Testing

All tests involving routes/general navigation use the school layout as an example layout, which has 295 nodes (visible and invisible). The routes are selected as follows:

* Using a random number generator, a number between 1 and 295 is selected.
* If this number does not correspond to a visible node, it is discarded.
* This is repeated until 20 such visible nodes are selected.
* These form 10 pairs of origin/destination test nodes (suitably representative of typical path queries).

The resulting pairs are as follows:

|  |  |  |
| --- | --- | --- |
| **Pair #** | **Origin node** | **Destination node** |
| 1 | Science Café | B4 |
| 2 | Photocopying room | Art 3D store |
| 3 | Surmaster’s office | B5 |
| 4 | 17 | 302 |
| 5 | P6 | Computing Laboratory |
| 6 | P1 | C6 |
| 7 | 2 | 25 |
| 8 | 303A | P2 |
| 9 | Science Tech Area | Geography Resource Centre |
| 10 | C4 | B3 |

A ‘long’ pair was also formed, consisting of rooms 5 and E1 as origin and destination respectively, to act as an example of a relatively long-distance route. This pair will be referred to as ‘Long’.

1. Shortest Path Generation

**Overview**

Ensuring that the algorithm for determining shortest path (Dijkstra’s) is correctly implemented, that is for a given origin and destination the sequence of nodes generated by the app should be the same as if done manually.

**Test performed**

The test considers the route generated between each of the randomly selected node pairs as described above. The sequence of nodes in the shortest path of each pair are first calculated by hand (using Dijkstra’s algorithm), then input into the app (also using Dijkstra’s). If the implementation of the shortest path algorithm is correct, the two sequences should match.

**Purpose**

Ensure that the path returned by the app is indeed the shortest path between two nodes, thus minimising travel required between two nodes and maximising utility (also in order to prevent erroneous data from being generated that would not correctly function with the rest of the app).

**Results**

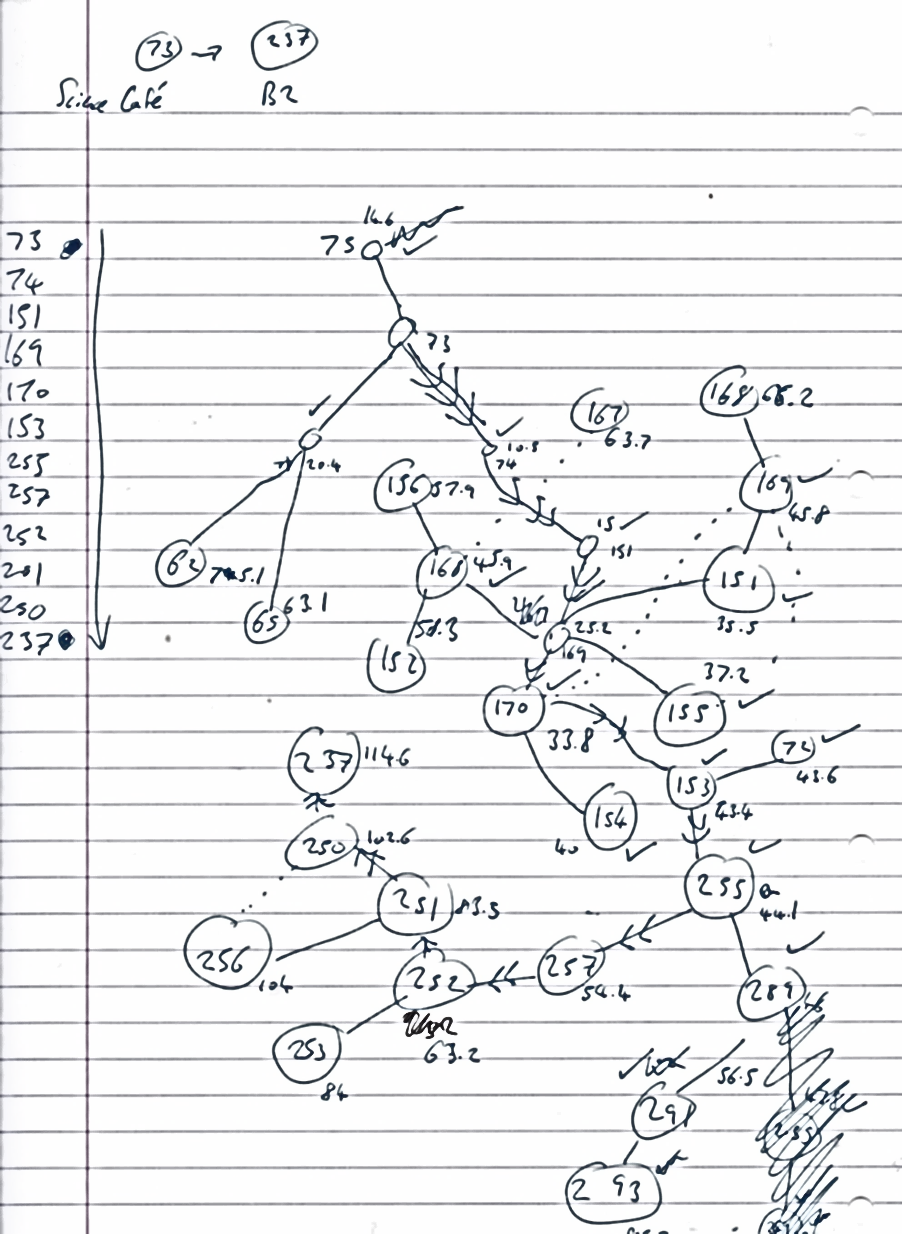
Test results are shown below. Nodes are written as index integers, as some nodes do not have string names on account of being invisible.

