LUMA-GIS Thesis nr 16

Nairobi City Journey Planner

An online and a Mobile Application

David Kanyari

2011
Dept. of Earth and Ecosystem Sciences
Division of Physical Geography and Ecosystem Analysis
Centre for Geographical Information Systems
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

A Master thesis presented to

Department of Physical Geography and Ecosystem Analysis

Centre for Geographical Information Systems

of

by

David Kanyari

in partial fulfilment of the requirements

for the degree of Master in Geographical Information Science

Supervisors:

Irene Rangel, Lund University

ABSTRACT

Public transport plays a vital role in enabling people to move from one place to another especially in urban areas. The provision of information related to public transportation enables travelers to pre-plan their journeys and even save time that would have been spent asking around or looking for this information from other sources. The city of Nairobi has a population of about four million people, most of whom rely on public transportation commonly known as *matatus* as the preferred mode of transport to work and to school. The public transport sector in Nairobi lacks proper management and organization. Currently, there exists no reliable source of public transport information that is easily accessible to public.

This project aimed at developing a journey planner application: an online and a mobile application to assist travellers to preplan their journey by giving them a platform where matatu information and routes could be accessed. This information included bus information, route followed from origin to destination, trip duration, total trip distance and a route map.

The application utilizes matatu routes data, Dijkstra routing algorithm, relational database and a Java ME mobile application. Both online and mobile application shares a common database and a routing algorithm hosted on a web server.

After the application was developed, user tests were conducted. The users expressed the desire to enrich the application by adding more stations and more matatu routes as part of future improvement. The users regarded the application as a vital tool that makes their lives easy in commuting around the city.

Keywords: Dijkstra algorithm, GIS, Matatu, Routing, Nairobi public transport, Journey planner.

ACKNOWLEDGMENTS

It is a pleasure to thank those who made this thesis possible such as my supervisor Irene Rangel who helped me with technical guidance, ideas and most importantly in the useful critics in regard to the development of the final application.

My appreciation also goes to professional input from Petri Vesikivi and Kari Salo from Helsinki, Finland. Their assistance went a long way in the programming, project brainstorming and guidance in development of the mobile application of journey planner.

Many thanks to the Kenyan organizations and mapping institutions that assisted me with matatu routes information, research material and the road data. Without this data, the project would have been difficult to carry out and to realize.

Finally, I thank all the participants who took part in the user testing of the application and for their useful feedback.

CONTENTS

ABSTRACT .... **i**

ACKNOWLEDGMENTS .... **ii**

CONTENTS .... **iii**

LIST OF FIGURES .... **v**

LIST OF TABLES .... **vi**

LIST OF ABBREVIATIONS .... **vii**

1. INTRODUCTION **1**

1.1 Public Transportation in Nairobi 2
1.2 Towards Better Public Transportation System 3
1.3 Development of Journey planner Application 4
1.4 Statement of Purpose 4
1.5 Objectives of the Study 6
1.6 Organization of the Report 6

2. GIS AND ROUTING ALGORITHMS **8**

2.1 What is GIS? 8
2.1.1 GIS in transportation 8
2.1.2 Representing the world in GIS 9
2.2 Vector Data Models 10
2.2.1 Spaghetti data model 10
2.2.2 Topological data model 11
2.3 GIS and Network Analysis 13
2.3.1 Network data model 13
2.3.2 Vehicle routing within a network 14
2.4 Databases 15
2.4.1 Types of DBMS models 16
2.4.2 Using Databases 17
2.5 Web and Mobile device Technologies 17
2.5.1 Mobile Information Device Profile (MIDP) 18
2.5.2 Developing a Mobile phone application 18
2.5.3 Overview of MIDP User Interface 19
2.5.4 Data Interchange between Mobile Client and Web Server 20
2.6 Routing Algorithms 21
2.6.1 Dijkstra Algorithm 22
2.6.1.1 How Dijkstra’s Algorithm Works 23
2.6.2 A* search Algorithm 25
2.6.3 Comparison between Dijkstra’s and A* algorithms in road networks 25
2.6.4 Analysis of Algorithms 27
2.7 Web Mapping 28
2.7.1 Web map service technology 29
2.7.2 Google Maps API 29
2.7.3 KML 30

2.8 Application Testing ... 31
2.8.1 Conducting a usability test ... 31
2.8.2 User testing methods ... 32
2. Overview of Related or Previous work ... 34

3. METHODS AND MATERIALS ... 37

3.1 Project Definition and Conceptualization ... 37
3.2 Project Work Flow ... 38
3.3 Requirement Specifications ... 39
3.3.1 User stories ... 39
3.3.2 What does the user do? ... 40
3.4 System Architecture ... 41
3.5 Web -Based Journey Planner Application ... 42
3.5.1 Mobile Journey Planner Client ... 42
3.5.2 How does the Mobile Journey planner work? ... 43
3.5.3 Things to consider when designing the mobile journey planner client ... 43
3.6 Project Design ... 44
3.6.1 Data Acquisition and Preparation ... 44
3.6.2 Network Design and Routing Algorithm ... 45
3.6.3 Network Modelling ... 46
3.6.4 Database Modelling ... 48
3.6.5 Entity – Relations Model ... 48
3.7 Final Application ... 49

4. ANALYSIS AND DISCUSSION ... 54

4.1 The Demo version ... 54
4.2 Route Map ... 57
4.3 The Mobile Client ... 59
4.3.1 Using the Mobile Journey Planner ... 60
4.3.2 Choice of Data Interchange Format ... 62
4.4 Application Testing ... 65
4.4.1 Testing methods ... 66
4.4.2 Results of user feedback ... 67

5. CONCLUSION AND FUTURE WORK ... 71

References ... 73

Appendices ... 78

Appendix 1: Mobile client connecting to a web server ... 78
Appendix 2: Dijkstra algorithm class ... 80
Appendix 3: Adjacency matrix ... 82
Appendix 4: Google maps API example loading map on a web page ... 83
Appendix 5: Java ME mobile client ... 84
Appendix 6: Questionnaire ... 89
Appendix 7: A Collection of Nairobi Matatu Photos ... 93
Appendix 8: Using Social Media (Face book) for user feed back ... 96

LIST OF FIGURES

Figure 1: Modeling the real world in GIS.. 9
Figure 2: Spaghetti data model.. 10
Figure 3: Topological data model. 12
Figure 4: Example of Node-arc Model.. 14
Figure 5: An example of vehicle routing problem in a typical city transport network. 15
Figure 6: Structures of Hierarchical DBMS. 16
Figure 7: Basic structure of Network DBMS. 16
Figure 8: Example of relational DBMS structure. 17
Figure 9: The life cycle of a Midlet.. 19
Figure 10: The MIDP Graphic user interface (GUI) Components.. 19
Figure 11: Connection mechanism between mobile device and Web server.. 20
Figure 12: Dijkstra‘s shortest route algorithm.. 23
Figure 13: An example of a KML files containing Place mark information for a place 30
Figure 14: Project work flow 38
Figure 15: User requirements abstraction 40
Figure 16: Conceptualized System Architecture. 41
Figure 17: Mobile journey planner client system architecture. 42
Figure 18: Modeling the network: Arcs and nodes. 46
Figure 19: Entity relations model. 49
Figure 20: Final System Architecture (for online and mobile clients). 50
Figure 21: Final mobile journey planner architecture. 53
Figure 22: Route Search form. 54
Figure 23: Selecting origin and destination stations from the drop down menu. 55
Figure 24: Route Search Results. 56
Figure 25: Route map generated by JavaScript. 57
Figure 26: Overlaying KML on Google maps to show the route and stations. 58
Figure 27: About page for the mobile client 60
Figure 28: Route information. 60
Figure 29: Route search form (empty) 61
Figure 30: Route search form (Filled) 61
Figure 31: Choosing Origin from a list. 62
Figure 32: Choosing Destination from a list. 62

LIST OF TABLES

Table 1: Adjacency Matrix 24
Table 2: State Matrix 24
Table 3: Overview of related projects.. 35
Table 4: Supported features and functionality of the final application. 59
Table 5: Age distribution of the Users 68
Table 6: User feedback results in brief 68
Table 7: Phone ownership information for the users in application testing 69

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSS	Cascading Style Sheets
CBD	Central Business District
DDL	Data Definition Language
DML	Data Manipulation Language
DOM	Document Object Model
GIS	Geographical Information Systems
GPRS	General Packet Radio Service
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JAVA ME	Java Micro Edition
JAVA SE	Java Standard Edition
JSON	JavaScript Object notation
KML	Keyhole Markup Language
MIDP	Mobile Information Device Profile
SQL	Structured Query Language
USB	Universal Serial Bus
XML	Extensible Markup Language

INTRODUCTION

As people tend to move to urban environments in search for jobs and other facilities, urban population tends to grow and hence the demand for better public and social services increases. Sommer et al. reports that by year 2004, Nairobi city had a population of about 3 million with annual population growth of 2.8% and a population density of 38 persons per hectare [1]. Furthermore, Nairobi has the highest growth rates compared to other cities in Africa. The city is regarded as a center for road, rail and air transport networks. Poor services and infrastructural constrains have slowed down the economic development around the city thus worsening the urban poverty situation [1]. According to Kingori et al., the city of Nairobi is the most populous city in the East African region carrying a population of over 3.5 million residents [2]. Nairobi is also seen as one of the fastest growing cities after Guadeloupe, Mexico City and Maputo [2]. It is estimated that 29% of the total population of Nairobi city lives below poverty line. From the economic growth perspective, about half of the National GDP comes from capital city, Nairobi [3]. With such rate of growth, the city faces many concerns like poor public services and overburdened infrastructure and other problems [3].

The public transportation in Kenya is mostly run by *matatu(s)*. *Matatu* is a local term used to refer to local buses that do not follow any time schedule and constitute the main form public transport around the city. Matatu is the informal para transit industry in Kenya that provides public transportation services to a majority of Nairobi residents every day. Matatus are the main mode of transport for average city residents and thus constitute the greatest share in city public transportation services. The public transportation system around the city of Nairobi does not provide ordered and scheduled services. The industry is mostly run by private enterprises that do not maintain consistent and reliable bus schedules. The primary mode of transport around the city is by public means and walking.

In the light of communication, mobile phone usage and ownership is picking up day by day among most of the city population. Most people own and use mobile phones in their day to day activities. Currently, there is an upward growth in access to internet resources among the working class and interested mobile phone users. The growth of the city is directly related to the free movement, transportation and communication channels available to use. As the city strives to keep in pace with technological developments in ICT infrastructure, also comes the need to access and share information. There have been some attempts to undertake projects like street addressing, street lighting etc. but nothing has been done yet so far to establish a system where people can plan routes to various places without the old fashioned way of asking for directions from the streets, or even using proper informative maps. In the course of this research, it was found that most people rarely use maps in Kenya due to their unavailability.

1.1 Public Transportation in Nairobi

Kingori et al. gives an insight into the traffic situation in Nairobi and terms it as alarming with over 7,500,000 person trips per day and about 46% of this constitutes home bound trips, 25% work related and 29% for other trip types [2]. Out of these person trips, the percentage proportions per mode are: 29% by matatus, 3.7% by bus, 0.4 % by rail, 47% by walking, 15% by private car while other means make about 4.9% of the total trips [2]. Most of these trips are carried out across and within the city center.

