**Assignment 2 Documentation**

RobERT HURST

2017

**Contents**

[Contents 1](#_Toc484626007)

[OPSC7311 - Assignment 2 Documentation 2](#_Toc484626008)

[Introduction 2](#_Toc484626009)

[Specifications 2](#_Toc484626010)

[Flowcharts 3](#_Toc484626011)

[Card Adapter Method 3](#_Toc484626012)

[Read From File 4](#_Toc484626013)

[Write to File 5](#_Toc484626014)

[UML 6](#_Toc484626015)

[Class Diagrams 6](#_Toc484626016)

[Help File 6](#_Toc484626017)

[Home Screen 7](#_Toc484626018)

[File Based System (Restaurant) 8](#_Toc484626019)

[File Based System (Restaurant)(Add) 9](#_Toc484626020)

[File Based System (Restaurant)(Add)(Validation) 10](#_Toc484626021)

[File Based System (Restaurant)(View Item) 12](#_Toc484626022)

[Database System (Game) 13](#_Toc484626023)

[Database System (Game)(Add) 14](#_Toc484626024)

[Database System (Game)(Edit) 17](#_Toc484626025)

[Screenshots & Test Data 18](#_Toc484626026)

[Test Data 1 19](#_Toc484626027)

[Test Data 2 20](#_Toc484626028)

[Test Data 3 21](#_Toc484626029)

[Conclusion 22](#_Toc484626030)

**OPSC7311 - Assignment 2 Documentation**

**Introduction**

This is the second assignment for 3rd year students currently Studying the Diploma in Information Technology: Software Development course. This assignment was designed to test a student’s knowledge and understanding of File and Database systems in an Android Application environment. The Assignment looked at generating lists based on the information that is stored in either formats of a text file, or a SQLite database found locally on the smart phone. The Assignment was also further split into two separate applications or intents in this instance. This document will go into the various development processes I took to get to my final product, and the analysis of my own work that I produced, and the outcome. This document will also contain flowcharts and various hand drawn diagrams that I used in the development of my application code named "Muffin".

**Specifications**

The specification document that was provided stated that the following criteria had to be met :

Question 1: Choose any problem to solve with means of a file based system.

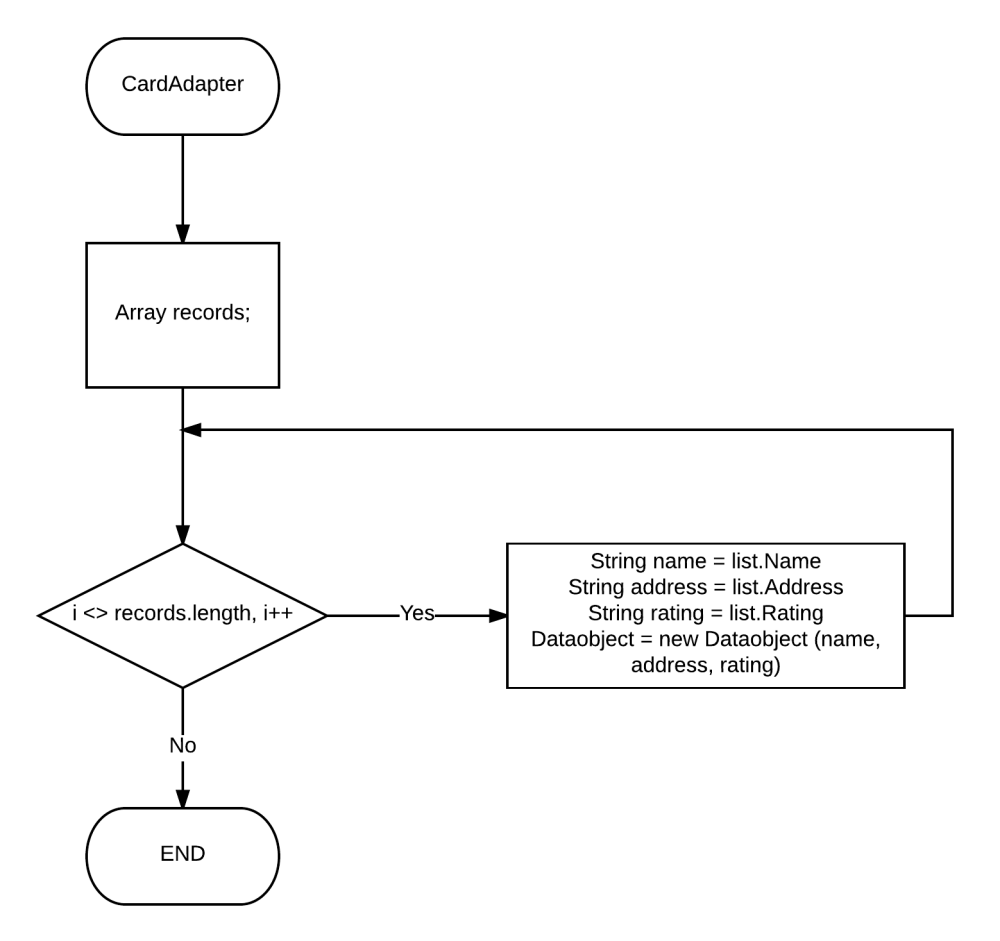
* Design the files required by your system.
* The file must contain records.
* Design a GUI interface via means of XML.
* Allow user to save data to the text file.
* View TextFile.
* View List.
* Confirm that the text file has been loaded successfully.
* Add extensive error handling.
* Research why your app does not need a traditional exit button.

