A Navigation System for Wheelchair Users

Supervised by Dr T W Chim

Interim Report

WONG, Chun Yin Jonathan, BEng(CS)4 (2012569450)
KWOK, Yun Tung Wizley, BEd&BSc(Chem)5 (2012539431)

2017-01-21
Summary

The Main Campus of HKU comprises of buildings situated at different levels. Travelling between buildings or levels within a building is convenient for pedestrians, but routes planned for wheelchair users require exclusion of steps and roads due to safety concerns. Accessibility information for the Main Campus is currently inadequate and scattered, and discourages wheelchair users from travelling within the campus.

A navigation system for wheelchair users is being developed for this Final Year Project to address the above issues. Geographical data of the Main Campus was transformed into a 3-dimensional connected graph that is routable with Dijkstra’s algorithm. At the current stage, the intermediate deliverable is a web application that provides primitive route suggestions within the Main Campus at coarse precision. The deliverable will later be transferred to the Android platform and made accessible via smartphones with extended scope and enhanced features.

Acknowledgements

The Project team has adapted floor and section plans of Main Campus buildings, publicly available from websites of Estates Office and Safety Office, for constructing the spatial database that serves the wheelchair navigation system. We wish to thank the staff of these service units for preparing and making available such information, which facilitated our collection of geographical data with efficiency and accuracy.

The intermediate deliverable of our project is currently developed and maintained on departmental servers. We would like to thank the Department of Computer Science for providing these computing facilities, which enabled us to implement and test the project deliverable at best convenience.
Table of Contents

Acknowledgements............................................................................................................................................. 1
Terminology......................................................................................................................................................... 3
1. Introduction..................................................................................................................................................... 3
2. Background..................................................................................................................................................... 4
3. Motivations and objectives............................................................................................................................ 6
4. Project scope.................................................................................................................................................. 7
5. Methodology and deliverables......................................................................................................................... 7
6. Challenges.................................................................................................................................................... 20
7. Progress ....................................................................................................................................................... 25
8. Conclusion.................................................................................................................................................... 25
9. References................................................................................................................................................... 26
Appendix A. Organisation of source files in project folder.................................................................................. 27
Appendix B. Workflow for administration interface operations........................................................................ 33
Appendix C. Operation instructions for user interface...................................................................................... 47
List of figures ................................................................................................................................................... 55
List of tables..................................................................................................................................................... 55
1. Terminology

The following terms are assigned specific meanings in this interim report:

- **A step** refers to a position along a path where abrupt difference in level exists.
- **A location** refers to a place that can be geographically identified by latitude, longitude, and altitude.
- **An accessible path** refers to a path, along which no steps exist.
- **A route** refers to a sequential combination of paths, connecting two locations.
- **A building** refers to a sheltered single- or multi-storey structure.
- **An open area** refers to a sheltered or unsheltered region that can be entered from its boundary at any point.
- **An entrance** refers to a location that provides step-free access to the interior of a building or an open area, with or without privileges of access.
- **A junction** refers to a location where three or more paths converge.
- **A text-based or web-based webpage** refers to an HTML page that is generally static and comprises HTML elements that render text.
- **A map-based webpage** refers to an HTML page that runs methods of the Google Maps JavaScript API and displays Google Maps on part or whole of the webpage.
- **A marker** refers to a Marker class instance of Google Maps JavaScript API, and is used to indicate locations on a map using images.
- **A line** refers to a PolyLine class instance of Google Maps JavaScript API, and is used to indicate paths on a map using straight lines.
- **A bubble** refers an InfoWindow class instance of Google Maps JavaScript API, and is used to annotate locations or paths using floating boxes.

1. Introduction

To promote usage of accessible facilities at HKU, our project team is currently developing a wheelchair navigation system which will plan accessibility-friendly routes within the Main Campus of the University of Hong Kong. The final deliverable of this project will be an online service to provide route suggestion based on user queries, and an Android application client that provides description of intermediate locations and graphical route in response to these queries.

The online service is currently available as an intermediate deliverable in the form of a web application, that is accessible via web browsers on desktop computers and mobile devices. Users can specify origin and destination at the precision of buildings or named open areas.
within the Main Campus. Route suggestion is presented in a text-based sequence of intermediate stops and a graphical route shown on Google Maps.

The current deliverable is a primitive service which provides target users with very basic information, and requires further enhancements before it can be finalised and is publishable. By the end of Semester 2, the wheelchair navigation system will be made available on the Android platform, with service coverage extended to other campuses of HKU, indoor locations made available, and user interface streamlined with better readability.

2. Background

2.1. Existing campus map

The Main Campus of HKU is situated along the hillside of Lung Fu Shan and Victoria Peak, where the steep landscape makes it necessary to have buildings rooted at different elevations. Many campus buildings are connected by walkways and stairs, making it convenient for students and staff to travel across levels within the campus. The campus map of HKU clearly illustrates the positions of these passages [1].

This campus map, however, is not beneficial to wheelchair users’ ease of access, as many of the illustrated paths are inaccessible to them (Figure 1). For example, there are two paths highlighted in yellow, one connecting May Hall with Eliot Hall and another connecting Chong Yuet Ming Amenities Centre with Knowles Building. These paths actually consist of steps that are not annotated on this campus map. Wheelchair users will instead need a map that shows accessible paths and facilities, such as lifts and ramps, while steps and stairs should be excluded.

![Figure 1 – Part of the map of HKU Main Campus, where passages involving steps are highlighted](image-url)
2.2. Accessible campus map

The HKU Estates Office had published a variant of the campus map with supplemented information regarding these facilities [2]. This map was, however, a mere screenshot of the online campus map annotated with locations of lifts, ramps and step-free paths. Two problems persist despite introduction of this annotated map.

First, accessible paths shown on the map may exist in one or more levels of respective buildings. With all such information flattened on single map, there are no means to distinguish between two passages of different levels which actually overlap (e.g. Haking Wong Building bridges the MTR station at 5th floor, and Kadoorie Biological Sciences Building at ground floor, Figure 2).

This leads to the second issue, for which floor numbers assigned to individual buildings are not standardised across campus with respect to absolute levels. One observation is that along the upper University Street from the west end to the east end, buildings are connected to the Street at different levels. For example, the Ground level of the Centennial Campus, the 1st floor of Composite Building, the 5th floor of Haking Wong Building and the 2nd floor of Chong Yuet Ming Amenities Centre are connected at the same level (Figure 3). This confuses many visitors and even newcomers of HKU, for they may consequently take incorrect paths for unintended destinations.

Figure 2 – Passages highlighted in blue are indistinguishable by levels

Figure 3 – Floor numbers of respective buildings with connection to the Upper University Street
2.3. Third-party route planners

Making use of online maps for route planning is a common practice in Hong Kong. Many digital maps, such as Google Maps [3] and Bing Maps, provide the function to plan routes designated for pedestrians, cyclists and drivers. However, these suggested routes are not necessarily safe and accessible, especially in cases when a route involves use of stairs, roads or passes across kerbs. These digital maps also share with the accessible campus map the problems of overlapping paths and lack of vertical information.

The Hong Kong Society for Rehabilitation also hosts a route planner for wheelchair users [4], with the available locations mostly being tourist attractions and shopping arcades, but schools and universities are not on the lists. The incomplete collection of locations makes its utilisation unpopular among residents and commuters.

Given the above observations, there currently lacks a proper route planner that can effectively assist wheelchair users with travelling within the Main Campus of HKU, while considering altitudes and floor levels.