The public transportation sector faces many challenges including poor infrastructure, poor traffic regulatory systems and poor traffic policies [4] and the tendency that access to good infrastructure is dependent on income levels rather than on population density [5]. Commuters in Nairobi continue to suffer physical and financial stresses e.g. sudden fares hikes, unreliable transportation by matatus and so on in the course of the reporting to work due to incompetence in planning by the authorities [6]. These problems are related to how Public Service Vehicle (PSV) transport is managed and organized. In this context PSV comprises of matatus, buses and taxis [6].

It is worrying to reckon that poor planning of city services including transportation and construction is likely to plunge the city into planning crisis in the near future [7]. However, there are plans to introduce *Nairobi Metropolitan Mass Rapid Transit*

Programme as proposed in metro 2030 strategy [8] on various transportation corridors and around CBD. The programme also includes a rapid rail, non-motorized transport systems and improved parking services.

Geographical Information Systems (GIS) is a vital tool that can be used in planning and modeling of matatus routes for Nairobi area. Network analysis could be implemented in optimum path finding. This is discussed in detail in chapters two and three of this report.

1.2 Towards Better Public Transportation System

It is evident that some kind of restructuring of the public transportation systems in Kenya specifically with reference to the city of Nairobi is needed. The relevant authorities need to take an initiative and implement proper transportation planning. If the current situation allowed prevailing, the chaotic situation witnessed now is likely to choke the city and plunge it into absolute planning crisis [7]. In the course of the research, it was found that traffic jams in city roads can take as long as one hour or more especially at rush hour in the morning along Thika and Jogoo roads. This means that many man hours are wasted. Wasted man power translates into wasted human resources.

According to Kumal et al., urban public transport depends on coordinated measures geared towards the improvement of infrastructure and quality traffic management services [9]. Short term measures like enforcing of existing traffic rules, vehicle inspections, traffic management and adopting quality transportation services and standards could help to improve traffic management [9]. These include standard fares and schedules in addition to reliable route structures [9]. Ndong'a argues that some efforts have been put in trying to decongest the city and improving the traffic situation in Nairobi [10]. For example, through introduction of a 'Smart Bus' that has scheduled trips and assigned routes [10].

1.3 Development of Journey planner Application

In the light of the above problems, this project sought to develop an application that could assist users to pre-plan their journey and to have information related to their journey prior to the start of the journey or 'on the fly'. An online application running on web browsers was developed based on the matatus routes network. This application aimed to provide information about buses, routes and route numbers for all matatus routes around the city of Nairobi. People need information about getting from place to place so then it becomes easier to plan their journeys well in advance and save time and money.

In connection to the online application, a mobile journey planner application was developed as most people always carry phones with them, thus a reliable and easily accessible service would be a great asset to use. The aim of this project was to develop a journey planner application that reduces the data traffic and maximizes the performance and efficiency. The mobile application was made compatible to most of the phones available in the Kenyan market especially the low end phones which constitute the greater part of the Kenyan mobile market.

The application employed a routing algorithm in determination of the shortest routes and connected this information to matatus route numbers. At the time of this study, every route around the city of Nairobi had a bus (matatus) number assigned to it. Normally, the route numbers and routes are assigned by Transport Licensing Board of Kenya. A route map showing the path followed from origin to destination was plotted on a map. This was only possible for the online version as the small screens sizes, limited bandwidths and processing power for targeted low end mobile phones were the limiting factors in the development of the application.

1.4 Statement of Purpose

Transportation plays a major role in moving people from one place to another e.g. from home to work and back to home. An efficient transportation system is one that makes it easier for people, goods or services to move from one location to another easily, effectively and without wasting much time. To achieve a reliable transportation system, information about routes, stations or bus stops, connection to the stations, bus route information and bus timetable and also accessibility to that information were essential.

Currently, the public transportation system in Nairobi is not well organized and it is mostly run by private enterprises in a ‘cut throat′ competition fashion. The demand for better transportation services still ranges on. The private entrepreneurs are usually prioritizing huge marginal profits even if it means offering very poor services. There were no operating bus schedules in place found during the time of this research. This means that a traveler would have to go by ‘trial and error’ to find a bus at some terminus somewhere. There are no clear bus stops and bus route information or maps freely accessible to the public. This in turn means that it becomes a task to plan any journey around the city and it is impossible to plan it in advance or ‘on the fly.’

There is a big problem in both accessibility and availability of information about how a person could navigate from one place to another and what bus to use to go to the desired destination. The quickest solution is usually to ask people nearby for information. The reliability of this information is limited to availability of the person to ask from and also on that person’s knowledge and the willingness to share that information plus the level of accuracy of that information.
This research project proposes a mobile and an online journey planner application as a solution to the above. The application is called ‘*Nairobi city journey planner’*. The application is limited to only matatus routes and stations as well as matatus route numbers around the city of Nairobi.

In the application, the user is able to choose an origin as well as a destination station. The application then returned comprehensive route and matatu information to the user. The request returned route information showing a list of stations by name from origin to destination and what bus number(s) to take. The application used Dijkstra routing algorithm to compute optimal routes. Station information was stored in a relational database on the server side. On the client side was a browser or a mobile phone device.

1.5 Objectives of the Study

This project aimed at developing a cell phone and web application that enabled users to get shortest route connections and the bus route information from any possible point to another and visualize the route on Google map or on Google Earth application.

The following were the objectives of this research project:

- To map all existing matatus routes.
- To map all bus stops along these routes including junctions, intersections, termini and destinations.
- To map major bus termini in the city.
- To develop an online and a mobile journey planner application that offers the following capabilities and services:
    - Compute optimum routes from any station to another and link this route with bus route numbers and distance information.
    - The resulting route and station names from the origin to destination station can then be visualized on a map. However map service was not implemented on the mobile phone due to the device limitations on targeted low end phones.
- To test the application with real users and acquire user feedback.

However, walking edges, bus schedule and driving directions were beyond the scope of this research as this project only considered matatus routes. In the research, no existing and functional bus schedule or timetable was found in use in the whole of Nairobi city.

1.6 Organization of the Report

This report is organized into four chapters. The first introduces the topic or the subject of research based on prevailing background information of the study area. It serves also as a brief introduction of the research topic and the problem being addressed. In this case, background information about Nairobi city and the state of the city’s public transportation (see appendix 7). The goals and objectives of the project are stated and defined in the first chapter. As the reader might notice, the names project, research and thesis have been used interchangeably to refer to this entire work. Literature review makes the second chapter. This gives the theoretical background of the project in relation to the available previous work. Moreover, the technologies used in the project are discusses there in brief.

The third chapter covers materials and methodologies employed in addressing the issues raised in the statement of purpose. This describes the procedures followed, data used and all the necessary steps followed towards development of the proposed application. The analysis of the results and discussion follows materials and methodologies. Application testing and collection of user feedback are discussed in this chapter. The review of the whole project, testing, user feedback, future plans and development, difficulties and challenges surrounding the project are also discussed. The final chapter is the conclusion in which various recommendations and final remarks are discussed. A collection of various appendices relating to this report have been provided e.g. codes excerpts, photos and questionnaires.

GIS AND ROUTING ALGORITHMS

This chapter gives a very basic overview of the technologies employed and the theoretical background behind this research. However, some parts of the literature have not been discussed in details and only brief details are given. Moreover, some parts of this chapter might look rather basic for a reader because only basic background information, concepts and technologies has been discussed.

2.1 What is GIS?

Korte et al. defines GIS as a computer system that has the capability to store all information found on a map [11] e.g. in a traditional map. It can store and show information about geographical features e.g. roads, buildings, cities, vegetation and other features. GIS is superior to a traditional map in data search and analysis.

GIS is a powerful tool that can be used in road networks, resource mapping and environmental monitoring. GIS can answer questions about 'what' is 'where' [12]. \`What´ refers to the feature type and its attributes and 'where' refers to a geographical location in the real world. The project modelled the public transportation routes in Nairobi using a GIS approach. This report also describes the approaches employed including GIS concepts used in case of future developments and improvements by other interested developers not necessarily the researcher who did this application.

2.1.1 GIS in transportation

The major components of a GIS are software, data, hardware, procedures and people. The software captures, analyses, stores and displays the spatial information. According to Hanson, GIS can address complex tasks for example constructions and transportation planning [13]. Transportation planning involves visioning, plan implementation, programming and finally maintenance of the whole system. Establishing goals or objectives of the transportation planning creates the visioning aspect of the project while planning aims at producing transportation map. Programming entails all the procedures involved from the start to the end of project including the project lifecycle, scope, design and construction. System maintenance refers to monitoring of performance of the system [13] and checking whether relevant updates are necessary to be made to improve on the current existing system. Spatial data is the core of any GIS. This data must be linked to attribute data for full GIS exploitation [13]. Road data (spatial data), application development procedures and technological expertise were employed in this research.

2.1.2 Representing the world in GIS

Geographical data describe locations and characteristics of the real world. Geographical data models include raster and vector data models. Geographical data models are mathematical models used to represent geographical objects and surfaces. Vector data models describe discrete features e.g. features are represented as points, lines or polygons, while raster data models are used mainly to depict continuous data like elevation and vegetation types [14]. This study focused on vector data structure as it formed the core of the research and therefore raster data models and structures are not discussed in depth.

Figure 1 shows modeling of geographical data in GIS. Physical reality is complex to represent and this calls for simplification and generalization. For instance, roads are represented as lines. Sometimes symbols are used to represent features in order to distinguish them from closely related features e.g. a highway and a foot path would be represented as lines but with different levels of generalization and symbology.

The matatu route networks was modelled as points and lines (vector data), hence knowledge of what GIS data model to be used for this project was necessary.

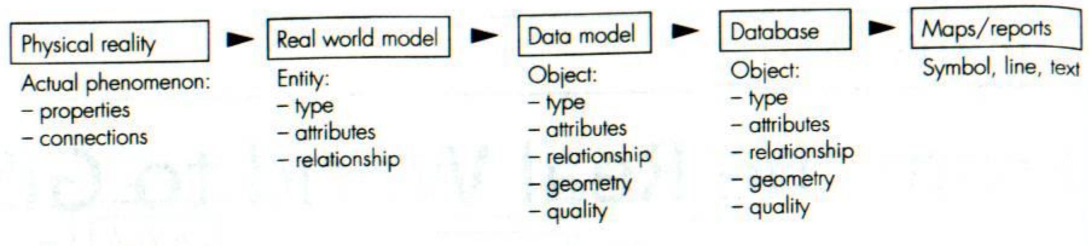


Figure 1: Modeling the real world in GIS. Source: [15].

