Mobile phone app: formal spec

This spec describes the CycleStreets iPhone app as currently implemented to version 1.01. It has been written to make creation of CycleStreets apps on other platforms more easy. It forms a template from which apps for other platforms can be created. These apps will not be a direct copy, since each platform has its own norms (e.g. menu handling, dialog styles) but this spec forms a basic framework for the design of these apps.

1. Overview

The app is basically a front-end client which talks to the CycleStreets API.

It implements both the Journey Planner and Photomap components of CycleStreets, and both should be considered essential for a release. The Journey Planner aspect is higher priority and should be given most attention in terms of avoiding bugs and interface deficiencies.

It assumes a touch-screen interface.

2. API

The API is documented at www.cyclestreets.net/api .

An API key must be obtained from www.cyclestreets.net/api/apply/ and is stored in the application binary. The key must not be put within any public source code distribution; it should remain private at all times to the developer. If a key is accidentally revealed, please contact CycleStreets.

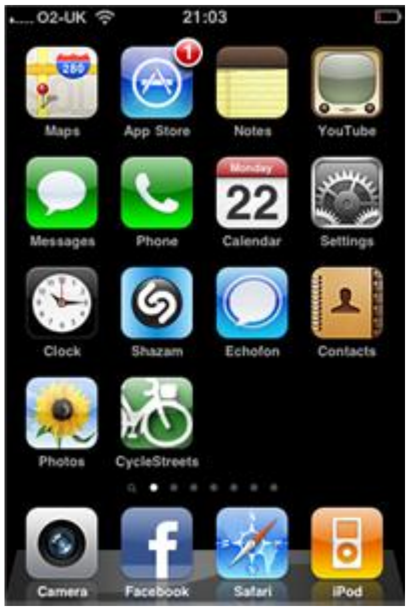
Undocumented calls must not be used and should not be required. If new calls or options are required for future feature ideas, the developers are pleased to consider adding these.

3. Labelling style

Throughout the application, labels use the British English “Sentence case” rather than Americanised “Capitalising Each Word” case.

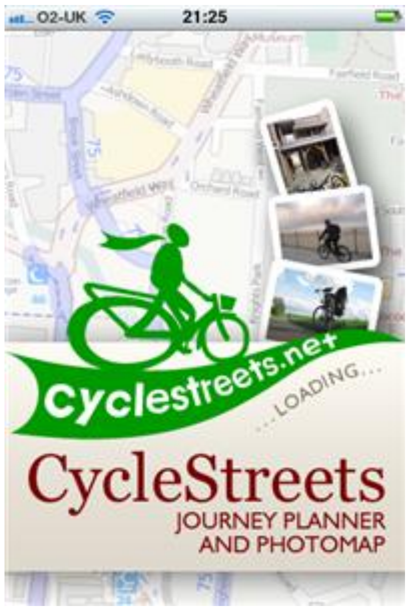
4. Application name, icon and loading screen

The icon used in the current app has been designed professionally, but the next version should use the icon used on the main site at http://www.cyclestreets.net/favicon.ico which, after some deliberation, is considered more stylised. This screenshot shows a mockup home screen using the proposed new icon:



The application name which should appear next to the icon is 'CycleStreets'. Note the capitalised S in the middle.

A splash screen is used upon load. This has been professionally designed. We can contact the designer to produce an amended version to the required screen size. The dimensions of the iPhone original are 320×480px.



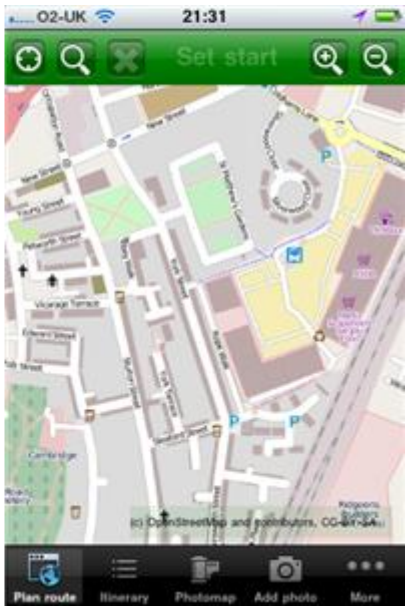
5. Overall layout

The iPhone app uses the standard metaphor of a menu along the bottom. There are five functions: "Plan route", "Itinerary", "Photomap" (note: the central 'm' is not capitalized), "Add photo", "More". Each includes an icon above it. The currently-selected function is visibly active by having a blue background.

Only upright (vertical) mode is currently supported. Horizontal mode could be added.

The app does not currently run as a native iPad app, though this could be added. An iPad will run the app in iPhone emulation/upscaling mode.

The Operating System's top bar remains visible at all times in white (rather than black) mode.



Green is used as the general colour theme.

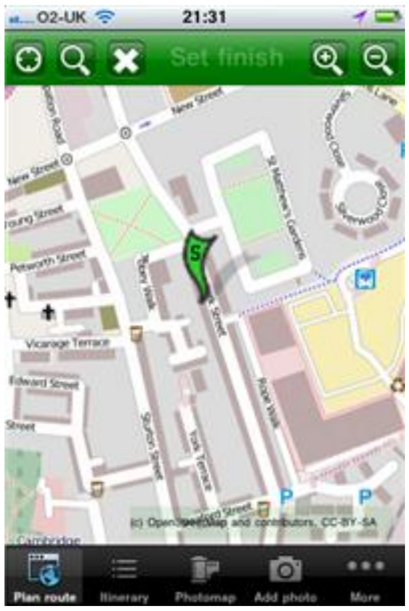
6. Journey planner

6.1. Overview

The Journey Planner enables planning of a cycle route from A-B anywhere within the UK. The user must select a start point and a finish point so that they can then plan the route. (Note that we always use the term 'finish' rather than 'end', as the latter term can be confused with the 'ends' of the route.) Once a route is planned, it is shown as a line on the main screen, and is loaded into the Itinerary option in the menu.

6.2. Screen layout

The screen layout is as follows:



The top control bar includes controls from left to right: Geolocation (using the standard design representing geolocation), Placefinder (magnifier class), Step-back (x), then a section to provide an instruction (referred to as the commentary panel), then buttons for zoom-in (magnifier with +), zoom-out (magnifier with -).

The map panel is made as large as possible after allowing sufficient space for reliable touch of the controls. The map panel contains an attribution; the attribution is detailed in a later section.

The main menu remains visible.

6.3. Setting the start / finish points

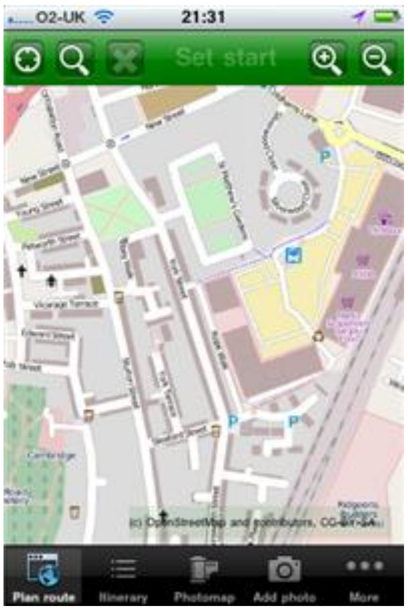
The predominant interface mode is an innovative three-taps concept: touch on the map to set the start point, touch to set the finish point, and click to plan. This avoids the complexities associated with a drop-pin approach. However, this UI is subject to review (see discussion below).

The Start point, when set, is represented by a green wisp icon containing the letter S. The Finish point likewise is red and contains the letter F.

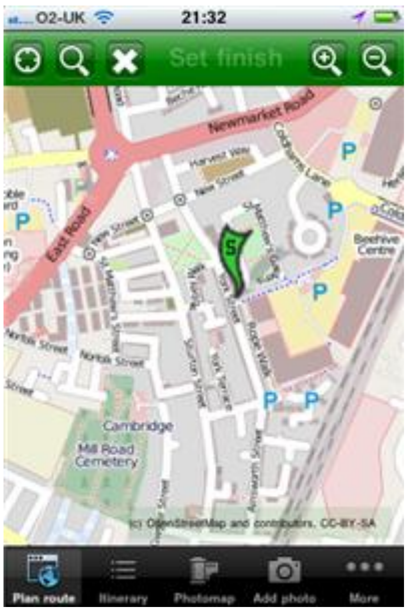
There are three ways to set the start and finish points. Either by touching directly on the screen as noted above, or by geolocating the user's current geographical location, or by using the placefinder.

6.4. Three taps, step by step, resulting in a planned journey

When no journey is displayed, i.e. in a default state, the screen shows "Set start" as the instruction in the commentary panel. This is referred to in this documentation as the 'cleared initial state'.



The user taps the screen. The start marker is then displayed. The shadow is part of the icon. The centre-point of the icon is the point where the user tapped. The instruction in the commentary panel is changed to “Set finish”.

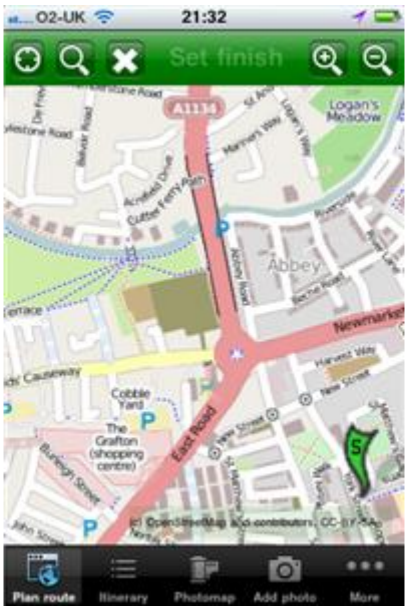


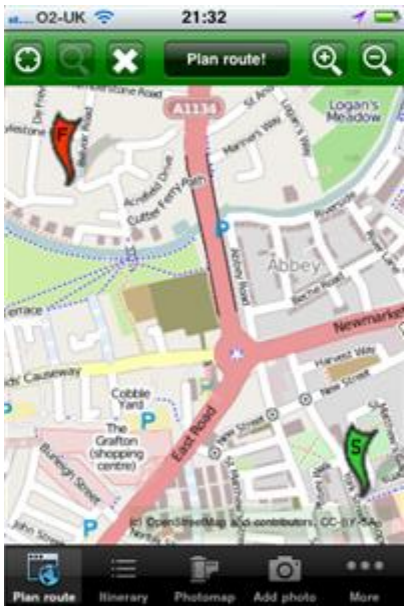
The marker is now ‘stuck’ to the map background. The user can scroll the map. The user can pinch the map to zoom out. When zoomed out, the marker remains at the same size (it does not change scale).

When zooming or panning, it is essential not to set a marker accidentally. A touch must be an explicit depression preceded by the finger not being on the screen. In practice this disambiguation is achieved by setting a timer of 0.20s after a zoom/pan operation; once the timer has expired, a touch can then be accepted. (Note that in the iPhone implementation the Route-Me library source was modified to accomplish this, as it wasn’t handling double taps properly; see DOUBLE_TAP_DELAY in RMMapView.m.)

The zoom out/in buttons have the same as using a two-finger pinch or reverse pinch respectively.

A second touch is treated as setting the finish point. The instruction panel then changes to a button "Plan route" and the geolocation button moves to an inactive state:

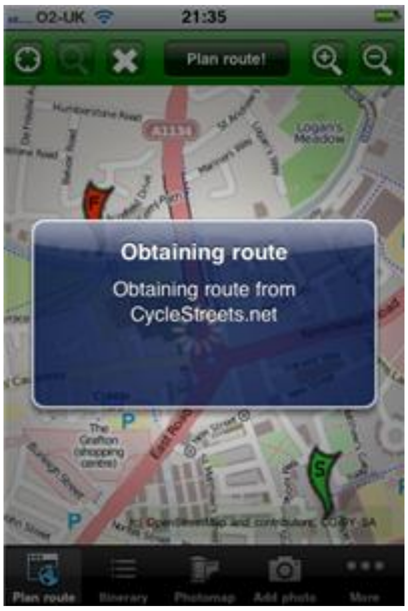




The map position does not change after this tap (i.e. it does not attempt to show both markers if that was not naturally the case) – it stays exactly where the user last left it. In general the application works on the basis that the map should not try to auto-pan or move unless the user has explicitly requested that (by touching or using the geolocation/placefinder; these two functions are explained later).