3. Motivations and objectives

To improve campus accessibility information provided to wheelchair users, our project aims to introduce a wheelchair navigation system whose service will initially cover the Main Campus of HKU, enabling wheelchair users to plan journeys via web browsers. As students participating in this project, we hope to achieve the following:

- Manage spatial data on relational database
- Model the Main Campus of HKU into connected graphs
- Apply pathfinding algorithms on the graph to obtain best path suggestions
- Develop web- and map-based administration interfaces for service management
- Develop Android-based user interface for geolocation and route planning
- Construct graphical routes on top of existing digital maps
- Enforce project architecture to ensure code maintainability

For the target users of this service, i.e. wheelchair users, we hope that they will be able to:

- Obtain information regarding campus accessibility without tedious searches
- Avoid any steps or stairs, which are unsafe to wheelchair users, in their planned journeys
- Travel point-to-point within campus with minimal difficulty or at minimal time
4. Project scope

The development of project is being carried out incrementally in two main stages, corresponding to two semesters respectively. The project has a minimal scope for Semester 1 and will use an extended scope for Semester 2, depending on practicability of introducing new features. Refer to Table 1 for the comparison of features available in each deliverable.

<table>
<thead>
<tr>
<th>Table 1 – Scope of the project deliverable in two stages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semester 1 (Intermediate deliverable)</strong></td>
</tr>
<tr>
<td>• The service coverage spans exclusively the Main Campus of HKU</td>
</tr>
<tr>
<td>• Only accessible paths between buildings and open areas are evaluated.</td>
</tr>
<tr>
<td>• Campus locations are identifiable to the precision of buildings and open areas.</td>
</tr>
<tr>
<td>• Users can input origins and destinations via dropdown lists of available locations or through geolocation only.</td>
</tr>
<tr>
<td>• The spatial database is maintained exclusively by the project team.</td>
</tr>
<tr>
<td>• The best path between two locations is determined by Dijkstra's algorithm.</td>
</tr>
<tr>
<td>• The service is available as a web application for desktop computers and mobile devices.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semester 2 (Final deliverable) *</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The service coverage will be extended to other campuses of HKU</td>
</tr>
<tr>
<td>• Accessible paths within buildings will be evaluated.</td>
</tr>
<tr>
<td>• Indoor locations such as rooms will be identifiable.</td>
</tr>
<tr>
<td>• Users will be able to arbitrarily pinpoint locations on map as origins and destinations.</td>
</tr>
<tr>
<td>• Users will be able to report updates on availability of accessible facilities.</td>
</tr>
<tr>
<td>• Suggested route will vary by introducing other pathfinding algorithms and with additional search criteria.</td>
</tr>
<tr>
<td>• The service will be made available as an Android application.</td>
</tr>
</tbody>
</table>

*: It is not guaranteed that all features mentioned in this stage will be implemented.

5. Methodology and deliverables

5.1. Completed tasks and intermediate deliverable

5.1.1. Conceptual design

5.1.1.1. Project requirements

The project deliverable was initially constructed as a multi-tier web application with the following functional requirements:

- The project team can maintain the database of spatial information and accessible paths by directly manipulating the database or through data entry on web- and map-based pages.
- The target users can use the wheelchair navigation system to specify required origins and destinations efficiently, and view the routing results as a list of intermediate stops along the suggested route, or an annotated route shown on a digital map.
- The service can be accessed via both web browsers and smartphones.
5.1.1.2. Project architecture

Figure 4 shows an overview of the project architecture. The coloured boxes denote completed items constructed for the intermediate deliverable. Detailed explanation of the architecture is described in Appendix A, while design and implementation of the features are discussed in the following sections.

For the convenience of ad hoc development and maintenance, the server-side scripts of the project deliverable are currently run on servers provided and maintained by the Department of Computer Science:

- **sophia.cs.hku.hk**, installed with MySQL database server 5.1.35.
- **i.cs.hku.hk**, installed with Apache HTTP server 2.4.7 and PHP 5.5.50.

The client-side webpages of the deliverable can run on modern web browsers with JavaScript support. Google Chrome 54 on Windows 8.1 is currently used for testing these webpages.

The project codes of this deliverable are written in the following languages:

- **SQL**. Stored routines are written for and stored in the MySQL database. Most of these routines serve to manipulate spatial data while enforcing data integrity. The rest of them serve to execute search queries efficiently as native SQL statements. Programming of these functions as SQL routines also encapsulates the database schema so external scripts accessing the database are not required to keep track of low-level database particulars such as table structure and data types.
- **PHP**. Scripts written in this language are executed on the Apache server for rendering HTML pages. These scripts are organised in layered architecture to
separate operations that carry out database access, data conversion, webpage rendering and form submission handling.

- **JavaScript.** Scripts written in this language are executed on web browsers. The JavaScript codes serve the purpose of directly manipulating webpage elements while reducing page loads during client-server communications, such as narrowing down location options during selection of origin and destination for planned routes. The jQuery library is imported in most of these scripts to simplify codes. The Google Maps JavaScript API [5] is imported to render Google Maps on webpages.

### 5.1.2. Geographical model

The geographical environment of the Main Campus was initially modelled into a connected graph as shown in Figure 5, where locations such as building entrances and junctions were represented by nodes, and accessible paths were represented by weighted bidirectional edges. Special nodes were defined to represent buildings and open areas, connecting all locations that belong to the building or area. The hierarchy of locations assigned logical relationships between nodes and made it easy for users to select locations at their preference of precision.

![Figure 5 – Conceptual model of the Engineering buildings as a connected graph, with (a) Building entrances and junctions in orange, (b) buildings in red, and (c) paths in blue](image)

A major drawback of this model was that the abstraction of paths to atomic straight lines of edges significantly deviated from actual shapes of these paths and made it hardly interpretable by map readers in general. In the worst case, a path within an area could comprise of twisted segments, while the project on map would merely be a straight line. The lack of information would not effectively assist wheelchair users with finding the right paths along a route.

An approach to improve path projection was to define locations and paths analogous to planning of tactile paths for visually impaired persons. Tactile paths can be defined in straight lines with explicit junctions and end points, making them suitable as a form of path representation on Google Maps API for the purpose of this project.
5.1.2.1. Mapping scheme

As the project aims to provide detailed routing services for users to navigate between buildings, clear indication of exits and entrances of each building is necessary. To achieve this purpose, the spatial data should be:

- Able to differentiate different exits at the same level of height
- Able to differentiate exits of different level of height
- Able to differentiate overlapping paths at different altitude

Several mapping schemes were found online. Most of which treat locations as a mesh of nodes while each of them gives different definition to nodes and edges, resulting in different running complexities of the product. The team examined several of these representations and came up with an adaptation of the Polygon Movement scheme [6] for this project:

Table 2 – Adaptation of Polygon Movement as mapping scheme for the project, with comparison

<table>
<thead>
<tr>
<th>Polygon Movement scheme</th>
<th>Project adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
</tbody>
</table>
| Brown nodes were located inside every grid, where some of which has no meaning to users. Some nodes having only 2 neighbouring nodes can be eliminated as they must be a point on the path connecting neighbouring nodes. | The nodes were set based on their physical meaning:  
- Green Nodes – Waypoints  
- Brown Nodes – Path Junctions  
- Yellow Nodes – Entrance to Buildings  
Nodes of different meaning will be processed differently |

To realise this mapping scheme, geographical coordinates (latitudes and longitudes) must be used as the representation scheme for spatial positions. As precision of these data directly affects the storage cost of the database, the decided precision was set at 5 decimal points for longitude and latitude degrees, or the distance of approximately 1 metre at the equator. This precision is sufficient for differentiating geographical positions of same altitude at building level.
5.1.2.2. Location representation

Each location is a unique combination of latitude, longitude and altitude. Since these geographical coordinates are interpretable only when referenced on a map, it is required that each location be further identified by an address-like combination of names, such as building name and room number to enhance location readability. Some difficulties encountered during this stage of work were discussed in section 6.1.

5.1.2.3. Path representation

Each path is a directed connection between two nodes, and represent an accessible path. To be considered accessible, the path must involve no steps and does not lie within busy roads, so ramps and walkways are examples of such paths. Paths between any two connected nodes are always defined in pairs. Each path is weighted by a cost function suited for mountaineering [6], so the costs differ for ascending and descending paths. These weights directly determine whether paths would be included in a route suggestion or not.

Stairs, walkways involving steps, road crossings with busy traffic and paths with steep gradient are considered inaccessible in this project, and therefore are not defined on the map. In general, any access beyond the defined network of paths in this project are assumed inaccessible.