|  |  |  |
| --- | --- | --- |
| **Test Data (pair #)** | **Expected outcome (hand-done traversal)** | **Actual outcome** |
| 1 | 73, 74, 151, 169, 170, 153, 255, 257, 252, 201, 250, 237 | Matches |
| 2 | 76, 72, 65, 60, 34, 25, 42, 43, 44, 51, 112, 111, 205, 201, 203 | Matches |
| 3 | 110, 135, 90, 36, 62, 71, 73, 74, 151, 169, 168, 152, 253, 251, 238 | Matches |
| 4 | 105, 141, 136, 140, 145, 147, 85, 225, 261, 270, 263 | Matches |
| 5 | 155, 169, 151, 74, 73, 71, 62, 36, 52, 4, 40, 88, 136, 140, 145, 147, 126 | Matches |
| 6 | 159, 164, 165, 166, 167, 168, 152, 253, 251, 254, 288, 287, 278 | Matches |
| 7 | 18, 37, 91, 186, 230, 229, 191 | Matches |
| 8 | 266, 270, 261, 225, 226, 227, 228, 229, 230, 186, 91, 150, 164, 158 | Matches |
| 9 | 79, 65, 62, 36, 52, 54, 55, 56, 23 | Matches |
| 10 | 276, 285, 283, 292, 243, 249, 242 | Matches |
| Long | 15, 56, 59, 51, 49, 50, 45, 47, 6 | Matches |

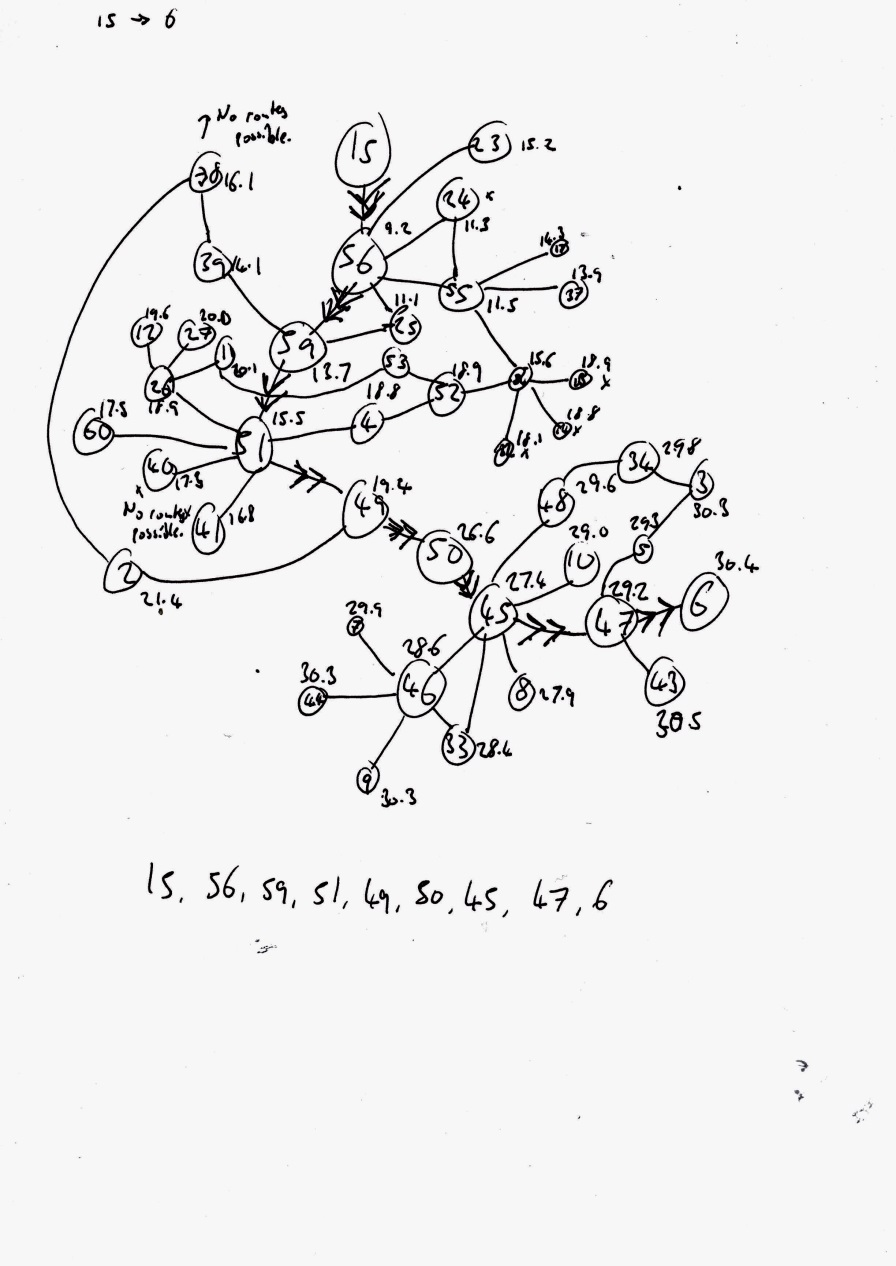
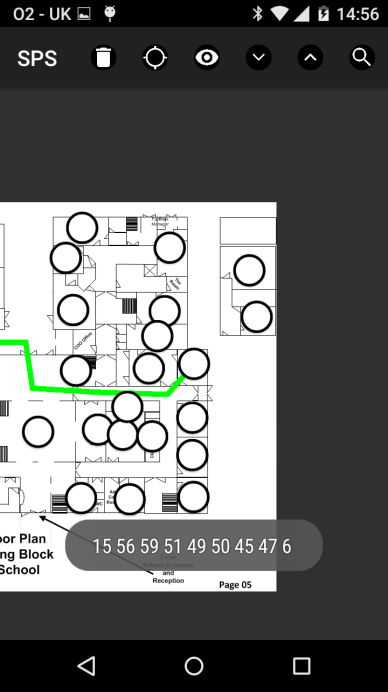
**Example evidence**

