## Android Fundamentals Project Self-Evaluation

**Instructions:** Once you’ve completed your Final Project, please evaluate it against the components of the rubric below. For each criteria that you met, put an “X” in either the “Does Not Meet Specifications” or the “Meets Specifications” box. For some criteria, we ask you to provide an explanation of where and how it was implemented in your app. This is a chance for you to briefly explain to the grader your thought-process during development. Once you are done, include this with the source code and accompanying files you are submitting. Then, give yourself a pat on the back for making a great app!

### Required Components

To “meet specifications”, your app must fulfill all of the criteria listed in this section of the rubric.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Does Not Meet Specifications** | **Meets Specifications** |
| Standard Design |  |  |
| App does not redefine the expected function of a system icon (such as the Back button). |  | X |
| App does not replace a system icon with a completely different icon if it triggers the standard UI behavior. |  | X |
| App does not redefine or misuse Android UI patterns, such that icons or behaviors could be misleading or confusing to users. |  | X |
| App includes a tablet layout which takes advantage of the additional space (if possible). |  | X |
| App includes at least two distinct views and uses intents properly to move between these views. |  | X |
| **Navigation** |  |  |
| App supports standard system Back button navigation and does not make use of any custom, on-screen "Back button" prompts. |  | X |
| All dialogs are dismissible using the Back button. |  | X |
| Pressing the Home button at any point navigates to the Home screen of the device. |  | X |
| **Permissions** |  |  |
| App requests only the absolute minimum permissions that it needs to support core functionality. |  | X |
| App does not request permissions to access sensitive data or services that can cost the user money, unless related to a core capability of the app. |  | X |
| **Please elaborate on why you chose these permissions:**  In addition to basic network permissions, the app uses coarse location. Location is used to determine a set  of nearby restaurants. Coarse location was chosen (as opposed to fine location) because the exact location  of the user is not that important. Within a city block or so, is perfectly fine. |  |  |
| **Performance and Stability** |  |  |
| App does not crash, force close, freeze, or otherwise function abnormally on any targeted device. |  | X |
| **ContentProvider** |  |  |
| App implements a ContentProvider to access locally stored data. |  | X |
| If it regularly pulls or sends data to/from a web service or API, app updates data in its cache at regular intervals using a SyncAdapter.  If it needs to pull or send data to/from a web service or API only once, or on a per request basis (such as a search application), app uses an IntentService to do so. |  | X |
| App uses a Loader to move its data to its views. |  | X |
| **1) What's the content provider called, and how is it backed?**  data.DataProvider  The data is stored in a sqlLite database consisting of a single table named “restaurant”. That table contains name, address, phone number, and location data.  **2) What backend does it talk to? What is the SyncAdapter called? What mechanism is used to actually talk over the network?**  Restaurant data is pulled from the Yelp! API. This is done in the services.SearchService IntentService. The SyncAdapter is ui.WheelAdapter.  The SearchService uses Yelp!'s API to retrieve data and populate the restaurant table. The Yelp! API uses Oauth authentication, and provides the data in JSON format. The SearchService parses the JSON data and inserts records into the database using the content provider's bulkInsert() method.  **3) What loaders/adaptors are used?**  An anonymous CurserLoader is created in activities.MainActivity. The loader gets initialized with a query that will return all rows in the restaurant table. The loader passes this result to WheelAdapter as a cursor. WheelAdapter is bound to WheelView which uses the data in WheelAdapter's cursr to draw the Lunch Wheel. |  |  |
| **User/App State** |  |  |
| App correctly preserves and restores user or app state. |  | X |
| When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used. |  | X |
| When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state. |  | X |
| **Please elaborate on how/where your app correctly preserves and restores user or app state:**  When the app is launched from the Home screen, the user is always taken to the main page. This is as intended. There is no attempt to return them to the results page. When re-started from All Apps, the user will be taken to either the Home screen, or the results page depending on where they left off.  All other applications state is maintained in either share preferences (location info, max distance, min rating) or  the sqlite database's restaurant table. The restaurant table has a column to mark the more recently selected restaurant. This allows the user to click on the widget and be taken directly to the result. |  |  |

### 

### Optional Components

To receive “exceeds specifications”, your app must fully implement all of the criteria listed under at least two of the four categories below (e.g. Notifications, ShareActionProvider, Broadcast Events, and Custom Views).

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Does Not Exceed Specifications** | **Exceeds Specifications** |
| Notifications | N/A |  |
| Notifications do not contain advertising or content unrelated to the core function of the app. |  |  |
| Notifications are persistent only if related to ongoing events (such as music playback or a phone call). |  |  |
| Multiple notifications are stacked into a single notification object, where possible. |  |  |
| App uses notifications only to indicate a context change relating to the user personally (such as an incoming message). |  |  |
| App uses notifications only to expose information/controls relating to an ongoing event (such as music playback or a phone call). |  |  |
| **Please elaborate on how/where you implemented Notifications in your app:**  Did not implement notification. |  |  |
| **ShareActionProvider** |  |  |
| Uses ShareActionProvider to share content with an outside application. |  | X |
| Makes use of Intent Extras to send rich content (i.e. a paragraph of content-specific text, a link and description, an image, etc). |  | X |
| **Please elaborate on how/where you implemented ShareActionProvider:**  When a restaurant is chosen, a share icon appears on the results page. The share message contains the restaurant's name, address, and phone number. |  |  |
| **Broadcast Events** | N/A |  |
| App intercepts broadcast events. |  |  |
| App responds to Broadcast events in a meaningful way. |  |  |
| **Please elaborate on how/where you implemented Broadcast Events:**  Did not implement broadcast events |  |  |
| **Custom Views** |  |  |
| App creates and uses a custom View. |  | X |
| App uses a novel View that couldn’t sufficiently be satisfied by the core Views in Android. |  | X |
| **Please elaborate on how/where you implemented Custom Views:**  ui.WheelView is a custom View that shows nearby restaurants in a carnival wheel style. When the user performs a “fling” motion, or clicks the spin button, the wheel begins spinning. After a few seconds, the wheel stops, and the chosen restaurant is displayed.  WheelView can show between 1 and approximately 50 items. It performs some basic math to ensure that the label is centered within each slice and that the name does not extend outside the bounds of the wheel. It also re-calculates it's size depending on the size of the canvas.  There's a lot more that could be added to this View including:   * accepting an array of colors for the background of the slices * specifying the text style * message if there is no data, or too much data to display * pegs around the outside edge, and a flipper to catch on the pegs |  |  |