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 has been 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. The application has been developed for both web and Android platforms, and allows users to query for possible route suggestion according to the input origin and destination.

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 deliverables of our project have been 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.

Usage of third-party frameworks and libraries

Several development frameworks and libraries have been imported for implementing the wheelchair navigation system:

- Google Maps JavaScript API
- jQuery library, version 3.1.1
- AngularJS framework, version 1.6.2
- Bootstrap framework, version 3.3.7
- Apache Cordova project / PhoneGap framework, version cli-6.5.0

We wish to thank the developers of these frameworks and libraries, for their works facilitated implementation of the project deliverable in more simplified manners.
# Table of Contents

Acknowledgements .......................................................................................................................... 1  
Usage of third-party frameworks and libraries ............................................................................. 1  
Terminology .................................................................................................................................... 3  
1. Introduction .................................................................................................................................. 3  
2. Background .................................................................................................................................. 4  
3. Motivations and objectives .......................................................................................................... 6  
4. Project scope ................................................................................................................................ 6  
5. Methodology and deliverables ..................................................................................................... 7  
6. Challenges and limitations .......................................................................................................... 22  
7. Progress ....................................................................................................................................... 24  
8. Review and possible future developments ................................................................................. 25  
9. Conclusion .................................................................................................................................... 26  
10. References ................................................................................................................................... 26  
Appendix A. Workflow for administration interface operations................................................... 28  
Appendix B. Operation instructions for Android application ......................................................... 43  
List of figures .................................................................................................................................... 46  
List of tables ...................................................................................................................................... 46
Terminology

The following terms are assigned specific meanings in this 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 access to entrance of a facility, 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 **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.

1. Introduction

To promote usage of accessible facilities at HKU, the project team has delivered a wheelchair navigation system which suggests accessibility-friendly routes within the University of Hong Kong campus. The deliverables of this project are (1) an online service to provide route suggestion based on user queries, and (2) 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 a single-page web application that is portable to the Android system. It can be accessed via web browsers on desktop computers and mobile devices, or directly run as an application on the Android system. Users can arbitrary specify locations as origin and destination as long as such locations have been identified in the system. Route suggestion is presented in a text-based sequence of intermediate stops and a graphical route shown on Google Maps. Users are informed about unavailability of accessible route if a search is unsuccessful.
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)

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](image1.jpg)

![Figure 3 – Floor numbers of respective buildings with connection to the Upper University Street](image2.jpg)

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 aimed to introduce a wheelchair navigation system whose service cover the University of Hong Kong campus, which enables wheelchair users to plan journeys via web browser or Android application. As students who participated in this project, we hoped to achieve the following:

- Manage spatial data on relational database
- Model the University of Hong Kong campus into connected graphs
- Apply pathfinding algorithms on the graph to obtain best path suggestions
- Develop text- 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 hoped that they would 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 had been carried out incrementally in two stages, corresponding to two semesters respectively. Refer to Table 1 for the comparison of features available in each deliverable.
Table 1 – Scope of the project deliverable in two stages

<table>
<thead>
<tr>
<th>Semester 1 (Intermediate deliverable)</th>
<th>Semester 2 (Final deliverable) *</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The service coverage spans exclusively the Main Campus of the University of Hong Kong</td>
<td>• The service coverage is extended to Sassoon Road Campus of the University of Hong Kong</td>
</tr>
<tr>
<td>• Only accessible paths between buildings and open areas are evaluated.</td>
<td>• Accessible paths within buildings are evaluated.</td>
</tr>
<tr>
<td>• Campus locations are identifiable to the precision of buildings and open areas.</td>
<td>• Indoor locations such as rooms are identifiable.</td>
</tr>
<tr>
<td>• Users can input origins and destinations via dropdown lists of available locations or through geolocation only.</td>
<td>• Users can arbitrarily pinpoint locations on map as origins and destinations.</td>
</tr>
<tr>
<td>• The spatial database is maintained exclusively by the project team.</td>
<td>• Users can report updates on availability of accessible facilities.</td>
</tr>
<tr>
<td>• The service is available as a web application for desktop computers and mobile devices.</td>
<td>• The service is also made available as an Android application.</td>
</tr>
</tbody>
</table>

