**Lyle Simpson: S1436436**

**Mobile Platform Development - Design Report**

UI & Architecture Design Principles

The goal of the user interface was to present information about earthquakes occurring in the UK in an engaging manner, and a manner which allows the important data relating to these earthquakes to be easily and quickly identified by the user. The interface was to make effective use of colour in order to quickly derive important metrics from the data, such as using colour to quickly make judgements about attributes such as the magnitude of an earthquake. A key goal of the interface was to present information succinctly to the user, to provide informative feedback to user actions such as loading the data and performing searches, and to allow for smooth navigation between the activities in the application.

The user interface favoured a simple, minimalistic design intended to succinctly summarize important seismic information for each earthquake event. This adheres to Nielson’s usability heuristic which states that applications should favour an aesthetic and minimalist design. This is accomplished on the earthquake list view by making appropriate use of the Android padding and margin attributes when displaying the list of seismic events, and also by pruning out less relevant data from the earthquake model – for example the “link” attribute. These design considerations allow the important information displayed on the page to have sufficient room, allowing for very quickly identification by the user. This avoids the list view being too cluttered to identify important data, giving each seismic event in the list the appropriate level of visibility to the user. Reducing clutter and information overload, as well as improving visibility, are key HCI principles which have been adhered to in the application.

The application also attempted to use the “recognition rather than recall” heuristic in order to allow users to easily learn how to navigate and use the application. By consistently placing toolbars on each activity, and by having a bottom navigation bar attached to each activity, the user does not need to learn new information on each screen, and can instead rely on tools they have already used. The icons used to support these toolbars are standard Android icons, using Material Design themes, further promoting usability. By keeping things consistent and well designed, the user is more likely to remain engaged by the application and try out a greater variety of its features, and helps avoid them leaving due to a confusing or overly-convoluted user interface.

Another important HCI principle considered in the building of the application is that of providing communication and feedback to the user. An example of this is when the user makes the HTTP request to the XML endpoint. Network activity can slow down the application, and thus the execution of this code is done on a background thread. While the information is being gathered, a spinner user interface element is shown, providing feedback to the user that the work is in progress. This is opposed to leaving the user in the dark, and then suddenly updating the user interface when the network request is finished. Providing feedback such as this makes for a more engaging and informative user experience, and this is a key HCI and usability principle. Additionally, to prevent network latency overly affecting the application, the data is cached to a SQLite database when downloaded, and when an update is available, the user is asked if they would like to download the update. This mechanism prevents the data from being reloaded when it is not necessary, allowing the user to peruse the application without needless reloads. In addition, by caching the data locally to an SQLite database, the application gains some offline capabilities, and can work in a situation where no network connection is available – albeit without the periodic updating of the earthquake list.

The application took steps to deal with the differences in orientation that can occur for Android apps. Separate layouts were created for the landing page and the earthquake detail-view based on whether the device was in landscape orientation, or in portrait orientation. The landing page had an issue in landscape mode where the image was overlapping the text and buttons, and thus a new layout file was created to accommodate this. Similarly, for the individual earthquake page in landscape mode, the informational text was not being shown to the user, and was instead hidden underneath the map. This is because the map and data were displayed in vertical LinearLayouts, and thus on the smaller vertical viewport in landscape mode, the data was pushed under the map. This was remedied by creating a new layout for landscape mode, and splitting the screen into two halves (horizontally) – the left hand side showing the map view of the earthquake, and the right hand side showing the data. This allowed the user to view everything effectively on one page, without having to scroll. Minimizing the effort the user has to go to is another key principle of good design, and creating alternative layouts for different orientations was a way to achieve this and create a more professional final deliverable.

The use of the MVVM pattern in the application allowed the screen orientation to change without the loss of the application data. Since changes to screen orientation typically destroys and recreates an Activity or Fragment, if the data was fetched directly in the Activity/Fragment, then it would also be wiped out by the change in orientation. By using a ViewModel class to hold and manage UI data, data can survive configuration changes such as screen rotations. The ViewModel class is an integral part of the MVVM pattern, and this pattern is a recommended software architecture for developing Android applications, with an implementation of the class now present in the Android Architecture Components libraries. The use of this pattern allows activities in the application to reuse the existing data when orientation changes, preventing the costly process of re-fetching the data from interrupting the user experience. This is an example of how architectural considerations can positively impact the user experience. It also has the benefit of separating data retrieval logic from the user interface logic, which is also good practice. The Repository pattern was also used, which mediated between the data store and the ViewModel, providing an in-memory representation of the earthquake data and defining a variety of re-ordering methods along with associated comparators, filtering methods, grouping methods, and database access and insertion logic.

The application processes dates and times using Java 8’s new LocalDate, LocalTime, and LocalDateTime classes, to process temporal data in the correct manner.