Question 2: Choose any problem to solve with means of a database system.

* Create the DBAdapter class.
* DB can be programmatically created.
* Create a how to activity.
* Have 4 CRUD functions.
* Make use of Methods and Parameter passing.
* GUI designed by XML.
* Validate all input.
* Keep application simple.

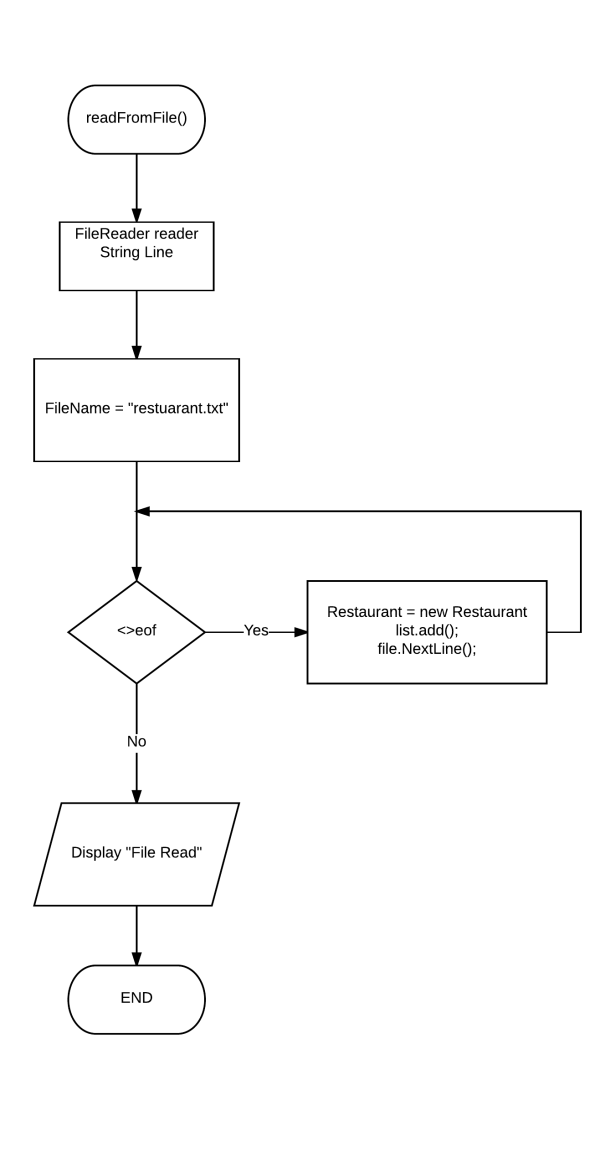
**Flowcharts**

**Card Adapter Method**



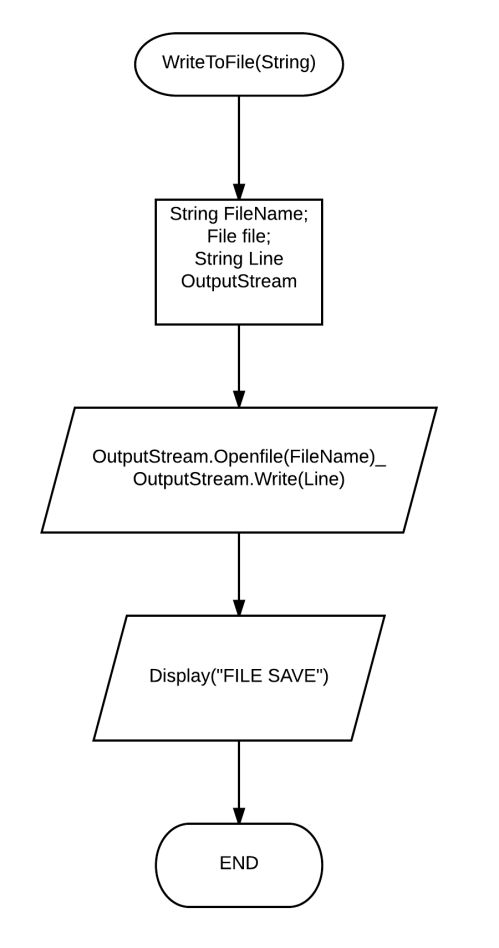
This is the method that I designed to make objects of generic data that is read in either the File or Database application. I wanted to use one class that will make use of all data. This will generate the objects based on how many objects are returned from either the database or file application