The app prevents the user directly setting (touching down) the finish marker within (the equivalent of) a few metres of the start marker. A special case is also handled to deal with this from the result of geolocation or namefinding (see below).

Now that the "Plan route!" button is available, the user can plan the route. Pressing the button brings up a modal dialog, worded exactly as as follows:



Obtaining route

Obtaining route from CycleStreets.net

6.5. Route API request and display

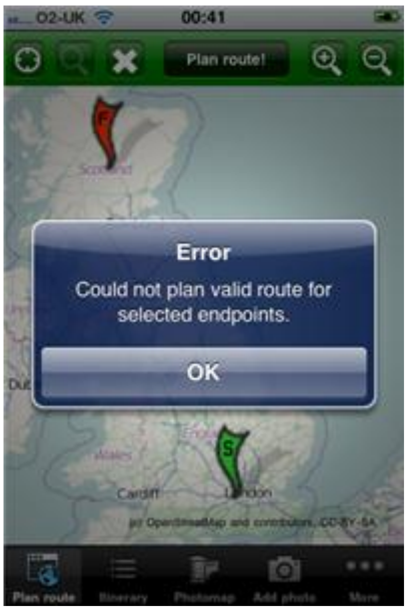
The route is then requested via the API. An example call is:

```
GET
http://www.cyclestreets.net/api/journey.xml?key=<key>&start_longitude=0.141442&start_lati
tude=52.205569&finish_longitude=0.142884&finish_latitude=52.209257&plan=balanced&speed=20
&useDom=1
```

<key> should be substituted with the license key. plan= and speed= come from the values in the settings (see later).

The XML API call timeout is whatever the underlying protocol provides for, i.e. it has not been set. It feels like 30s; 15 seconds is probably a better value for future implementation.

If the API responds with an error, or the site is unreachable, this should be displayed to the user. (Usual reasons are the route being too far away, points being too close, or an unroutable network for some reason. The server could also be unresponsive.)

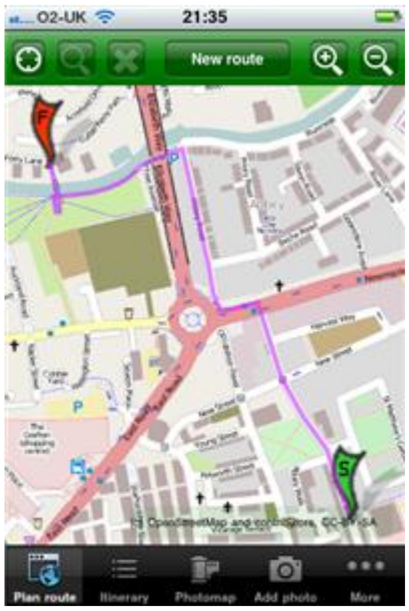


Error

Could not plan valid route for selected endpoints.

OK

Upon the route solution being received, the modal state is cleared, returning to the previous state, and the line is drawn. The line is 4px with colour {Red:0.8 green:0.2 blue:1.0 alpha:0.8}; there is then an additional 0.75 alpha filter, i.e. ending up with 0.6 alpha. (The double-alpha is an implementation quirk, and it is believed this code was never cleaned up in time.) This line design is similar to the Google Maps application shipped with the phone. The colour was chosen to ensure that it does not obscure map details such as street names.



The line should be drawn over both the map and the markers, i.e. it has the highest z-index.

The map position is redrawn such that the markers are both shown, with a little space between both such markers and their respective screen edges, i.e. pan and zoom is done.

The geolocation button remains available, but the placefinder and step-back buttons are disabled. The central button now reads "New route".

The user can pan/zoom if they wish. They can click on the geolocation button to get the current state. (See later description of geolocation functionality.)

At this point, the route solution is saved to "My saved routes". This includes the names of the start and finish points, the total time, and each part of the route. These are saved to the device and are not re-requested in the case of route recall from My saved routes.

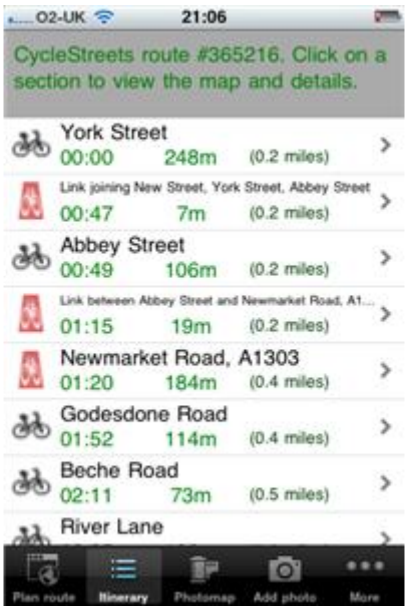
The planned route remains persistent upon returning to the Plan route screen, e.g. when closing the app or switching to another menu item and returning.

6.6. Itinerary view

A planned route is also dumped into the Itinerary screen, i.e. the second menu entry. This provides turn-by-turn details.

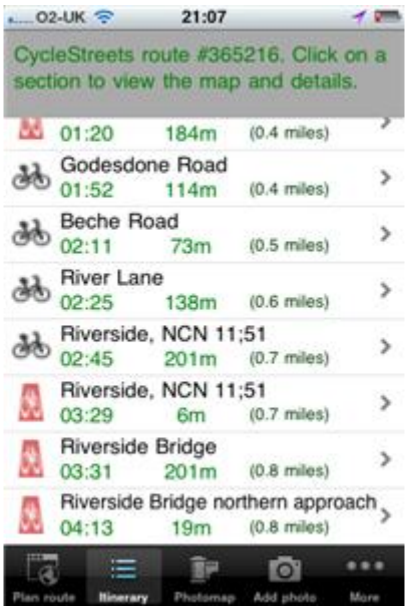
The current itinerary implementation is as follows:

There is a list of stage-by-stage entries as supplied by the API, with the route number being shown at the top, and a scrollable panel showing each part of the journey in a table view.



CycleStreets route #365216. Click on a section to view the map and details.

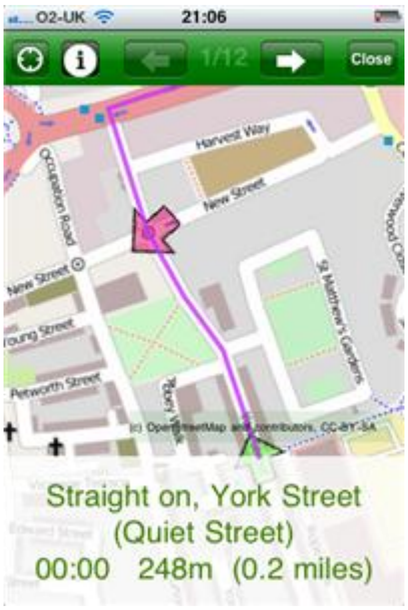
York Street 00:00 248m (0.2 miles)
Link joining New Street, York Street, Abbey Street 00:47 7m (0.2 miles)
Abbey Street 00:49 106m (0.2 miles)
Link between Abbey Street and Newmarket Road, A1... 01:15 19m (0.2 miles)
Newmarket Road, A1303 01:20 184m (0.4 miles)
Godesdone Road 01:52 114m (0.4 miles)
Beche Road 02:11 73m (0.5 miles)
River Lane



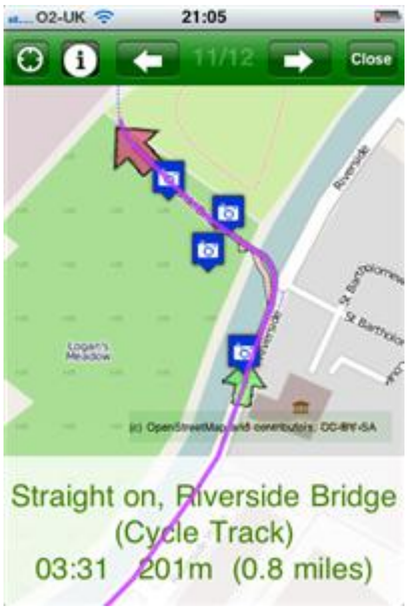
CycleStreets route #365216. Click on a section to view the map and details.

01:20 184m (0.4 miles)
Godesdone Road 01:52 114m (0.4 miles)
Beche Road 02:11 73m (0.5 miles)
River Lane 02:25 138m (0.6 miles)
Riverside, NCN 11;51 02:45 201m (0.7 miles)
Riverside, NCN 11;51 03:29 6m (0.7 miles)
Riverside Bridge 03:31 201m (0.8 miles)
Riverside Bridge northern approach 04:13 19m (0.8 miles)

Selecting an entry takes the user to more details of that part of the journey. Once the map is shown, the user can click left/right to advance through, with the stage shown (e.g. 11/12 for the 11th of 12 stages). (Note that in the first screenshot here, the arrow points left because of a very tiny turn; this is a deficiency of the API that it should try to avoid tiny turns in this way. The 'Close' button returns the user to the table view above.



Straight on, York Street (Quiet Street) 00:00 248m (0.2 miles)



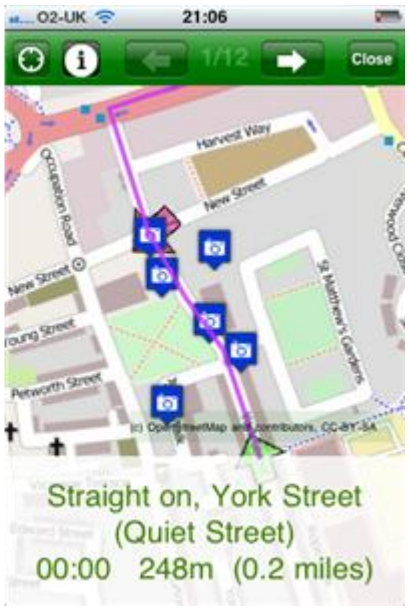
Straight on, Riverside Bridge (Cycle Track) 03:31 201m (0.8 miles)

The details of the route are shown at the bottom of the screen. The purple line is drawn over this at present, to give more context, though this looks a little scrappy, so should be reconsidered if the itinerary stays.

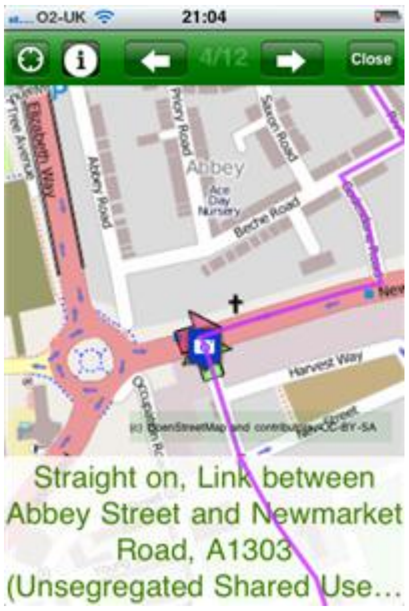
Note that the main menu has disappeared, which is not ideal but does create more space.

A proposed change is to combine the street type and distance together on one line, using a smaller font, to increase the amount of available screen space.