5. Methodology and deliverables

5.1. Conceptual design

5.1.1. Project requirements

The following parties are involved for the wheelchair navigation system:

- Project team: they are responsible for development of functions required by the system, and maintenance of geographical data required for constructing accessible routes.
- Target users: these may be wheelchair users or any user of the system, who request for accessible route suggestion by providing required origin and destination.

The project deliverable has been constructed as web application with the following functional requirements:

- Administration interface: 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.
- User interface: 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.
5.1.2. Project architecture

Figure 4 shows an overview of the project architecture. Design and implementation of the features are discussed in the following sections.

The lowermost layer represents the database, which stores raw geographical data of the University of Hong Kong campus and its routing information. Separate tables were defined for locations, accessible paths connected by any two locations, temporary records of accessible routes and search history.

The layer above database represents its routines. Two common approaches exist for retrieving data from database: calling routines that are executed on database server, or executing SQL statements from web server. It had been observed that the former approach would be more efficient when large amounts of data had to be evaluated for one process, hence the introduction of routines to the database.

The middle layer represents functional requirements for the wheelchair navigation system. Separate functions are specified for the project team and for target users respectively, and are implemented on an application server.

The layer above functional requirements represents the web interface. Both administration and user interfaces are required in the form of web applications in order to be accessible via web browsers.

The top layer represents the Android application. It is an addition to the web-based user interface and is required for native execution on smartphones that use the Android platform. The application requires Internet access for frequent communication with the application server.
5.2. Choice of development platforms

The project was developed as a web application that allows maintenance of geographical data by the project team through web interface, and query for accessible routes by target users through both web interface and Android application.

For the convenience of spontaneous development and maintenance, the server-side scripts of the project deliverable are designated to 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.

Both administration and user interfaces are expected to be accessed via web browsers. Target users also have the choice to access the wheelchair navigation system via the Android application client.

While Android OS is the target platform of the client application for the wheelchair navigation system, it is a challenging task and causes overhead preparation to develop the Android application from scratch, since imported libraries and completed codes implemented for the web application of the system cannot be effectively converted to Java codes running on Android platform. In order to reuse web-based codes, Apache Cordova was chosen to serve as a conversion engine that exports the web application to Android executable. For this development platform, the input must be a single-page application written in pure client-side scripts, i.e. HTML, CSS and JavaScript.

The client application is accessible on both modern web browsers with JavaScript support (tested on Google Chrome 57) and Android platform (tested on Android 4.0.4).

The project codes for the deliverable were 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 originally intended for execution on web browsers. Several libraries and frameworks have been imported to render map content and make the application more suitable for mobile platforms:
  o Google Maps JavaScript API. This library is required for displaying annotated digital map on user interface.
  o jQuery library. Functions of this library allow simplified coding dedicated to manipulating web-based elements on user interface. It is also required by the Bootstrap framework.
  o AngularJS framework. This framework is useful for automatically controlling content displayed on user interface according to application state, without the need of having codes that explicitly instruct the manner that certain page content is to be displayed.
  o Bootstrap framework. This framework serves as substitute to a dedicated CSS stylesheet for the application. It also features responsive design that is suitable for screens of mobile devices.

5.3. Geographical model

The geographical environment of the Main Campus was initially modelled into a connected graph as shown in Figure 5 Error! Reference source not found., 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.

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.

### 5.3.1. 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><strong>Polygon Movement scheme</strong></th>
<th><strong>Project adaptation</strong></th>
</tr>
</thead>
</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.

5.3.2. 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.

5.3.3. 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.

