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: The app uses the following permissions that are required for it's core functionality. 1. INTERNET – Required for fetching news data from the remote server. 2. ACCESS_NETWORK_STATE – Required by Google Play Services being used by the app. 3. READ_SYNC_SETTINGS – Required by the sync adapter implemented in the app. 4. WRITE_SYNC_SETTINGS – Required by the sync adapter implemented in the app. 5. AUTHENTICATE_ACCOUNTS – Required by the sync adapter implemented in the app. 6. RECEIVE_BOOT_COMPLETED – Required to receive the boot completed event by the broadcast receiver implemented in the app.		
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? The content provider is called NewsProvider and is backed by a SQLite database. **2) What backend does it talk to? What is the SyncAdapter called? What mechanism is used to actually talk over the network?** The content provider reads and writes data from/to a local SQLite database. The sync adapter, called NewsSyncAdapter, periodically fetches news data over the internet from the New York Times Developer API. It uses the HttpURLConnection class within it's onPerformSync() method to connect to and get data in JSON format from the remote server. This data is then parsed within the getNewsDataFromJson() method and inserted into the database using the content provider. **3) What loaders/adaptors are used?** The app uses cursor loader to load data from the content provider and cursor adapter to create, and provide the views.		
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: 1. In the MainActivity class, the app stores it's instance state into the out bundle in the		

onSaveInstanceState() method and restores it in the onCreate() method. 2. In the HeadlineFragment class, the app stores it's instance state into the out bundle in the onSaveInstanceState() method and restores it in the onCreateView() method.		

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		
Notifications do not contain advertising or content unrelated to the core function of the app.		X
Notifications are persistent only if related to ongoing events (such as music playback or a phone call).		X
Multiple notifications are stacked into a single notification object, where possible.		X
App uses notifications only to indicate a context change relating to the user personally (such as an incoming message).		X
App uses notifications only to expose information/controls relating to an ongoing event (such as music playback or a phone call).		X
Please elaborate on how/where you implemented Notifications in your app: 1. The app sends a notification to the user whenever fresh news data arrives from the server. This is vital as it provides the user with timely intimation and the opportunity to launch the app and check the latest news updates, if he/she chooses so. 2. The notification is implemented inside the NewsSyncAdapter class, which upon completion of a sync		

operation, creates and sends a notification within it's sendNewsNotification() method using the NotificationCompat API. 3. These notifications provide a ticker, are stacked and not persistent. The user can tap the notification to launch the app and need not explicitly dismiss them.		
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: 1. The share functionality provides the user the ability to share the news story headline, summary and related uri with others. 2. The ShareActionProvider is implemented inside the DetailsFragment class and shows up in the options menu. 3. It is created inside the onCreateOptionsMenu() method using the MenuItemCompat API. 4. The intent for the ShareActionProvider is set once the data has been bound to all the views in the fragment. 5. Additionally, the app also implements a floating action button for the share functionality as it is one of the key features of the app.		
Broadcast Events		
App intercepts broadcast events.		X
App responds to Broadcast events in a meaningful way.		X
Please elaborate on how/where you implemented Broadcast Events: 1. The app intercepts the ACTION_BOOT_COMPLETED and ACTION_REBOOT events from the system and triggers an immediate fetch of fresh news data from the server. This makes for a more engaging user experience as he/she is greeted with a notification on fresh news hot off the blocks whenever the device		

boots or restarts! 2. The app implements a broadcast receiver called NewsReceiver that listens for these events and then responds by invoking the NewsSyncAdapter to start a sync operation in it's onReceive() method.		
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:		