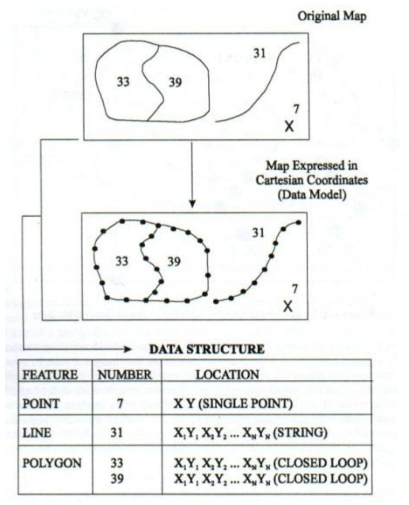
2.2 Vector Data Models

This is one method of representing geographical data where spatial locations, relationships between objects are expressed explicitly. Lines are represented as connected coordinates while points are represented by one coordinate. Polygons or areas are represented by lines but with first and last coordinate's being the same. The attribute data for this feature model is kept in a database. Bernhardsen adds that lines and polygons are difficult to represent in a database because of the differences in the number of points composing them [15]. Some polygons could be made of numerous numbers of points. Below are some examples of vector data models:

- Spaghetti data model.
- Topological data model.
- TIN

2.2.1 Spaghetti data model

This type of data models involves storage of each feature as a separate entity from the others. Polygon features with common boundary are stored separately meaning the boundary data is stored twice. No feature to feature relationship information is stored explicitly. GIS operations like overlaying and network analysis, point in polygon problems are difficult due to unlinked data model [15]. Figure 2 shows an example of a spaghetti data model.

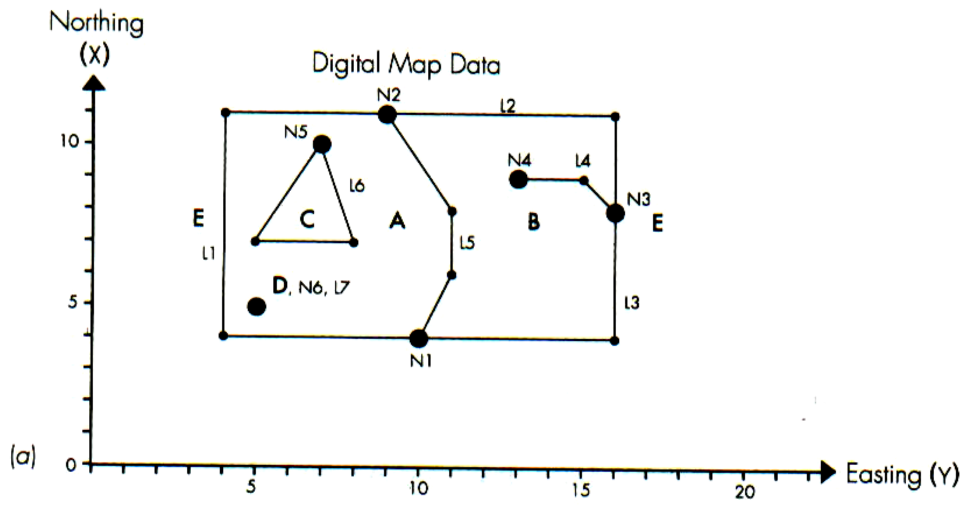


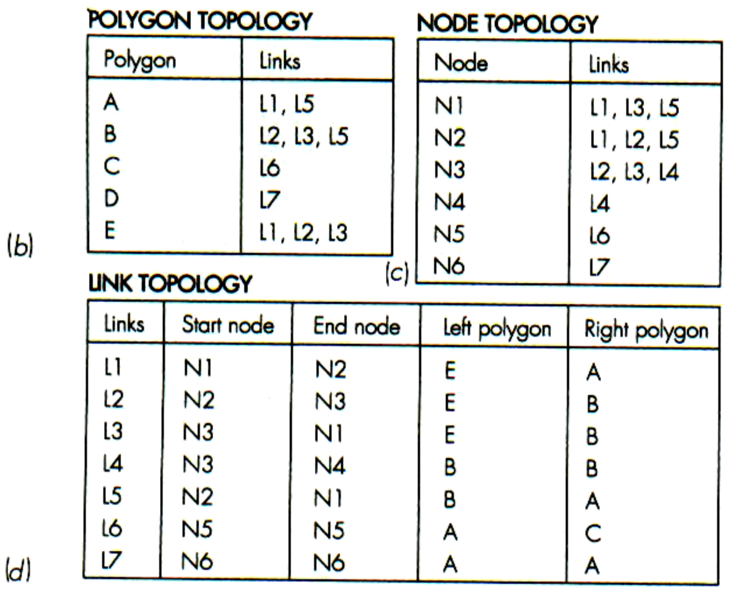
FEATURE	NUMBER	LOCATION
POINT	7	X Y (SINGLE POINT)
LINE	31	X_1Y_1 X_2Y_2 ... X_NY_N (STRING)
POLYGON	33 39	X_1Y_1 X_2Y_2 ... X_NY_N (CLOSED LOOP) X_1Y_1 X_2Y_2 ... X_NY_N (CLOSED LOOP)

Figure 2: Spaghetti data model. Source: [16].

2.2.2 Topological data model

Topology examines those geometrical properties that undergo no change when a geometrical object is subjected to transformations e.g. stretching, twisting or bending. Topological data model defines connectivity and relationship between objects independently from their coordinate’s data [15]. A topological model contains nodes, arcs or links. A node is a point where two lines intersect. A link joins two nodes. Sometimes links are referred to as edges. A network is composed of several links where some links share nodes, see Figure 3 (a). A closed polygon is made of alternating links and nodes enclosing an area [15].Unlike in spaghetti model, common boundaries between polygons are stored only once.




(b)

POLYGON TOPOLOGY

Polygon	Links
A	L1, L5
B	L2, L3, L5
C	L6
D	L7
E	L1, L2, L3

(c)

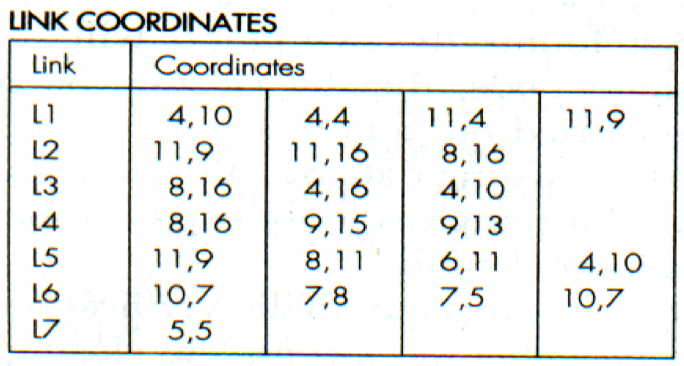
NODE TOPOLOGY

Node	Links
N1	L1, L3, L5
N2	L1, L2, L5
N3	L2, L3, L4
N4	L4
N5	L6
N6	L7

LINK TOPOLOGY

(d)

Links	Start node	End node	Left polygon	Right polygon
L1	N1	N2	E	A
L2	N2	N3	E	B
L3	N3	N1	E	B
L4	N3	N4	B	B
L5	N2	N1	B	A
L6	N5	N5	A	C
L7	N6	N6	A	A



LINK COORDINATES

Link	Coordinates			
L1	4,10	4,4	11,4	11,9
L2	11,9	11,16	8,16	
L3	8,16	4,16	4,10	
L4	8,16	9,15	9,13	
L5	11,9	8,11	6,11	4,10
L6	10,7	7,8	7,5	10,7
L7	5,5			

(e)

Figure 3: Topological data model: Source: [15].

In Figure 3 above, polygon topology lists the links that make up a polygon while node topology shows the links that intersect at a given node. Link topology links the start and end node defining any link in addition to adjacent polygons on opposite sides of the link, normally referred to as right and left polygons. A topological model maintains that connectivity and relationships between various features remain undistorted and are represented independently of the coordinate information. It is also possible to perform overlays, network analysis if all lines are connected and all loose ends removed [15].

Other vector models include Triangulated Irregular Networks (TIN) used to model three dimensional surfaces e.g. terrain models. Longley et al. adds that TINs are used in creation of surfaces in GIS by use of non-overlapping triangular elements [17, 18].

From the above discussion about the vector data models, either spaghetti or topological data models could be used in the development of the route planner. However, for the purpose of network analysis needed for route computations for the matatu routes, topological data model was preferred and therefore adopted as discussed in chapter 3 of this report. Spaghetti models topology information and creates more redundancies and overheads in data storage.

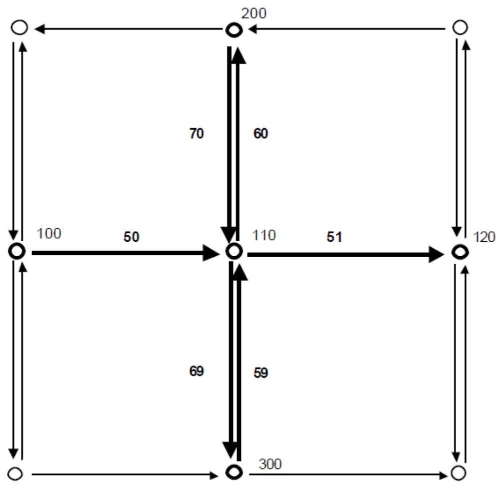
2.3 GIS and Network Analysis

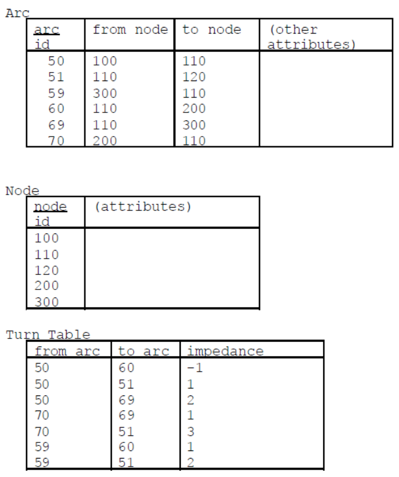
Application of geographical information science and systems to solve transportation problems is referred to as Geographical *Information Systems for Transportation* (GIS-T). GIS-T requires special data structures to represent complexities in transportation networks and application of networks algorithms for determination of optimum paths. Transportation network data model includes use of turntables, linear referencing, traffic lines and non-planar networks. A linear referencing system enables storage and maintenance of information or events that take place in a given transportation network e.g. road quality and traffic flow [20]. Hensher illustrates some of the network routing problems including Travelling Salesman Problem (TSP), vehicle routing problem, shortest route problem and time constrained routings [21]. Furthermore, a network is said to be a network if only topology and connectivity between nodes and arcs exists. Transportation networks are usually represented as nodes and links. This network could represent flow of people, goods or vehicles. A network can be represented graphically as directed links as arcs and undirected links as edges and arcs intersect at a node [20].

However, most GIS softwares only utilize simple geometric entities such as points, polygons and lines and cannot handle data such as underpasses, overpasses, origin destination flows, complex paths and intermodal transfers like example motor way to railway line [20]. A network differs from a graph by the fact that a network can accommodate weights or costs assigned to various arcs. Planar networks require that all arcs intersect at a node. The relationship between nodes and arcs is referred to as network topology [21]. In making the matatu route networks; network analysis was used where arcs intersecting at nodes were used. Costs or weights were also assigned to the arcs.