The assembly of nodes and directed edges forms a connected graph that is three-dimensional and separable by altitudes (Figure 6). This data representation of the Main Campus makes possible the utilisation of many pathfinding algorithms, such as Dijkstra’s algorithm, for planning accessible routes between Campus locations.
5.1.3. Data management on database

The wheelchair navigation system stores all above-mentioned spatial data in sophia.cs.hku.hk. To avoid data anomalies such as location duplication and headless paths, stored routines were implemented to minimise occurrence of such anomalies. For example, one of the routines forbids double entry of a location with identical combination of latitude, longitude and altitude.

Additional routines were also implemented to execute frequently used queries, such as listing of paths. These routines were written to accept few parameters so complex SQL statements need not be handled by PHP scripts. Table structures in the database are encapsulated by these routines so dependency is minimised between PHP scripts and the database. In other words, there is no need to maintain information about tables and attributes in PHP scripts that access the MySQL database.

5.1.4. Administration interface

Given the significant amount of campus locations and accessible paths evaluated on the Main Campus, it is generally impracticable to manually input all the evaluated information into the database as raw data of coordinates, due to the risk of erroneous data entry and difficulty to verify the input data. To simplify the process of data entry,
functions and corresponding administration interfaces for managing locations and paths were developed, in both web and map views.

5.1.4.1. Web view

The web view generates HTML pages which allow us to search for locations and paths in sorted tables, to input or maintain individual locations by entering primitive data of coordinates and names, and to define or maintain individual paths by indicating origin and destination from the list of existing locations.

5.1.4.2. Map view

The map view allows data entry and maintenance directly on Google Maps (Figure 7), on which existing locations and paths are displayed as markers and lines. Clicking an unmarked position on the map allows input of a new Campus location, while clicking on a marker allows search and maintenance on existing locations, or entry of new paths. Similarly, clicking on a line allows search and maintenance of existing paths.

The above administration operations required implementation of workflow for both views. The workflow for web view was as simple as specifying several types of HTML form submissions, but map views required webpages to keep track of states and transitions in workflow for smooth operations on data entry. The challenges we faced during implementation of workflow on map view was discussed in section 6.2.

5.1.5. Pathfinding algorithm

Findings from the Internet suggest that Dijkstra's algorithm and A* search are the most common pathfinding algorithms for most online routing services. Refer to the comparison of these two algorithms below:

![Figure 7 – Map view of path management with existing markers, lines, and paths projected on a line](image-url)
Table 3 – Comparison of Dijkstra and A* Search pathfinding algorithms

<table>
<thead>
<tr>
<th>Nature</th>
<th>Dijkstra’s Algorithm (directional, binary heap)</th>
<th>A* Search Algorithm (directional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>Greedy Algorithm</td>
<td>Heuristic Search</td>
</tr>
<tr>
<td></td>
<td>Assured</td>
<td>Local minimum may occur</td>
</tr>
<tr>
<td>Time Complexity</td>
<td>O((E + V) log V)</td>
<td>O(E + V)</td>
</tr>
<tr>
<td></td>
<td>On binary heap</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• E = number of edges</td>
<td>• E = number of edges</td>
</tr>
<tr>
<td></td>
<td>• V = number of vertices</td>
<td>• V = number of vertices</td>
</tr>
<tr>
<td>Space Complexity</td>
<td>O(log n) on binary heap</td>
<td>O(log n) on binary heap</td>
</tr>
<tr>
<td></td>
<td>On binary heap</td>
<td></td>
</tr>
<tr>
<td>Advantages</td>
<td>Non-exhaustive search</td>
<td>Short running time when size of vertices and edge is large (due to the use of heuristics)</td>
</tr>
<tr>
<td></td>
<td>Always provide optimal solution efficiently when compared to other algorithms</td>
<td></td>
</tr>
<tr>
<td>Disadvantages</td>
<td>Performance highly influenced by increase in data size</td>
<td>Performance heavily dependent on heuristic function</td>
</tr>
<tr>
<td></td>
<td>• Search is uninformed (i.e. searching in all possible directions)</td>
<td>• Optimal solution not always guaranteed (local minimum)</td>
</tr>
</tbody>
</table>

For the purpose of this project, where only the HKU Main Campus was of interest at the current stage, number of edges and nodes is limited and the concern over correctness of solution outweighs that over the speed of searching (i.e. The search will not take too long as the data amount is small). Therefore, the team decided to implement Dijkstra’s algorithm as the base pathfinder.

The pathfinder was implemented on the database server as a stored routine. Given two nodes corresponding to origin and destination, this algorithm finds a combination of paths, with smallest sum of weights, that exists between these two nodes. The algorithm was chosen for its versatility and procedural simplicity, and had been modified to be compatible with the modelled graph representation of the Main Campus. Compared to implementing the algorithm in PHP script, the stored routine approach guarantees faster execution of this algorithm when natively implemented on the database, and costs minimal networking, memory and processing loads on the Apache server.

The pathfinding routine also takes only two parameters that represent the two ends of a required route. The returned result is either a sequence of paths that bridge the two ends, along with node and edge attributes, or an error message in case no possible route exists between these two ends. This routine has been tested on the wheelchair
navigation system with guaranteed route suggestions, and is being adjusted to construct routes that are more practicable from target users’ perspective.

5.1.6. User interface

The web-based user interface of the wheelchair navigation system comprises of 3 views: query form, route suggestions in text description and graphical route displayed on Google Maps. We attempted to replicate the most common design features of route planners that are available from popular digital maps.

The user interface is now available in two layouts, mobile and desktop. The following sections discuss individual views presented on the mobile version, while the desktop version is in general a combination of the 3 views on the same webpage, which is suitable for display on desktop screens (Figure 8).

![User Interface Example](image)

Figure 8 – Desktop mode of the user interface for route search

5.1.6.1. Query form

The query form allows users to specify origin and destination from locations defined in the database (Figure 9). Options include open areas, junctions, buildings, and building entrances at different floors if available. For each set of input fields for origin and destination respectively, users are required to specify locations in two phases:
1. Users type the name of the required building or area in the text box, and upon display of auto-completion options below the text box, they select the desired option to filter campus locations. If a user accesses the service via a smartphone and want to plan routes to or from their current location, they can use the geolocation function to approximately determine the building or area that they are situated in, and the operation is equivalent to inputting building or area name in the text box.

2. The available options from the dropdown list are updated with locations belonging to the building or area specified in the text box. Users choose from the options in dropdown list to specify their required origin or destination. These selected values are submitted for processing by the pathfinding algorithm.

![Two-phase entry of location selection for origin and destination](image_url)

*Figure 9 – Two-phase entry of location selection for origin and destination by typing name of building (top) and selecting location within a building (middle)*

5.1.6.2. **Text-based results**

The internal results returned for a query are either a sequence of paths, if a route exist, or an error message if no possible route can be constructed. For successful search, the result is displayed to the user as a table of intermediate locations along the suggested route (Figure 10). Each location is described by its name and duration to reach from the previous location. A hyperlink is also displayed on this page, leading the user to the graphical route presented on a separate page.
An unsuccessful search returns an error message, indicating that no route can be constructed from existing paths. Graphical route is not available for unsuccessful search.

![Total time: 22 min 4 sec](image)

0. Starting point

*Entrance at Asia Global Institute (Room G03-G05),
Main Building (G/F)*

1. 17 sec

*Path Junction at Asia Global Institute (Room G03-G05),
Main Building (G/F)*

2. 34 sec

*Path Junction at Plaza,
Main Building (G/F)*

3. 17 sec

*Path Junction at Upper ramp exit,
Main Building (G/F)*

*Figure 10 – Text-based search result with intermediate stops*

5.1.6.3. **Map-based results**

Suggested routes, if available, are displayed on Google Maps as series of markers and lines (Figure 11).

