen Chiong’s guidance when coding the event handlers for each button of each row in the aforementioned dynamic table.**
* He modified the design and code for our “SignUp” activity, as originally, our project had 2 signup fragments(one for email and password, and another for the user’s personal information, such as name) which were already working with our *Firebase* database, but we decided that having 2 fragments was unnecessary, so he combined both into one fragment and developed the code for adding users to our *Firebase* database.
* He developed a *Custom Text View* for password text fields which allowed users to see they password by clicking on a small icon on that field. At this point, this feature was working but it was a bit “bugged” as sometimes it would not run properly. It was later fixed by Alina.
* He developed the code to allow the “Room” activity to store data in our *Firebase* database. This was later worked on by Alina.
* He developed the code to allow the “Room Service” and “Activities” activities to store data into our *Firebase* database.
* He modified the group’s presentation to better organize its content, as well as helped to organize the group’s presentation.
* He was responsible for formatting, structuring and writing the final content for the Interim Report.
* He was responsible for proof-reading for the final report.

## Volodymyr Pavlovskyi

Admin Page

Was created for managing and deleting user`s information and reviewing related bookings as well as all content of database for hotel manager sorted by categories.

Features:

* Implementation of reading, update and delete features of Firebase. The page gives the ability to review all data which is stored in the database sorted by categories. Also, there is an ability to update users’ personal information such as first name, last name, and region. Admin can review and delete all data related to a particular user.
* Dynamic page content design. The content of the page (such as user inputs) is changing depending on the option chosen. For every user input, the page content will adapt to fit the query results from the database. Also, there are different kinds of validation on user inputs. For example, if the manager will try to search a user which doesn`t exist in the database, the page will pop-up the appropriate message. All queries are placed on separate event listeners, so shown information updates in real time.
* Visibility only for Admin. The page itself is visible on the Navigation Bar only for Admin or Manager. In order to see it, the proper credentials should be inputted during the login process ([a.matterhotel@gmail.com](mailto:a.matterhotel@gmail.com) for email and android1234 for password). The Admin page won`t be shown for any other users.
* Error handling. All interactions with the database are placed in proper try catch statements to prevent page crashing due to internet problems or any other issues. In case something goes wrong, the appropriate message will be shown.

Activities Page

Activities page gives the ability to book different activities for a user. The user can book activities for himself or a group of people for any date after current date.

Features:

* ExpandView. The expanded view allows a user to see and hide full description of each activity by clicking hide and expand symbols which are shown as up and down arrows. This was made to keep page design consistent.
* For every activity, there is corresponding “Book” button which sends the user to the next booking activity where all activity details should be entered. Moreover, when the user clicks on the “Book” button the name value of the selected activity sends to the next page to store it in the database.
* Fragment. There is also a fragment which appears on the Activities Page after booking was made. The fragment itself appears as a screen wide red rectangle on the top of the page height enough to fit confirmation text. After 2 seconds the fragment closes itself automatically. Triggered by clicking the “Book” button on the Booking page.

Booking Page

Designed to input details of chosen activity booking and write entered data to Firebase. Works as an extension of Activity page to make activity booking easier for the user and to prevent the page being too complicated.

Features:

* Date picker. Was made to make date input user-friendly and intuitive. There is a restriction set which validates user input by making impossible to book activity for past time. The active dates which might be chosen are the only current date and beyond.

**Additions:**

- Made user`s email to be shown on top of navigation bar by sending the email value from the login page to the Main page.

- Fixed app crashing after home button being pressed by storing the email value on separate variable and passing to every page chosen every time user enters other app pages.