2.3.1 Network data model

Data models represent real world as abstracted and organized entities in databases. Modeling data involves use of conceptual, logical and physical data models where the conceptual data models do not consider any implementation aspect of it. The geo-relational data model is a widely adopted logical data model used in networks representation [21]. These models are used where attribute and spatial data are separated into different data models. RDBMS (Relational Database Management System) tables are used to store topological relationships and geometry and other associated information. The relation data structure that supports planar network model consists of arc node relation information [21]. Figure 4 shows an example of an arc – node model of a network.




Arc

arc id	from node	to node	(other attributes)
50	100	110	
51	110	120	
59	300	110	
60	110	200	
69	110	300	
70	200	110	

Node

node id	(attributes)
100	
110	
120	
200	
300	

Turn Table

from arc	to arc	impedance
50	60	-1
50	51	1
50	69	2
70	69	1
70	51	3
59	60	1
59	51	2

50 -	ArcID
100 -	NodeID

Figure 4: Example of Node-arc Model. Source: [20].

Nodes and arcs were used to represent the matatu stations or bus stops and routes respectively by using this Arc-node model.

2.3.2 Vehicle routing within a network

There are many procedures employed in solving networking routing problems in GIS-T. Usually these problems are simple to comprehend and to state but sometimes very hard to model and solve mathematically.

For the purpose of this study, only node-arc routing problem is discussed where key events in the routing occur at nodes e.g. bus transfer points, passengers alighting and boarding at bus stops. The performance of GIS-T application is dependent on the comprehensiveness of the data model adopted and how well nodes, links and other related information are organized and stored in GIS database. Figure 5 shows an example of vehicle routing in a typical city transportation network. The figure depicts one way streets, intersections, junctions and turns just like the street networks in Nairobi. These features were considered in the network modeling.

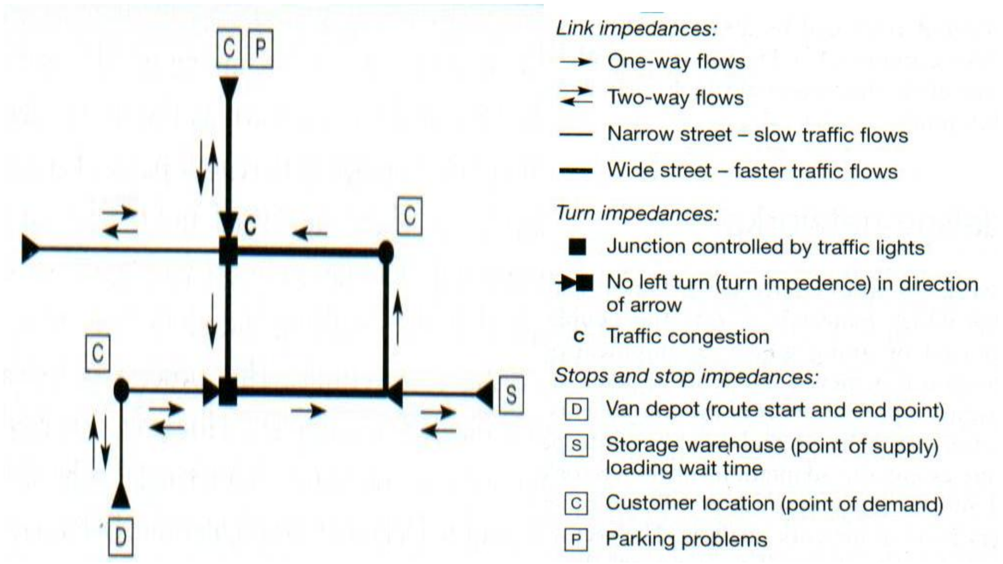

C
P
C
C
C
S
D
c
Link impedances:
One-way flows
Two-way flows
Narrow street – slow traffic flows
Wide street – faster traffic flows
Turn impedances:
Junction controlled by traffic lights
No left turn (turn impedence) in direction of arrow
c Traffic congestion
Stops and stop impedances:
D Van depot (route start and end point)
S Storage warehouse (point of supply) loading wait time
C Customer location (point of demand)
P Parking problems

Figure 5: An example of vehicle routing problem in a typical city transport network. Source: [22].

2.4 Databases

A database is an integrated data set about a certain subject. A spatial database is one that has geographic data of a particular subject. A Database Management System (DBMS) creates, maintains a database as well as managing access to it [23]. Databases reduce costs of maintaining data, avoids data redundancy and offers sharing of data. To maintain integrity of data base constraints and relationships can be enforced to the data. However, databases could be complex and expensive to maintain [23]. In the next section, various types of database models that could be used in this project are briefly discussed. Databases form a very wide subject area to be discussed here and so only very basic discussion has been presented in this report.

2.4.1 Types of DBMS models

- **Hierarchical**: This data model organizes data as a tree structure in a parent- child format. This translates to redundancies especially for child data segments. Creating of links between records needs the establishment of a parent-child relationship by use of trees [23]. Figure 6 shows a basic structure for hierarchical database.

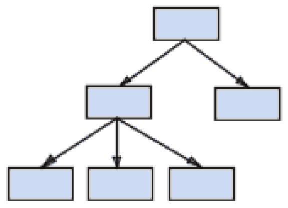


Figure 6: Structures of Hierarchical DBMS [24].

- **Network**: This is more or less like hierarchical model only that it allows many to many relationships and network of relationships presented as pair wise sets [23] as shown in Figure 7 below.

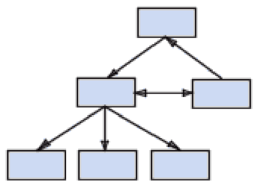
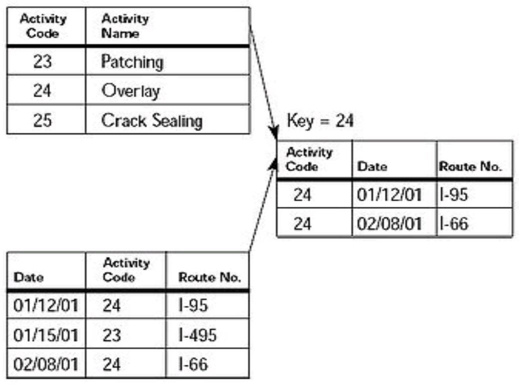


Figure 7: Basic structure of Network DBMS [24].

- **Relational**: This allows for data structures definitions, retrieval and storage operations. It also allows enforcing of constraints by use of relationships between tables [23]. The table stores rows of records with their attributes stored in fields. Joining of tables is done by using a common field or foreign keys that link common fields into two or more tables. Each record is unique and every table should have at least a foreign key to enable an unambiguous retrieval of a record. An example of a relational database is given in Figure 8 below.



Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Figure 8: Example of relational DBMS structure [24].

Other types include Object- Oriented and Object - Relational databases. In this project, relational database was used since it assisted in relating the different entities for example stations, roads and matatu numbers as this was the purpose of the project.

2.4.2 Using Databases

In order to use any database, it is necessary to connect to it and to run queries in database requires use of SQL (Structured Query Language). SQL has the following statements:

- Data Definition Language (DDL) - Used to create tables e.g. *CREATE TABLE* functionality.
- Data Manipulation Language (DML) – This is used to retrieve and manipulate data. It includes the following modules: *SELECT, DELETE, UPDATE* and *INSERT*.
- Data Control Language (DDL) – This enforces security on data in the database e.g. *GRANT, CREATE USER,* and *DROP USER* [24].

Since the application to be made required a database driven approach, therefore, several SQL commands or queries will be used in the application to fetch information about stations, roads and bus information (see appendix 1).

2.5 Web and Mobile device Technologies

In developing a Java based mobile journey planner, understanding the components and features available in mobile devices as well as the mobile computing environment is important. A journey planner application was developed to run on mobile phones in addition to the online version. Only brief overview is given to give the reader some clues about the terminologies as well as technologies employed in the project. Later in the chapter, web technologies used in the development of the browser based application have been mentioned but not in deep details.

2.5.1 Mobile Information Device Profile (MIDP)

The Mobile Information Device Profile (MIDP) enables the development of mobile application and services for most mobile phones. It runs in a Java runtime environment in most common phones available today. It functions in connection to Connected Limited Device Configuration (CLDC) that sets the base for application programming interfaces and a virtual machine for phones. When MIDP is used in combination with CLDC, it offers a platform to develop applications for mobile phone devices [25].

2.5.2 Developing a Mobile phone application

MIDP defines the behavior of the application to be made. The application must inherit a *Midlet* (A java class that defines the life cycle of the application, see appendix 5). A *Midlet* acts like the main method in standard java applications. It can have the following states: loaded, active, paused and destroyed [26] as shown in Figure10 below.

In the development of the mobile version, a journey planner *Midlet* was developed. The components of the Midlet included a user interface that included forms, choice groups and commands. These are described in brief in the next section and also in Figure 10.The journey planner Midlet followed the architecture described in Figure 9 below. More details concerning the implementation and documentation of the Java ME *Midlet* remains outside the scope of this project. Only skeletal information about the technology to be used is discussed here.

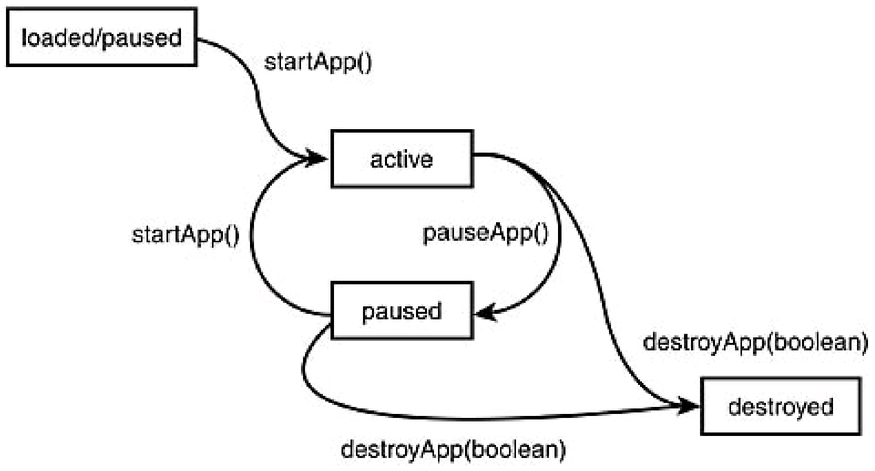


Figure 9: The life cycle of a Midlet. Source: [26].

2.5.3 Overview of MIDP User Interface

The MIDP user interface can either be low level API or high level API [10]. High level API contains components like text fields, choices and gauges while low level API contains canvas and enables full control of the Midlet to the pixel level unlike high level API. Figure 10 shows the component of MIDP graphical user interface. These are the components that were to be used in the development of the mobile application user interface. The user interface consisted of forms, text fields, choice groups and alerts. This is a brief overview about developing a mobile application in Java in a MIDP environment and details about MIDP functionalities, system architecture and other information are beyond the scope of this project.