**Read From File**



This is the method that I designed to read and make objects to populate a list, that will be used to create card view objects for the list. It is very basic logic where essentially the loop will continue until file cursor is at the end of the file. The file name needs to remain constant otherwise the data will not be read from anywhere. The loop also ensures that the file reader moves onto the next line with each iteration.

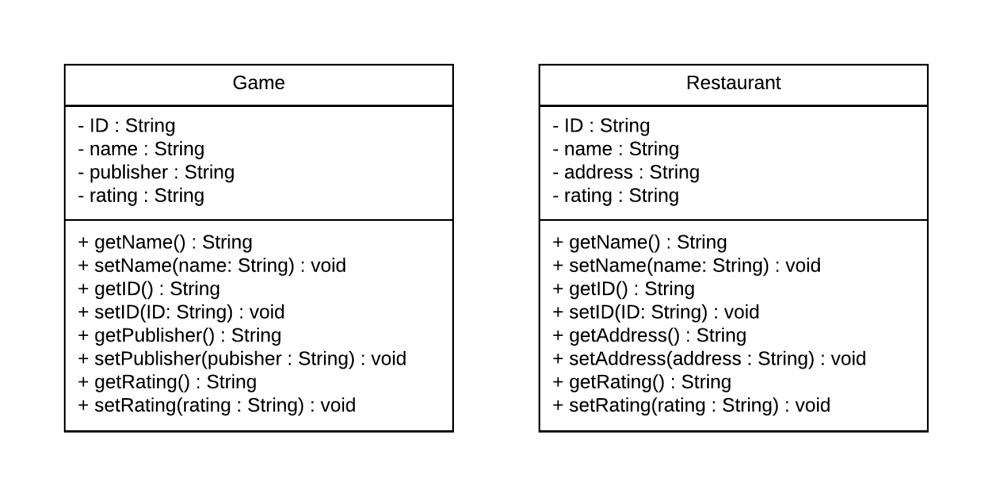
**Write to File**



This bit of logic is very important to writing to the file. Or appending to the end of it. It essentially just creates a writer object that will pass in one line, that is made up of the various fields that will be populated. The user is then displayed a messaged if the write has been successful.