Each marker corresponds to an intermediate location, listed in the table mentioned in the previous section. Since it is difficult to identify location by placement of a marker, each marker is programmed to display a pop-up balloon showing the location name whenever this marker is clicked. Pagination buttons are provided in the balloon, allowing the user to travel step-by-step along intermediate stops on the map.

The lines correspond to the accessible paths included in a suggested route. The lines are coloured differently for paths of different nature: blue for flat paths, green for descents, and red for ascents. Grey lines represent abstraction of indoor paths and cannot be followed, as the project scope for Semester 1 does not cover indoor paths.
5.2. Pending tasks for final deliverable

Since the wheelchair navigation system will be made available on Android devices in Semester 2, some user interfaces and service features will be transferred to the Android platform. Some additions will be made to the functional requirements and project architecture. The service coverage will also be extended accordingly.

5.2.1. Functional requirements for Android application

With the introduction of service to the Android platform, there are many platform-specific features that can be exploited, given the Android device is equipped with appropriate sensors:

- Geolocation can be carried out dynamically through Wi-Fi, GPS or cellular signals.
- Phone orientation can be measured by compass.
- Elevation can be determined by GPS or barometer.

If feasible, the functional requirements will be supplemented to the navigation system accordingly as follows:

- Geolocate on smartphones, and plan routes from or to users’ current location.
- Report to the project team when users find any accessible paths unusable, or when they discover new accessible paths.
- Keep record of the routes that are frequently used, and possibly share the routes with other users.
The user interface for the Android application will be like that accessed via web browsers, so users can switch between platforms without significant difference in experience.

5.2.2. Update to project architecture

Refer to the grey boxes of Figure 4 for pending items, corresponding to the Android layer of project architecture. The Android application will require frequent access to the wheelchair navigation system via the Internet, so additional PHP scripts will be appended to the project for handling data requests from Android devices. Layered architecture will also be enforced for this extension to facilitate code maintenance.

Java is the programming language for developing Android applications. The Android application will be built first with a user interface similar to that of the web application, then additional features will be introduced incrementally. Depending on the file system of Android application projects, there will still be attempts to enforce layered architecture on Android project codes.

5.2.3. Extension to service coverage

The current scope of the wheelchair navigation system is confined to the Main Campus of HKU. We will extend the collection of accessibility information to other campuses of HKU, so students and staff travelling within university premises will be able to access the navigation service while outside the Main Campus.

The current proposal is to expand our service to the Sassoon Road Campus. This is the second major campus of HKU where the Faculty of Medicine and many residential halls reside. The process of data collection will be similar to that done for the Main Campus.

5.2.4. Indoor accessible paths

The accessible paths defined for the intermediate deliverable excluded indoor areas, therefore some paths for a suggested route would be shown in grey on Google Maps. These paths in grey are not well-defined and do not accurately project the indoor paths within buildings.

Floor plans of campus buildings are publicly available from the Safety Office website [8], which can assist us with defining locations and accessible paths for each building. The consideration of altitudes will still be enforced so locations of different elevations situated at same geographical coordinates will still be distinguishable. Upon completion of path construction, it is expected that the suggested routes displayed on Google Maps can be accurately projected onto actual paths.
6. Challenges

6.1. Determination of coordinates and altitudes

Determining the exact position of a campus location is not an efficient task. Even with GPS-equipped smartphones, the readings may significantly deviate from actual coordinates. The determination of coordinates for campus locations evaluated in Semester 1 were carried out in 3 approaches: collection of GPS readings, geolocation via Google’s Location Service and collection of coordinates from satellite view of Google Earth. These 3 approaches usually return inconsistent coordinates for same location.

6.1.1. GPS

A GPS-enabled smartphone relies on signals from several satellites to determine its geographical position, and the readings are most accurate when geolocation takes place at unsheltered areas. By comparison, most of the campus locations are either located indoors or very close to buildings, making it difficult to intercept signals from satellites as these signals are blocked or reflected by structures.

6.1.2. Google’s Location Service

Location Service by Google enables a smartphone to have its location determined via Internet connection. Although this Service can sometimes provide better approximation to actual positions than GPS, the determined positions returned for a fixed location may vary for different trials due to factors such as route taken for sending geolocation request through the Internet and signal strength of Internet connection.

6.1.3. Google Earth

Google Earth provides the most accurate projection of campus locations to geographical coordinates, as satellite view is available in high-definition images (Figure 12). Google Earth is also capable of providing altitude readings at surfaces of three-dimensional buildings. Most of the latitude, longitude and altitude readings for campus locations were collected from Google Earth, assuming these values are exact and accurate.
Given the above approaches for collecting geographical data, it was expected that coordinates and altitudes of locations could be recorded at best accuracy. However, when such data was visualised on Google Maps, these locations still appeared deviating from their expected positions. Our later discussions suggested that the canvas of Google Maps is itself inaccurate and geographical features presented on the map are incomplete, resulting in such issues.

Figure 13 shows the marker of the Landscape Garden on Google Maps, whose geographical coordinates are directly obtained from the screenshot of Google Earth in Figure 12. It is clearly seen that the hand cursor resides within the area of Haking Wong Building in Google Earth, while the marker representing the same location lies just outside the boundary of the same building on Google Maps. The discrepancies in the visualised positions of locations make it difficult to decide how locations should be placed on the Google Maps canvas.
The project team had discussed which approach to take for recording positions of campus locations. Eventually, it was agreed that placements of markers would initially be set to actual geographical coordinates, then adjusted to assimilate into spatial relationships of geographical features as displayed on the Google Maps canvas. We believed that the coercion of geographical coordinates with the map canvas would ensure consistent route interpretation among target users of this service.

6.2. Administrative workflow on Google Maps

Although the Google Maps API allows content being dynamically displayed on the map canvas, some features of the API do not favour interactions between users and elements displayed on the map. One of these features is the InfoWindow class, which is responsible for displaying bubble messages pointing to a position on the map.

The process of data management on map view was intended to simulate that of the web view, except that manual typing of geographical coordinates was replaced by mouse click on the map canvas, and that selection of locations and paths from HTML tables or dropdown lists was replaced by mouse click on markers and lines. The InfoWindow class of the JavaScript API was therefore used to display floating windows for inputting details of locations and paths.

6.2.1. Example workflow for location maintenance

The map view of location maintenance provides operations to list, add, edit and delete locations directly on Google Maps. Locations that exist in the database are displayed
as markers on the map canvas, while management of locations is carried out via InfoWindows.

Each of these operations require different contents displayed on the same InfoWindow at different stages of workflow. Figure 14 shows an example to modify details of a location on map view. When a marker on a map is clicked, an InfoWindow appears anchored to the marker and shows the locations defined at the position of the marker, in descending order of altitudes. Upon clicking a hyperlink corresponding to a location, content of the InfoWindow is replaced with the input form of location attributes. After editing the form values and clicking Save, a message is shown to confirm successful update to the location.

6.2.2. Example workflow for path maintenance

The map view of path maintenance provides operations to list, add, edit and delete paths directly on Google Maps. Paths that exist in the database are displayed as lines between two markers on the map canvas, or none if the path connects two vertically displaced locations that are located at the same marker.

The content displayed on InfoWindows differ among the operations on paths. Figure 15 shows an example to add a new path. When a marker is clicked on the map, an InfoWindow appears anchored to the marker and shows the locations defined at the marker position, as well as any vertical paths lying beneath the same marker. After clicking the button corresponding to a location, content of the InfoWindow is replaced and prompts for another marker to be clicked. Upon clicking a second marker (which could be the same marker), the InfoWindow is displaced and anchored to the second marker, and shows the locations available at this marker position. After selecting the second location, an input form is displayed for entering path attributes. After data entry and Save is clicked, a confirmation message is shown on the InfoWindow.
As seen from the above observations, the InfoWindow elements are responsible for displaying dynamic contents across different stages of workflow. However, the method for manipulating contents of an InfoWindow only accepts static HTML codes, making it difficult to use InfoWindows for data entry and user interaction.