5.3.4. Data collection on campus
Most of the locations recorded in the database had been proposed after evaluation on site. This helped ensure that the process of data collection would not accidentally propose accessible paths that involved steps or obstacles.

The project team had consulted service departments of the University for the standards of recognising accessible paths. In general, accessible paths require gentle gradients, considerable width that provides sufficient space between a wheelchair and side boundaries of the path, and absence of vehicular traffic.
At the Main Campus, accessible paths can be easily proposed due to their abundance, hence many locations within the Main Campus are connected by one or more accessible routes.

On the other hand, the Sassoon Road Campus is situated at two sides of Sassoon Road with steeper gradient. While it is apparent that this gradient is steeper than normal standards would allow, the step-free paths along the sides of Sassoon Road are the only available ones in the proximity of the Campus – most neighbouring paths are staircases, and some locations are completely inaccessible by wheelchair which have already been excluded from the system – thus making it inevitable to propose these as accessible paths.

All the geographical data collected from campus are input to the system via the administration interface and database routines. The connection of locations and accessible paths forms one or more connected graphs that are three-dimensional and separable by altitudes (Figure 6). This data representation of campus makes possible the utilisation of pathfinding algorithm, such as Dijkstra’s algorithm, for planning accessible routes between campus locations.

Figure 6 – Accessible paths in blue connects buildings of altitudes of 91m (Upper University Street, above) and 85m (Lower University Street, below) respectively
5.4. 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.

5.5. 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.

5.5.1. 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.

5.5.2. Map view

The map view allows data entry and maintenance directly on Google Maps (Figure 7.Error! Reference source not found.), 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.
Figure 7 – Map view of path management with existing markers, lines, and paths projected on a line

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.

5.6. 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:

Table 3 – Comparison of Dijkstra and A* Search pathfinding algorithms

<table>
<thead>
<tr>
<th></th>
<th>Dijkstra’s Algorithm (directional, binary heap)</th>
<th>A* Search Algorithm (directional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature</td>
<td>Greedy Algorithm</td>
<td>Heuristic Search</td>
</tr>
<tr>
<td>Correctness</td>
<td>Assured</td>
<td>Local minimum may occur</td>
</tr>
<tr>
<td>Time Complexity</td>
<td>O((E + V) log V) On binary heap</td>
<td>O(E + V) On binary heap</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>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 University of Hong Kong campus was of interest, 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.

5.7. Android application client

An Android application client had been developed as final deliverable for the project, which provided a user interface with better presentation and more intuitive operations for target users. The Android application shares most of the features available from the web-based user interface.

5.7.1. Development platform

The project team initially intended to develop the Android application using its native development tools, namely Android Studio, SDK and related libraries. After attempts to install these tools, however, we concluded that the performance of computing resources deployed by ourselves were insufficient for running these tools, hence obstructed our activities to develop Android application on this development platform.

The project team was also concerned about the efficiency of project conversion from web application to Android application. For the web application, three to four webpages were written in PHP scripts supported with JavaScript functions, which brought uncertainty as to how similar functional behaviour could be replicated on the Android platform. Apache Cordova, an open source mobile development framework, was
eventually chosen as the development platform for the Android application. This framework enables web developers to convert single-page web applications to cross-platform mobile applications. To fit the web application in this framework, we converted it to a single-page version that did not involve any server-generated HTML codes.

5.7.2. Presentation of user interface

In the intermediate deliverable, styles applied to the web application were primitive and only served the purpose of allowing target users to identify functions from different sections on the user interface (Figure 8). The side panel did not provide adequate information to guide users through different stages of user operations, and layout of the webpage was not responsive in design, making it difficult to fit in screens of varying sizes.

The web application also exhibited problems for combined use of PHP and JavaScript for generating HTML codes (Figure 9). Although functions with such coding practice would generally work in expected manner on static webpages, it would be impracticable to implement the same functions for the Apache Cordova platform, due to the constraint that only client-side codes are supported. Server-side values can only be retrieved via AJAX requests.
Bootstrap framework had been used for providing neat visual presentation to users of the wheelchair navigation system, making it intuitive to understand roles of different parts of the Android application. The assignment of page elements to classes also made the styling process simpler. The page itself is therefore responsive as well (Figure 10).