Photos-en-route are also shown. However, this can lead to a rather confusing/crowded screen:

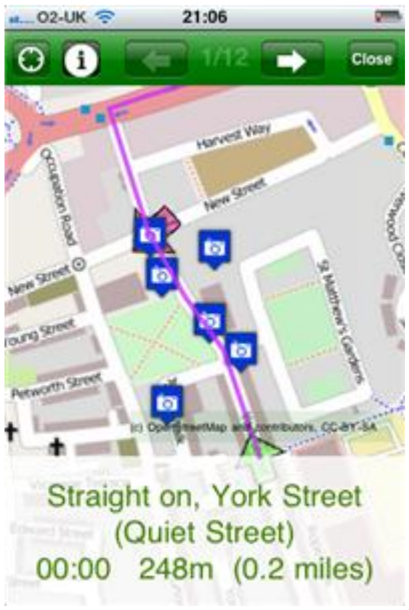


Straight on, York Street (Quiet Street)
00:00 248m (0.2 miles)

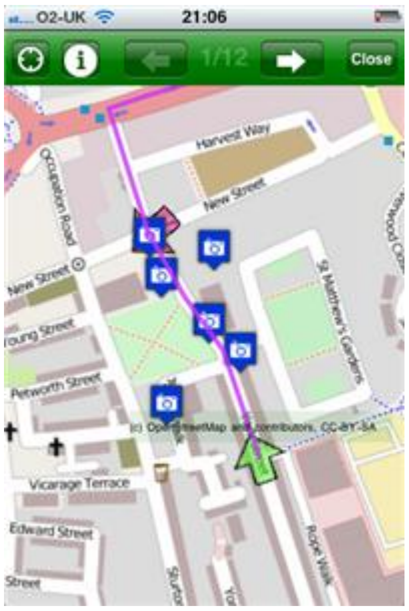


Straight on, Link between Abbey Street and Newmarket Road, A1303 (Unsegregated Shared Use…

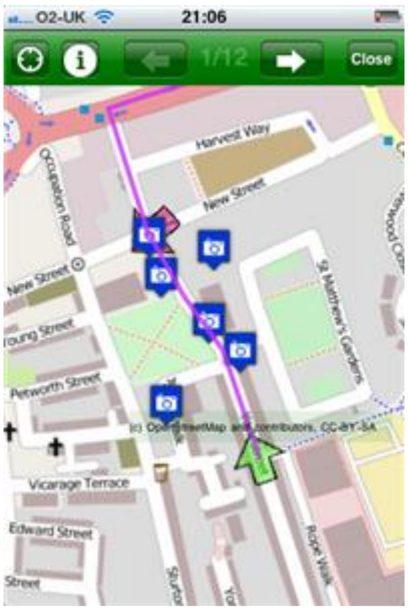
The user can remove the text area at the bottom by clicking on the (i) (for info) button at the top. Note that a bug currently leaves the map attribution in the middle of the screen rather than redrawing it at the bottom. Currently the info button does not change its design in any way; this could be amended in a future release.

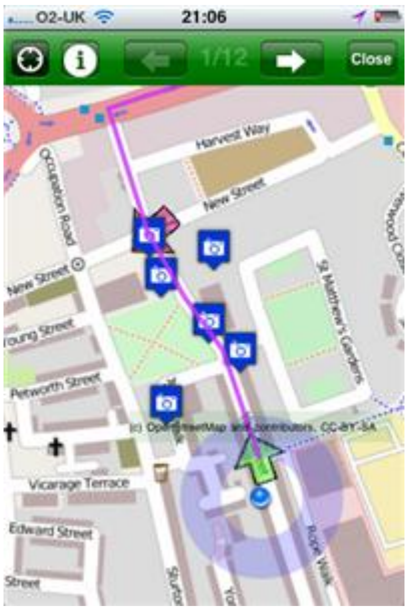


Straight on, York Street (Quiet Street)
00:00 248m (0.2 miles)



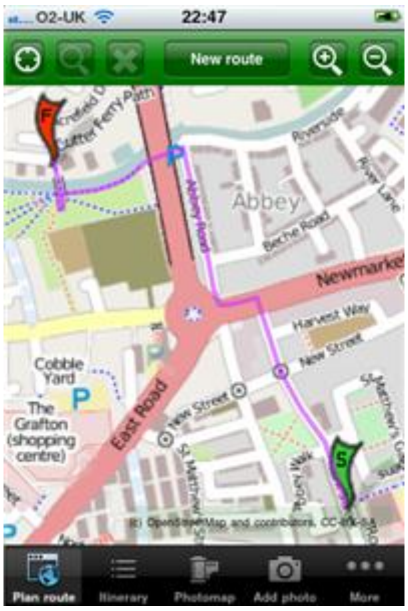
The user can also switch on display of their current location using the geolocation button. In this mode, the blue dot then follows the user. This works whether or not the info panel is on. Currently it seems to drain the battery more than is ideal, though this may be a standard iOS occurrence.

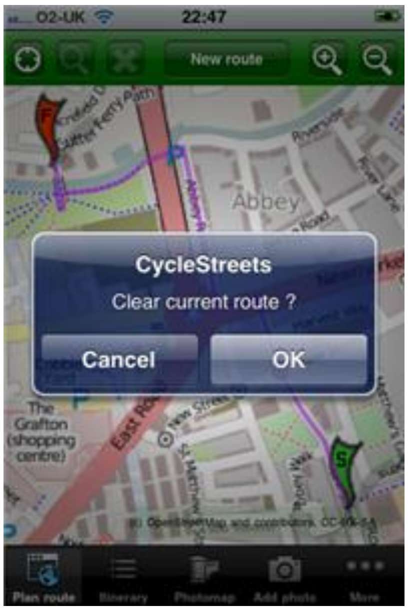




6.7. Planning another route

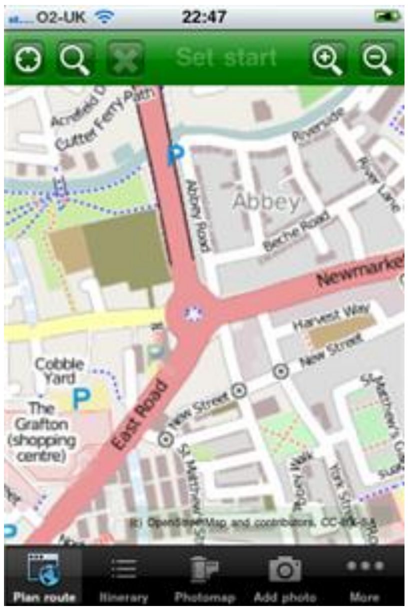
As the route remains, the user must explicitly clear it, by clicking the ‘New route’ button. A modal dialog is shown as follows, with Cancel returning to the persistent state. OK returns things to the cleared initial state, i.e. removes the dialog, removes the two markers and removes the route line but leaves the map in the same position.




CycleStreets

Clear current route ?

Cancel OK

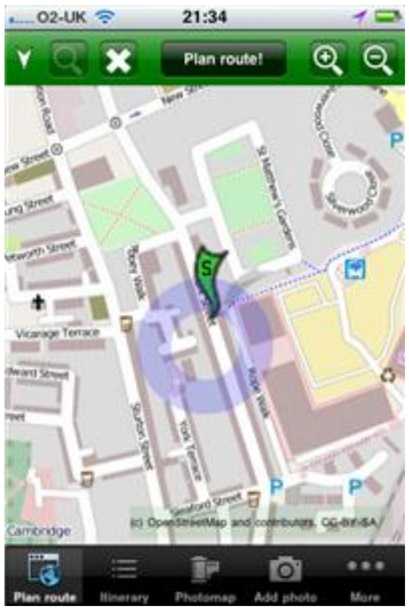


6.8. Geolocation

The Geolocation finds the user’s current location on-street.

It is a one-shot control, not a toggle. Leaving it on would drain battery life very badly so this is not done.

If the start/finish point is not set, geolocation has the effect of being like setting the marker. Otherwise, i.e. when a journey is on the screen, it shows the users’ current location.

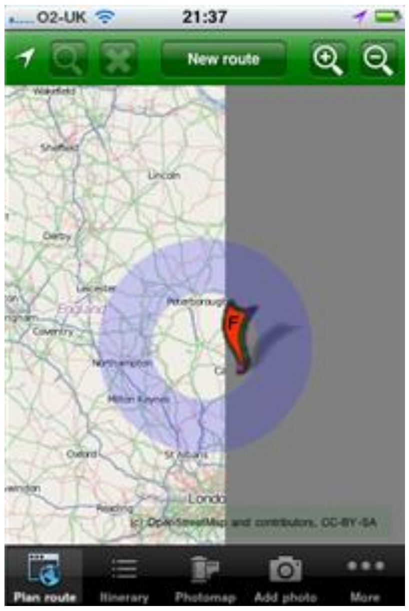


Pressing the geolocation button results in the location being searched for. The routine requests the location from the OS. During this request phase, the geolocation button shows a spinning arrow animation (at present this has 8 states, 45-degrees apart, but ideally this would be a smooth animation). Once the location is within a good level of accuracy, geolocation is considered 'successful'.

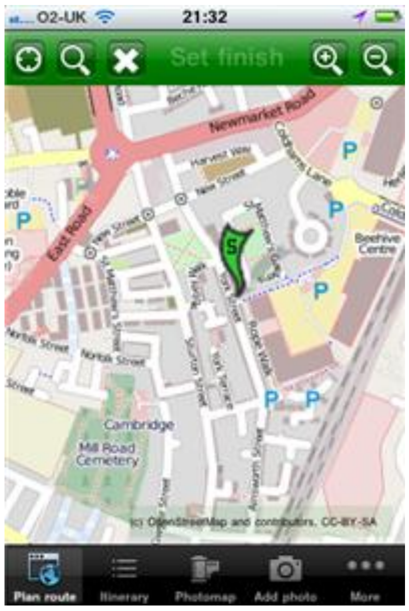
The developer explains the determination of a 'good level of accuracy':

```
Location is callback driven, not polled; you register a listener and get updates on the
location. When I get accuracy at one of the 2 levels below (in metres), I set a timer to
switch off listening n-seconds later:

static CLLocationAccuracy ACCURACY_OK = 100;

static CLLocationAccuracy ACCURACY_BEST = 20;

static NSTimeInterval LOC_OFF_DELAY_BAD = 30.0;

static NSTimeInterval LOC_OFF_DELAY_OK = 3.0;

static NSTimeInterval LOC_OFF_DELAY_BEST = 1.0;
```

If the user's location is not within the current view of the map, the map is moved to show it, with the geolocation result (at each pulse, or upon success) being in the very centre of the map panel. For the result of each such pulse, the relevant marker is set if required, and a blue circle surrounds this marker. Of course, the map background may not be in the tile cache, so a gray or partially gray screen can result given the speed of geolocation compared to the speed of download.

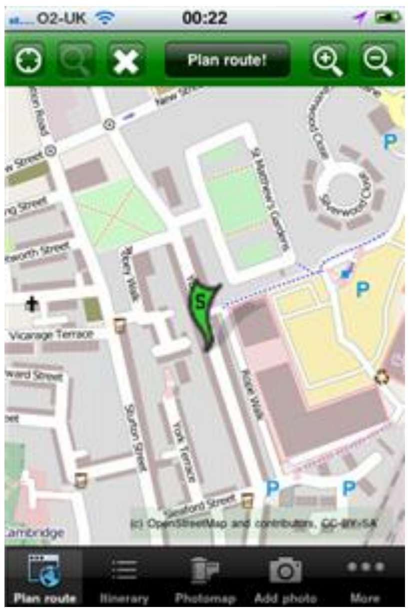


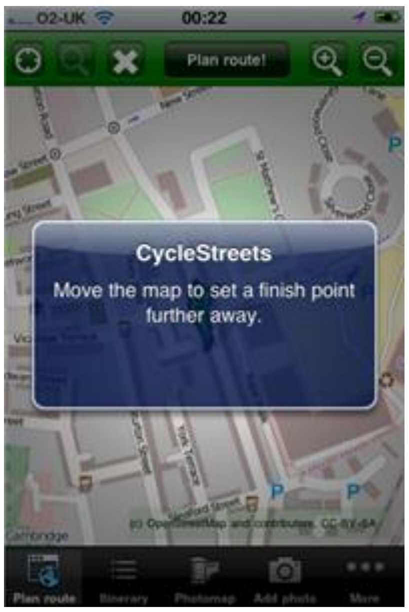
When a ‘successful’ response is achieved, the geolocation button proceeds back to its default state, with the set marker remaining.