**UML**

**Class Diagrams**

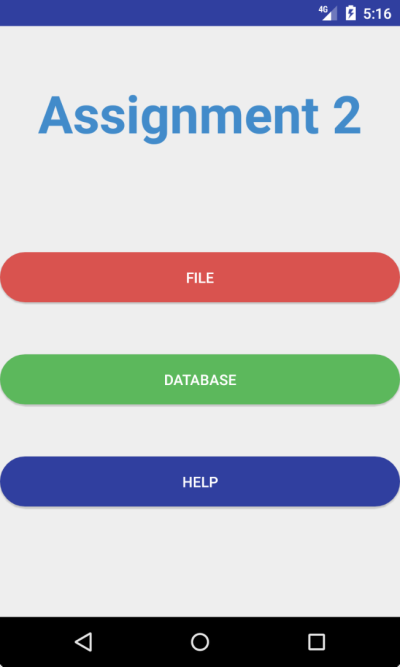


These are the class diagrams for the Game and Restaurant classes, that would be used to create objects to in turn populate the list views. Both of the classes are structured in a similar fashion as to increase code usability throughout the application. The classes show the various getters and setters that will be used to set and gain information.

**Help File**

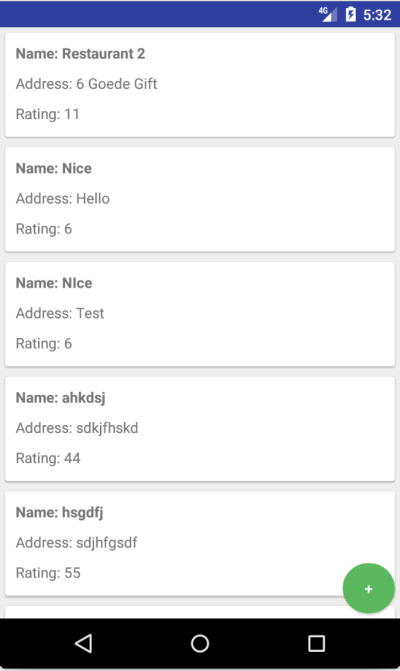
Welcome to 'Muffin' a program that allows users to enter either a small review of a Game or Restaurant. The application is used for small entries that most likely happen when the user on the go going from restaurant to restaurant, or finishing games. The Application will also display the reviews that they have created, and will also the user to View and edit. The home screen that can take you to either application looks like the following.

**Home Screen**



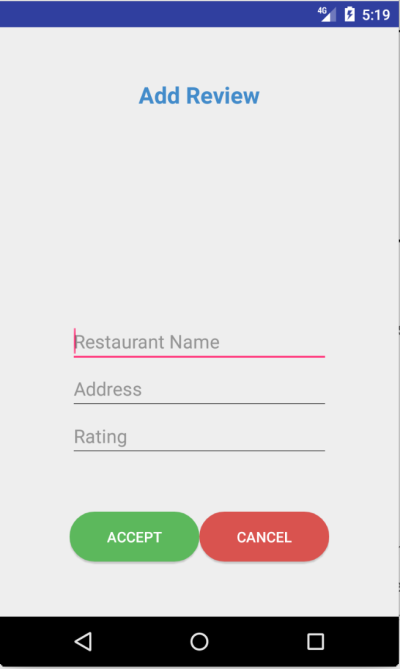
This is the home Screen. The home screen will allow the user to Access the File based sistem, or the Database system. There is also an additional help screen to guide the user on the fly. The next screen is an example of what screen will appear when pressing the File button.