Below: Hand-done graph traversal, followed by smartphone app printing sequence of shortest path. Only examples from the first pair and the long pair are shown in order to limit volume of data. For the purposes of testing, the app outputs a list of nodes comprising the route taken.

First Pair

Second Pair

1. Layout Retrieval

**Overview**

Test the various methods of retrieving a map layout and saving, where success is defined as the layout being saved to storage.

**Test performed**

For each method of retrieval, the method is executed; if the layout is successfully added, the test is passed, and is otherwise failed.

**Purpose**

Ensure that the user is able to retrieve layouts using all (or as many of) the stated methods as possible, to facilitate distribution.

**Results**

Test results are shown below. In each case, the expected outcome is the layout being added to storage, i.e. the layout name (corresponding to the method) displaying in the map list of the home screen. Before each test, the app is reset to default such that no layouts are in storage, resulting in this display:



|  |  |  |
| --- | --- | --- |
| **Method** | **Outcome** | **Result** |
| In-app URL entry | H:\img\retr\Screenshot_2017-03-15-12-16-46.png H:\img\retr\Screenshot_2017-03-15-16-32-04.png  URL of .lay file host is entered to this dialog box.  Layout appears in map list, signifying that it has successfully downloaded and passed integrity checks. | Success |
| Hyperlink (outside of app) | H:\img\retr\Screenshot_2017-03-15-16-49-47.png H:\img\retr\Screenshot_2017-03-15-16-49-42.png  Hyperlink in another application is clicked, passing a URL intent to Android which (after querying user) is redirected to this app.  Layout is added to map list, implying successful download. | Success |
| QR code (outside of app) | H:\img\retr\Screenshot_2017-03-15-13-06-41.pngH:\img\retr\Screenshot_2017-03-15-16-35-32.png  QR code is scanned using a third-party app, prompting the user to navigate to the listed URL. From here, continues as in the case of a hyperlink.  Layout is successfully added to map list. | Success |
| File selection (outside of app) | H:\img\retr\Screenshot_2017-03-15-12-17-58.png H:\img\retr\Screenshot_2017-03-16-16-50-52.png  File is selected using a third-party application, passing a file intent filter to Android and subsequently this app (only app suited to opening .lay files).  Layout is successfully added to map list. | Success |

1. Calculation/Navigation time

**Overview**

Recording time taken for each critical element of the app (node search, route generation, overall navigation).

**Test performed**

The time taken for the app to complete each of the following processes is recorded, by starting and stopping a Stopwatch instance (part of the System.Diagnostics namespace) before/after the process returns its content. The time elapsed is printed to the screen.