The Android application being a single-page application introduced the tendency to implementing all contents of the application and execution logic in one HTML file and one JavaScript file. AngularJS framework had been used for enforcing the display logic of page elements according to application state. For example, the application has three screens – main page, route planner and about page – for which only one screen should be displayed at a time. The AngularJS directives attributed to selected page elements.
decide which page to be displayed. This serves as a reliable simulation to swapping activities on Android applications.

5.7.3. Operations for specifying locations

In the web application for the navigation system, users can only specify origin and destination locations via geolocation or two-phase location input (Figure 11). The input form for location input had been reported to be difficult to use, as most users were unable to proceed from the first phase (building / area input box) to second phase (location dropdown list). Most users confused the Geolocation button with a hypothetical Filter button, while actually the only way to populate options in the location dropdown list was to focus off the input box. Some users also mistook the input box to be sufficient for the purpose of specifying a location, and would click Search straight away.

![Two-phase location input form](image.png)

*Figure 11 – Two-phase location input form*

The Android application improved the process of location input by introducing step-by-step verbal instructions, enhanced workflow for two-phase input, and a third option to pinpoint desired origin and destination on map.
Figure 12–Android application with three sections
(top) location input form, (middle) map canvas, (bottom) text instructions

- Step-by-step instructions are sequentially shown according to the stage of user input, and is regulated by AngularJS directives. Users are therefore informed of the missing information that is pending input.
- The enhanced two-phase input now combines both input box and dropdown list in one field, which makes it easier to distinguish between origin and destination input fields (Figure 12, Figure 13). The Search button is disabled by default when the input form is partially completed, and is enabled only when both input fields show dropdown list with origin and destination options specified.
- The option to pinpoint endpoints by clicking on map is considered the most important feature enhancement to the wheelchair navigation system. While it still requires users to select exact locations as origin and destination from dropdown lists, users are able to relate geographical positions with location
names, making it far more intuitive than requiring knowledge of locations names before being able to input origin and destination locations.

Figure 13 – Location input form with dropdown list enabled (top) and pinpointing on map (middle, labelled by ‘S’)

Similar to the web application, the Android application actively updates positions of origin and destination markers as a user changes selection to origin and destination locations on dropdown lists. The only feature removed from the input form is the data list prompted during text input, as its values can only be generated from PHP scripts.

5.7.4. Visualisation of route suggestion

For each route suggestion, the web application showed text description and graphical path as separate information. Users also reported that the shift buttons shown on the message window above each intermediate stop along a route – for the purpose of moving the focus and message window to a previous or next stop – gave an impression of allowing users to pan the screen in horizontal directions (Figure 14).
In the Android application, message windows are no longer used for displaying route information. It had been observed that they might conflict with message windows that are displayed upon clicking on landmarks embedded on Google Maps. Navigation footers have instead been introduced, for displaying information regarding each intermediate stop. Buttons for panning to previous and next stops are explicitly labelled, which prevent users from interpreting these buttons as panning controls.

A Report Data Error button has been introduced to let users send email to the project team whenever they suspect incorrect information in a route suggestion. The button leads to a hyperlink for starting an email client, and carries origin, destination and query time into email content as technical information, which allows the project team to easily locate the query from journal records logged in the database (Figure 15).

Figure 14 – Route suggestions in graphical paths (middle) and text description with buttons for panning along stops and report data error (bottom)
The Google Maps canvas embedded on web application was designed to be destroyed and re-instantiated each time a new route search was requested. In the Android application, the same map canvas is used through the entire application execution cycle, and annotations on the map canvas are always systematically cleared before a new route search is requested, followed by adding new annotations to the same map canvas. This reduces memory usage on devices and facilitates effective variable re-usage.