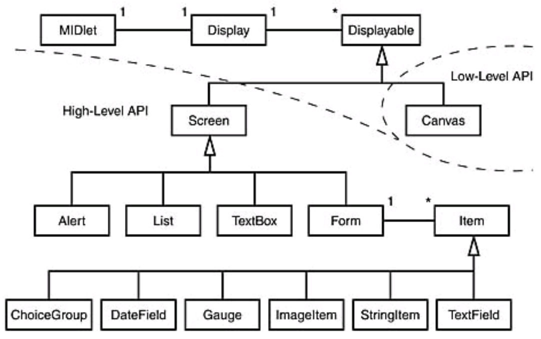


Figure 10: The MIDP Graphic user interface (GUI) Components. Source: [26].

2.5.4 Data Interchange between Mobile Client and Web Server

Sometimes a mobile application needs to retrieve content or resources from a web server. In a MIDP environment, connection to web server is carried out via HTTP Connection. HTTP (Hypertext Transfer Protocol) is the protocol of data communication on the internet [27]. HTTP Connection uses an '*Input Stream*´ and an `Output *Stream*´ in one connection. However, HTTP connection has three states: *open*, *connected* and *closed*. However, for a successful connection, connection parameters can be set such as the request e.g. GET, POST and HEAD methods [28]. For the purpose of this research, HTTP protocol and request methods and how they work are not discussed and are beyond the scope of this research.

In some situations, a mobile phone needs to communicate with a remote web server to retrieve data and show it on the device. For example, a mobile application that needs to connect to a PHP page in a web server, then several considerations should be made concerning the type of data interchange formats to use [29]. Figure 11 shows the interaction between a mobile client and a web server.

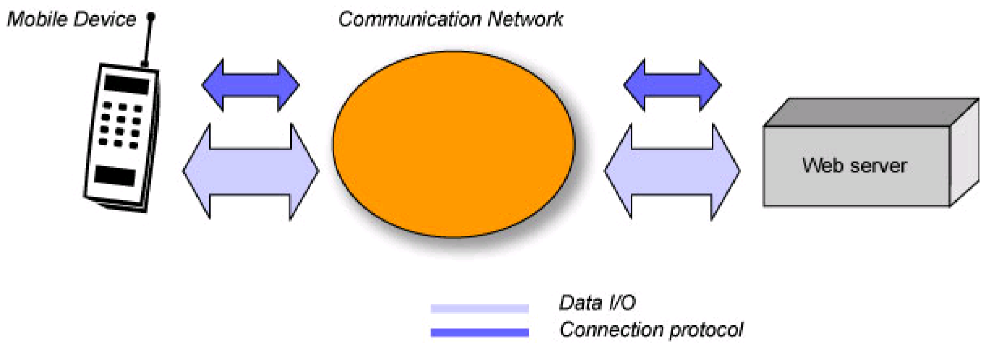


Figure 11: Connection mechanism between mobile device and Web server. Source: [30].

The journey planner application for the mobile phone required HTTP connection to the web server for route computation request and response communication. This will be discussed again in the following chapters. The choice of any data interchange format to be used depends on issues about lightweight, efficiency, lower computational intensity and lower operational costs [31]. The most commonly used data interchange formats are:

- **JSON** (JavaScript Object Notation)–This is a popular format used for communicating between a (Java Me) client and an application server [32]. It is an alternative to XML. It is a text based, human readable, light weight data interchange format [33]. It uses simple syntax data structure such as collections of name and value pairs, and ordered list of values [31]. The disadvantage of JSON is that it is nonstandard and non-operable, in some circumstances e.g. in non-textual data [32]. It is more compact since it is a combination of objects and array literals for storing data. Nicholas et al. argues that JSON does have variables assignments or equality but only represents data [34]. It is based on JavaScript syntax; hence, it can be incorporated into JavaScript files and can be accessed without parsing unlike XML.

- **XML** –This stands for Extensible Markup Language. It acts as a set of rules for encoding documents, a format readable by machines [35]. It is text based and document oriented and it offers generality, simplicity and usability. However, it is heavy and verbose for data representation [34] and it needs to be parsed by an XML parsers [30] in order to extract data in a MIDP application.

- **Custom Protocol** –This serves as an alternative to JSON or XML especially for simple and short strings by using string tokenizers [36] or string delimiters in the encoded string in the server response. A string tokenizer allows an application to break a string into tokens. By using string manipulation, contents of a long string can be parsed into different elements in the MIDP application.

A comparison of these formats and a justification of the choice adopted are given in the analysis section. JSON, XML and custom protocols were experimented as the possible data interchange format options between the mobile client and the web server.

2.6 Routing Algorithms

Routing means searching for what path to take while a routing algorithm searches for the optimum path between an origin and a destination. For example, given scattered cities that are linked to each other through a complicated road network such that there is more than one alternative route from one city to another and motorist just wants to know the cheapest or the quickest route between any two of those cities. This is an example of a network problem in which optimum routes need to be computed. In the context of this research, the following routing algorithms were considered:

- Dijkstra shortest Path algorithm
- A* search algorithm.

Network analysis in GIS assists indetermination and making of choices with regard to optimum route, finding nearest social amenities and service points. Sanjeev et al. remarks that GIS route finding modules and heuristic algorithms are employed in search for the best solutions to optimum path problem [40]. These routing algorithms could be further customized to address problems of selecting a route to a given destination in intelligent transport systems [40]. An optimum route search for destination is the most important goal in network analysis. Minimum travelling times or distances are used in definition of the optimum route. According to Mainali et al., sometimes the user may not always be interested in the minimum time or distance given in the shortest route but may prefer using certain routes even if it means adding to the cost of travel [41].

In this project, route computations for user selected origin and destination stations required an implementation of a routing algorithm. A brief discussion of two of the most widely used routing algorithms is given in this report. These are Dijkstra's and A* search algorithms.

2.6.1 Dijkstra Algorithm

Dijkstra's shortest path algorithm computes the lowest cost path from one point (node) to all others in a network graph composed of weighted edges [42, 43]. For example, the weighted edges can be thought of as interconnected road network and the nodes as cities.

Dijkstra's algorithm is an exact algorithm but does not guarantee that realistic deadlines will be met. Furthermore, genetic algorithms may be used as they provide alternative routes using other solutions in the shortest time. In the past, genetic algorithms were used to find easiest-to- drive and quasi shortest route to reach a destination within a given time [40]. This method can be used to produce and choose possible routes that guarantee the meeting of deadlines and satisfy constraints regarding ease of driving [40].

2.6.1.1 How Dijkstra's Algorithm Works

Beginning at a start node, the algorithm sets the cost to this source as 0 and to all other existing and unvisited nodes as infinity. The algorithm then proceeds to look for adjacent nodes that are connected to the source node. It updates the visited nodes to new value which is the sum of the value of previously visited node and weight of the adjacency from this previous node to the current node. The current node is then set as visited and the algorithm searches for any other existing unvisited nodes connected to the current node and the value at the visited nodes is updated. When the value of adjacent edge plus node value becomes less than the value of adjacent node, the adjacent node is changed to this value. The algorithm continues until all existing nodes are marked as visited and their values updated. The value of the nodes that the algorithm returns denotes the least cost of the travel from start node to that node [43]. An example of a Dijkstra shortest path problem can be visualized in using Figure 12 below.

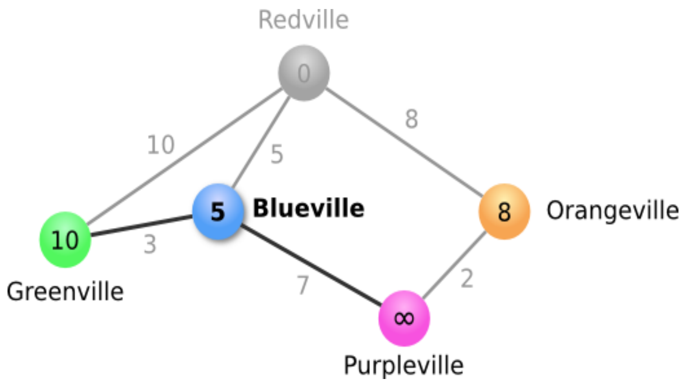


Figure 12: Dijkstra‘s shortest route algorithm. Source: [21].

Starting at a source node, Redville and setting its value as 0 and all other nodes set to infinity since paths to them have not yet been computed. From the diagram, the start node shares adjacencies to Greenville, Blueville and Orangeville [43]. These adjacencies are shown in the Table 1 as an adjacency matrix.

Table 1: Adjacency Matrix

Node	Redville	Greenville	Blueville	Purpleville	Orangeville
Redville	0	10	5	–	8
Greenville	10	0	3	–	–
Blueville	5	3	0	7	–
Purpleville	–	–	7	0	2
Orangeville	8	–	–	2	0

By checking the value of weighted edge and node value of current node as less than value of adjacent node, then the Greenville, Blueville and Orangeville nodes all get an updated value (previous values were infinity).

Table 2: State Matrix

Node	Distance	Path	Visited
Redville	0	Redville	Yes
Greenville	10	Redville	No
Blueville	5	Redville	No
Purpleville	∞	Redville	No
Orangeville	8	Redville	No

This means that Greenville gets value 10 (node value at start node (Redville + edge value) = 0+10=10), similarly Blueville gets value 5 and Orangeville gets 8 as its new updated value [43]. This information can be presented as state matrix in Table 2. Distance column denotes the current shortest distance from starting node to this node where path column denotes the previous node in the shortest route to that node. The visited column denotes whether the node has been 'visited' or not. In this case, Redville has been visited. A node is not marked as 'visited' when it is seen from other nodes and their paths and distances values updated. A node will not be visited until it is the node with shortest distance value and not yet visited. The algorithm proceeds by setting current node as Blueville and Redville as visited since it has less value compared to other unvisited nodes and the procedure is repeated from the start until all nodes are visited and their values updated including state matrices for all visited nodes until the shortest path is found.

2.6.2 A* search Algorithm

This is another routing algorithm that utilizes priority queue in the computation of least cost paths. It is an improvement of the efficiency of the Dijkstra's algorithm. If a chosen path contains higher cost than another visited path segment, it leaves the former in preference to the later until the destination is reached. It uses the following concept [44]:

$$f'(n) = g(n) + h'(n)$$

Where:

g(n) - Distance from start to current node.

h'(n) - Estimated distance from current node to destination node created using a heuristic function [44] .

f'(n) - Current estimated shortest path.

The heuristic function **h'(n)** is an arbitrary function and the better its estimate the better the accuracy of the least cost path solution [44]. A* uses an estimator or heuristic function in estimation of the shortest path between source and destination nodes. According to Sanjeev et al. A* algorithm performs better than Dijkstra algorithm.

2.6.3 Comparison between Dijkstra's and A* algorithms in road networks

For many years, researchers have been bothered by the development of a shortest path algorithm possessing reasonable theoretical time bounds, as many algorithms are not always fast enough for large networks. According to Gutman, A* shows a reduction in priority queue insertion in proportion to reduction in computation time hence the cost of the heuristic function is very significant [45]. Tests carried out using two road networks in Asia, showed a similar reduction in computation times of A* algorithm from Dijkstra ranging from five to ten times [45]. The computation times for Dijkstra algorithms for road networks increase with the square of distance. To decrease time, a heuristic approach is normally used [45].