After some studies on the behaviour of InfoWindows, a solution was designed and implemented to enable workflow on Google Maps using code segments from 3 scripts. First, content displayed on InfoWindows are generated, depending on type of operation and parameters passed, in an external PHP script and then injected into the InfoWindow via AJAX requests. Second, event handlers are registered to the generated content of InfoWindows so JavaScript methods can be triggered when some action buttons or hyperlinks are clicked. Finally, the JavaScript methods called by InfoWindow elements are implemented in the HTML page that contains the Google Maps canvas. Additional variables are also declared in this HTML page to keep track of user inputs during stages of workflow. Thus, a complete workflow for an operation on location or path management involves continuous passing of function calls and AJAX requests among PHP scripts, the InfoWindow elements and the scripts embedded on the page that displays the Google Maps canvas.
7. Progress

Table 4 lists tasks involved in the project and their progress.

Table 4 – Progress of project tasks

<table>
<thead>
<tr>
<th>Item</th>
<th>Completion date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semester 1 (Intermediate deliverable)</strong></td>
<td></td>
</tr>
<tr>
<td>Study of Google Maps API</td>
<td>28 September 2016</td>
</tr>
<tr>
<td>Service implementation</td>
<td>28 September 2016</td>
</tr>
<tr>
<td>Deliverables of Phase 1</td>
<td>2 October 2016</td>
</tr>
<tr>
<td>• Detailed project plan</td>
<td>2 October 2016</td>
</tr>
<tr>
<td>• Project webpage</td>
<td>2 October 2016</td>
</tr>
<tr>
<td>First data entry</td>
<td>12 October 2016</td>
</tr>
<tr>
<td>Verification of pathfinding algorithm</td>
<td>26 October 2016</td>
</tr>
<tr>
<td>Rendering on Web application</td>
<td>23 November 2016</td>
</tr>
<tr>
<td>Preliminary testing</td>
<td>30 November 2016</td>
</tr>
<tr>
<td>Deliverables of Phase 2</td>
<td>28 December 2016</td>
</tr>
<tr>
<td>• Preliminary implementation</td>
<td>28 December 2016</td>
</tr>
<tr>
<td>• Detailed interim report</td>
<td>28 December 2016</td>
</tr>
<tr>
<td><strong>Semester 2 (Final deliverable)</strong></td>
<td></td>
</tr>
<tr>
<td>Feedback evaluation</td>
<td>21 December 2016</td>
</tr>
<tr>
<td>Enhancement study</td>
<td>4 January 2017</td>
</tr>
<tr>
<td>Development of Android application</td>
<td>18 January 2017</td>
</tr>
<tr>
<td>Second data entry</td>
<td>1 February 2017</td>
</tr>
<tr>
<td>Enhancement design and implementation</td>
<td>22 February 2017</td>
</tr>
<tr>
<td>Intensive testing</td>
<td>8 March 2017</td>
</tr>
<tr>
<td>Final review and amendments</td>
<td>29 March 2017</td>
</tr>
<tr>
<td>Deliverables of Phase 3</td>
<td>16 April 2017</td>
</tr>
<tr>
<td>• Finalised tested implementation</td>
<td>16 April 2017</td>
</tr>
<tr>
<td>• Final report</td>
<td>16 April 2017</td>
</tr>
<tr>
<td>• Final presentation</td>
<td>16 April 2017</td>
</tr>
<tr>
<td>• Project presentation</td>
<td>16 April 2017</td>
</tr>
</tbody>
</table>

Note: Due dates other than departmental deadlines are subject to changes

8. Conclusion

Development of the wheelchair navigation system is currently complete as an intermediate deliverable for web platforms. Although the service only provides minimal features, it is guaranteed that a successful search results in successful presentation of path sequence and intermediate stops in a table and graphical route displayed on map canvas. The current service covers buildings and open areas of the Main Campus, and includes only outdoor paths that connect these buildings and areas.

For the final deliverable to be completed in Semester 2, we will develop the user interface of the service on Android platform. The service coverage will also be expanded outside the Main Campus of HKU. Above all, this project aims to encourage wheelchair users to travel within
the HKU campus with greater convenience by effectively utilising the accessible facilities that are currently available.

9. References


Appendix A. Organisation of source files in project folder

The project folder comprises PHP, JavaScript sources that provide administrative and routing functions, as well as supporting CSS and image files that enhance webpage presentation on browsers. The hierarchy of folders is limited to at most one level, for the ease of locating and maintaining source codes within the same directory. Some of the folders and files are listed as follows:

- / (Root directory)
  - constants.php
  - db.php
  - index.php

- /service/
  - portSvc.php
  - pathSvc.php
  - routeSvc.php

- /webForm/
  - portForm.php
  - pathForm.php
  - routeForm.php

- /webHandler/
  - portHandler.php
  - pathHandler.php
  - routeHandler.php

- /webList/
  - portList.php
  - pathList.php

- /map/
  - portMap.php
  - pathMap.php
  - routeMap.php

- /comms/
  - portJSON.php
  - portInfoWin.php
  - pathJSON.php
  - pathInfoWin.php
  - routeFormJSON.php
  - routeMarkerJSON.php
  - routeLineJSON.php

- /ui/
  - index.php
  - style.css

- /js/
  - routeForm.js
  - routeMap.js

- /icons/

Note: For a complete directory and file listing, please refer to the deliverable submission.

Since dependencies exist between script files located within different folders, behaviour and dependencies of these files of concern are discussed in the following sections individually.

A.1. / (Root directory)

The root directory is a main reference location for all PHP and JavaScript files. For any script file located within a directory, reference to other script files require the suffix ‘../’ included in the include path to denote the root directory, before drilling down to the required directory one level beneath.

A.1.1. constants.php

The wheelchair navigation system requires frequent access to the database server. Login credentials for database access, namely server name and port, login username and password, are stored in plain text PHP script, placed at the root directory and referenced by /db.php whenever database access is required. It is assumed that the access rights to this file are limited to authorised persons such that it would be safe to keep this file at the root directory of the project folder.
A.1.2. db.php
This PHP script provides functions to access MySQL database through execution of low-level MySQL queries. The only function in this script retrieves the login credentials to the database server from /constants.php, builds the query of a routine call by accepting a routine name and an array of routine parameters, executes the query, and returns a result set of dynamic structure depending on the caller function. The result set is then returned to caller functions of PHP scripts located in /service/.

A.1.3. index.php
An administrative interface is constructed on this PHP file, with links pointing to Port and Path management, and route search pages dispersed in /webForm/, /webHandler/, /map/ and /ui/ folders.

A.2. /service/
The /service/ folder holds functions corresponding to functional requirements of the project deliverable. These functions facilitate access to the database by providing routine names and arranging function parameters in array form, then calling the function in /db.php by passing in the routine name and the array as parameters. The returned result set is appropriately rearranged and retrieved by corresponding caller functions in higher-level scripts for further manipulation.

A.2.1. portSvc.php
/service/portSvc.php holds functions for calling stored routines that manipulate individual Port records and list Port records in sorted or unsorted manners.

A.2.2. pathSvc.php
/service/pathSvc.php holds functions for calling stored routines that manipulate individual Path records and list Path records in sorted or unsorted manners.

A.2.3. routeSvc.php
/service/routeSvc.php holds functions for calling stored routines that list areas and Port names belonging to a specified area.

A.3. /webForm/
The /webForm/ folder holds HTML forms for inputting and editing details of Port or Path records, serving the initial method for primitive data collection and input. To facilitate more efficient data entry, JavaScript helper functions are implemented to preload relevant data from /service/ scripts and assist with input of certain fields.
A.3.1. portForm.php
/webForm/portForm.php provides HTML form for creating, editing or deleting individual Port record. JavaScript helper function is provided to parse geographical coordinates copied from Google Earth and copy values to latitude and longitude fields respectively.

A.3.2. pathForm.php
/webForm/pathForm.php provides HTML form for creating, editing or deleting individual Path record. Available Ports are listed as options in dropdown lists of origin and destination fields respectively.