If the user moves the map (via pan/zoom, whether using the button or touching), the geolocation system is immediately cancelled and the geolocation button returns to its unused state. If a marker has been set during the geolocation phase it is left, however inaccurate. This is because a user map move is considered to be an acceptance of the geolocation result.

If the finish marker ends up being automatically set very near (equivalent of a few metres) to an existing start marker, then the following special-case dialog is given; it goes away automatically after a few seconds and the finish marker is not set.





CycleStreets
Move the map to set a finish point further away.

6.9. The step-back button

The step back button is used during the phase of setting the start/finish marker. If the start marker is set, it has the effect of unsetting it and thus moving back one stage in the state flow. If the finish marker (and therefore the start marker also) is set, it removes the finish marker and retains only the start. As usual, the map itself is not moved, irrespective of whether the marker(s) is/are visible.

Currently, a marker can only be moved by using the step-back button, and not by dragging it also. A future release should add drag support.

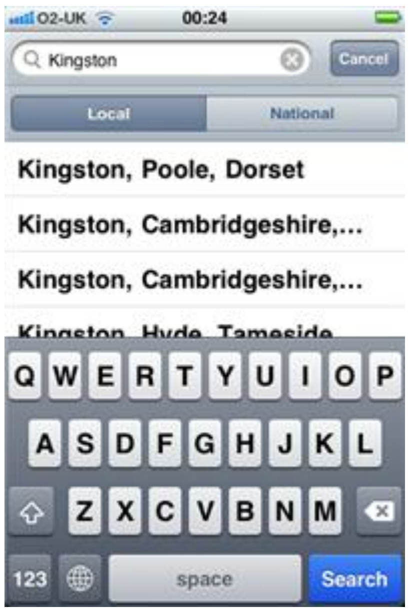
6.10. Namefinder

Rather than clicking on the map, users can use the namefinding function to search for places/postcodes/streets/towns/etc. This uses the `geocoder.xml` call in the CycleStreets API. Clicking on a chosen name inserts the underlying lat/long point as a start/finish marker in the same way as above.

Example call:

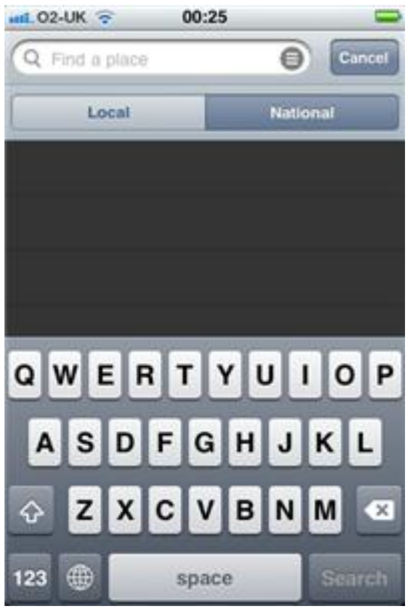
```
GET http://www.cyclestreets.net/api/geocoder.xml?key=<key>&street=Kingston&w=-
0.118818&n=52.454547&e=0.381182&s=51.954547&zoom=16
```

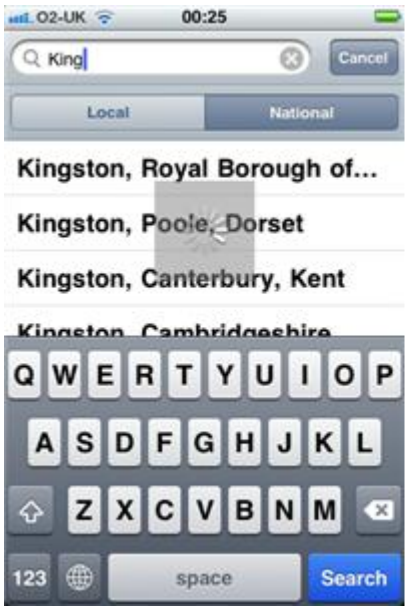
Note that names (i.e. the ‘street’) should be URL encoded, e.g. Bar%20Hill for ‘Bar Hill’.



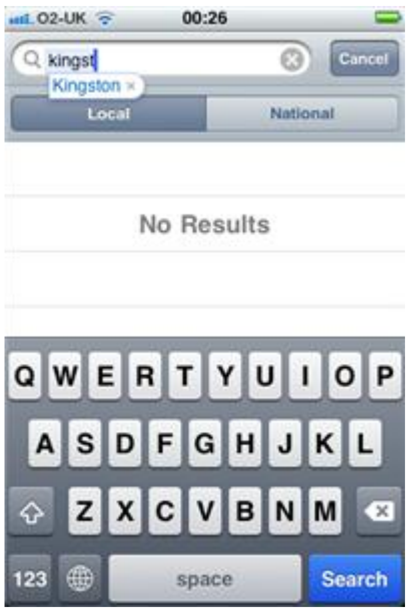
The screen for this starts off blank (currently the background is dark gray, but this should be white). As the user types, the app sends off a request to the API to fetch the results. A ‘loading’ icon overlay is shown while requests are being sent.

At present, there is no timer to determine whether the user is still typing. Instead, the algorithm is that a request is fired off once the user has entered four characters. Each subsequent keypress results in another request fire-off once the previous request returns. This implementation may need reconsideration.





If no results are available, this is indicated to the user:



Currently auto-correct is on but (due to a bug) this should not be present unless the UI spell-checker has a large resource of built-in names.

The user can choose to favour national searches rather than more local ones. The relevant API calls are:

Local (geocoder bounding box is 0.25 degrees either side of the current map centre, with zoom 16):

```
GET http://www.cyclestreets.net/api/geocoder.xml?key=<key>&street=Kingston&w=-
0.118818&n=52.454547&e=0.381182&s=51.954547&zoom=16
```

National (geocoder bounding box is 4 degrees either side of the current map centre, with zoom 6):

```
GET http://www.cyclestreets.net/api/geocoder.xml?key=<key>&street=Kingston&w=-
3.868818&n=56.204547&e=4.131182&s=48.204547&zoom=6
```

(The requirement to send both a specified bounding box and a zoom, and the use of a known set of values for the British Isles are recognised deficiencies in the API, which may be addressed as time allows.)

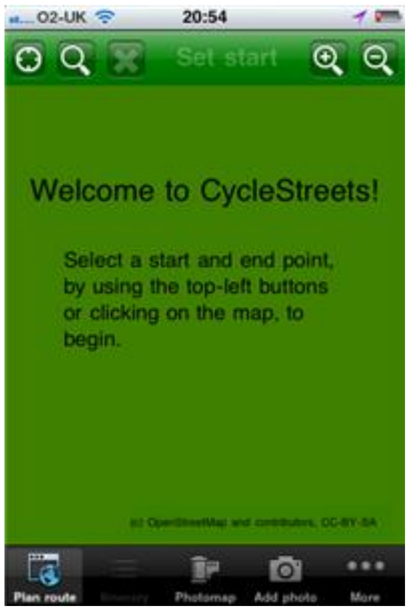
The results sent back include fields <name>, <near>, <latitude>, <longitude>. At present the <name> (e.g. Kingston) and <near> (e.g. Royal Borough …) fields are shown to the user separated by a comma. It is hoped that a future release will show the <near> field in a smaller font.

As with the geolocation marker-setting mode, if the finish marker ends up being automatically set very near (equivalent of a few metres) to an existing start marker, then the special-case dialog is given; it goes away automatically after a few seconds and the finish marker is not set.

6.11. Initial walk-through dialogs

A set of dialogs are used to help walk the user through the required operations the first time the app is used, as described below. Once they are shown, they are never shown again. These are therefore stored as part of the app's local data.

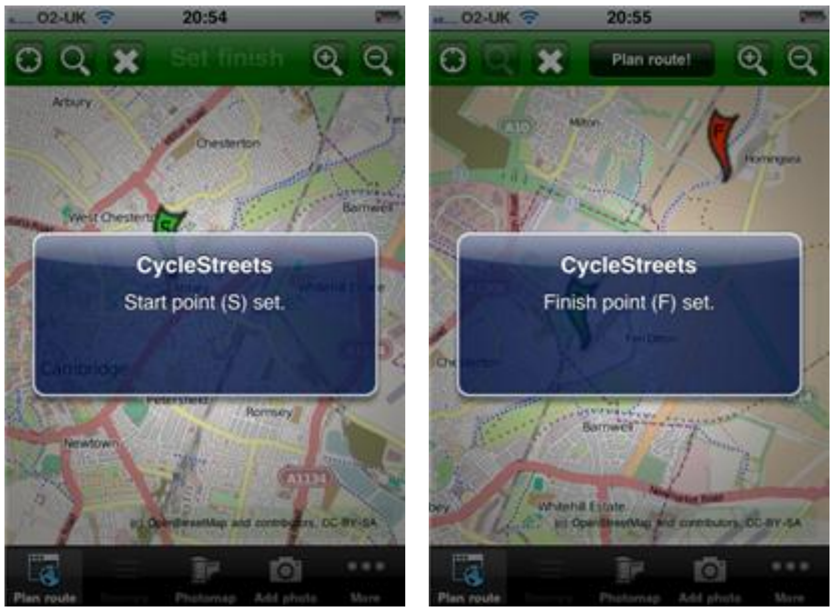
On first load, this screen is shown, which fades out after a few seconds. (It could be replaced with a standard modal dialog.)



Welcome to CycleStreets!

Select a start and end point, by using the top-left buttons or clicking on the map, to begin.

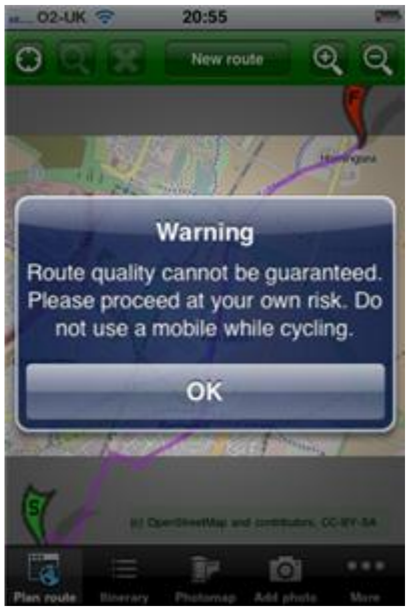
Immediately after setting the start point (whether via touch, namefinder or geolocation), a modal dialog confirms this. The same happens with the finish point also:



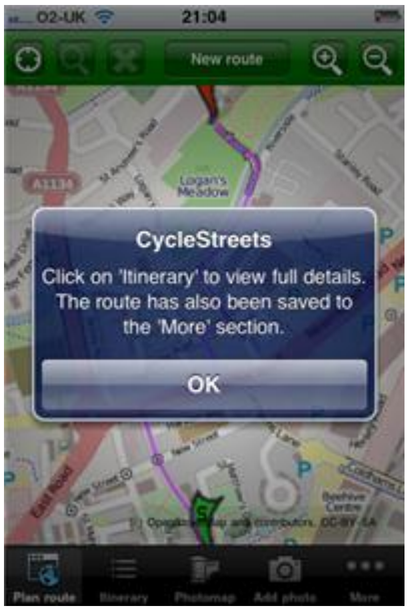
CycleStreets
Start point (S) set.

CycleStreets
Finish point (F) set.

After successfully planning their first route, the user receives two modal dialogs, one after the other, immediately following the point the route is drawn on the map:


Warning

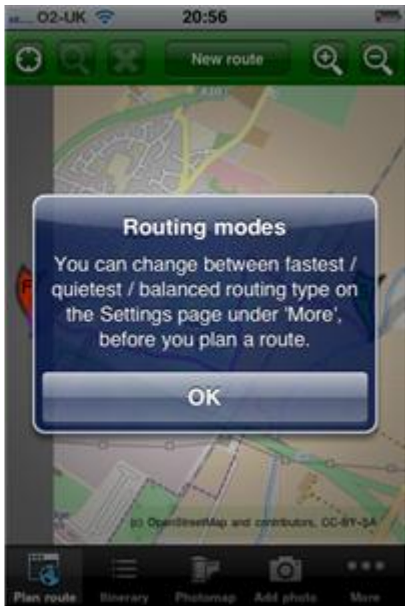
Route quality cannot be guaranteed. Please proceed at your own risk. Do not use a mobile while cycling.