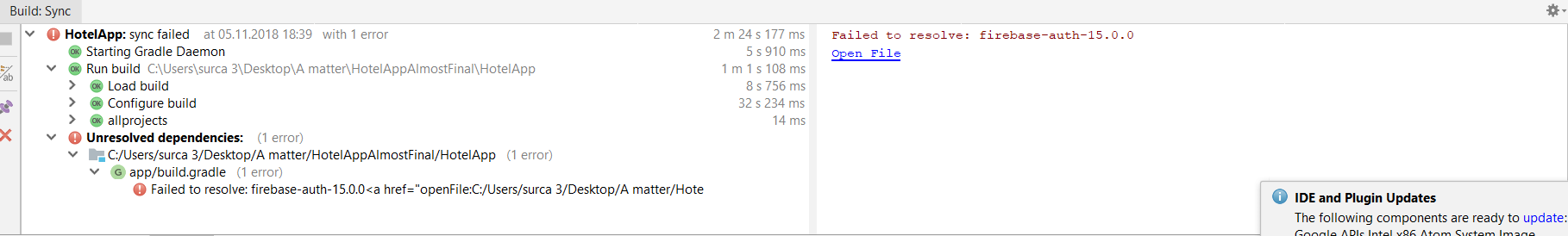
- Made notification being sent every time booking information was changed in the database by connecting notification sending method to the database event listener.

**Final Documentation:**

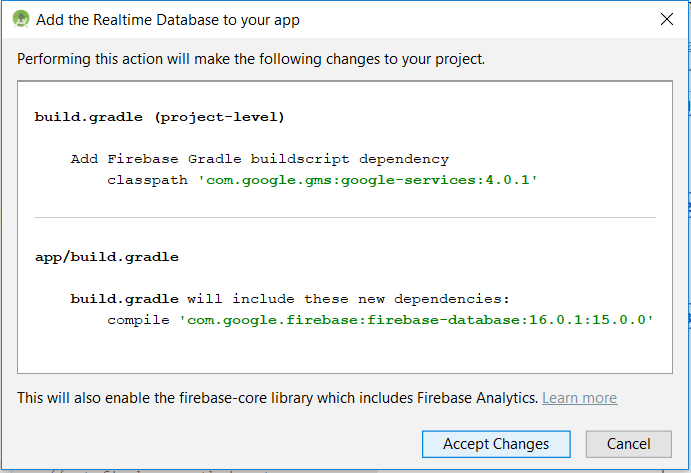
* Additional challenges.
* Functionalities.
* Incomplete functions.
* Individual parts of the project.

# Appendix B – List of figures

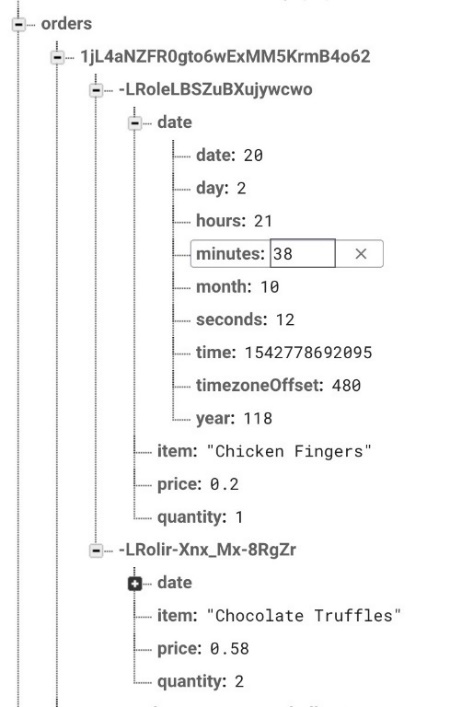
## Figure #1 – 1st Issue (Could not connect the app to Firebase authentication)