6. Challenges and limitations

6.1. Maintenance of campus accessibility data

The original database schema designated separate tables for storing locations (otherwise known as Ports) and abstraction of lifts / buildings / open areas (otherwise known as Portals). While the database table for Ports was defined at an early stage, for which administration interface for Port management were rapidly developed, the table for Portals – with mandatory 1:1 relationship with Ports table – had been defined at a relatively late stage that made it difficult to integrate Portal management as a new function in the
administration interface. Database procedures had been developed for all operations on data manipulation.

As a result, all Ports can be maintained on web-based and map-based administration interfaces, while all Portals have to be maintained through calls to database routines. A few potential problems are observed from this current practice:

- The operations provided by the administration interface are not comprehensive. When a lift, a building or an open area has to be defined for abstraction of indoor paths, one or more routine calls must be executed from a database client.
- As each Portal record must have a corresponding Port record, the administration interface can inevitably access Port records associated with Portals. Accidental modification of Portal-based Port records can lead to unpredictable data error. Similar situation is found for accessible paths with or without connection to Portals.
- On the map-based administration interface, Portal-based Port markers are not distinguishable from Ports, and therefore identification of Portals is extremely difficult (Figure 17). Similar situation is found for indistinguishability between accessible paths with and without connection to Portals.

As possible measures to resolve the above issues, either the Portal entity had to be removed to prevent any potential data errors, or an administration interface had to be developed for the Portal entity, so that Portals would be made distinguishable from Ports.
6.2. Coding issues on Android application

The Android application, in the form of single-page web application, made heavy use of framework functions for implementing features concerning presentation and workflow. Several issues had been encountered during implementation:

- While the codes for webpage content and script execution logic had been implemented in separate files, which was a desirable outcome, contents of these two files could not be further split into smaller components for easier code maintenance, because such separation leads to execution errors. Given a small-scale application developed for this project, it is maintainable at low difficulty, but the difficulty always increases with scale of application.

- Some lines of code written for the AngularJS framework do not exhibit expected behaviour after execution (Figure 17). In particular, an assignment of variable value to an ng-model variable does not immediately get reflected in an ng-bind directive for an HTML element.

```javascript
focus = $scope.$apply(function() {
    $scope.raw.focus = next = 0;
    return $scope.raw.focus;
});
```

Figure 17 – Variable assignment that could not be captured by AngularJS, if not enclosed by the $apply block

While many AngularJS scripts seen in learning resources appeared to be implemented in single file, there exist alternatives to maintaining webpage contents as separate files by routing to pages through AngularJS. This may serve as an improvement to maintainability of web applications in future developments.

As for the issue with unexecuted assignments on AngularJS ng-model variables, after some research on reports of similar issues [9][10], it could be learned from the design of AngularJS that updates to ng-bind values are triggered when a monitor function of the library discovers an explicit change to ng-model value. While the cause of lack of change for the ng-model value in this Android application remains unknown, a workaround of explicitly applying the ng-model change can be done that guarantees an updated value for the ng-bind directive.

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></td>
<td>Anticipated</td>
</tr>
<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><strong>Deliverables of Phase 1</strong></td>
<td>2 October 2016</td>
</tr>
<tr>
<td>• Detailed project plan</td>
<td></td>
</tr>
<tr>
<td>• Project webpage</td>
<td></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><strong>Deliverables of Phase 2</strong></td>
<td>28 December 2016</td>
</tr>
<tr>
<td>• Preliminary implementation</td>
<td></td>
</tr>
<tr>
<td>• Detailed interim report</td>
<td></td>
</tr>
<tr>
<td>• First presentation</td>
<td></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><strong>Deliverables of Phase 3</strong></td>
<td></td>
</tr>
<tr>
<td>• Finalised tested implementation</td>
<td>16 April 2017</td>
</tr>
<tr>
<td>• Final report</td>
<td></td>
</tr>
<tr>
<td>• Final presentation</td>
<td></td>
</tr>
<tr>
<td>• Project presentation</td>
<td></td>
</tr>
</tbody>
</table>

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

### 8. Review and possible future developments

While it had been a successful attempt to develop a simple web application with Android application client that provides route searching service for wheelchair users, this service is still primitive and lacks more powerful functions for better revealing campus accessibility information to target users.