CycleStreets

Click on 'Itinerary' to view full details. The route has also been saved to the 'More' section.

[NB These screens were taken from two different screenshotting attempts; the map underneath should naturally be the same.]

There is also a modal dialog box that appears upon the successful planning of the user's second journey, again, immediately after the point the route is drawn on the map.


Routing modes

You can change between fastest / quietest / balanced routing type on the Settings page under 'More', before you plan a route.

There are no other dialogs.

These six screens will thus only ever be seen once by the user, though a future release of the app could have an option to reset all help dialogs, which could be useful when needing to demonstrate the app to friends, etc.

7. Journey planner: discussion of future changes to core usage

7.1. Three-taps interface concept

The three-taps interface is novel and therefore non-standard, though when the user is used to it is extremely efficient. The initial run dialogs are an attempt to get round this but indicate that the system is still not as self-explanatory as it should be. The step-back system is not entirely self-explanatory. There is also evidence in the Guardian review that the commentary panel has been construed as a button rather than a piece of advice to the user. A final problem is that the order of setting has to be start-finish-plan; changing the start point requires undoing the finish also, and the finish cannot be set before the start.

For all these reasons, the three-taps interface needs reconsideration or refinement.

However, the alternative, a drop-pin approach, is often awkward given the need to set both a start and finish point. One other routing application does not enable the user to set both and the pin setting order feels non-intuitive. The Google Maps app shipped with the device requires the user to touch down for a full second before a pin is dropped, which is also non-intuitive. In other words, all current interfaces seem to have deficiencies in the way they set two points.

One possible solution is to retain the three-taps system but have two boxes in the top bar that represent the individual states of the start/finish markers, with the focus of these two boxes somehow making clear that the user is now changing the currently selected box (i.e. the start or the finish). This would enable the commentary panel to removed, along with all the guide-through dialogs. Some wire-framing and usability testing is needed before changing this UI.

An additional issue is that the current interface is totally unsuitable for setting waypoints.

7.2. Switching between map strategies (fastest/quietest/balanced/shortest)

Four map strategies (fastest/quietest/balanced/shortest) are supported by the API, with balanced being the default in the app (as per the website). Only the first three are supported in the app, and ideally the shortest mode should be added in as an option, even though it is in practice of little interest to users as it ignores hills.

At present the app takes the strategy chosen in the settings. The user cannot currently switch between different strategies easily to compare them, which is a strongly-desired user requirement. The app needs to implement this, which may have the effect of considerable changes to the top bar UI area.

7.3. Itinerary

It is arguably of limited use, based on only minimal feedback and needs rethinking for a future release (following user surveying). A satnav view is the longer-term objective for use of this menu entry, basically just taking the main screen view and rotating the map tile background to the angle of the phone, and the user's current location being followed by moving the map panel, keeping the direction of the map always forward to the user.

8. Photomap

The Photomap enables users to view photos of cycling-related infrastructure and to upload their own photos, which will be automatically geolocated and given captions and categorisation.

8.1. Viewing/browsing mode

The browsing mode is the third of the five main menu entries.

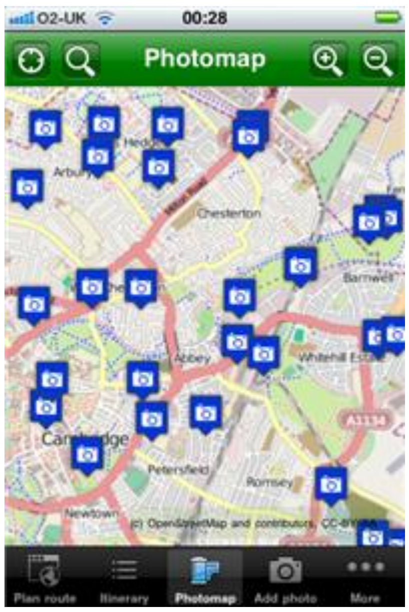
The map is centred on the user's location by default. The user can pan/zoom the map as usual.

The journey planner and Photomap sides save their current locations (for use when returning to them having gone to a different screen) independently.

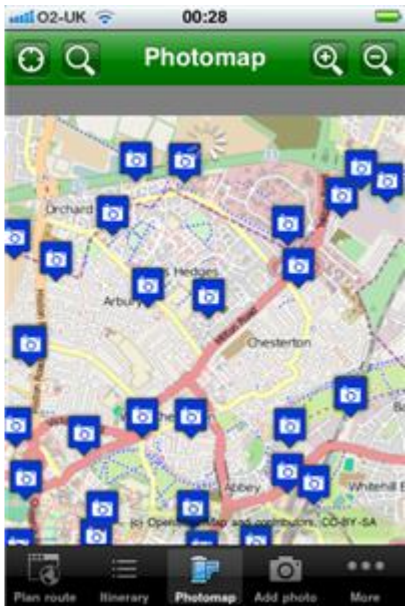
Immediately after a map move, an API call to /api/photos.xml is made, e.g.:

```
GET
http://www.cyclestreets.net/api/photos.xml?key=<key>&longitude=0.141997&latitude=52.20517
4&n=52.224478&e=0.169463&w=0.114531&s=52.185870&zoom=13&useDom=1&thumbnailsize=300&limit=
25&suppressplaceholders=1&minimaldata=1
```

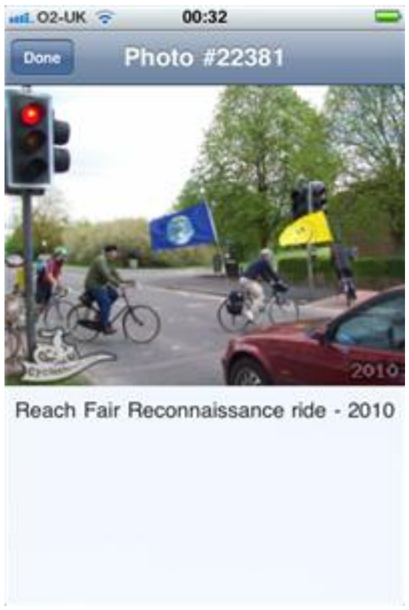
which specifies the current bounding box/zoom and sets a limit of 25 photos to be sent, with the fields in the XML minimized to avoid unnecessarily larger downloads. It also suppresses placeholders (points in the CycleStreets Photomap which as yet have no photo but have a caption).



As the map is moved, the 'loading' UI icon is shown near the top of the screen as an overlay:

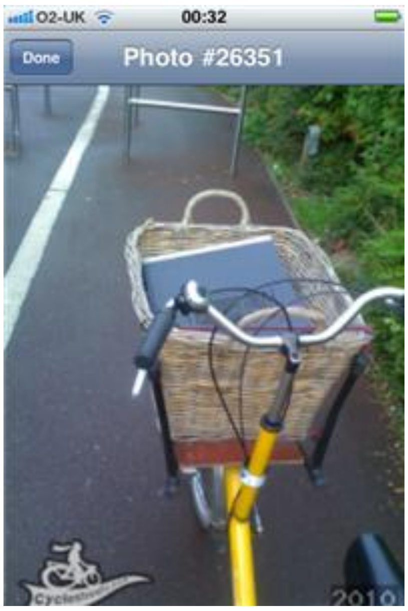


Touching on a marker results in the photo being loaded (which is specified in the XML returned) and the caption. The photo number is shown at the top. The user can return to the browsing screen by clicking on 'Done':



Reach Fair Reconnaissance ride - 2010

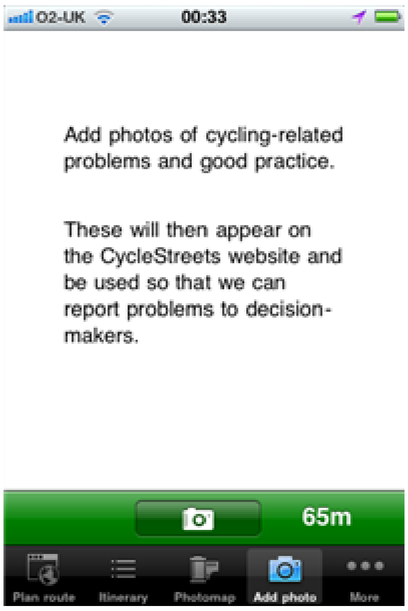
Currently a bug results in vertical-alignment photos not been scaled down, such that the caption is then scrolled off beyond the visible part of the screen:



8.2. Adding a photo

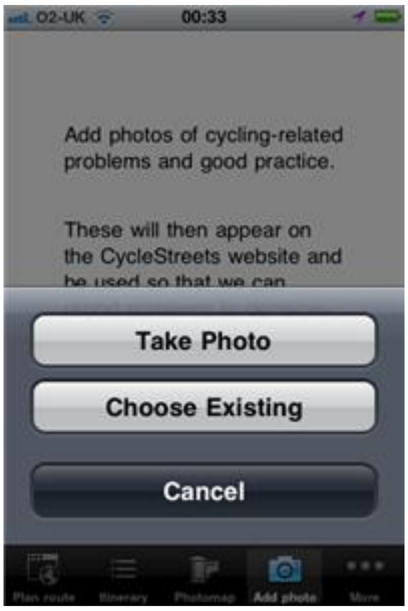
This is done using the fourth main menu entry. By way of an overview, this functionality requires selection of a photo (or taking one live), setting of a caption (optional), setting of a category (mandatory) and confirming the upload. The photo is then sent to the server. The server responds with XML indicating whether the upload failed or was successful, with the latter sending the photo number.

The default screen, as loaded from the fourth menu button is shown as follows. The current geolocation resolution is shown, though this should be removed (or given with an explanation) in a future release. Pressing the camera button brings up the standard UI dialog giving options to Take Photo or Choose Existing.

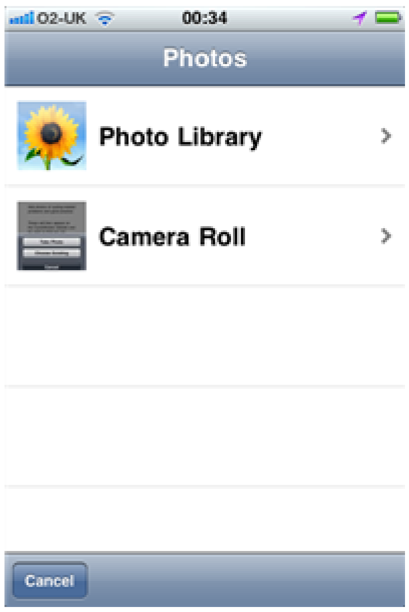


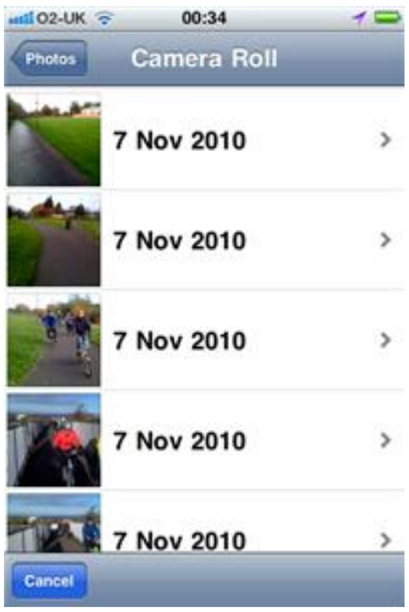
Add photos of cycling-related problems and good practice.

These will then appear on the CycleStreets website and be used so that we can report problems to decision-makers.