Mainali et.al. proposes a dynamic optimum route search algorithm that considers changes in traffic situation and multiple criteria e.g. user preferences when travelling times for certain road sections change [41]. A* is similar to Dijkstra but with reduced computation time done by use of a heuristic function. Further, in changing road situations both A* and Dijkstra fail to exploit available information from previous searches for new computations with minimal computations but do calculations anew [41]. However, most dynamic routing algorithms are inefficient as they need separate computations for increase or decrease of travelling time or distance [41]. A fastest route with minimum travelling time might not be what the user prefers since users may want to use a route they are comfortable with.

Rolf et al. suggests a technique to speed up the computation for shortest path for large networks and sparsely directed graphs with positive arc weights by using arc flag approach based on Dijkstra algorithm [46]. This method involves partitioning of graphs into regions and taking considerations of whether a given arc lies in the shortest path. Moreover, this method in combination with appropriate partitioning achieves a considerable speed up in relation to standard Dijkstra thus narrowing the search area for Dijkstra shortest path for long distance shortest path finding [46].

Since the road network for the project was small, Dijkstra’s algorithm was preferred and implemented in the project. The road network used consisted of 143 stations and 173 road sections (only very small part of Nairobi was considered). A* Algorithm also needed additional hosting costs since it would have been developed from a Java Enterprise Edition platform that required an application server in addition to a web server.

2.6.4 Analysis of Algorithms

In the choice of the algorithm to be used, it is important to understand how an algorithm behaves with respect to large input data. This is with respect to computation time when subjected to large input data sets say for example, given a road network for an entire country and the task is to compute cheapest path from any two points. Traffic information systems are the most widely used applications of Dijkstra algorithm for shortest paths. A study was conducted on public railway transport in Germany where numerous queries were made to the servers by customers seeking optimum connections in railroad traffic [47]. The study recommends a satisfactory compromise to be found between the speed of an algorithm and the quality of results. This research based its work on local transport where timetables were regular and speed up techniques were based on strict periodicity of trains, buses and ferries [47].

The running of algorithms typically grows with the input size. CPU (Central Processing Unit) times are strongly sensitive to details of the implementations and the total number of steps, procedures and operations in the priority queues. CPU times and priority queues could be used to determine the complexity of an algorithm [47]. The computations for optimum connections in rail road traffic information system seek for best combination for total travel time [47].

Run time analysis is usually carried out given the relationship between run time and amount of inputs [48]. For example, some programs may take longer than others to execute a task depending on the amount of resources e.g. inputs and processing power and which algorithms they use.

Bauzer et al. comments that the computational complexity of Dijkstra algorithms is given as $O(n^2)$ where n is the number of nodes in the network [49]. This means that for very large networks computation for shortest path problem, this algorithm becomes inefficient. Now, the relation between the complexity of an algorithm and the size of input data could be understood. Since the project's dataset was small, Dijkstra algorithm was used in optimum path computations. Perhaps in future as the project becomes bigger, alternative routing algorithms might be considered.

2.7 Web Mapping

After the routes are computed, the application needs to display the computed route on a map on the internet. Some aspects of web mapping were considered as this helped in understanding the concepts behind displaying a map on the internet. Web mapping is the process of designing, generating and publishing maps on the internet. Web mapping offers real time maps, frequent update of maps and sharing of geographical information. Moreover, internet offers a good platform and a medium to facilitate launching of map services on the net [50]. However, these services face several challenges e.g. low display resolution, limited bandwidths, slow internet connections especially for mobile phones. Currently, fully interactive maps as well as static maps services are available on the World Wide Web.

Advantages of Web Maps

- Ability to deliver up to date information.
- Affordable software and hardware infrastructure e.g. Open source web server hardware [50].
- Run on web browsers on the client side.
- Enables collaborative mapping e.g. Open street map and personalization of content.
- Multimedia content like videos, animations can also be used in the web map application [50].

Disadvantages of Web Maps

- Band width limitation especially for mobile phones.
- Limited screen sizes e.g. to show a large map on a small screen, scaling it to screen size might render it unreadable.
- Copyright and privacy concerns.
- Many web maps have poor cartographic quality [50].

In this study, web maps were used in showing the route map as the online application had the functionality of web map. The goal was to give as much information to a user including information that was viewable on a route map.

2.7.1 Web map service technology

To understand the technology and concept in web map services, it is worth to consider what really happens behind the scenes in order to have a map delivered on the client's browser or machine after request is passed to the web server. When a HTTP request for a map is launched to a web server, this server communicates with map application server that in turn returns a map to the client. The map server is interfaced with a spatial database. On the client side, a web browser with supporting technology e.g. Java script loads the map on the client's browser.

In this project, a route map for the user chosen origin and destination stations was plotted on the client browser so then the user was able to view the plotted route for the journey. In the internet, there are various web mapping services available including Bing maps, Yahoo and Google Maps.

Due to the flexibility in use, popularity, map quality and better customization features, the research preferred Google maps web mapping service to the others. This meant that the journey planner application development would employ Google maps API. In the following section, Google maps API is discussed in brief.

2.7.2 Google Maps API

Google maps API is the most widely used web mapping service across the globe. It utilizes advanced geocoding capabilities and delivers secure map content over the browser [51]. Google maps offer a variety of APIs enabling powerful functionality in delivering map content into various applications. Personalized maps can be embedded in personal websites as the Maps API is a free service. Further the API offers capabilities for manipulating maps through JavaScript functions [52]. In order to display feature data on Google maps or Google earth applications, the application support use of KML as geographical data file format. For the purpose of testing and comparing of performance, route maps was drawn using either KML or JavaScript technologies with the former made by server scripting while the latter implemented on client browser. KML technology is discussed in the following section.

In this application, Google Maps was used to show the route map. Further, route data in KML format could also be downloaded and viewed in Google Earth application. An example of implementation of Google maps API to show the route map is given in appendix 4.

2.7.3 KML

Keyhole Markup Language (KML) is an XML schema that expresses geographical information for internet based maps and for 3D visualization in Google earth [53]. It specifies feature types (place mark, polygon and polyline) and their attributes as well as coordinates in degrees for latitude and longitude. Open Geospatial Consortium (OGC) approved KML to be the de facto standard format for geographical data format on the internet [54].

A KML file is parsed like XML file in order to extract the contents and show it on the map. Attributes such as icon styles, camera positions and labels can be customized to enhance the visualization of KML data when overlaid on a map. However, this information needs to be included in the KML element tags [55]. An example of KML file containing one placemark as point data is shown below (see Figure 13).

```
<?xml version="1.0" encoding="UTF-8"?>
<kmlxmlns="http://earth.google.com/kml/2.2">
<Placemark>
<name>Far Place</name>
<description>This is a good place to live
</description>
<Point>
<coordinates>-2.78453,32.18745
</coordinates>
</Point>
</Placemark>
</kml>
```

Figure 13: An example of a KML files containing Place mark information for a place

The KML technology was used in packing of route data for the user selected route. This route KML could be overlaid on Google maps or viewed on Google Earth. Recall that KML is like an XML file that carries geographical information for 3D or 2D visualization. In the application, stations and paths information (location data) for user selected routes was obtained as a KML file.

2.8 Application Testing

Application testing with real users is the most fundamental usability method since it offers a chance to gather the users experience for a website or an application. User testing is sometimes irreplaceable since it gives information about how people used the product and the problems they encountered while using a product. In this application, several aspects of usability testing were considered in the light of the application developed. Usability testing assists in understanding how real users experience the application. A well designed user test measures actual performance of the application with respect to goal-critical tasks [37].

For the purpose of this thesis, the application was tested with the real users in Nairobi. The testing was looking for user experience in view of the functionality, design and user interface of the application made.

2.8.1 Conducting a usability test

In conducting user test, it is necessary for a researcher to read and direct participants through the tasks. It is advisable for the researcher to read from the same “script” as the participants to avoid biased scenarios [37]. At the end of user test, the researcher should be able to analyze participants’ facial expressions thereby noting the number of mouse clicks and navigation paths followed [38]. This would help to determine the most frustrating tasks from the user’s perspective and this would help in improving the user interface for future application developments [38]. However, it is suggested that usability testing should be done iteratively to ensure usability requirements are met in the final application [39].

2.8.2 User testing methods

Some of the methods that could be used in user test are described in this chapter. However, only brief discussions of the methods used are given. These methods include:

a) One-on-One Interviews

Structured interviews help researchers to learn user's attitudes and beliefs about the application and the functionalities and features supported by the application [38]. The interview should consist of questions that follow some kind of order clarifying what the tasks entails and how a user feels about the tasks. The negative aspect of interviews is that the output is not always statistical [39]. Oral interviews were conducted in an informal or formal session where preliminarily testing of the application was done. This required actual physical contact with the target users in the streets.

b) Participatory Design

This method engages the targeted end users in the process of solving design problems whereby the participants bring up their views of the problem and offer ideas or solutions on user needs and preferences not evident in the initial project design [38]. Later the users may be asked to state why they would like the application to be designed the way they envision it. This gives the researcher more understanding of user needs and preferences [38].

c) Manual testing

Good software testing involves the use of manual testing in order to determine the effectiveness of the application [39]. Furthermore, manual testing is one of the oldest and most effective ways in which one can carry out software testing [39]. Software testing is one of the most important tasks in any software design. However, manual testing is deemed to be tedious and laborious but does not employ any automated form of testing.

d) User centered design

This approach keeps the targeted users at the center of project design and implementation [37]. This is achieved by taking user key points in the project’s design and ensures the application meets the users’ requirements. In this case, user stories are gathered where users express their expected outcome or functionality of the application [37]. User requirements dictate the goal and design of the application. This requires user requirements and specifications gathering, design and user based assessment of the application.

e) Questionnaires

These are used to ask users for their responses to a predefined set of questions. Questionnaires constitute a good way of generating good statistics on user feedback and also assist in demonstrating the credibility of the testing exercise from a scientific point of view [38]. However, unbiased questions are preferred. Other methods of user testing include surveys, focus groups and others [37]. In the test, only user centered design, oral interviews and use of questionnaires were considered to test the application with real users to acquire user feedback. This user feedback could be used to evaluate the effectiveness and the performance of the proposed application [38]. In addition to the final testing, participatory design was used in the updating and maintenance of the final systems since it would be necessary to fine tune the system for future improvements and future research. With the system already running, system update and refinement could be periodically done to improve on the relevance, reliability and efficiency of the final application.

The purpose of the user testing for the application to be made was to test the functionality and to assess whether the application fulfilled the user requirements set in the methodology. At the beginning of user testing, a choice for target group is made. This target group consists of all possible users who would be interested in using the proposed application. The users should be as representative as possible to the intended target users [38]. However, some background information using criteria such as education background, computer literacy levels and other necessary factors is necessary depending on the goal of the user testing. In the application testing, oral interviews and questionnaires were used in the testing of this application [37]. Typically, participants would perform a set of tasks within a test session. Tasks should represent the most common user goals like route finding and bus information and since this was the most important goals of the website or application in the researcher's perspective.