**Purpose**

To determine whether the app is in a state appropriate for fast navigation to allow users to quickly access routes/nodes, as per spec point 1.

**Results**

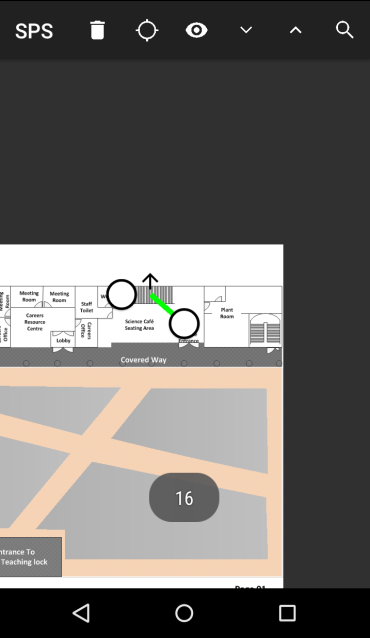
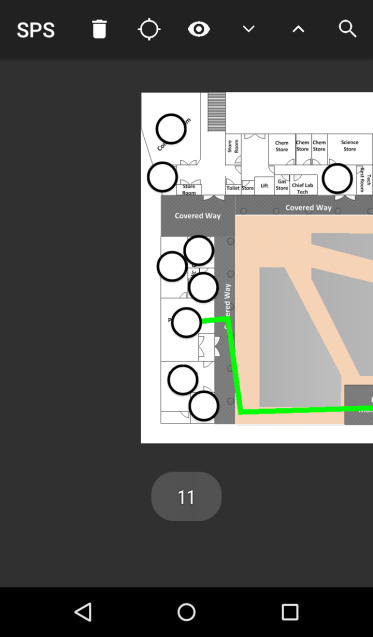
Key: s = seconds, ms = milliseconds.

* The overall navigation of the app took 13.55 seconds compared to the desired result of 90 seconds, leading to **success**.

|  |  |  |
| --- | --- | --- |
| **Test Data (pair #)** | **Expected results (upper bound of acceptable time for search/route generation)** | **Actual outcome** |
| 1 | 3s/5s | 1ms/16ms, success |
| 2 | 3s/5s | 1ms/11ms, success |
| 3 | 3s/5s | 2ms/6ms, success |
| 4 | 3s/5s | 1ms/11ms, success |
| 5 | 3s/5s | 1ms/3ms, success |
| 6 | 3s/5s | 1ms/5ms, success |
| 7 | 3s/5s | 1ms/2ms, success |
| 8 | 3s/5s | 1ms/5ms, success |
| 9 | 3s/5s | 1ms/3ms, success |
| 10 | 3s/5s | 1ms/15ms, success |
| Long | 3s/5s | 2ms/16ms, success |

**Example evidence**

Times elapsed for pairs #1 and #2 respectively. For the purposes of testing, the app displays the time taken for route generation.

1. User Interface Navigation

**Overview**

Attempting to navigate the user interface surrounding the routes generated between the pairs listed in the introduction.

**Test performed**

Attempt to use the app to navigate the layout after each of the routes listed above are displayed, emulating ‘following’ the route in the app. The test will succeed if the app functions as expected, and fail otherwise, for instance in the case that:

* + Floors transition incorrectly
  + The route is not visible
  + Arrows do not appear at the end of a route on one floor signifying whether the floor of the proceeding node is above or below the current one
  + Arrows do not appear at the beginning of a route on one floor signifying whether the floor of the preceding node was above or below the current one
  + Re-centring the view affects the orientation of the route.