## Figure #2 – 2nd Issue (Could not connect the app to Firebase Real-Time database)

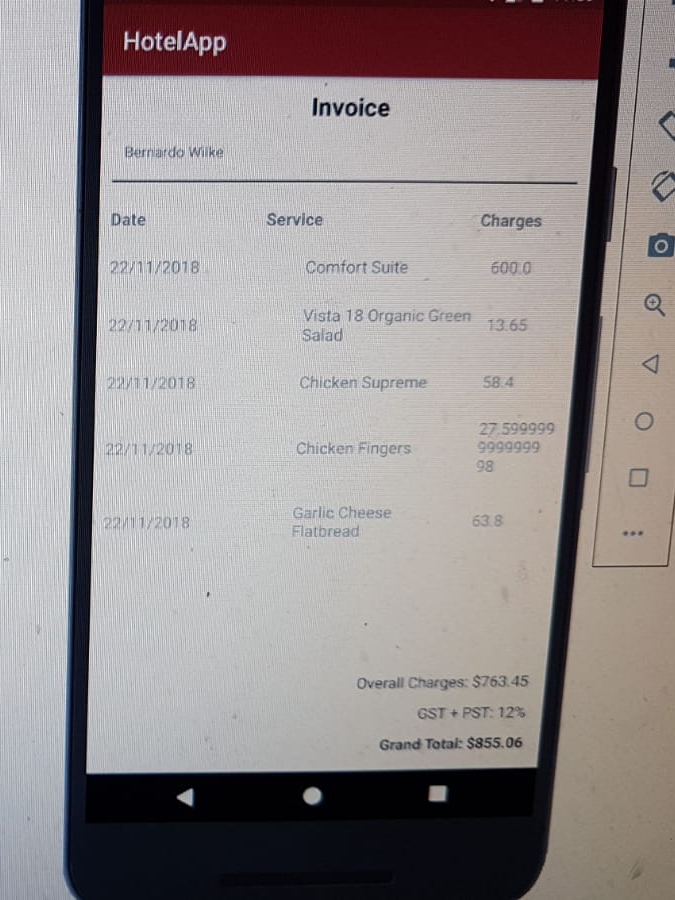
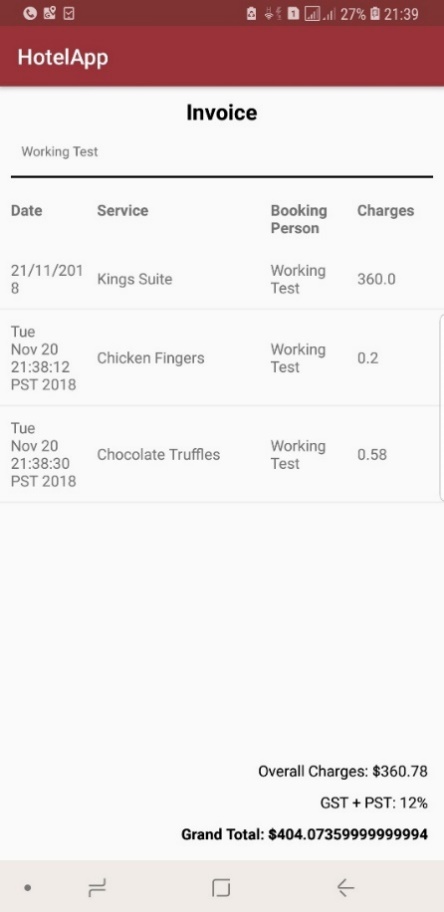


Figures #3 and #4 – 4th Issue (“date” field was not stored correctly in the db)

