Calorific: The Calorie Counting App

Calorific is designed to be a user-friendly application for users to track their calorie intake over the period of a day. A catalogue of regularly consumed items can be built up from input by the user, who can name and photograph the items for their in-app database. Another tab shows the total calories consumed for the selected day, and a diary option allows the user to track calorie consumption over time.

Why it is suited to be a mobile application

Quite simply, the user can add to their records on the go, conveniently. They no longer need to return to a workstation to enter the data or keep a notebook which can be easily lost and requires a writing implement. The app also has access to the devices' onboard camera, which helps the user take ownership of the app by establishing a personal connection with the data. The best applications in my research established a sense of ownership of the process of participation, which I have attempted to emulate. By capturing images of the items the user consumes, the application creates a personal link to the user's world.

The process of adding to the tracker is seamless, the user simply finds (or if necessary adds) the item by swiping through the available items and clicks the green '+' button. Likewise, if an error is made, the user can hit the '-' button to subtract from the count. Taking advantage of the devices' touch screen in this way is efficient and effortless. The layout is fun and colourful, also uses contrast to highlight varying functions such as increment, decrement, add new item etc. This makes it a pleasurable experience to use the app. In addition, today's mobile devices carry very large screens. The application therefore places the primary navigation tool – the navigation bar – at the bottom of the screen where it can be more easily accessed.

The most pertinent data is also quickly available; highest calorie items rise to the top of the 'Daily Total' section, and the 'Food & Drink' area is sorted alphabetically for easier finding of items. Likewise 'View Diary' entries are sorted by date. According to my research, users become quickly frustrated when the data and functionality they are trying to access is hidden and difficult to find, or fails in use. This is a common reason for users to abandon a platform altogether.

In summary, the on-board camera creates a sense of ownership, the use of mobile touch screen technology and design makes it easy to use and navigate, the data is readily available and editable, and the mobile format means that the application is always to hand when required.

How it has been designed for mobile use

There are a number of design decisions that have been made in order to ensure a user-friendly experience, optimised for mobile devices:

1. Content is allowed to run past the end of the screen. The user can easily scroll to view ‘hidden’ content, and this preserves space to make reading and viewing of the content much easier.
2. Mobile friendly controls have been chosen, with only the absolute minimum of two text fields utilised in the entire app. There are no drag-drop or tiny interfaces which can make interaction on a mobile device difficult. I have deliberately focussed on a simple, easy to understand UI, for example using buttons to increment and decrement instead of presenting a clumsy text field.
3. Relatively little interaction is required – browsing is smooth, and interacting with the data often requires little more than a touch, particularly after the user has setup most of their commonly used items.
4. File size of pictures taken is minimised in order to save limited space and ensure a faster load time by preserving memory. Picture quality suffers a little, but is not noticeable at the thumbnail scale.
5. The application requires no network access, and so uses none of the users’ mobile data plan. It is therefore accessible in all connection status scenarios.
6. The buttons use contrasting but complementary colours to consistently highlight different functions across the application, aiding in quick, efficient navigation.
7. Content is appropriately spaced which makes it much easier to find and digest information.
8. The navigation is simple and easy to understand and follow. There are only four unique pages in the application, with only a handful of linkages between. This makes it easier for the user to understand how to get to where they need to be.
9. The interface follows common design patterns, such as the footer navigation. This is easily within reach of the user’s thumbs and fingers in use, regardless of screen size. Additionally, the date is presented in a text field, in order to encourage users to attempt to edit it. They will find that the devices’ date selector dialogue will be called, which is optimised for the mobile experience.
10. Users expect that devices ‘simply work’ the way they want. There was some thought made to how the device loads the date. I have supposed that a user would not want to have to select today’s date every morning, however I imagined that they would equally be frustrated if the date changed back to today’s date every time they changed page or re-opened the app. Thus, I have implemented a mechanism whereby upon first opening the application on any given day, today’s date will be selected. Otherwise, the currently selected date will continue to be used.