User Interaction

This section will document the ways in which the application can be used.

The user is greeted with a landing page upon first load. This is a styled page consisting of an image, and a button allowing them to navigate into the main application. When this button is clicked, the application will attempt to load data from the remote XML API. While this network request is being performed, a spinner will display, informing the user to wait until this process is completed.

Once the data is loaded, the user is shown a RecyclerView containing all of the earthquakes returned by the API. Details about each earthquake are displayed, and if the user clicks an individual earthquake, they are taken to a page which shows more comprehensive information about that earthquake, including a map view of where the earthquake occurred (indicated by a Marker). The list view also has a search bar that allows the user to search the list of earthquakes by location, and the list will dynamically filter based on the user’s input. There is a date-picker option, where the user selects a date and all earthquakes that occurred on that date are shown. Finally, there are a wide variety of re-ordering options – the user can order by location name, location position, date, magnitude, and depth. These can all be in either ascending or descending order.

As well as the list and detail views, there are some “summary” views within the application. Firstly, there is a map-view, which shows a Google Map of all earthquakes that’ve been recovered from the API, by their geographical coordinates. These are denoted with Markers, and tooltips which display to the user the location name and the date on which the earthquake occurred. This page contains a map as its primary user interface component, along with the navigation bar along the bottom that allows the user to navigate elsewhere in the application. There is also a statistics activity, which shows some interesting summary statistics, such as:

* Number of earthquakes
* Deepest earthquake
* Highest magnitude earthquake
* Date with most earthquakes
* County with most earthquakes
* Hourly time period where most earthquakes occurred.

This page is one of the main areas of the application which can be improved, as the styling implemented is very minimal. However, it presents some interesting facts to the user.

Along the bottom of each activity is a navigation drawer that allows navigation between the earthquake list activity, the earthquake map activity, and the earthquake statistics activity. This provides a consistent, attractive mechanism by which the user can navigate between activities.

Graphical Components

The application used standard Android user interface widgets in order to deliver an engaging experience to the user, using standard components they are already familiar with from other applications. These include the following:

* TextView – for delivering text to the user interface. These widgets are populated by values held within Model classes, formatted in a user-friendly manner – for example, dates and times are stored in the Model as LocalDateTime objects, and their native representation is converted to a more readable format with help from Java’s DateTimeFormatter class.
* SearchView – a standard widget for searching through the list of Earthquakes, along with the magnifying glass search icon. This allows the user to enter search queries.
* BottomNavigationView – a navigation bar attached to the bottom of each Activity in the application, which uses a Material Design theme and allows navigation through the application
* MapView – shows a Google Map representation of where an individual Earthquake occurred, or a Map which shows where all the earthquakes occurred. The map uses markers to denote the focal point of each earthquake, and is embellished with tooltips that display more information about the earthquake when clicked – for example, the location and the date of the earthquake.
* RecyclerView – A view that allows a large collection of data – in this case, an ArrayList of Earthquake instances – to be efficiently scrolled through. The RecyclerView provides a window into the collection, reusing existing “view-holders” as the user scrolls through the data. The RecyclerView uses a LinearLayoutManager to manage the size and positions of each item in the RecyclerView, and uses a subclass of RecyclerView.Adapter to manage populating each item within the RecyclerView with its data. Using a RecyclerView provides performance benefits when compared to a normal ListView, and for that reason was chosen for use in this application.
* DateTimePicker – A standard Android widget that allows a user-friendly mechanism for selecting a date, using a calendar-like visual representation.
* Buttons – a simple button is used to kickstart the application and invoke the first web request.

Different layout types were used to deliver these user interface elements to the end user – notably the ConstraintLayout and the LinearLayout. The ConstraintLayout allowed the use of relatively position user interface widgets, without resulting in deeply nested view hierarchies – an advantage it provides over using the conventional RelativeLayout. The LinearLayout allowed simple linear position of user interface elements, both in horizontal and vertical orientations. For example, within the RecyclerView, a LinearLayoutManager allows each earthquake item within the list to be positioned linearly from top to bottom.

Conclusion

I believe the Earthquake Tracker application was successful in using the above user interface elements, and successful in implementing the aforementioned HCI principles, in order to provide an engaging user experience. The application provides a fluid, easy-to-use user interface that allows for the browsing of earthquake data in the UK. It also allows engaging summary information to be viewed, notably map views of all earthquakes, charts showing earthquake frequency, and summary statistics such as highest magnitude earthquakes, deepest earthquakes and more. I believe the application was enhanced by the architectural decisions to store data in a SQLite database, and the decision to use the MVVM and the Repository patterns to separate user interface code from data retrieval code, thereby allowing data to be reused upon orientation changes. Overall, the application meets the functional requirements determined by the spec, meets the targeted HCI heuristics, and provides an engaging user experience.