A.3.3. routeForm.php
/webForm/routeForm.php provides HTML form for inputting origin and destination in 2 phases. JavaScript helper functions are implemented to filter Port options after area input and rejecting invalid inputs.

A.4. /webHandler/
The /webHandler/ folder holds PHP scripts that handle form submissions triggered by pages from /webForm/ and AJAX calls by scripts from /map/ and /js/. These handlers provide redirection back to /webForm/ or /webList/ pages, or display page content directly generated by handler scripts.

A.4.1. portHandler.php
/webHandler/portHandler.php collects data submitted by /webForm/portForm.php and /map/portMap.php for inserting, updating and deleting Port records.

A.4.2. pathHandler.php
/webHandler/pathHandler.php collects data submitted by /webForm/pathForm.php and /map/pathMap.php for inserting, updating and deleting Path records.

A.4.3. routeHandler.php
/webHandler/routeHandler.php displays text-based route suggestions according to specified origin and destination of a requested route submitted from /webForm/routeForm.php or /ui/index.php.

A.5. /webList/
The /weblist/ folder holds functions for listing text-based Port and Path records on webpages in web view, by calling functions provided in scripts within /service/ folder. Hyperlinks are provided to point to pages kept in /webForm/ for creating and updating respective records. These listing functions also enable sorting of records by required column, as this functionality is made available in corresponding stored routines.
A.5.1. portList.php
/webList/portList.php displays details of all Port records in tabular form, by calling function from /service/portSvc.php.

A.5.2. pathList.php
/webList/pathList.php displays details of all Path records in tabular form, by calling function from /service/pathSvc.php.

A.6. /map/
The PHP files in /map/ folder provide webpages for viewing and managing geographical data on Google Maps. For data management, these webpages provide the workflow for listing, viewing, adding, editing and deleting records with resulting operations equivalent to those on /webForm/ and /webList/. These operations are handled by scripts from /comms/ and /webHandler/.

A.6.1. portMap.php
/map/portMap.php provides functions to view, add, edit and delete Port records directly on Google Maps, without the need to manually input coordinates of Port positions. Altitude range for Ports to be manipulated can be specified to hide unwanted Markers. Markers representing Port locations are pre-loaded on the map through AJAX requests to /comms/portJSON.php. Workflow of operations on Port records are carried out on HTML forms generated by /comms/portInfoWin.php. Vertically distributed Ports located at the same geographical position are distinguishable due to consideration of altitudes for Port records. Markers are also dynamically created and removed as appropriate during manipulation of Port records.

A.6.2. pathMap.php
/map/pathMap.php provides functions to view, add, edit and delete Path records directly on Google Maps by clicking on Markers for Port selection and clicking on PolyLines for Path selection. Altitude range for Paths to be manipulated can be specified to hide unwanted Markers and PolyLines. Markers and PolyLines, representing Ports and Paths respectively, are pre-loaded on the map through AJAX requests to /comms/portJSON.php and /comms/pathJSON.php. Workflow of operations on Path records are carried out on HTML forms generated by /comms/pathInfoWin.php. PolyLines are dynamically created and removed as appropriate during manipulation of Path records.
A.6.3. routeMap.php

\texttt{/map/routeMap.php} presents the map view for route suggestions displayed on 
\texttt{/webHandler/routeHandler.php}. Route suggestions are displayed on map as Markers and PolyLines with relevant details through AJAX requests to \texttt{/comms/portJSON.php}, \texttt{/comms/routeMarkerJSON.php} and \texttt{/comms/routeLineJSON.php}.

A.7. /comms/

The \texttt{/comms/} folder provides scripts facilitating map-based data requests through AJAX calls. PHP functions from \texttt{/service/} folder are called to retrieve the relevant data as requested. There are 2 major types of data: HTML codes for Google Maps InfoWindows and JSON data for placement of Google Maps Markers and PolyLines on map canvas of \texttt{/map/} and \texttt{/ui/} pages.

A.7.1. portJSON.php

This handler sends geographical positions of Port records to \texttt{/map/portMap.php}, \texttt{/map/routeMap.php} and \texttt{/ui/index.php} in JSON format, after collecting data from function call in \texttt{/service/portSvc.php}.

A.7.2. portInfoWin.php

This handler generates HTML forms for manipulation of Port records on \texttt{/map/portMap.php}, after collecting data from function call in \texttt{/service/portSvc.php}.

A.7.3. pathJSON.php

This handler sends geographical positions of Path records to \texttt{/map/pathMap.php} in JSON format, after collecting data from function call in \texttt{/service/pathSvc.php}.

A.7.4. pathInfoWin.php

This handler generates HTML forms for manipulation of Path records on \texttt{/map/pathMap.php}, after collecting data from function call in \texttt{/service/pathSvc.php}.

A.7.5. routeFormJSON.php

This handler retrieves Portal and Port information according to data provided by \texttt{/webForm/routeForm.php} and \texttt{/ui/index.php} in JSON format, after collecting data from function call in \texttt{/service/routeSvc.php}.

A.7.6. routeMarkerJSON.php

This handler retrieves geographic positions of the intermediate stops for a suggested route and sends the data to \texttt{/webForm/routeMap.php} and \texttt{/ui/index.php} in JSON format, after collecting data from function call in \texttt{/service/routeSvc.php}. 

31
A.7.7. routeLineJSON.php
This handler retrieves geographic positions of the intermediate stops for a suggested route and sends the data to \texttt{/webForm/routeMap.php} and \texttt{/ui/index.php} in JSON format, after collecting data from function call in \texttt{/service/routeSvc.php}.

A.8. /ui/
The \texttt{/ui/} folder holds files for presenting the desktop version of route search function. It shares most of the handler functions with pages from \texttt{/webHandler/} and \texttt{/comms/} with the mobile version of the route search function.

A.8.1. index.php
\texttt{/ui/index.php} is a desktop version of the route search functions with three sections: search form, text-based search results and map-based search results. Helper functions from \texttt{/js/} are required for two-phase location input and display of route on Google Maps.

A.8.2. style.css
CSS rules are implemented to ensure all page elements can fit on a desktop screen.

A.9. /js/
The \texttt{/js/} folder provides essential scripts, written in JavaScript, that assist with location input and route presentation of the route search function, by initiating AJAX calls to \texttt{/comms/} scripts for data submission and retrieval. Since the mobile and desktop versions make use of the same set of script functions for operations, this folder provides a unified source for execution of these functions with easy maintenance.

A.9.1. routeForm.js
\texttt{/js/routeForm.js} provides helper functions to assist with two-phase input of origin and destination. The results are used by \texttt{/webHandler/routeForm.php} and \texttt{/ui/index.php}.

A.9.2. routeMap.js
\texttt{/js/routeMap.js} generates Google Maps embedded on \texttt{/map/routeMap.php} and \texttt{/ui/index.php}.