**Purpose**

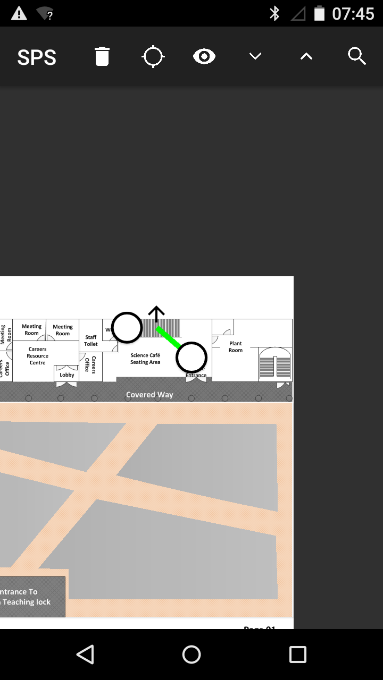
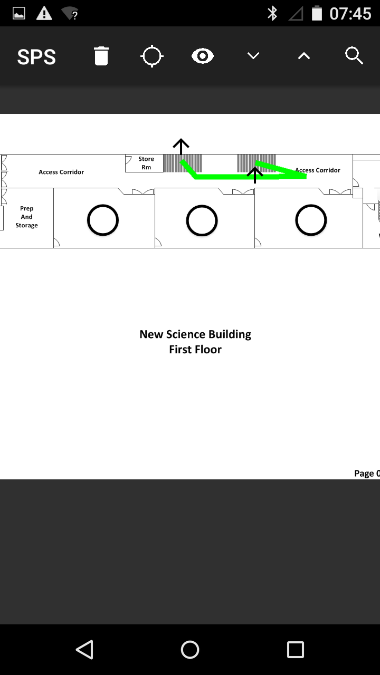
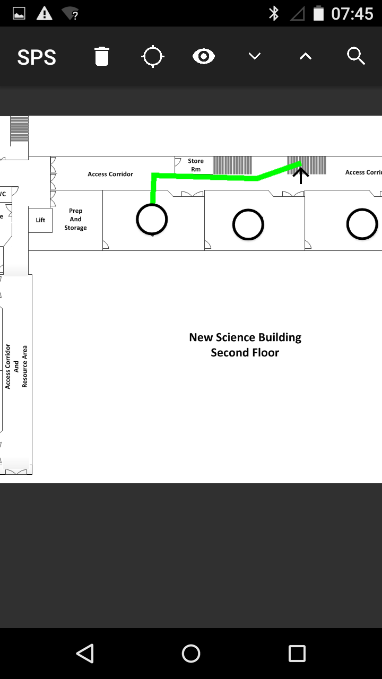
To demonstrate that the app user interface generally works as intended, and that the interface is clear and accessible on the user end as specified in spec point x.

**Results**

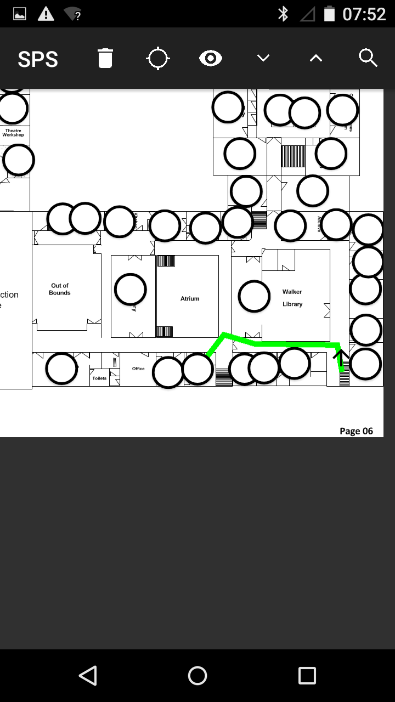
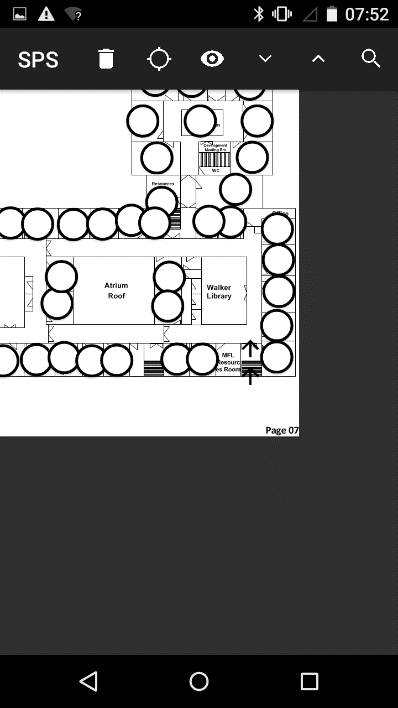
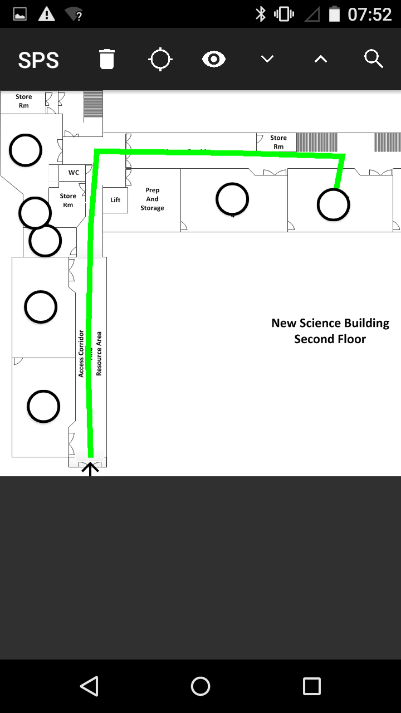
No problems were encountered in the navigation of the routes, leading to **success** (for details, see evidence).

**Example evidence**

Navigating route formed from pair #1.

Navigating route formed from pair #2.