How it works (a simple explanation of the code)

I used the PhoneGap emulator together with the ‘Brackets’ text editor and GitHub source control to develop and build the application. Having started out with an Android Studio project, I found instead that PhoneGap was ideal for my intended set of features; use of web technologies also enabled a much speedier deployment. I built the app in HTML, CSS and JavaScript, together with the Bootstrap framework which I used to standardise the layout. Using a core structure of four pages and five script files, I created a consistent design and coding style in order to ensure efficient maintenance. A shared, ‘main.js’ file helped to ensure that I was only coding functions once.

Each page was built out of the same underlying template to ensure a consistent style. This consistency was reinforced elsewhere: the feeds in the ‘Food & Drink’ and ‘Daily Total’ pages were actually built using the same underlying repeater function; merely differentiated by supplying a different SQL statement for an argument. By minimising the code footprint in this way, I have made it easier to locate and solve bugs.

The layout is robust, I have used CSS to apply a proportional (rem) margin at the top and bottom in order to prevent the ‘fixed’ header and footer from obscuring the content. Otherwise, the app is built up in it’s static and dynamic parts by using containers, rows and columns from the bootstrap framework to ensure flexibility and scalability (re-orientating the device from portrait to horizontal simulates a larger screen).

IDs and onclick / onchange attributes are used in the HTML to hook into from the JavaScript layer. These onclick/onchange attributes often refer to shared functions, and in the case of dynamically generated content are dynamically given an argument on loading the page. Most data-handling is done by WebSQL using a database object, although some simple objects such as ‘state’ and ‘date’ are held in the ‘localStorage’ object. Initially, I wanted to use the ‘sessionStorage’ object for the date, so that it could be assigned fresh on startup. However, I found that this was not persistent across the app and is in fact emptied between pages. Instead, the ‘localStorage’ variable called ‘today’ is checked with the function ‘initialise’. If this is not equal to today’s date, then it and the ‘date’ variable (representing the currently selected date) are changed. This means that a more advanced procedure is followed, that the user can continue with his currently selected date until the following morning when the date will be refreshed.

Pictures are retrieved using the camera API. This proved simple to use and implement. On returning the file-path to the saved image, the ‘src’ attribute of the thumbNail image is used to contain the file-path. This can then be updated into the ‘index’ table of the WebSQL database when the tick button is clicked. The detail page, which handles editing and adding new items, is controlled by the ‘localStorage’ variable ‘state’, which is in turn set by either clicking on edit (‘state’=EDIT), in the feed, or add (‘state’=ADD), from either the ‘Food & Drink’ tab or the ‘Daily Total’ tab (index and counter pages, respectively).

I implemented the application using only two tables, index and counter. Index records the commonly consumed food and drink items, and counter matches this data using the ‘key’ field, to a date and quantity. Interesting composite data can be drawn out of this so that for instance the ‘View Diary’ screen can show the total calories consumed for a day by summing the products of quantities and calories of respective items consumed only on each day. As an interesting quirk, I also implemented a feature that the ‘View Diary’ tab will show as the picture in the feed for each entry, the item that contributed the most calories on that day, using a similar SQL query.

Implementing the increment and decrement feature of the items in the feed proved an interesting challenge, in two respects. When adding an item that did not yet exist for that day, it could not simply be edited using an update query, likewise, it should not be possible for the user to decrement a quantity below zero. A solution was found by using an insert query if adding a new item for the selected date; similarly deleting each item from the date when subtracting the last of the item.

Performance is somewhat sluggish in the application. My primary thoughts are the application could be slowed by the size of images or by inefficient SQL calls. While I have attempted to mitigate this by reducing picture quality to 10% and avoiding sub-queries where possible (preferring instead to use more efficient inner and outer joins) further testing, analysis and debugging may reveal potential performance improvements.