A.10. /icons/
The \texttt{/icons/} folder keeps several PNG files required for presenting locations on Google Maps as Markers. These Markers represent Ports and Portals for map view of administration pages, and origin, intermediate stops and destination for map view of the user interface.
Appendix B. Workflow for administration interface operations

B.1. Port management

B.1.1. Add first Port at position

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click on map to specify position</td>
<td><img src="image1" alt="Screenshot" /></td>
</tr>
<tr>
<td>2. Click <strong>Add</strong> to create new Port</td>
<td><img src="image2" alt="Screenshot" /></td>
</tr>
</tbody>
</table>
| 3. Input Port details  
  a. Input **Altitude** of Port  
  b. Input **Name** of Port  
  c. Input **Area** of Port  
  d. Select **Port Type** from options | ![Screenshot](image3) |
| 4. Click **Save** | ![Screenshot](image4) |
| 5. Success message is shown for successful Port entry  
  6. Click **Return** to view list of existing Ports found at Marker position | ![Screenshot](image5) |
### B.1.2. Add Port at position

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Marker on map</td>
<td><img src="image1.png" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Check list of existing Ports&lt;br&gt;3. Click <strong>Add</strong> to create new Port at same position</td>
<td><img src="image2.png" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>4. Input Port details (Refer to Step 3, B.1.1)&lt;br&gt;5. Click <strong>Save</strong></td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>6. Success message is shown for successful Port entry&lt;br&gt;7. Click <strong>Return</strong> to view list of existing Ports found at Marker position</td>
<td><img src="image4.png" alt="Screenshot 4" /></td>
</tr>
</tbody>
</table>
### B.1.3. View / Edit Port details

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Marker on map</td>
<td><img src="image1.png" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Check list of existing Ports</td>
<td><img src="image2.png" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Port ID to view Port details</td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Edit details of Port as necessary</td>
<td><img src="image4.png" alt="Screenshot 4" /></td>
</tr>
<tr>
<td>5. Click <strong>Save</strong> to update Port details</td>
<td><img src="image5.png" alt="Screenshot 5" /></td>
</tr>
<tr>
<td>6. Success message is shown for successful Port update</td>
<td><img src="image6.png" alt="Screenshot 6" /></td>
</tr>
<tr>
<td>7. Click <strong>Return</strong> to view list of existing Ports found at Marker position</td>
<td><img src="image7.png" alt="Screenshot 7" /></td>
</tr>
</tbody>
</table>
## B.1.4. Delete Port at position

<table>
<thead>
<tr>
<th><strong>Step</strong></th>
<th><strong>Screenshot</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Marker on map</td>
<td><img src="image1.png" alt="Map with Marker" /></td>
</tr>
<tr>
<td>2. Check list of existing Ports</td>
<td><img src="image2.png" alt="Port List" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Port ID to view Port details</td>
<td><img src="image3.png" alt="Port Details" /></td>
</tr>
<tr>
<td>4. Click <strong>Delete</strong></td>
<td><img src="image4.png" alt="Delete Confirmation" /></td>
</tr>
<tr>
<td>5. When prompted to confirm deletion (along with any associated Paths), click <strong>OK</strong> to proceed or <strong>Cancel</strong> to abort</td>
<td><img src="image5.png" alt="Confirmation Dialog" /></td>
</tr>
<tr>
<td>6. Success message is shown for successful Port deletion</td>
<td><img src="image6.png" alt="Success Message" /></td>
</tr>
<tr>
<td>7. Click <strong>Return</strong> to view list of existing Ports found at Marker position</td>
<td><img src="image7.png" alt="Return Button" /></td>
</tr>
</tbody>
</table>
### B.1.5. Delete last Port at position

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Marker on map</td>
<td><img src="image1.png" alt="Screenshot" /></td>
</tr>
</tbody>
</table>
| 2. Check list of existing Ports  
3. Click on hyperlink of Port ID to view Port details | ![Screenshot](image2.png) |
| 4. Click **Delete** | ![Screenshot](image3.png) |
| 5. When prompted to confirm deletion (along with any associated Paths), click **OK** to proceed or **Cancel** to abort | ![Screenshot](image4.png) |
| 6. Success message is shown for successful Port deletion | ![Screenshot](image5.png) |
### B.2. Path management

#### B.2.1. Add first Path between Markers

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required origin Marker on map</td>
<td><img src="image1" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Select required Port as origin</td>
<td><img src="image2" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3. Click required destination Marker on map</td>
<td><img src="image3" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Select required Port as destination</td>
<td><img src="image4" alt="Screenshot 4" /></td>
</tr>
</tbody>
</table>
| 5. Input Path details  
  a. Input **Description** of Path  
  b. No need to input **Difficulty** as its value is evaluated automatically | ![Screenshot 5](image5) |
| 6. Click **Save** to create bi-directional Path records | ![Screenshot 6](image6) |
### B.2.2. Add overlapping Path

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required origin Marker on map</td>
<td><img src="image1" alt="Screenshot" /></td>
</tr>
<tr>
<td>2. Select required Port as origin</td>
<td><img src="image2" alt="Screenshot" /></td>
</tr>
<tr>
<td>3. Click required destination Marker on map</td>
<td><img src="image3" alt="Screenshot" /></td>
</tr>
<tr>
<td>4. Select required Port as destination</td>
<td><img src="image4" alt="Screenshot" /></td>
</tr>
</tbody>
</table>

7. Success message is shown for successful Path entry
Step | Screenshot
---|---
5. Input Path details  
   a. Input **Description** of Path  
   b. No need to input **Difficulty** as its value is evaluated automatically  
6. Click **Save** to create bi-directional Path records

| 7. Success message is shown for successful Path entry |

**B.2.3. Add vertical Path**

Step | Screenshot
---|---
1. Click required Marker on map

| 2. Select required Port as origin |

...
<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Click same Marker on map</td>
<td><img src="image1" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Select required Port as destination</td>
<td><img src="image2" alt="Screenshot 4" /></td>
</tr>
</tbody>
</table>
| 5. Input Path details  
  a. Input **Description** of Path  
  b. No need to input **Difficulty** as its value is evaluated automatically | ![Screenshot 5](image3) |
| 6. Click **Save** | ![Screenshot 6](image4) |
| 7. Success message is shown for successful Path entry | ![Screenshot 7](image5) |
### B.2.4. View / Edit Path details

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Line on map</td>
<td><img src="image1.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>2. Check list of existing Paths</td>
<td><img src="image2.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Path ID to view Path details</td>
<td><img src="image3.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>4. Edit details of Path as necessary</td>
<td><img src="image4.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>5. Click <strong>Save</strong> to update Path details</td>
<td><img src="image5.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>6. Success message is shown for successful Path update</td>
<td><img src="image6.png" alt="Screenshot" /></td>
</tr>
</tbody>
</table>
### B.2.5. View vertical Path details

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Marker on map</td>
<td><img src="image1.png" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Check list of existing Paths</td>
<td><img src="image2.png" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Path ID to view Path details</td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Check Path details</td>
<td><img src="image4.png" alt="Screenshot 4" /></td>
</tr>
</tbody>
</table>
### B.2.6. Delete vertical Path

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Marker on map</td>
<td><img src="image1.png" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Check list of existing Paths</td>
<td><img src="image2.png" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Path ID (of either direction) to view Path details</td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Click <strong>Delete</strong> to remove bi-directional Path records</td>
<td><img src="image4.png" alt="Screenshot 4" /></td>
</tr>
</tbody>
</table>
### B.2.7. Delete overlapping Path

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Line on map</td>
<td><img src="image1.png" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Check list of existing Paths</td>
<td><img src="image2.png" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Path ID (of either direction) to view Path details</td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
</tbody>
</table>

5. Success message is shown for successful Path deletion
### B.2.8. Delete last Path between Markers

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Click required Line on map</td>
<td><img src="image1.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>2. Check list of existing Paths</td>
<td><img src="image2.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>3. Click on hyperlink of Path ID (of either direction) to view Path details</td>
<td><img src="image3.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>4. Click <strong>Delete</strong> to remove bi-directional Path records</td>
<td><img src="image4.png" alt="Screenshot" /></td>
</tr>
</tbody>
</table>
Appendix C. Operation instructions for user interface

C.1. Input origin and destination

C.1.1. Manually type Building / Area name

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Select <strong>Building/Area</strong> field for <strong>Origin</strong> section</td>
<td></td>
</tr>
<tr>
<td>2. Type in full or partial name of the required origin building or area</td>
<td></td>
</tr>
<tr>
<td>3. Ignore the displayed suggestion list and click any blank region</td>
<td></td>
</tr>
<tr>
<td>4. Options for the <strong>Location</strong> dropdown list are automatically loaded</td>
<td></td>
</tr>
<tr>
<td>5. To choose a location other than the default option, click the <strong>Location</strong> dropdown list</td>
<td></td>
</tr>
</tbody>
</table>
### Step

6. Click the required **Location** option from the dropdown list

7. Repeat steps 1-6 for inputting location in **Destination** section

8. Click **Submit** to submit query, or click **Reset** to re-input fields

### C.1.2. Select Building / Area by providing keyword

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><img src="image1.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>Select <strong>Building/Area</strong> field for <strong>Origin</strong> section</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td><img src="image2.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>Type in full or partial name of the required origin building or area</td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Screenshot</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>3. Select required building or area from the displayed suggestion list</td>
<td><img src="image-url" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Click any blank region</td>
<td><img src="image-url" alt="Screenshot 4" /></td>
</tr>
<tr>
<td>5. Options for the <strong>Location</strong> dropdown list are automatically loaded</td>
<td><img src="image-url" alt="Screenshot 5" /></td>
</tr>
<tr>
<td>6. To choose a location other than the default option, click the <strong>Location</strong> dropdown list</td>
<td><img src="image-url" alt="Screenshot 6" /></td>
</tr>
<tr>
<td>7. Click the required <strong>Location</strong> option from the dropdown list</td>
<td><img src="image-url" alt="Screenshot 7" /></td>
</tr>
<tr>
<td>8. Repeat steps 1-7 for inputting location in <strong>Destination</strong> section</td>
<td><img src="image-url" alt="Screenshot 8" /></td>
</tr>
</tbody>
</table>
### Step 9
Click **Submit** to submit query, or click **Reset** to re-input fields.