The accessible facilities found at both Main Campus and Sassoon Road Campus are observed to be in sharp contrast:

- Nearly all buildings and their interiors at the Main Campus are accessible via lifts and ramps. For the inaccessible floors of these buildings, many of them do not have actively used rooms and therefore do not require extra accessible facilities.
- Most buildings at the Sassoon Road Campus are connected by step-free footpaths at sides of Sassoon Road. However, the drawbacks are steep gradient and the necessity
to cross the road for reaching some buildings. In the worst case, Sassoon Road is often busy at daytime. Also, several buildings are not equipped with lifts, such that rooms located at higher floors are unreachable.

While many campus locations are reachable by vehicle, it would be more convenient if wheelchair users could safely travel on their own to travel within the campus. It is therefore hoped that the university would consider introducing wheelchair-friendly facilities to buildings and their floor levels that are unreachable by wheelchair users, so as to make the campus genuinely and completely accessible.

Most web programming technologies had been put into appropriate use for this project. The same becomes instead a development constraint when the deliverable is an Android application. The conversion service provided by Apache Cordova is just a workaround for avoiding the motivation to learn Android programming, so that many features equipped with Android smartphones have not been put into effective utilisation. It is hoped that our future projects on Android development would no longer rely on web programming skills, but make actual use of Android programming skills instead.

9. Conclusion

Development of the wheelchair navigation system is now complete as a final deliverable for both web and Android platforms. Although the service only provides a routing feature, it is guaranteed that a successful search results in successful presentation of path sequence and intermediate stops in text and graphical information. The service now covers the Main Campus and Sassoon Road Campus of the University of Hong Kong, with some indoor accessible paths made available in the system.

Above all, this project aims to encourage wheelchair users to travel within campus with greater convenience by effectively utilising the accessible facilities that are readily available at the University of Hong Kong.

10. References


# Appendix A. Workflow for administration interface operations

## A.1. Port management

### A.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.png" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Click <strong>Add</strong> to create new Port</td>
<td><img src="image2.png" alt="Screenshot 2" /></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 3](image3.png) |
| 4. Click **Save** | ![Screenshot 4](image4.png) |
| 5. Success message is shown for successful Port entry  
  6. Click **Return** to view list of existing Ports found at Marker position | ![Screenshot 5](image5.png) |
### A.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</td>
<td><img src="image2.png" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3. Click <strong>Add</strong> to create new Port at same position</td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Input Port details (Refer to Step 3, A.1.1)</td>
<td><img src="image4.png" alt="Screenshot 4" /></td>
</tr>
<tr>
<td>5. Click <strong>Save</strong></td>
<td><img src="image5.png" alt="Screenshot 5" /></td>
</tr>
<tr>
<td>6. Success message is shown for successful Port entry</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>
## A.1.3. View / Edit Port details

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step</strong></td>
<td><strong>Screenshot</strong></td>
</tr>
<tr>
<td>1.</td>
<td>Click required Marker on map</td>
</tr>
<tr>
<td>2.</td>
<td>Check list of existing Ports</td>
</tr>
<tr>
<td>3.</td>
<td>Click on hyperlink of Port ID to view Port details</td>
</tr>
<tr>
<td>4.</td>
<td>Edit details of Port as necessary</td>
</tr>
<tr>
<td>5.</td>
<td>Click <strong>Save</strong> to update Port details</td>
</tr>
<tr>
<td>6.</td>
<td>Success message is shown for successful Port update</td>
</tr>
<tr>
<td>7.</td>
<td>Click <strong>Return</strong> to view list of existing Ports found at Marker position</td>
</tr>
</tbody>
</table>
### A.1.4. Delete Port at position