**File Based System (Restaurant)**



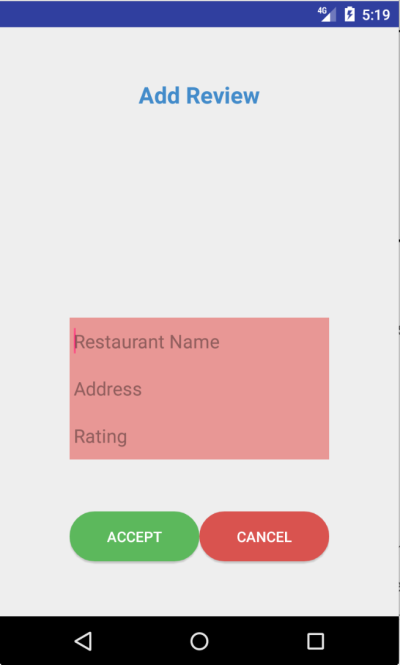
This is the list of restaurants that the user is presented with when accessing the file based system. The user can tap on any of these items and it will take them to another screen to view the review. There is also the ability to add a Review by pressing the green plus icon in the button right hand side of the screen.

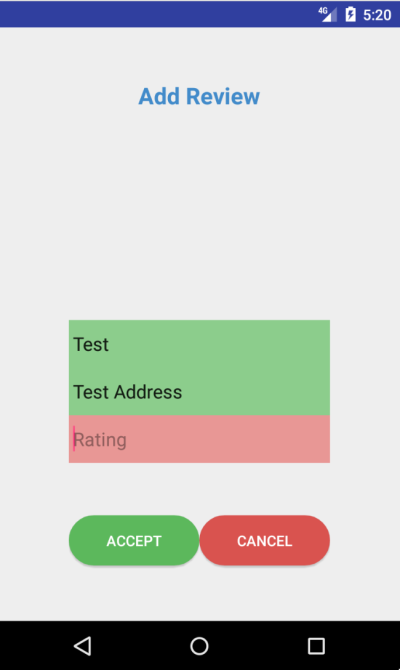
**File Based System (Restaurant)(Add)**



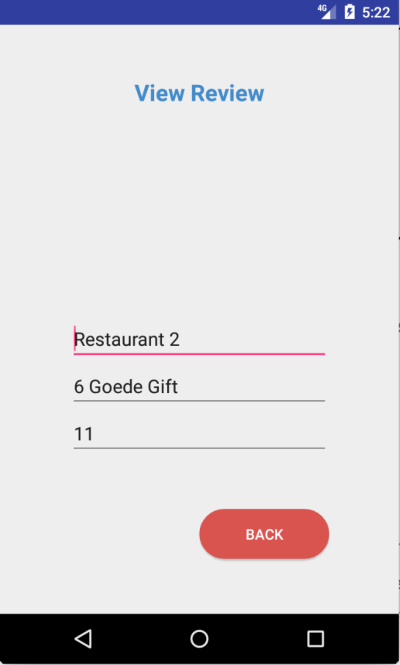
This screen is what the user will make use of to Enter the details in of a review. There are three fields that are provided. You cannot leave any of them blank. There is a check to see if the user has indeed done so. (Shown in the next Screen). There is also a check whether the user has Entered a valid rating out of 100.