2. Overview of Related or Previous work

Following an exhaustive research and relevant consultations regarding existence and the implementation of a similar or ongoing project in the City of Nairobi, no comprehensive information on matatus routing and journey planner project has ever been done or implemented. However, there have been several attempts towards development of an SMS direction service as student projects in Strathmore University, Nairobi [56]. However, more information about how it worked and whether it was released for use by the public could not be obtained. Furthermore, this service was not accessible by the researcher for hands on experience on its functionality. The relationship between these previous work(s) and this project is that, both share one goal of providing information on directions to the end user about moving from one place to another.

Other SMS direction services found were Google SMS direction service [57] and Arena SMS service [58]. Information about the popularity in active use, accessibility and availability of these SMS direction services was not adequately established in the course of this project. However, more information about these services could not be obtained thus making it hard to adequately compare and contrast them with this project. The services have only been mentioned, though more information about them e.g. how to use them, and if they work or not and whether the public can access them or not was not obtained. As far as this thesis is concerned, these services were treated as attempts to give direction services. Recall the main aim of the thesis was to provide information on routes and matatu for user chosen origin and destination search for directions and buses in addition to a route map.

Similar works and projects have been implemented in many developed countries. For example in Helsinki, Finland, a reliable and up to date bus information service was developed years ago and is currently still in use. According to Fleishman et al (2003), this bus information service acts as a journey planner service where all information needed in travelling could be found easily and in one package at any time when needed [59]. This application can be found online at:
http://www.hsl.fi/EN/Pages/default.aspx (visited on 16.02.2011).

This service allows a user to search for best transport connections between an origin and destination using all buses, trams, commuter trains and metros. It also includes walking edges in case a user needs to walk some distance to the first stop or from the last stop in order to reach the destination. This service also provides real time information (arrival and departure) at bus stops and transfer points. At the time of this research, this service provided additional services including incorporation of a route map showing the path followed from start stop to destination.

Table 3: Overview of related projects

Capability and service	Arena SMS	SMS Direction Service	Google SMS service	Finland Bus System	**Proposed Nairobi City Journey Planner**
Directions	Yes	Yes	Yes	Yes	**Yes**
Bus information	No	Yes	No	Yes	**Yes**
Trip distance	No	No	Yes	Yes	**Yes**
Route map	No	No	No	Yes	**Yes**
SMS	Yes	Yes	Yes	No	**No**
Browser based	No	No	No	Yes	**Yes**
Stand alone or downloadable	No	No	No	No	**Yes**
Travel time	No	No	Yes	Yes	**Yes**
Implemented?	Yes	No	Yes	Yes	**Yes**
Charges	Not free	Not free	Free	Free	**Free**

The above table (Table 3) shows the various issues considered in comparing the proposed project to the existing, previously implemented services. Yes and No denote the availability and unavailability of a given feature respectively. From the table, it can be seen that SMS based service has been a common platform for the Kenyan market. Maybe, this sheds light for the future in view of the proposed project. This project initially developed a standalone downloadable mobile client to be installed on a phone once and for all. There were opinions to develop the same on SMS platform but this would be one in future work if found appropriate.

So far the browser based application has had the least previous work done in the market as none existed previously. Only SMS related direction services have been attempted in the Kenyan market. In contrast, the proposed Nairobi journey planner did not provide bus time tables and real time connection information since it was based on basic bus stop and route information without consideration of time factor. There were no operational bus timetables found in use.

METHODS AND MATERIALS

This chapter deals with the methodologies, materials as well as technologies employed towards development of a mobile and web - based journey planner application.
The work flow for the entire project involved the following aspects and issues:

- Project definition and conceptualization.
- Data acquisition and preparation.
- Formulation of routing algorithm.
- Database design.
- Demo version.
- Website design.
- Mobile client design.
- Testing and collection of user feedback.
- Final version release.

3.1 Project Definition and Conceptualization

This involved collection of user requirements through assessment of several user stories. The user stories express the wish and the need of a potential user of the application to be developed. It also enlightens a developer on the application use from users' perspectives. User stories also assist in formulation of goals and objectives of the project in order to meet the user needs as well as benchmarking the application development process. Definition of a project usually forms the initial draft before any actual project work begins. It gives an overview of the proposed project and the methodologies to be employed to accomplish a successful project. A comprehensive project plan also ensures that everything required for the project works together towards realization of the goals and objectives of the project. These include data, technologies and their limitations as well as the steps or methods to be followed or to be employed to realize a successfully working project.

3.2 Project Work Flow

The diagram below shows the various steps followed in the execution of the project.

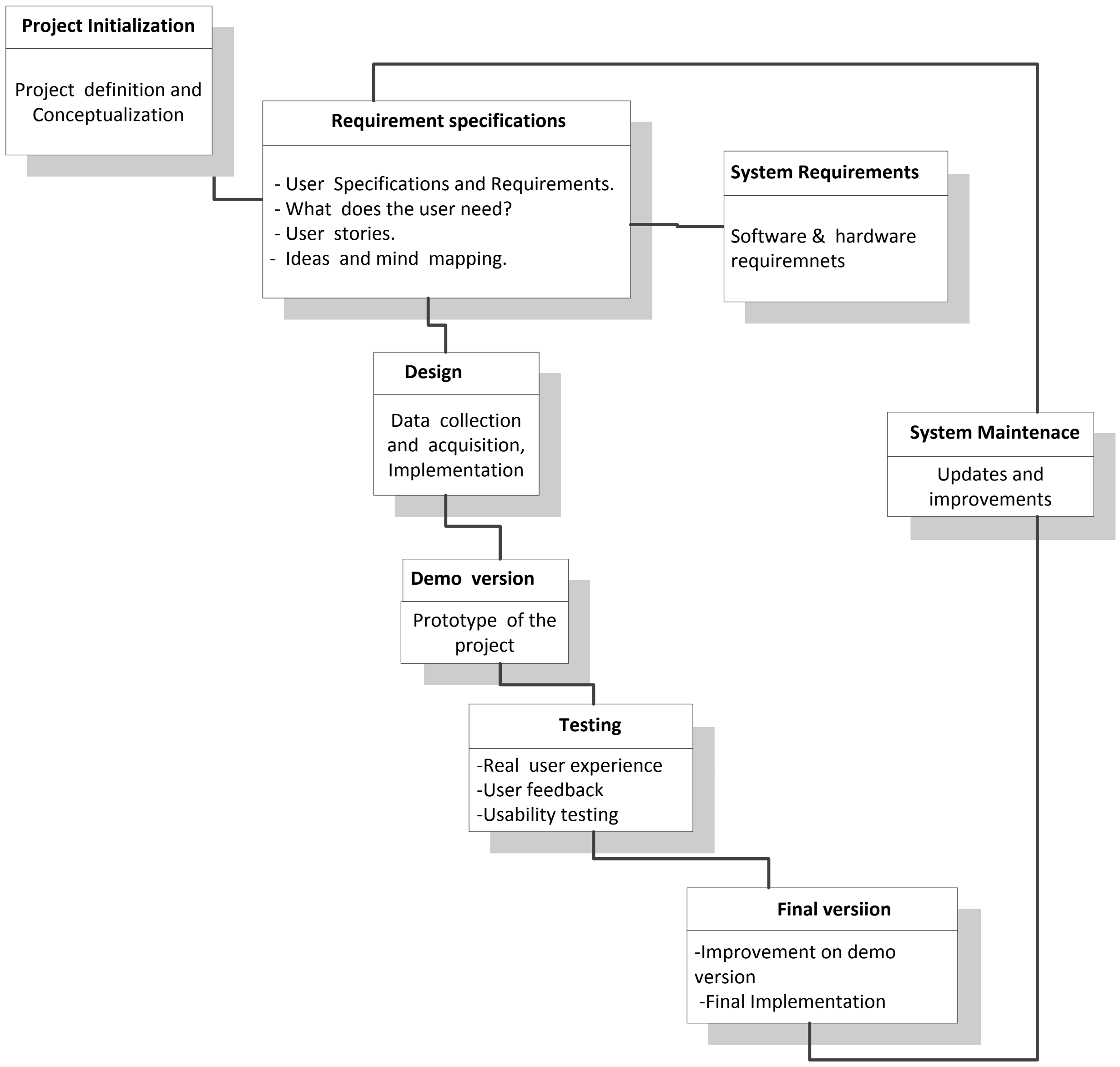

Project Initialization
Project definition and Conceptualization
Requirement specifications
- User Specifications and Requirements.
- What does the user need?
- User stories.
- Ideas and mind mapping.
System Requirements
Software & hardware requiremnets
Design
Data collection and acquisition, Implementation
System Maintenace
Updates and improvements
Demo version
Prototype of the project
Testing
-Real user experience
-User feedback
-Usability testing
Final versiion
-Improvement on demo version
-Final Implementation

Figure 14: Project work flow

Once the project definition was complete, user stories were gathered as the next task. Figure 14 above illustrates the various steps and procedures employed in development of the application. Starting with conceptualization of the project, user stories were collected to determine what the users expected the system to fulfill. To deliver an application that satisfies users, mind mapping, brainstorming of ideas and inputs from concerned parties and from the members of the community and even by use of online social media sites like face book (see appendix 8) were used. The design dealt with data collection and acquisition from transport industry and mapping institutions in Kenya. After the design, a prototype application was developed and preliminary testing done. From the user feedback an improved demo version would lead to a final version. Since the online version needed to be hosted and updated, system maintenance was to be done to ensure the system remained up to date and addressed user needs as much as possible.

3.3 Requirement Specifications

3.3.1 User stories

The proposed journey planner application idealizes a situation where a typical user who lives in Nairobi wanted to know how she or he can connect from one point to another through Nairobi's metropolitan city prior to or on the fly during a journey by public means of transport. The user had a rough background about various place names but did not know how to get from one place to another especially by public means. Every popular place or estate had a route number and bus stops that served it and 'matatus' (public means of transport) that plied that route.

The shortened version of the above becomes:

"As a Nairobian I want to plan my journey myself in advance to travel from one place to another by 'matatus' with minimum asking for information from anybody as it saves my time and helps me plan my journey better."

The above information was gathered (actually overheard) from one out of a collection of city residents who expressed their wish for a service that focuses on public transportation in the city during the researcher's stay in Nairobi, Kenya. The user stories motivated the researcher to think of a project that addressed the users' sentiments. The chaotic and disorganized transport in Kenya inspired the researcher to come up with this study. The user stories had been gathered over the years during the researchers stay in his home country. Most user stories were gathered from casual street interviews.

3.3.2 What does the user do?

This is what the user is expected to do and what the application delivers:

- Enters the name of the place she/he is in (preferably a popular place or known bus stop or name) or any another place the user wishes to connect from as origin. (See Figure 15).
- Enters the preferred destination name or a place of interest.
- Request sent to server via http connection and response generated and shown on the user's phone.
- Response could be: Bus route and connection information relevant to the user.