### Screenshot
![Search Route]

**Origin**
- Chong Yuet Ming Amenities Centre (2/F)
- Security post ▼

**Destination**
- Run Run Shaw Building (1/F)
- Lift lobby (1/F) ▼

Submit ▼ Reset ▼

---

#### C.1.3. Select Building / Area directly from suggestion list

### Step 1
Click **Building/Area** field for **Origin** section.

### Screenshot
![Search Route]

**Origin**
- Building/Area Locate ▼
- Location ▼

**Destination**
- Building/Area Locate ▼
- Location ▼

Submit ▼ Reset ▼

### Step 2
Select required building or area from the displayed suggestion list.

### Screenshot
![Search Route]

**Origin**
- Building/Area Locate ▼
- Location ▼

**Destination**
- Building/Area Locate ▼
- Location ▼

Submit ▼ Reset ▼

### Step 3
Click any blank region.

### Screenshot
![Search Route]

**Origin**
- Building/Area Locate ▼
- Location ▼

**Destination**
- Building/Area Locate ▼
- Location ▼

Submit ▼ Reset ▼
<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>Options for the <strong>Location</strong> dropdown list are automatically loaded.</td>
</tr>
<tr>
<td>5.</td>
<td>To choose a location other than the default option, click the <strong>Location</strong> dropdown list.</td>
</tr>
<tr>
<td>6.</td>
<td>Click the required <strong>Location</strong> option from the dropdown list.</td>
</tr>
<tr>
<td>7.</td>
<td>Repeat steps 1-6 for inputting location in <strong>Destination</strong> section.</td>
</tr>
<tr>
<td>8.</td>
<td>Click <strong>Submit</strong> to submit query, or click <strong>Reset</strong> to re-input fields.</td>
</tr>
</tbody>
</table>
C.1.4. Detect Building / Area via geolocation

<table>
<thead>
<tr>
<th><strong>Step</strong></th>
<th><strong>Screenshot</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Prerequisites</strong></td>
<td><img src="image1.png" alt="Search Route" /></td>
</tr>
<tr>
<td>• Geolocation support on browser</td>
<td><strong>Origin</strong> Geolocate</td>
</tr>
<tr>
<td>• Secure connection (HTTPS)</td>
<td><strong>Building/Area</strong></td>
</tr>
<tr>
<td>• Internet connection without Proxy</td>
<td><strong>Location</strong></td>
</tr>
<tr>
<td>1. Click <strong>Geolocate</strong> button for <strong>Origin</strong> section</td>
<td><img src="image2.png" alt="Search Route" /></td>
</tr>
<tr>
<td>2. Value of <strong>Building/Area</strong> field and options for the <strong>Location</strong> dropdown list are automatically loaded</td>
<td><strong>Origin</strong> Geolocate</td>
</tr>
<tr>
<td>3. To choose a location other than the default option, click the <strong>Location</strong> dropdown list</td>
<td><strong>Building/Area</strong> Knowles Building</td>
</tr>
<tr>
<td>4. Click the required <strong>Location</strong> option from the dropdown list</td>
<td><strong>Destination</strong> Geolocate</td>
</tr>
<tr>
<td>5. Repeat steps 1-4 for inputting location in <strong>Destination</strong> section</td>
<td><strong>Building/Area</strong> Knowles Building</td>
</tr>
<tr>
<td>6. Click <strong>Submit</strong> to submit query, or click <strong>Reset</strong> to re-input fields</td>
<td><img src="image3.png" alt="Search Route" /></td>
</tr>
</tbody>
</table>
C.2. View search result

C.2.1. Web view

<table>
<thead>
<tr>
<th>Screenshot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview</strong></td>
<td></td>
</tr>
</tbody>
</table>
| ![Map view](image.png) | - Link to map view of route suggestion  
- Estimation of journey duration total  
- Origin  
- Intermediate stop(s)  
- Destination |

| Stop description | |
|------------------| |
| 2. 1 min 40 sec  | |
| *Entrance at Loading bay.* Chow Yei Ching Building (G/F) | - Duration to travel from previous stop  
- Type and name of stop  
- Area of stop |

C.2.2. Map view

<table>
<thead>
<tr>
<th>Screenshot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Map view" /></td>
<td></td>
</tr>
<tr>
<td>Screenshot</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **Marker** | • Origin  
• Intermediate stop  
• Destination  

**InfoWindow display**  
Click on a Marker to show stop description |
| **PolyLine** |  
(Thin solid grey line) Indoor path (Abstraction without actual path projection)   
(Thick solid red line) Ascent   
(Thick solid green line) Descent   
(Thick line of arbitrary solid colour) Horizontal path (Varying colour by gradient) |
| **InfoWindow** | • Stop number, type and name  
• Stop area  
• Buttons for panning between stops  
  o [«]: Previous stop  
  o [»]: Next stop |
List of figures

Figure 1 – Part of the map of HKU Main Campus, where passages involving steps are highlighted ........................................................................................................................................................................4
Figure 2 – Passages highlighted in blue are indistinguishable by levels ........................................5
Figure 3 – Floor numbers of respective buildings with connection to the Upper University Street ..................................................................................................................................................................................5
Figure 4 – Project architecture for final deliverable, where grey boxes are items pending development ........................................................................................................................................................................8
Figure 5 – Conceptual model of the Engineering buildings as a connected graph, with (a) Building entrances and junctions in orange, (b) buildings in red, and (c) paths in blue .................9
Figure 6 – Accessible paths in blue connects buildings of altitudes of 91m (Upper University Street, above) and 85m (Lower University Street, below) respectively ...........................................12
Figure 7 – Map view of path management with existing markers, lines, and paths projected on a line ........................................................................................................................................................................13
Figure 8 – Desktop mode of the user interface for route search .................................................15
Figure 9 – Two-phase entry of location selection for origin and destination by typing name of building (top) and selecting location within a building (middle) ............................................16
Figure 10 – Text-based search result with intermediate stops ..............................................17
Figure 11 – Map-based search result with differing path colours and descriptions of locations, close-up at an intermediate stop .........................................................................................................................18
Figure 12 – Screenshot of Google Earth, with coordinates and altitude shown at bottom right ........................................................................................................................................................................21
Figure 13 – Location of Landscape Garden input with coordinates obtained from the previous figure ........................................................................................................................................................................22
Figure 14 – Screenshots for workflow of editing attributes of a location ................................23
Figure 15 – Screenshots, top-down from left, for workflow of adding path between two locations ........................................................................................................................................................................24

List of tables

Table 1 – Scope of the project deliverable in two stages ...............................................7
Table 2 – Adaptation of Polygon Movement as mapping scheme for the project, with comparison ........................................................................................................................................................................10
Table 3 – Comparison of Dijkstra and A* Search pathfinding algorithms .....................14
Table 4 – Progress of project tasks ..................................................................................25