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Click required Marker on map</td>
</tr>
<tr>
<td>2.</td>
<td>Check list of existing Ports</td>
</tr>
<tr>
<td>3.</td>
<td>Click on hyperlink of Port ID to view Port details</td>
</tr>
<tr>
<td>4.</td>
<td>Click <strong>Delete</strong></td>
</tr>
<tr>
<td>5.</td>
<td>When prompted to confirm deletion (along with any associated Paths), click <strong>OK</strong> to proceed or <strong>Cancel</strong> to abort</td>
</tr>
<tr>
<td>6.</td>
<td>Success message is shown for successful Port deletion</td>
</tr>
<tr>
<td>7.</td>
<td>Click <strong>Return</strong> to view list of existing Ports found at Marker position</td>
</tr>
</tbody>
</table>
### A.1.5. Delete last Port at position

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><img src="image1.png" alt="Screenshot 1" /> Click required Marker on map</td>
</tr>
<tr>
<td>2.</td>
<td><img src="image2.png" alt="Screenshot 2" /> Check list of existing Ports</td>
</tr>
<tr>
<td>3.</td>
<td><img src="image3.png" alt="Screenshot 3" /> Click on hyperlink of Port ID to view Port details</td>
</tr>
<tr>
<td>4.</td>
<td><img src="image4.png" alt="Screenshot 4" /> Click <strong>Delete</strong></td>
</tr>
<tr>
<td>5.</td>
<td><img src="image5.png" alt="Screenshot 5" /> When prompted to confirm deletion (along with any associated Paths), click <strong>OK</strong> to proceed or <strong>Cancel</strong> to abort</td>
</tr>
<tr>
<td>6.</td>
<td><img src="image6.png" alt="Screenshot 6" /> Success message is shown for successful Port deletion</td>
</tr>
</tbody>
</table>
A.2. Path management

A.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.png" alt="Step 1" /></td>
</tr>
<tr>
<td>2. Select required Port as origin</td>
<td><img src="image2.png" alt="Step 2" /></td>
</tr>
<tr>
<td>3. Click required destination Marker on map</td>
<td><img src="image3.png" alt="Step 3" /></td>
</tr>
<tr>
<td>4. Select required Port as destination</td>
<td><img src="image4.png" alt="Step 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 | ![Step 5](image5.png) |
| 6. Click **Save** to create bi-directional Path records | ![Step 6](image6.png) |
### A.2.2. Add overlapping Path

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

7. Success message is shown for successful Path entry
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

### A.2.3. Add 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" /></td>
</tr>
<tr>
<td>2. Select required Port as origin</td>
<td><img src="image2.png" alt="Screenshot" /></td>
</tr>
<tr>
<td>Step</td>
<td>Screenshot</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>3.</td>
<td>Click same Marker on map</td>
</tr>
<tr>
<td>4.</td>
<td>Select required Port as destination</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 |
| 6.   | Click **Save** |
| 7.   | Success message is shown for successful Path entry |
### A.2.4. View / Edit Path details

<table>
<thead>
<tr>
<th><strong>Step</strong></th>
<th><strong>Screenshot</strong></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 to view Path details</td>
<td><img src="image3.png" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4. Edit details of Path as necessary</td>
<td><img src="image4.png" alt="Screenshot 4" /></td>
</tr>
<tr>
<td>5. Click <strong>Save</strong> to update Path details</td>
<td><img src="image5.png" alt="Screenshot 5" /></td>
</tr>
<tr>
<td>6. Success message is shown for successful Path update</td>
<td><img src="image6.png" alt="Screenshot 6" /></td>
</tr>
</tbody>
</table>
## A.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>
### A.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>
### Step 5. Success message is shown for successful Path deletion