**  **

1. Miscellaneous tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Overview** | **Test performed** | **Purpose** | **Results** |
| **1. Disk space consumption**: Observing amount of space occupied by app with a suitably large layout stored alongside. | Observe the Android system’s value for the amount of storage occupied by the app and the school layout (with additional images embedded in the file to simulate all rooms having an image). | Ensure that the storage consumption by the app is reasonable and that users can retrieve/observe layouts without issues of storage, as per spec point 12. | The storage occupied by the app and the school layout, as shown below, is 13.66MB, well under the expected upper limit of 50MB and thus **passing** the test.  C:\Users\Oliver\Downloads\img\Screenshot_2017-03-16-21-31-33.png |
| **2. Internet connection**: Observing correct function when device is not connected to the internet. | Carry out all of the above tests (excluding layout retrieval) again, without internet connection. | Make sure that the app is navigable without internet connection, as per spec point 13. | All tests passed with almost identical results to the cases of internet connection, leading to **success**. |
| **3. Simultaneous layout storage**: Test whether multiple layouts can be stored simultaneously and used without interference. | Retrieve 5 layouts and determine whether they function as expected (i.e. route generation provides routes that appear sensible, rooms returned belong to the layout specified). | Allow the user to be able to retrieve/view multiple layouts without having to continually download them, as in spec point 3. | The app behaved as expected while all layouts were stored simultaneously, yielding sensible routes and appropriate search results, leading to **success**.  H:\img\retr\Screenshot_2017-03-16-11-10-29.png  Demonstration of multiple layouts stored simultaneously. |

Evaluation

1. Feedback

I received feedback from the 4th Form Undermaster and 4th Form pupils, the end-users of my programs. Given that the pupils are likely to only use the smartphone navigation app and not the data entry program, their feedback was only of the smartphone app, whereas that of the 4th Form Undermaster was of both the smartphone app and the data entry program.

1.1 Desktop Data Entry Program

1. **4th Form Undermaster**

Obviously the task of creating a floor plan for an entire building is inevitably time consuming and overall daunting, but the building creation program works well to reduce the amount of time spent on it. After getting to grips with how to create floors and rooms (and the connections between them), the burden of layout creation is decreased such that the only process that takes particular effort is entering the details of each room, since everything else is so simplified. I also like that the application is well equipped to update existing layouts in terms of rooms and their details, which could be required often.

1.2 Smartphone Building Navigation App

1. **4th Form Undermaster**

The smartphone application works well; it provides clear routes between rooms, which are easy to find by searching and moving around the map. The layout on the app (as well as on the building creator application) is easy to navigate thanks to the movable background which can be zoomed. Also useful is that as well as finding out how to get from one room to another, the app also gives the option to hide all room icons and just observe the general layout of the school, giving an idea of the overall layout. Overall impressive, although in terms of improvements, some form of location tracking would be useful to follow the route given by the app, otherwise pupils can get lost. It would also be good to have the app give instructions of how it should be used, as the functions of some icons weren’t immediately apparent, although I understood how to use it after first use. I think that people’s locations being visible on the app\* may have privacy concerns and should at least be optional. I agree with the concepts of colour-coding and locations of nearest instances of particular facilities, such as nearest printers\*.

\*See ‘4th Form pupils’ response below.

**2. 4th Form pupils** (amalgamated into a single response)

The app works well, but some things could be added to make it better: The ‘search by teacher’ feature is very useful, but for a similar idea it would also be good to search for the nearest water fountain, bathroom etc. since searching for ‘Bathroom’ in the search gives a lot of results. You could also have the colour of the room on the map match up with its purpose, for example rooms in different departments can be coloured differently. Showing you where you are in the school would also be very useful. Having said this it works really well as it is; the idea of having it be generic and not just for St. Paul’s is also good, and I like how easy it is to download the map through just scanning a QR code and instantly downloading it. The ability to locate other people using the app through the layout would also be good.

1.3 Discussion

The end users appeared to approve of the execution of the project, with for the layout navigation. Praise was drawn to the ease of altering pre-existing layouts, as well as ease of navigation, and was overall received well, matching the satisfaction of all of the specification points of the end users’ issue solution. One point of criticism was the lack of positioning in the school; as discussed in Analysis, implementation of this would have been ineffective, potentially costly for the school and at the very least difficult to implement, although if more time had been invested this could have been an avenue to experiment with in implementation. Another criticism was with in-app instruction; for the data entry application the Undermaster stated that she was able to use the program easily ‘after getting to grips’ with it, implying initial confusion, and similar sentiments were echoed for the smartphone app. Instructions that appear during first-time use would be another possibility for implementation if tried again. Further suggestions for the app’s functionality from the feedback are evaluated below. Other than feedback, I feel that the additional functionality implemented compared to Google’s Indoor Maps solution render it overall successful; although I acknowledge that positioning would impart the app with considerably greater functionality, I feel that the current execution is successful in efficiently addressing the issue posed by my end users.