**File Based System (Restaurant)(Add)(Validation)**





**File Based System (Restaurant)(View Item)**



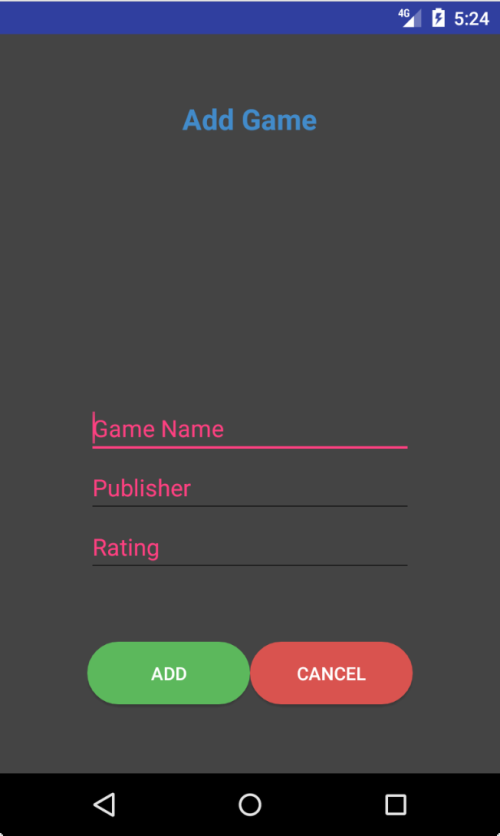
This screen allows the user to only view the entered information.

**Database System (Game)**

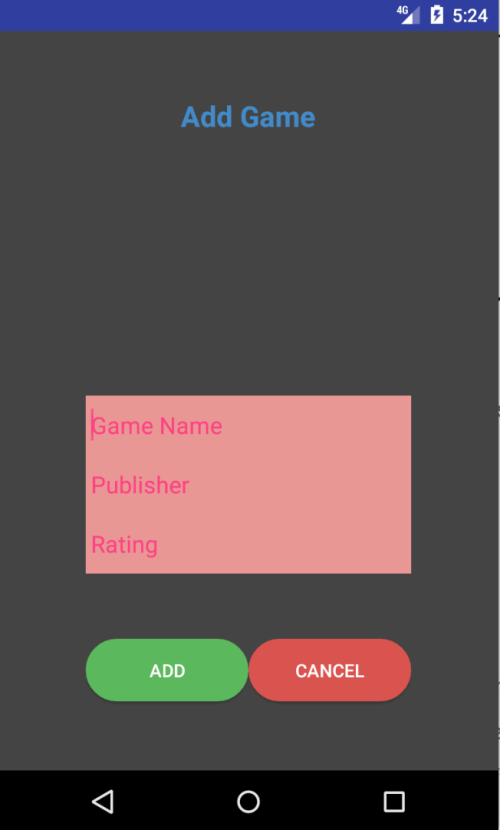


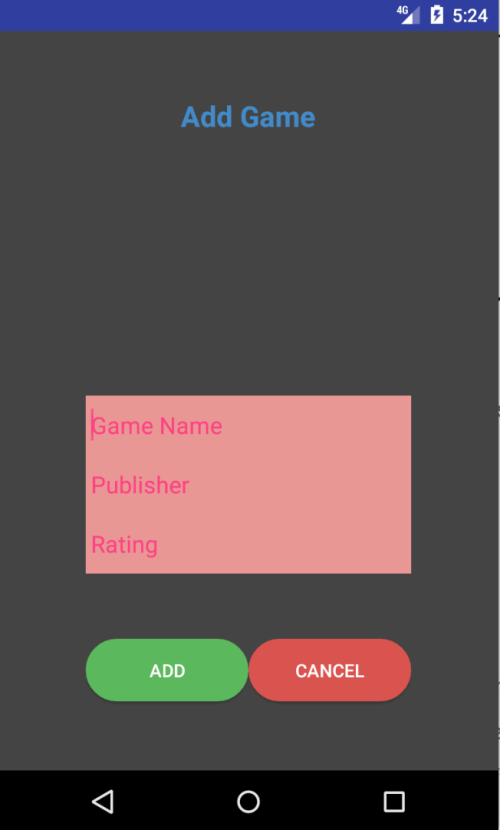
This screen shows the list of game reviews that the user has done. The List contains games that the user can tap on in order to view each item. This will allow them to edit or even to delete the item. To add a new item the user will tap the Green plus icon located on the bottom right hand side of the Screen. This will take you to the Add screen. This screen looks like the following.