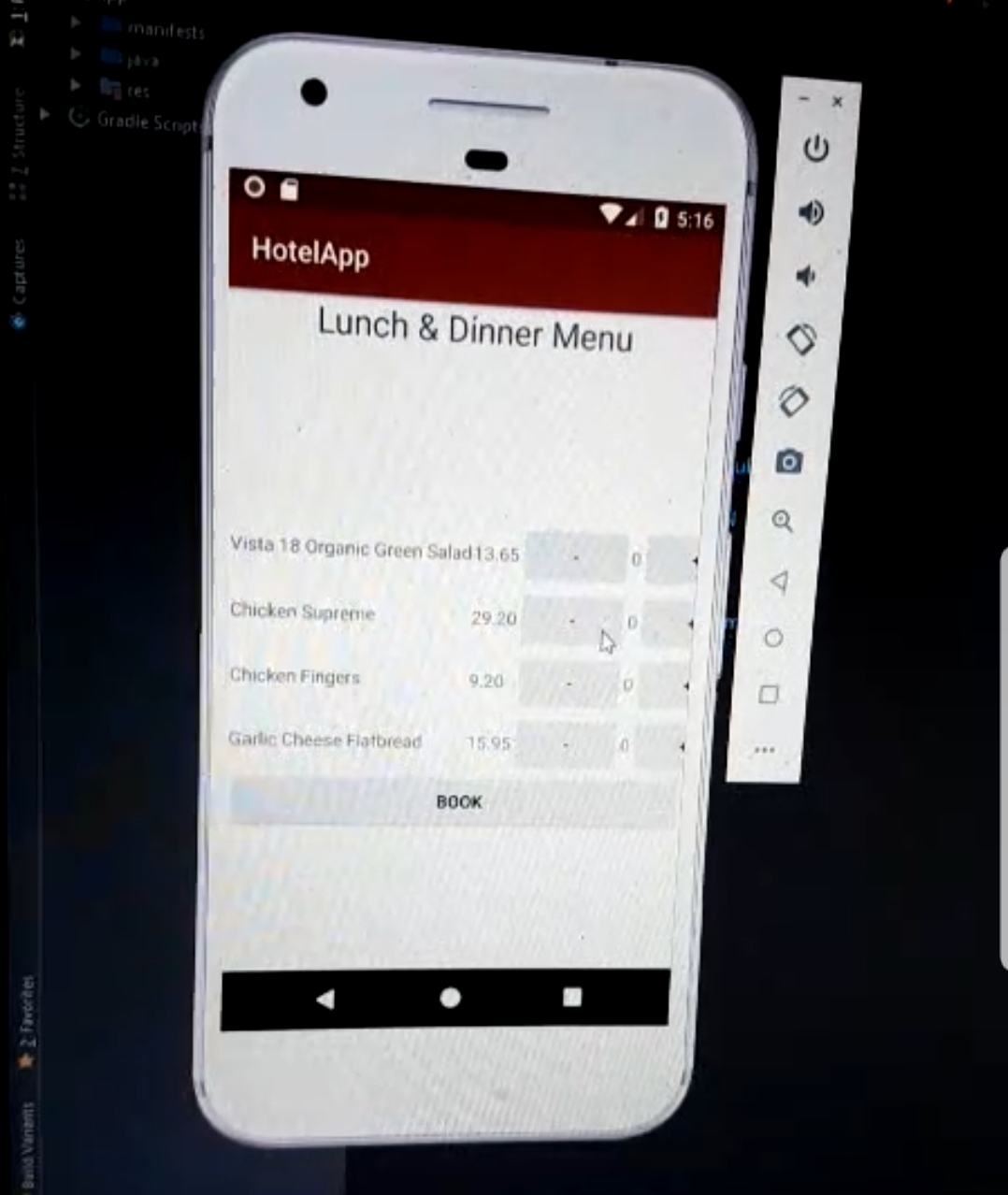


Error Fix

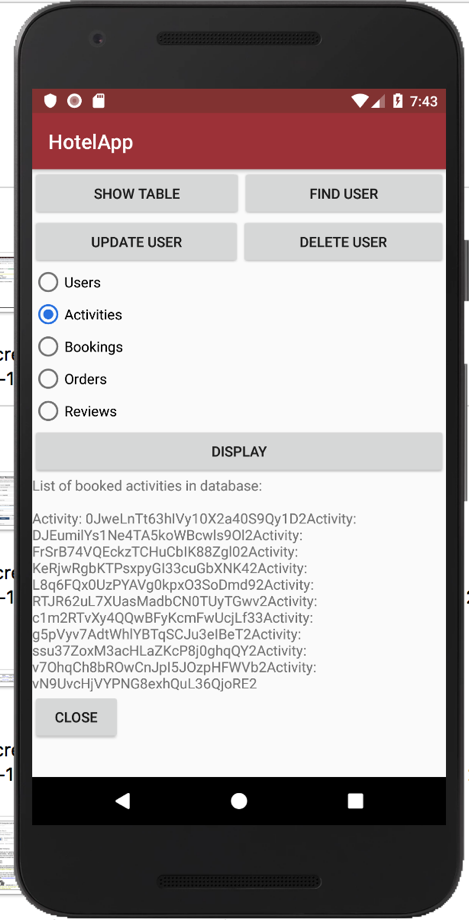
Figure #5 and #6 – 4th Issue and 5th Issue (Prices in the Invoice were not correctly formatted)



## Figure #7 – 6th Issue (The table in Room Service Booking page was cropped)



## Figure #8 – 8th Issue (Couldn`t read the information which stored under object which stored under another object)



## Figure #9 – 11th Issue (Database getGoogleApiForMethod() returned Gms and didn`t allow to do anything related to database)