1. Specification Achievement

|  |  |
| --- | --- |
| Spec. point | Met? |
| 1. Users will be able to download a building layout specific to a particular building, complete with details about each room and locations thereof, submitted by someone else through a submission platform. Users will be able to do this for multiple buildings/building layouts; for the sake of testing, at least five layouts should be able to be downloaded and stored without errors/issues arising. | **Yes**. Building layouts can be downloaded, allowing searching of rooms, display of room details, and generation/display of routes between rooms. From testing it can be seen that at least five map layouts can be saved on the device at the same time (see test 5.3). |
| 2. The system will be developed for the Android (smartphone) platform. | **Yes**, the app runs on Android (4.1 and above), through Xamarin for C#. |
| 3. A desktop data entry program should be available for creation of such layouts. For multiple floors of a building, users should be able to create room ‘nodes’ (imposed over a background image) and specify a room’s title, description, and associations, as well as connections with other nodes and room image. | **Yes**, this is all achieved in the desktop application. |
| 3. The layout in question to be retrieved and downloaded will either be specified by an ID entered directly into the app, or by scanning a QR code (using a third-party application) to specify which layout to download. | **Yes**. Layouts can be downloaded by entering an http(s) link to a .lay (layout) in the app, or by scanning a QR code that expresses that http(s) hyperlink as shown in test 2. Additional to specification, users can also download layouts by clicking hyperlinks (leading to such files) outside of the app (e.g. in a web browser), or by opening a .lay file on the system outside of the app. |
| 4. The desktop data entry program should display one floor layout at a time (which can be changed to other floors), and should allow users to drag the view around and zoom in, as well as allowing the user to add/remove floors/rooms. | **Yes**: floors are displayed individually, with the dragging/zooming view working as intended and the floor/node creation/removal working as intended. |
| 5. The layout in question to be retrieved and downloaded will either be specified by an ID entered directly into the app, or by scanning a QR code (using a third-party application) to specify which layout to download. | **Yes**: layouts can be retrieved by URL entry in-app and QR code scanning, as well as hyperlink/file selection outside of the app. |
| 6. A search function will be provided for users to search for rooms using the room name as well as labels associated with the room (defined during entry of the layout). Labels will consist of terms that better define the room (for testing, samples of such terms would be whether the room has a printer, the room’s current occupant, and societies associated) and aid as additional search terms. Performing the search should display room properties (e.g. description, image) as well as location. | **Yes**, rooms can be searched according to title, description and associated tags. The app view moves to the location of the room, and displays respective information. |
| 7. When found, rooms will be displayed with descriptions, associated labels, notes pertaining to the room, and an image illustrating what the room looks like from the surrounding area. The location of the room will also be illustrated by a ‘node’ being superimposed onto an image of the layout of the building, where the node lies on its respective location in the layout. | **Yes**, rooms can be tapped on the app to display name, description, associated tags **and associated image**. |
| 8. The process of the system taking the user’s search query and performing a search should be carried out in less than three seconds. | **Yes**, returning a list of rooms matching criteria supplied is **seemingly instantaneous** (<1ms) from a human perspective. |
| 9. A function will be provided for users illustrating the path occupying the least distance between two nodes (rooms) that the user has entered. This will be represented as a coloured line passing through other nodes (rooms) (some potentially hidden, serving only as navigational midpoints), as well as arrows below/beneath nodes signifying whether the user should be ascending/descending floors at that node. | **Yes**, a coloured path is displayed showing the path of least distance between two nodes. From testing it appears that these are indeed the paths of least resistance (see test 1). Arrows are shown illustrating change of floor correctly. |
| 10. The process of the system taking the user’s route query and generating/returning the optimal path should be carried out in less than five seconds. | **Yes**; from testing, time taken between entering origin/destination rooms and displaying the path of least distance took an average of **8.5 milliseconds** with standard deviation of 5.48 milliseconds (3.s.f.) (see test 3), well below the criteria of five seconds. |
| 11. The overall task of opening the app on a smartphone, navigating the user interface, entering a route or search query and receiving a response from the app should not take over 90 seconds overall. | **Yes**, the overall process of the app launching, searching for a destination room, entering an origin room and generating a visible shortest path took **13.55 seconds** (see test 3) (well within the criteria of 90 seconds). |
| 12. When generating a route to a room, ‘major’ areas being passed in reaching the room will also be highlighted in order to aid the user in understanding the location of the room. | **Averted**; I instead chose to have all nodes visible/highlighted while the path is displayed, with the option to hide all nodes to isolate the path. |
| 13. Before the user has specified whether to make a search/route query, the system should display the layout backdrop of the (default) floor complete with all nodes (rooms) superimposed (and the option to change floor), with tapping each node akin to having searched for it; this will allow for the user to browse the layout before making a specific query. | **Yes**, the layout backdrop is displayed with nodes superimposed is displayed, where floors can be changed. Visibility of all nodes can be toggled, to view the backdrop/shortest path more clearly. |
| 14. The default layout view should have the following options:   1. Users should be able to navigate the layout view by dragging the view around and zooming in and out (as with the desktop program), via touch gestures. 2. It should be possible to re-centre the view to avoid getting ‘lost’ by excessive manipulation of the view. 3. Users should have the option of hiding the representations of the nodes, leaving only the background layout image visible (for clarity). | **Yes**:   * The touch gesture view allows users to navigate layouts correctly. * A ‘re-centre’ icon is provided for the user to reset the view orientation. * A ‘hide nodes’ icon is provided for the user to hide room icons, leaving only background image and route (if shown) visible. |
| 15. The space occupied by the system in the user’s device storage should not be excessive; to this end, the space occupied by the app when only my school’s layout has been downloaded should not exceed 50MB. | **Yes**, the app with the school’s layout (with all rooms featuring images) takes up **13.66MB**  (see test 5.1) (well below the 50MB required). |
| 16. Besides retrieving/downloading the layout of each building, internet connection should not be required for usage of any aspect of the system; it should arise that, having downloaded a test layout, the app should perform perfectly as expected with no internet connection. | **Yes**; besides downloading layouts, all tests carried out above were conducted without internet connection, with perfect functionality as expected (see test 5.2). |