**Database System (Game)(Add)**

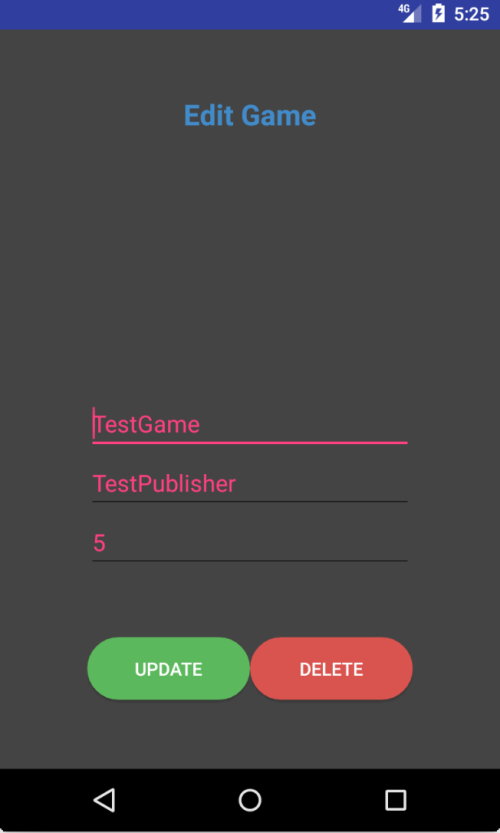


This screen is what the user will make use of to Enter the details in of a review. There are three fields that are provided. You cannot leave any of them blank. There is a check to see if the user has indeed done so.(Shown in the next Screen). There is also a check whether or not the user has Entered a valid rating out of 100.





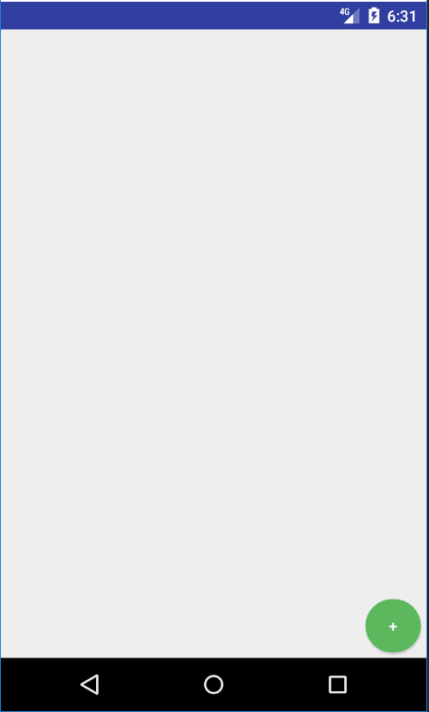
**Database System (Game)(Edit)**



This screen allows the user to only view and edit the entered information. It also allows the user to delete a record from the database.

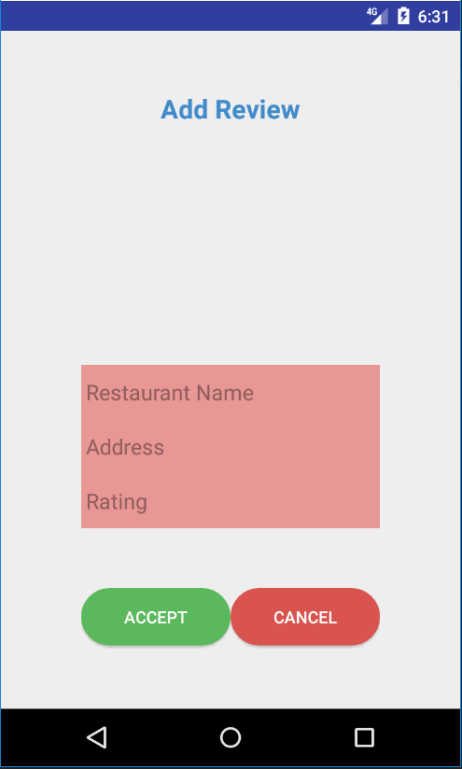
**Screenshots & Test Data**

The tests that i will be conducting for the File based system will be to Provide essentially three sets of data. The first set of Data is Blank (" "," "," "). The second set will be to prove that the Validation works for the first two field ("RestaurantTest","AddressTest", "Test"), this data will in turn result in a validation issue for the 3rd field. The third set of data will be completly valid. This will be the following ("Restaurant", "Address", "90"), this will result in an added record in the list.



