University of Sunderland
Faculty of Computer Science

CET325 Project: Assignment

Student Name: Thomas William Gaff
Student Email: bg88vx@student.sunderland.ac.uk
Student Number: bg88vx Programme: Computer Science

Design

The way the app is implemented is that the first screen a user comes across is the home screen, this home screen contains some information on the museum. From here the user can access the other features of app, which are the Store page for the ticket prices, the Painting List which is the list of some notable paintings and the RSS feed which contains an RSS feed of cultural news. The Store page has 2 text views which display the Adult Price and the Student Price, with a spinner which allows the user to change their currency settings which is saved. They can also edit the price for the adult and the discount for the student price, which is also saved. The paintings list contains a listview which displays the painting, its title, artist, year and the room. There is an RSS feed which displays the NY Times culture RSS feed and the user can change the RSS feed if they want to.

The storage for the app is all done through the SQLite database or shared preferences. There is no need for any data to be already installed on the phone for the app to work either.

There are 2 RESTAPIs that are used in this app, the first is fixer.io for the currency conversion and the other is RSStoJSON which is used in the RSS feed.

The home page is designed to be user friendly with buttons clearly labeled with words or symbols in the case of the store. The store is set out in logical manner with the spinner at it's logical spot on the bottom. The paintings list and the RSS feed, are set out in a very simple way with each item being identical in size. As for the storage, SQLite was used as it was inbuilt into Android and it is a good way of making sure that data is persistent. As for using shared preferences, they are the easiest way of storing variables and option selection that a user makes and restoring to those settings. Fixer.io was used for the currency as it was the suggested one but not only that, it is easy to use and is also easy to implement. As for the RSStoJSON I have used that api before so I was used to it and it makes processing RSS feeds a lot easier than doing the code yourself.

Evaluation-Core Requirements

1. Home screen

1.1. It is the main point of entry for your application. It should display some general information about the museum to the user when the application opens and provide access to the core functions of your app. **Y**

1.2. Use your creativity to create attractive screen. The UI should look professional. **Y**

2. Info screen

2.1. **Admission screen:** an icon button (always visible) in the Action Bar of the Home Screen should lead to screen displaying information about the ticket prices. **Y**

2.2. Supported currency should include GBP and the local currency. If the local currency is GBP then EUR should be the second currency. **Y**

2.3. When the app runs for the first time, it will display prices in the local currency. **Y**

2.4. The app should have a predefined exchange rate, but should be able to be update (automatically when the app starts, on request later) downloading the exchange rates from a data provider (for example Fixer.io a JSON API for foreign exchange rates). **Y**

2.5. The user should be able to change the currency and this setting should be persistent. **Y**

2.6. As soon as the user change the currency, the ticket prices should be displayed in the new currency. **Y**

2.7. Price for ticket for adults can be edited, and ticket for students is recalculated accordingly to a discount rate (editable). e.g. Adult: £ 10.00, Student discount: 30%, Student £ 7.00 **Y**

3. Data Structure and preloaded data

3.1. The app should be able to install without any data pre-stored on the device. **Y**

3.2. The app should create the database at the first run, and recreate if the database does not exist. **Y**

3.3. Your app should fill the database with at least 10 paintings, which should be displayed on the list of recommended sites. **Y**

3.4. Data should include the following files: “Artist”, “Title”, “Room” “Description”, “Image”, “Year” and “Rank”. Image should display the actual painting. “Year” is the year when the painting was completed. Rank is an integer between 1 and 5. (0 means not ranked) **Y**

4. Master view

4.1. The users should be able to display a list of a few (3-5) paintings (the rest of the list can be visible by scrolling). **Y**

4.2. Visible field in the list should include at least “Artist”, “Title”, “Image”, ”Year”. **Y**

4.3. All images displayed should have the same size and be small enough to accommodate the space in an efficient way, but not too small that they become unrecognisable and each element of the list should ideally occupy the same space on the screen (regardless the length of the text fields. **Y**

4.4. The user can rank the paintings during the visit. The app should be able to display (without need to open another activity) the list of paintings in the following modes: “all”, “ranked”, “not ranked” **Y**

4.5. The list can be sorted by the user by “rank” (decreasing), alphabetical order “title” and “artist, title” **Y**

4.6. The user should be able to add another painting. “Artist”, “Title”, and “Year” are mandatory fields (see detail view). “Image” should be a default “icon” until an image of the painting is taken (see desirable req. 1) **Y**

4.7. The user should be able to edit/delete a place/site he/she has added, but any not pre- loaded entries (except the “rank” field”, see 5.2/5.3)

4.8. Data should be stored persistently. **Y**

5. Detail view

5.1. If the user selects an element on the master view, a detail view will be displayed and the user will see the full information about a painting. **Y**

5.2. For preloaded data only “Rank” can edited. **Y**

5.3. For added records, any field can be edited. **Y**

6. Technical Requirements

6.1. IDE: Android Studio 2.3.3 **Y**

6.2. com.android.tools.build:gradle:2.3.3 **Y**

6.3. compileSdkVersion 23 v

6.4. buildToolsVersion "26.0.1" **Y**

6.5. AVD Nexus 5 Android 6.0 API23 **Y**

Evaluation-Desirable Requirement

I opted to go for the RSS feed. I have designed an activity in which the user can access an RSS feed of cultural news, it is defaulted to the NY Times culture section and the user can add another RSS feed to take data from although this overwrites the previous one they used. I used the RSS to JSON API to convert the RSS feed to a JSON as I have used this API before and was familiar with how to implement it.

Test Strategy

I ran monkey 5 times with 10,000 clicks each time and there was no crashes occurred during this.