1. Potential Improvements

Naturally, as with any project, a greater amount of time would allow for a higher quality project to be produced; although the current implementation meets/exceeds all requirements of the specification, more time would allow polish (particularly of user interface) to the standard of a professional for-profit endeavour, as well as allowing several further features to be implemented (listed below).

* The appearance of rooms on the layout can be altered to reflect their purpose, for instance colouring a particular department’s rooms with a particular colour (which could be optionally defined by the user in data entry).
* The arrow showing the user to ascend/descend floors was reported as being unclear, implying greater effort should be dedicated towards making this clear. The current implementation is that if the route requires the user to ascend/descend a floor, an upward/downward arrow is displays above/below the last node on the current floor. Similarly, if the user has just transitioned to the current floor from the one above/below, an arrow facing downwards/upwards is shown above/below the first node of the floor. Evidently this requires effort to comprehend; some form of in-app explanation may service this.
* Password protection could be utilised in layout files; although the end users were satisfied with the level of security via protected distribution (see Analysis 1.2, Undermaster’s interview), the option of security would be stronger and received better by other institutions wishing to create such layouts, wishing to keep such data private. This would be entirely feasible in terms of implementation (using asymmetric encryption and keys distributed by these institutions), but care would be required in correct implementation.
* The user could be prompted to install the app if download of a layout was attempted, although implementation would be difficult given that current distribution is file-related. This could be potentially achieved by shifting distribution to a centralized server that checks individual users for possession of the app.
* Use of a Fibonacci heap in place of a min-heap for the implementation of Dijkstra’s algorithm; although this would supposedly execute with time complexity of as opposed to current time complexity of where *E* is the number of edges and *V* is the number of vertices (nodes) thus increasing speed, it may lead to increased general case execution time due to involvement of large constants; in addition, the current execution (for relatively large layouts) is in the order of milliseconds compared to the requirement of 5 seconds, implying that the usage of min-heaps is sufficient.
* Positional data would be particularly useful, but also questionable in accuracy. An accurate solution would be Wi-Fi signal triangulation, but this would require network connection contravening specification point 13, and would require implementation from the institution side. A general solution would be GPS, but this is typically too inaccurate on the small scale as discussed in Analysis. Thus a general solution is difficult to find.
  + As an extension, positioning of fellow users of the app (for instance teachers) would suffer similar issues in terms of positioning, as well as issues of privacy. Privacy concerns would be subsided if the option to be located by other users was opt-in, as opposed to opt-out or mandatory. Provided that positioning is made possible, this system may prove useful.
* Currently, the desktop application assumes bidirectional connection between nodes during connection creation. Implementation of single-direction connections would be simple enough that the greater concern would be how to integrate it into user interface, although this could also be easily rectified.
* The timetable concept discussed in Analysis section 2.2 could have been implemented as an optional feature, greatly enhancing ease of traversal. Although not necessarily required, many institutions such as schools/universities would find this as useful as route navigation itself.
* Another possible functionality would be to demonstrate the direction in which the user is facing would be to rotate the app view to show what would lie in front of the user. This would aid in showing the user exactly where rooms lie as noted in Analysis 2.2 (regarding room appearance). However, given the nature of the app display, this would likely be considerably more difficult to implement than that which has already been done.