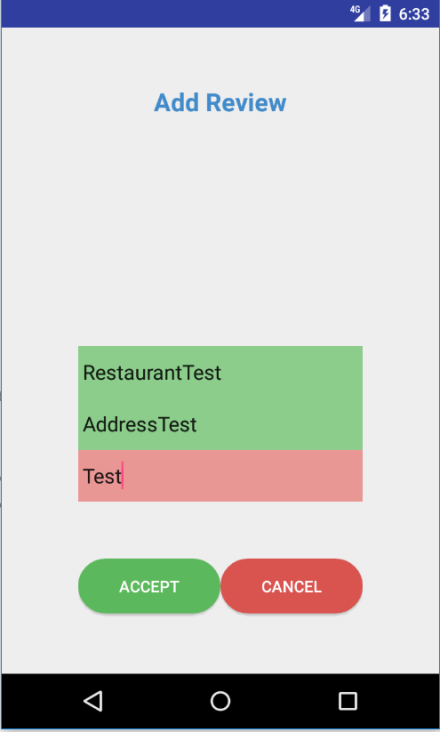
This is the blank list that we are expecting in this test. We will now press the green icon to start the creation process.

**Test Data 1**



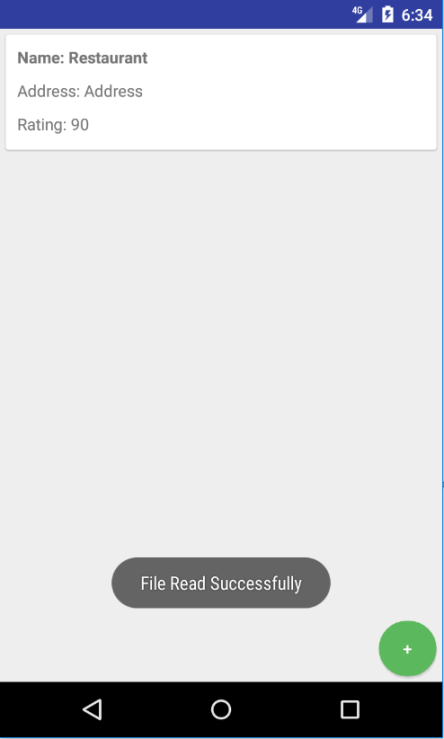
This screen takes our first set of test data, and shows the validation working for if the user inputs nothing into the fields. Not allowing them to create a record at all.

**Test Data 2**



This data shows that the validation works with the user inputting any string into the first two fields. The 3rd field expects a number but I supplied it with a string instead, resulting in an error not allowing the user to create a record.

**Test Data 3**



The third set of data is completely valid in that the user supplied data for all fields, and that the rating is not a string. This then created a valid record which in turn is then written to file. And the file is then read.

**Conclusion**

As a developer, this assignment has tested my knowledge of writing and reading to files, and learning about the various restrictions of writing to the SD card, and other places within the phone. I have also learnt the major advantage of making use of SQLite. SQLite had a lot more structure within it, allowing you to also make use of various relational ideologies that we see in other RLDB's. It also allows for better organization of data within the application. Also, the basic CRUD operations are a lot easier to make use of, as the methods are almost built into SQLite. As a developer, I am starting to look at things critically in that I am coming up with ideas for Applications that I would want to make use of. I am looking to fill a possible need for a problem. My application that I have developed had a lot of potential to be something better than what is currently is. There can be major changes to the amount of detail for each review. And possibly include additional functions such as photographic and geolocation. In terms of meeting the criteria for the assignment, I have essentially achieved what is required, and for that, it is a decent application. But for future projects something like this can evolve to something much better.