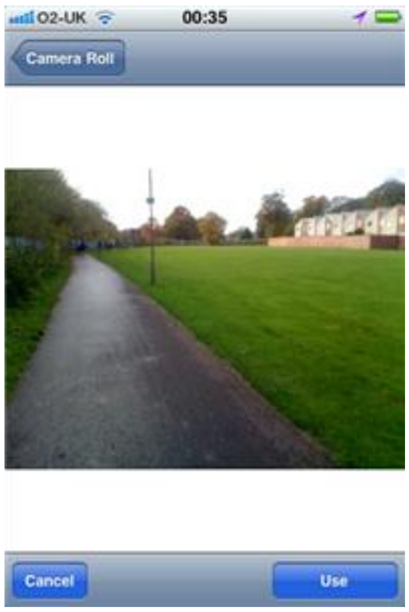


Choose Existing brings up a photo picker. At present the native UI picker is not used as this does not return the EXIF geolocation values within the photo handle. (However, it is believed that iOS 4.1 now supplies these so this may be implemented in a future release so that the picker library in use can be deprecated.) Note that at the local photo listing stage there also exists a Cancel button:

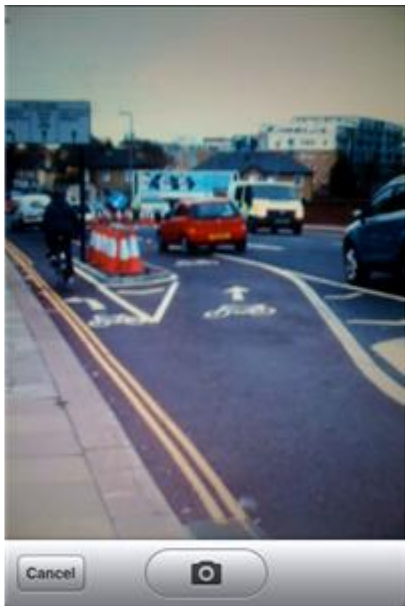




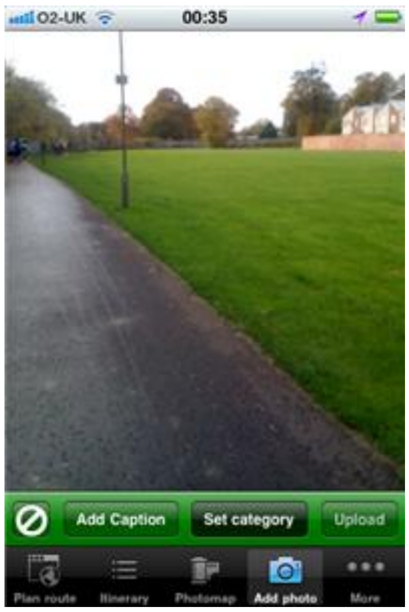
Selecting a photo shows a preview of it, with buttons to return to the Camera Roll and to Use the picture. There is currently also a 'Cancel' button but a future release will see this removed, as it is too easy to mistake with the Camera Roll (back) button.



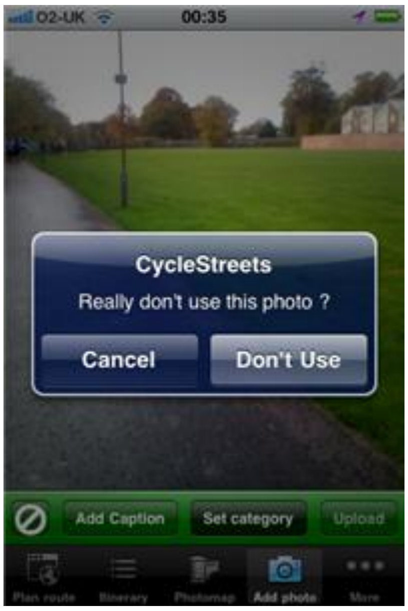
If the user instead chose to Take Photo, the standard Camera dialog is brought up. This should result in a photo taken at the standard resolution of the device, should include Geolocation EXIF tags, and should store the photo in the main camera roll (even if the CycleStreets app doesn't end up using it, i.e. the user backs out of the uploading sequence).



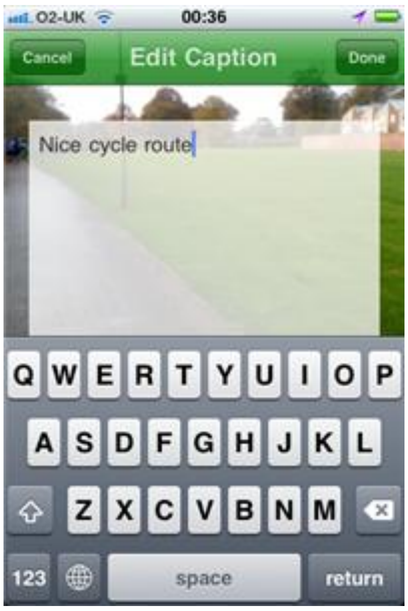
Having now got a photo, a screen is shown with the photo in place, and four buttons: a Cancellation icon, Add caption, Set category, Upload. Category is a mandatory field and thus appears visually more strong. Upload is not yet clickable as the category has not been set.



Using cancel returns to the default “Add photos of …” screen, though a modal dialog is used to prevent accidental clicks of the cancel button.

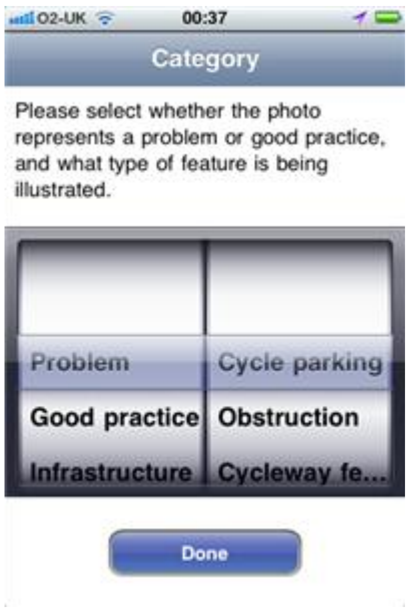


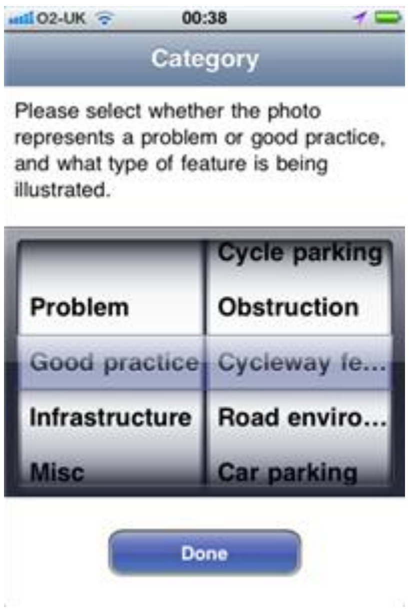
Assuming the user has not cancelled the photo selection, they can optionally then add a caption. When done, 'Done' returns to the previous screen.



At some point they must also select a category. This brings up a dual selection control using the standard wheel UI and contains explanatory text. The categories are fetched using the `photomapcategories.xml` API call. The response contains the meta-categories (left on screen) and categories (right on screen) plus a cache-until time in UNIXtime for this data.

```
http://www.cyclestreets.net/api/photomapcategories.xml?key=yourapikey
```

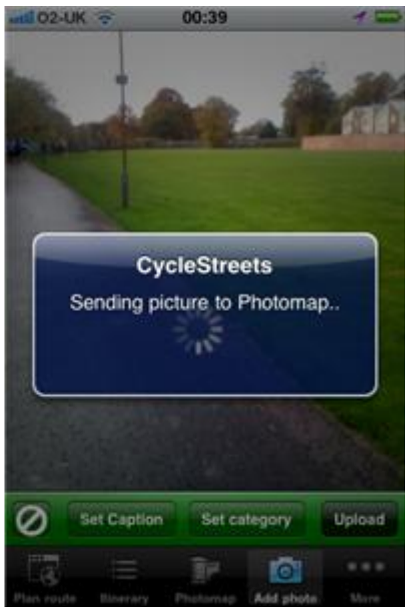

Please select whether the photo represents a problem or good practice, and what type of feature is being illustrated.


Please select whether the photo represents a problem or good practice, and what type of feature is being illustrated.

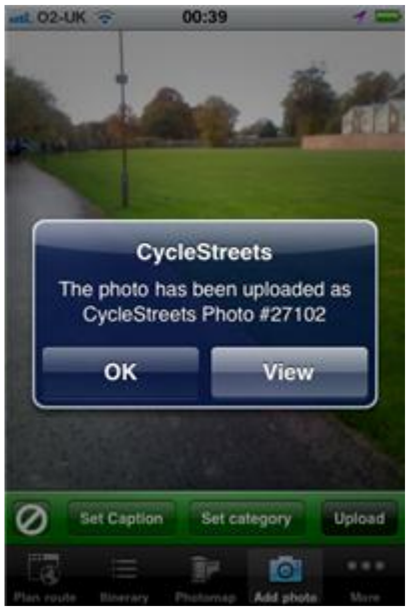
By default, Problem and Cycle parking are selected (i.e. the default is not a blank option). The user can change these. When selected, Done must be clicked.

The 'Upload' button now becomes available by being highlighted in the stronger style; 'Set category' now reverts to ordinary (lighter green) style. These changes attempt to make clear to the user that they should upload the photo now, as everything has been set correctly.

On depressing Upload, if the user is not signed in, they must do so first (see below). If they are signed in, the image is sent, with a modal dialog and a spinner:


CycleStreets
Sending picture to Photomap..

The XML returned is then used to determine the Photo number on success, or the failure message in the XML is piped to the user:



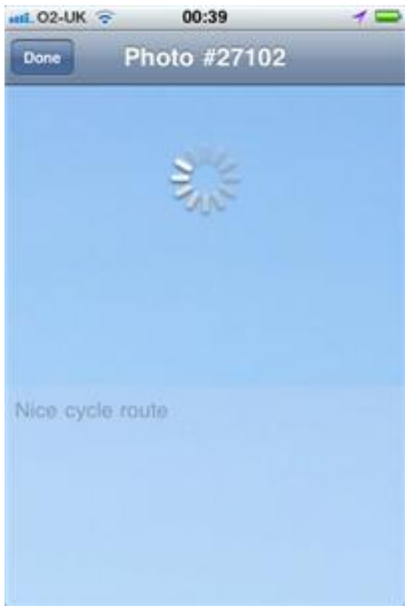
CycleStreets

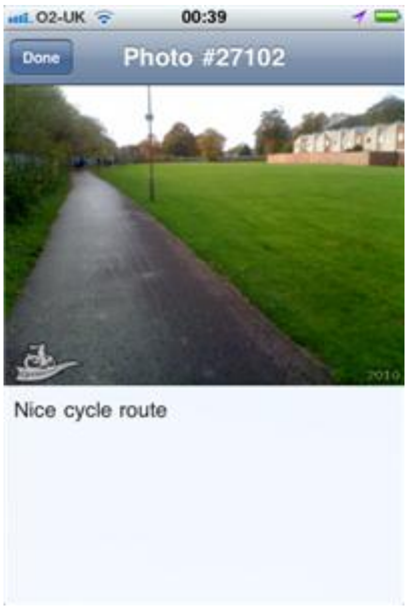
The photo has been uploaded as CycleStreets Photo #27102

OK View

[Failure screenshot needed]

Clicking on OK resets the screen to return to the default "Add photos of …" screen. Pressing View loads the Photo via the API (though this could be done using the photo in memory if preferred); again the Photo number must be displayed at the top:



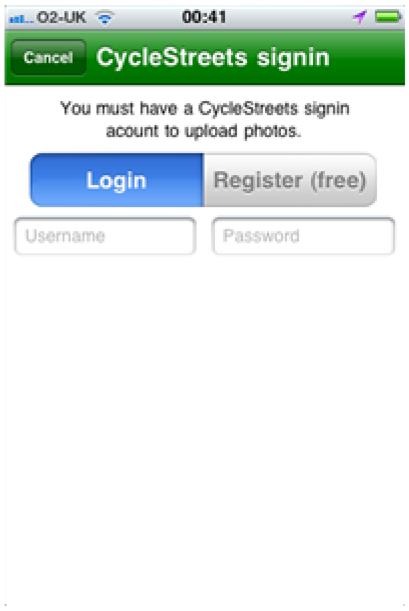


8.3. Signin process

Photos require a username to be associated with them. This requires a Signin account.