### A.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="image" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2. Check list of existing Paths</td>
<td><img src="image" 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="image" alt="Screenshot 3" /></td>
</tr>
</tbody>
</table>
### A.2.8. Delete last Path between Markers

<table>
<thead>
<tr>
<th>Step</th>
<th>Screenshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><img src="image1" alt="Screenshot 1" /></td>
</tr>
<tr>
<td>2.</td>
<td><img src="image2" alt="Screenshot 2" /></td>
</tr>
<tr>
<td>3.</td>
<td><img src="image3" alt="Screenshot 3" /></td>
</tr>
<tr>
<td>4.</td>
<td><img src="image4" alt="Screenshot 4" /></td>
</tr>
</tbody>
</table>

1. **Click required Line on map**
2. **Check list of existing Paths**
3. **Click on hyperlink of Path ID (of either direction) to view Path details**
4. **Click **Delete** to remove bi-directional Path records**
A.3. Portal management

Since no administration interface is available for managing Portals, database routines must be called instead. Portals must be maintained using designated procedures.

A.3.1. Create Portal
To create a Portal, execute ‘CALL ADDPORTAL (<1>, <2>);’, where:

- <1> is the name of Portal
- <2> is the type of Portal, either ‘Building’ or ‘Open Area’

The ID of Portal is returned for successful execution.

A.3.2. Add Port to Portal
To connect a Port to Portal, execute ‘CALL ADDPORTTOPORTAL (<1>, <2>);’, where

- <1> is the ID of Portal
- <2> is the ID of Port

A.3.3. Update Portal details
To change particulars of a Portal, execute ‘CALL UPDATEPORTAL (<1>, <2>, <3>);’, where

- <1> is the ID of Portal
- <2> is the name of Portal
- <3> is the type of Portal, either ‘Building’ or ‘Open Area’

A.3.4. Remove Port from Portal
To remove a Port from Portal, execute ‘CALL REMOVEPORTFROMPORTAL (<1>, <2>);’, where

- <1> is the ID of Portal
- <2> is the ID of Port
A.3.5. Delete Portal
To delete a Portal, execute ‘CALL DELETEPORTAL (<1>);’, where

- <1> is the ID of Portal

Appendix B. Operation instructions for Android application

B.1. Overview

B.2. Specify Origin and Destination

B.2.1. By typing keyword
1. Type keyword in Origin text box
2. Press [>>]
3. When message ‘—Select origin—’ is shown, open dropdown list
4. Select location as Origin
5. Repeat steps 1-4 for Destination input
6. Press Search

B.2.2. By pinpointing on map
1. Press position on map
2. When message ‘—Select origin—’ is shown, open dropdown list
3. Select location as **Origin**
4. Repeat steps 1-3 for **Destination** input
5. Press **Search**

**B.2.3. By geolocation**

1. Press 📢 for **Origin**
2. When message ‘—Select origin—’ is shown, open dropdown list
3. Select location as **Origin**
4. Repeat steps 1-3 for **Destination** input
5. Press **Search**

**B.2.4. Additional operations**

- **Map: Main Campus / Sassoon Road Campus** : Pan map to selected campus
- **Back** : Return to main screen
- **Reset** : Clear all input values and search results

**B.3. View search result**

**B.3.1. Overview**
B.3.2. Operations

- **Markers** (marker icons): Click to show information for intermediate stop
- **Report Data Error**: Submit report for inaccurate information in route suggestion
- **< Prev / Next >**: Pan to previous / next intermediate 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 .......................................................... 10
Figure 6 – Accessible paths in blue connects buildings of altitudes of 91m (Upper University Street, above) and 85m (Lower University Street, below) respectively ............................................................. 13
Figure 7 – Map view of path management with existing markers, lines, and paths projected on a line .................................................................................................................................................. 14

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 ............................................................................................................................................................. 11
Table 3 – Comparison of Dijkstra and A* Search pathfinding algorithms .............................................. 15
Table 4 – Progress of project tasks ........................................................................................................... 25