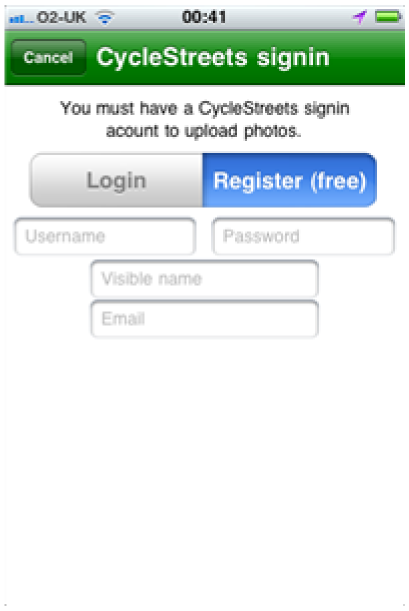
Note that the terminology used should always be 'Signin' and never 'Sign in' or 'Log in' or any variant. Signin should be capitalised at all times as it is regarded as a name.

This screen appears immediately after clicking 'Upload' during the user's first attempt to try adding a photo:

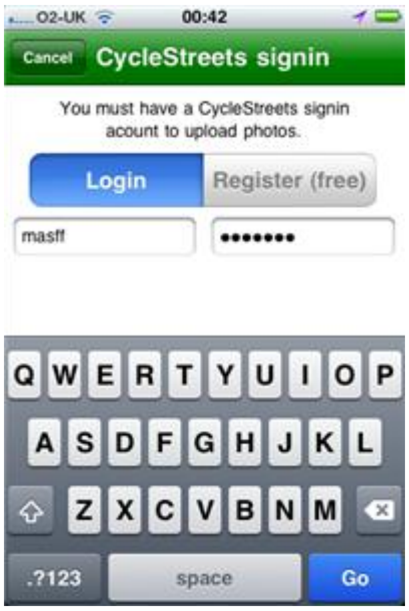


You must have a CycleStreets signin acount to upload photos.

The user can switch between two tabs: Login (which should read ‘Signin’; the current label is a bug) and ‘Register (free)’. In both cases a Username and Password must be supplied. Register mode has two extra boxes:



You must have a CycleStreets signin acount to upload photos.



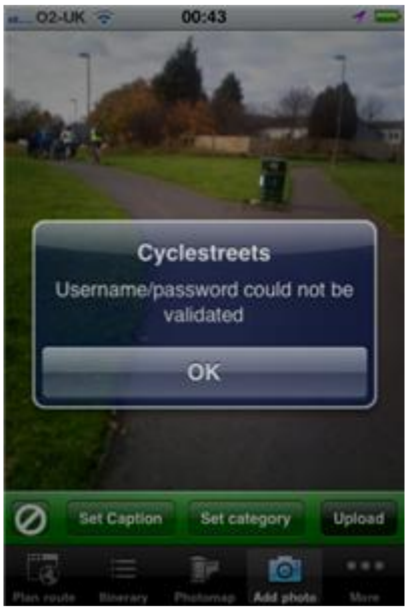
You must have a CycleStreets signin acount to upload photos.

Having entered an existing Username and Password, and clicking Go, these credentials are sent with the addphoto.xml call, which (note) is sent over HTTPS since it contains personal data.

```
API call
```

If correct, the credentials are then stored in the app’s local data store, which means that the Signin/Register page does not get shown in future upload attempts.

If the credentials are wrong, the addphoto.xml call will give a failure message, and a warning which should be passed to the user as a modal dialog box. (Note that there is currently a bug which results in the order being (a) return to the photo screen, (b) show error dialog; this should instead be (b) with dismissal button then (a).)



Cyclestreets

Username/password could not be validated

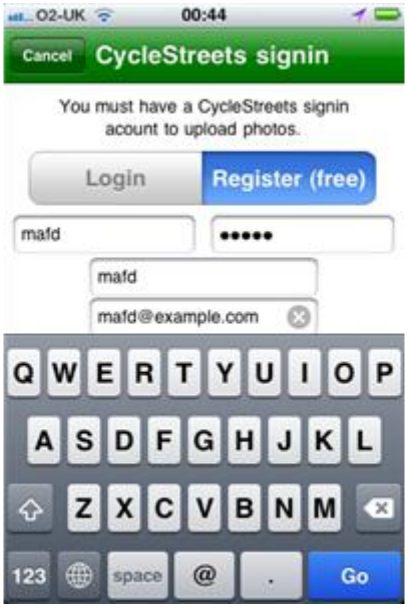
OK

Upon dismissal of the dialog this would return the user to the Upload screen, at which point the user can try again; the Signin/Register page is thus visible again as the credentials were not correct.

8.4. Signin – new user registration

Users may not already have a Signin account, and the app enables them to do this immediately.

The second tab in the Login/Register screen, Register (free) is the same as Login but contains additional fields:



CycleStreets signin

You must have a CycleStreets signin acount to upload photos.

mafd

•••••

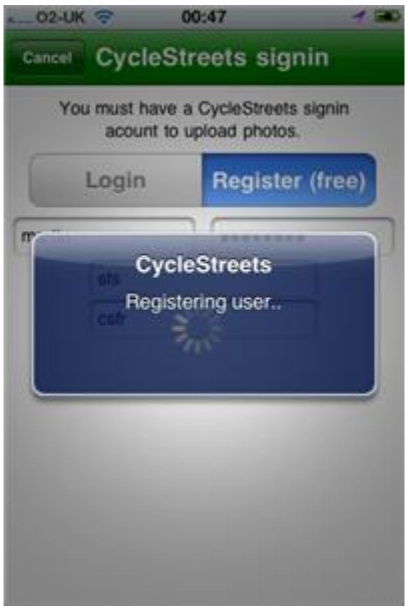
mafd

mafd@example.com

When all four are filled in, clicking on Go means to register the account then proceed with the upload.

Basic validation (e.g. e-mail syntax) should be performed on all fields before Go becomes available.

For registration, a call to usercreate.xml is made, with a temporary dialog showing that the registration is in process:

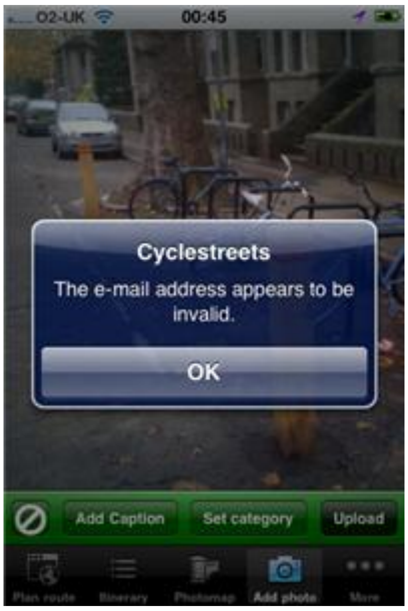


You must have a CycleStreets signin acount to upload photos.

CycleStreets

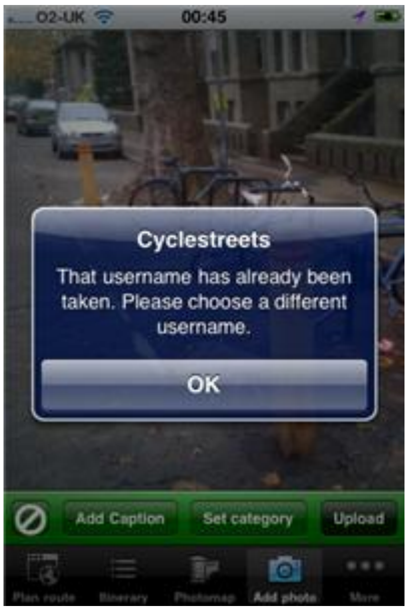
Registering user..

If an error occurred during registration, this must be piped from the API XML to the user. (Note that there is currently a bug which results in the order being (a) return to the photo screen, (b) show error dialog; this should instead be (b) with dismissal button then (a).)



Cyclestreets

The e-mail address appears to be invalid.

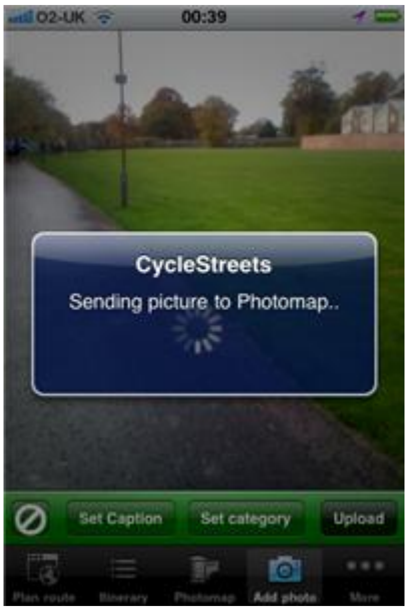


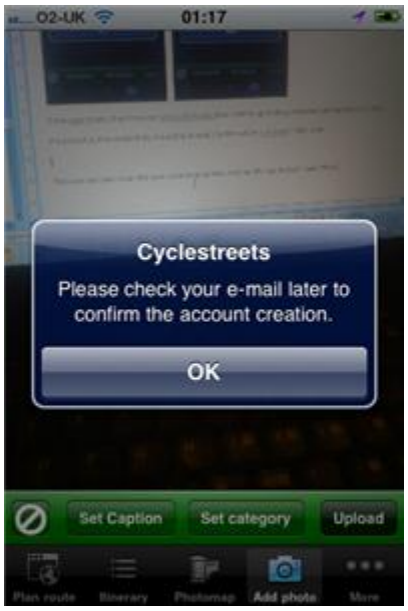
Cyclestreets

That username has already been taken. Please choose a different username.

If the signin fails, then the user should be taken back to the uploading screen enabling them to retry.

If successful, the credentials should be stored. Confirmation is shown to the user:



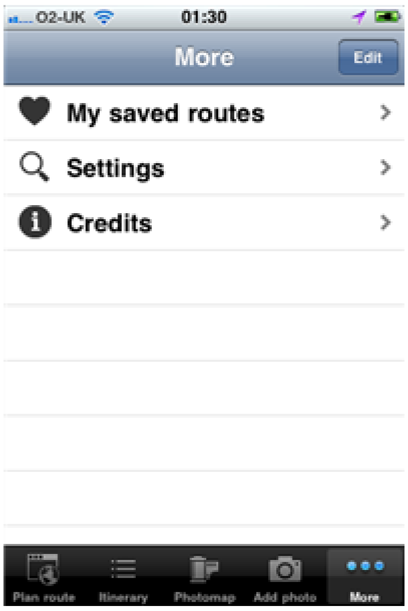


[NB These screens were taken from different screenshotting attempts; the photo underneath should naturally be the same.]

The user can later clear the now-saved credentials via the settings section (see below).

9. More (settings etc.)

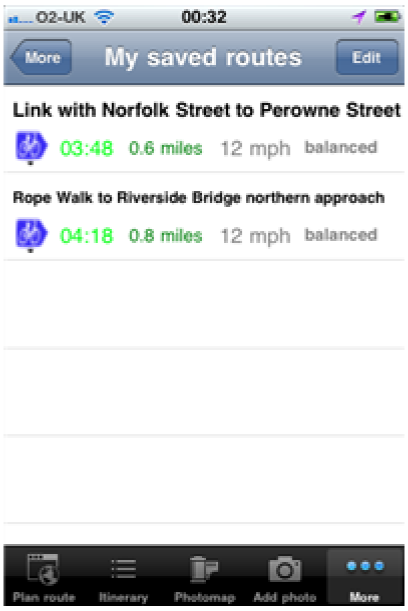
'More' is the label used for the fifth and less menu item. There are currently three facilities within this section:



9.1. My saved routes

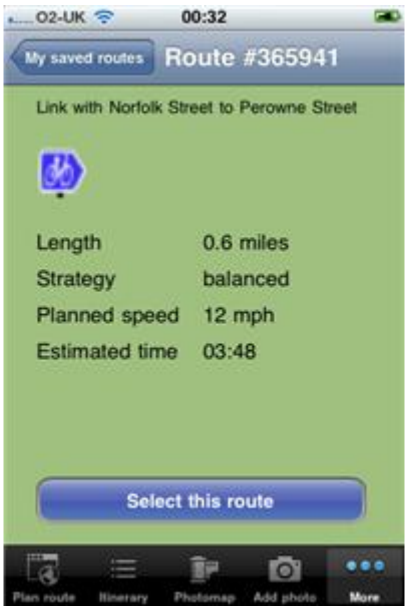
As noted above, successfully-planned routes are saved here. This area shows a list of the routes the user has planned via the app. (At present this is not tied into the signin system, which would enable fetching of routes planned on the website also. This should be implemented at a later date, and requires API support.)

Clicking on an entry brings up the summary of its key details, and a button 'Select this route':



Link with Norfolk Street to Perowne Street
03:48 0.6 miles 12 mph balanced

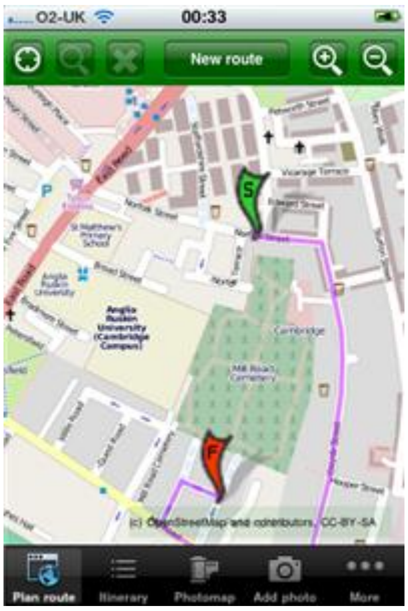
Rope Walk to Riverside Bridge northern approach
04:18 0.8 miles 12 mph balanced

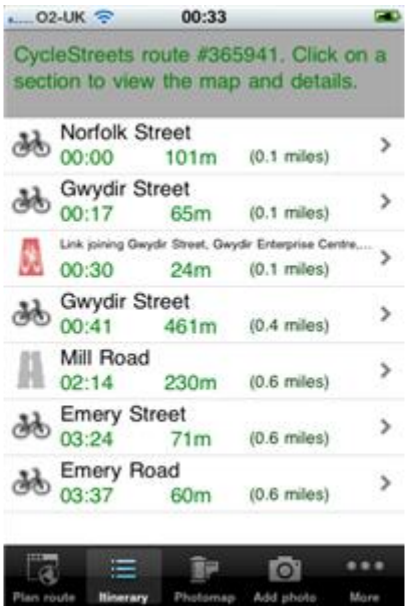


Link with Norfolk Street to Perowne Street

Length	0.6 miles
Strategy	balanced
Planned speed	12 mph
Estimated time	03:48

Selecting the route wipes what (if anything) is currently in Plan route and Itinerary (which will have been saved into My saved route) and loads the recalled journey's route and itinerary into those panels.


O2-UK
00:33
New route
Anglia Ruskin University (Cambridge Campus)
Mill Road Cemetery
(c) OpenStreetMap and contributors, CC-BY-SA
Plan route
Itinerary
Photomap
Add photo
More

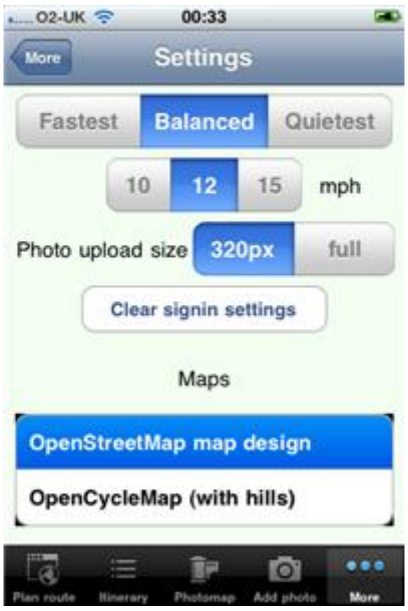


CycleStreets route #365941. Click on a section to view the map and details.

Norfolk Street			
00:00	101m	(0.1 miles)	>
Gwydir Street			
00:17	65m	(0.1 miles)	>
Link joining Gwydir Street, Gwydir Enterprise Centre,...			
00:30	24m	(0.1 miles)	>
Gwydir Street			
00:41	461m	(0.4 miles)	>
Mill Road			
02:14	230m	(0.6 miles)	>
Emery Street			
03:24	71m	(0.6 miles)	>
Emery Road			
03:37	60m	(0.6 miles)	>

9.2. Settings

A range of app settings can be changed:



Fastest/Balanced/Quietest should be joined by a fourth option, Shortest in a future release. This setting, and the speed (10/12/15mph) are used in the journey.xml call.

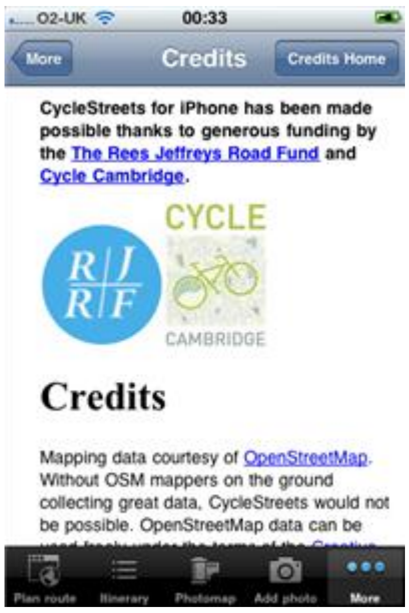
The Photo upload size determines what size is sent during the upload. It does not affect the actual taking of the photo.

The 'Clear signin settings' (which should read 'Clear Signin settings') clears any stored Signin credentials, meaning that the next 'Add photo' use will trigger the Signin/Register dialog sequence. Using the clearance button should result in a modal prompt 'Are you sure?' to prevent accidental deletion, though this is currently not implemented.

Lastly the map cartography design, i.e. the tileserver setting can be set here. This immediately affects all places where tiles are loaded, i.e. all of Plan route, Itinerary and Photomap. Map tiles are discussed below.

9.3. Credits

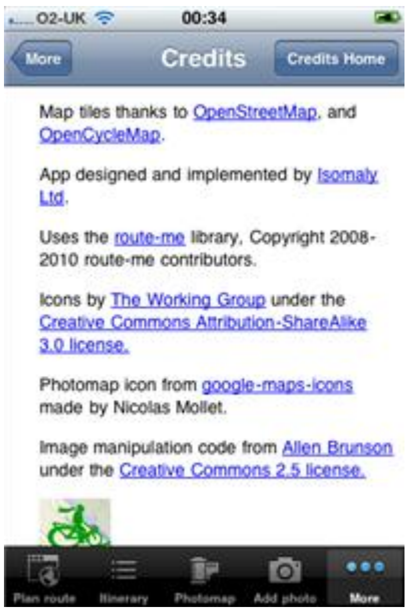
This is a simple HTML page as credits.html in the repository. At present, following the links results in a lack of scroll bars. This bug should be addressed in a future version.



CycleStreets for iPhone has been made possible thanks to generous funding by the The Rees Jeffreys Road Fund and Cycle Cambridge.

Credits

Mapping data courtesy of OpenStreetMap. Without OSM mappers on the ground collecting great data, CycleStreets would not be possible. OpenStreetMap data can be



Map tiles thanks to OpenStreetMap, and OpenCycleMap.

App designed and implemented by Isomaly Ltd.

Uses the route-me library, Copyright 2008-2010 route-me contributors.

Icons by The Working Group under the Creative Commons Attribution-ShareAlike 3.0 license.

Photomap icon from google-maps-icons made by Nicolas Mollet.

Image manipulation code from Allen Brunson under the Creative Commons 2.5 license.

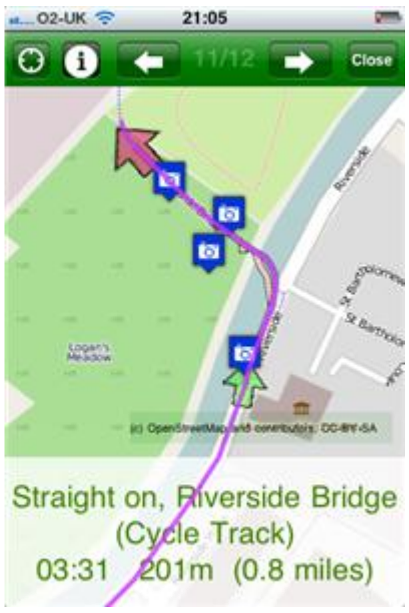
10. Map tiles and attribution

10.1. Attribution

It is essential to ensure the map cartography is attributed wherever it appears. It represents the hard work of thousands of people collecting data as well as work from cartographers and system administrators; as such attribution is both a legal requirement and an ethical obligation.

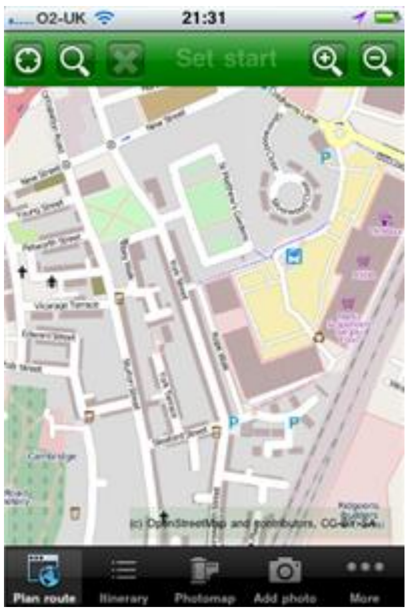
The attribution always appears above (in a z-index sense) above the map tiles and has a semi-opaque background with background fern (same green as elsewhere) and 10% opacity.

There should be a padding of a few pixels between the attribution text and its background. (At present there is a bug such that the left side does not have any padding.)



(c) OpenStreetMap and contributors, CC-BY-SA

Straight on, Riverside Bridge (Cycle Track)
03:31 201m (0.8 miles)



(c) OpenStreetMap and contributors, CC-BY-SA

10.2. Styles and attribution texts

Two map tile styles are currently supported.

OpenStreetMap website style: This the default and is called “OpenStreetMap map design” in the settings and has the attribution string “(c) OpenStreetMap and contributors, CC-BY-SA”. Tiles are served from http://{a|b|c}.tile.openstreetmap.org . At present usage of the app is low enough for this probably to be acceptable usage, but this situation should be monitored.

OpenCycleMap: This is called “OpenCycleMap (with hills)” in the settings and has the attribution string “(c) OpenStreetMap and contributors, CC-BY-SA;\nMap images (c) OpenCycleMap” (where the \n here is actually a newline). Tiles are served from http://{a|b|c}.tile.openstreetmap.org/tile/ .

We would like to use the RideTheCity III style available from Cloudmade but Cloudmade require payment for mobile use so this is not possible.

10.3. Map tile cache

Map tile caching avoids the user loosing the background unnecessarily as they pan/zoom or plan/view routes.

On iPhone, tile caching is implemented natively within the Route-Me library.

11. Restore after user pause or suspend

The app should aim to restore the user’s state as best as possible, particularly in the journey planning screens, where the user is likely to put the phone away and expect it to restore exactly as left, after unlocking if necessary.

The latest screen should be returned after an explicit return to the phone home screen. The latest planned/recalled journey should be loaded in to the first two menu entries (journey planner and itinerary screens).