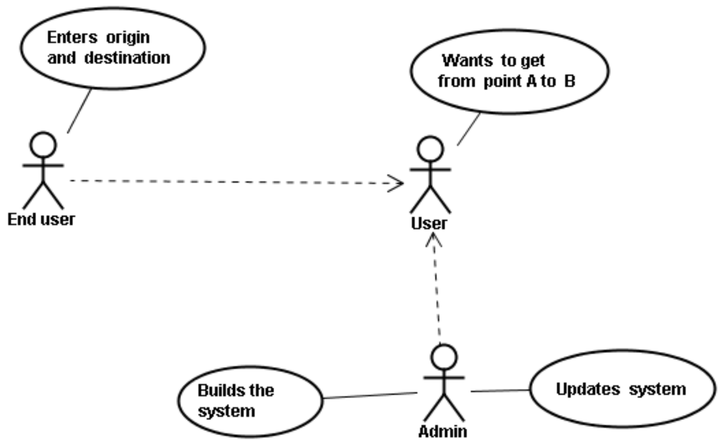


Figure 15: User requirements abstraction

Figure 15 above describes the mission of this project in an abstract manner as it roughly describes the concept of the project as given in the user story. An admin needs to maintain and update a system that addresses the user needs who needs to move from point A to B.

3.4 System Architecture

The overall system architecture is shown below in Figure 16.The diagram shows the components and the interaction of various components in the proposed system. The mobile user and online user are also shown in the sketch. This diagram was meant just to show the components in brief. Detailed system architecture is shown in the final system architecture.

The conceptual system architecture is given in diagram (Figure 16).It consists of the following features:

- Shared web server between browser based and mobile clients.
- Google maps server connection to overlay KML on map to plot route data.
- Routing algorithm: Dijkstra shortest path algorithm.
- Database (MYSQL) RDBMS.
- Graphical user interface where user interacts with the system.

These are the features that will be developed and interfaced to enable the realization of the project goals. The development of the application encompasses the development of each of the above features.

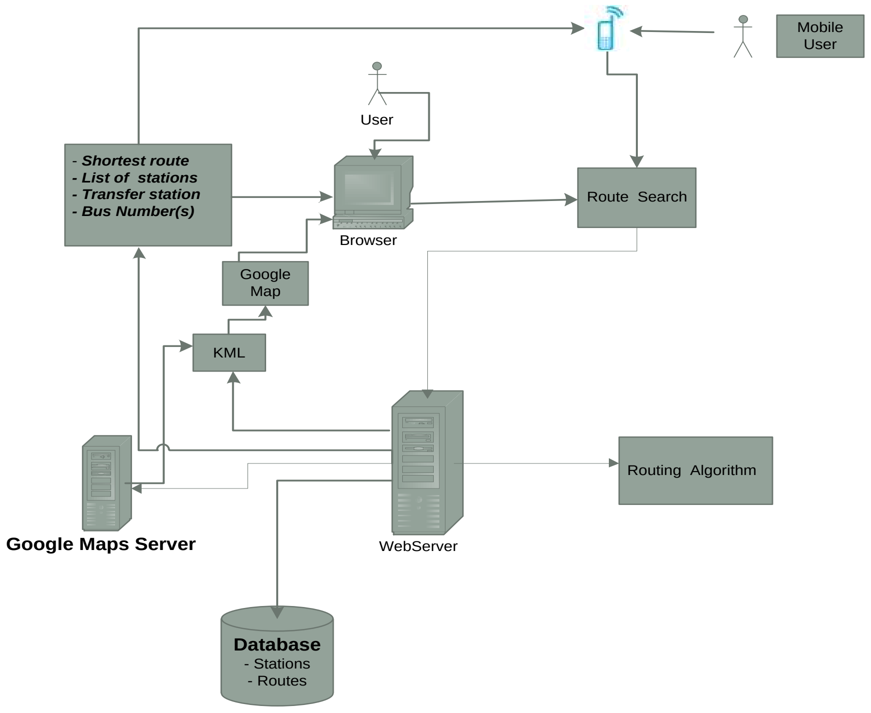


Figure 16: Conceptualized System Architecture.

3.5 Web -Based Journey Planner Application

This acted as a browser based application where a user visited the application URL on a browser (usually on a computer or mobile browser) and was able to search for matatus routes or bus route information. By selecting origin and destination stations a route was computed by the routing algorithm and with database connection, station names and bus numbers were presented to the user as the result. The resulting route could then be visualized on Google maps by overlaying of the KML file generated from the user selections and search. The map could be panned and zoomed for readability.

3.5.1 Mobile Journey Planner Client

Conceptualized system architecture for the proposed mobile journey planner client is shown in Figure 17 below. Figure 17 shows the components used in the design of the mobile client. The diagram serves to show a sketch of the mobile client in an attempt to give an oversight on what was used in final system architecture.

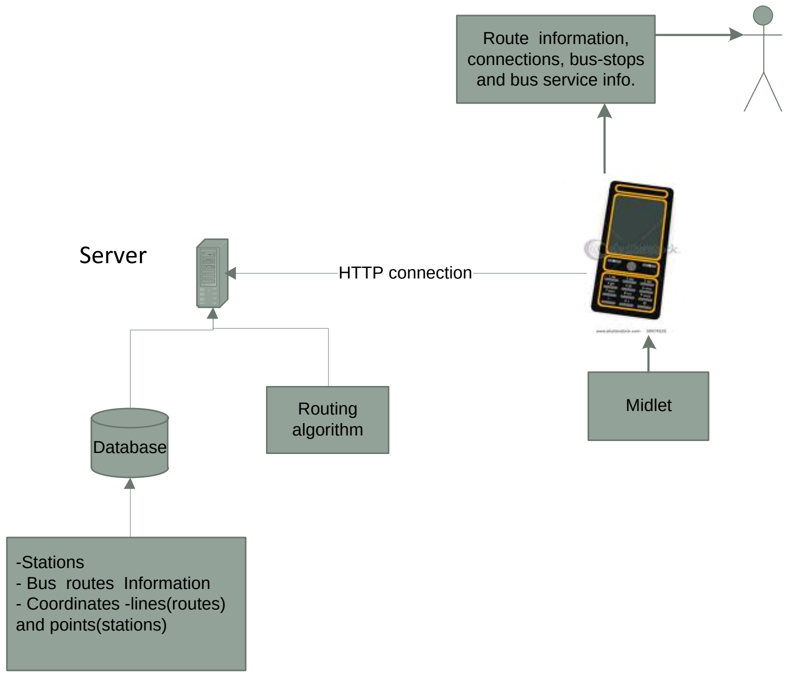

Route information, connections, bus-stops and bus service info.
Server
HTTP connection
Database
Routing algorithm
Midlet
-Stations
- Bus routes Information
- Coordinates -lines(routes) and points(stations)

Figure 17: Mobile journey planner client system architecture.

The Requirement Specification for Mobile Journey Planner client included:

- Platform: Java enabled mobile phone: Midlet application.
- User Input: Text based: Origin and Destination.
- Output: Response as text or static map.
- Target Devices: Nokia Series 40 as a starting point.
- GPRS enabled phone: For internet connection and data communication.
- MySQL database: Coordinates, names, route data, bus stops IDs etc.
- Web server: Apache.
- Routing Algorithm: PHP based.

These requirements focused on the common mobile platform or model used by a majority of Kenyans. The project aimed at developing an application that would work on these phones or at least for the majority of the Kenyan population.

3.5.2 How does the Mobile Journey planner work?

From Figure 17 above, a user could use a mobile phone to access the same services offered on the online application by first having the application installed on the mobile phone. The application allowed a user to search for matatus routes by keying in origin and destination station names. The application composed the request and sent it to the web server that had connection to databases and routing algorithm through HTTP GET method. The response from the server contained the route information, lists of stations from the start to the end station, total distance of the journey, bus transfer station if applicable and matatus numbers that the user can use in the journey.

The mobile journey planner application was a MIDP application: Midlet. This Midlet contained the station lists stored as *arraylist*. The user could key in both origin and destination station or select from the available stations listed in the station list viewable in the mobile application. Only available stations could be used in the route search.

3.5.3 Things to consider when designing the mobile journey planner client

Since the proposed mobile journey planner client was a thin client with limited bandwidth and limited processing power, the following issues were considered.

- Packet data arrangement availability for users for data exchange between web server and the mobile application.
- Optimizing data traffic by keeping the size of data to be exchanged across the HTTP connection to be a minimum as that translates to an added economic value.
- Limited storage on server in case data volume became huge.
- It is possible to plot a route on Google maps from KML. If the application could parse KML and plot on static map returned to user.
- Cost of internet connection in Kenya by mobile phone is expensive. Rates are based on amount of megabytes transferred
- If the application was needed to be in local language (Swahili) though English is commonly adopted language (Localization).
- Processing desired routes and returning a KML file with some information displayed on the map. The possibility and usefulness of incorporating maps on small screens of mobile phones.

3.6 Project Design

3.6.1 Data Acquisition and Preparation

The core of the project was dependent on the availability of data. The data used in the project included the following:

- **Road shape files (vector data)** - These were acquired from a government mapping agency in Kenya. This data consisted of all classes of roads, place names and their attribute information. This data was used in delineation of matatus route information and also plotting of the matatus route in the Google maps. The selected data from these shapefiles was exported as KMl file and viewed in Google Earth. In Google Earth, the location of bus stops and routes including intersections could be better viewed and adjusted. This meant a new KML file containing routes (lines) and station (points) and related attribute information was what was needed to hold the spatial information related to the study area. The information was later stored in a database. From KML file, coordinates data, station names and road names could be extracted and stored in the database. Metadata for data was given as follows:

Source: Kenya Roads Board
Projection: Universal Transverse Mercator (UTM)
Datum: WGS84
Units: Meters
Scale: 1:10000

- **Bus Route information** - This was data about buses and the routes followed by a given bus number. This made it possible to relate the spatial data to the bus route information thus it was possible to define any bus route by use of coordinate information. This was obtained from Kenya Bus Management Limited, Nairobi, a public transport Bus Company operating around Nairobi city though for only few routes in the study area.

- **Bus Station information** - This was information about terminus name, destination and route description of various bus numbers. Main terminus was where the passengers could board matatus. This was useful in the application in the determination of information about where to change buses, what terminus to go to and the bus number that plied to a given destination. This was obtained from Kenya Bus Management Company.

3.6.2 Network Design and Routing Algorithm

This involved the actual implementation and design of the network of nodes and links to be used in the routing algorithm for determination of the optimum paths. The nodes are the bus stops or stations while links or arcs are links between bus stops (see Figure 19). Bus stops were denoted as point data and links between stations as lines. Some curvilinear lines had to undergo generalization. Some roads were made of several links joined together but stored as separate entities.

3.6.3 Network Modelling

In order to formulate the network to be used in the development of the application, a hypothetical example of structure of the network adopted is given in Figure 18 below.

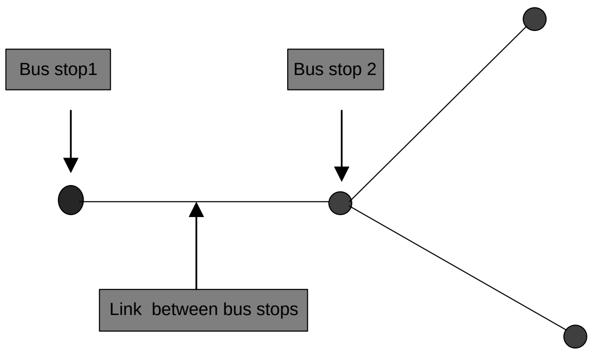


Figure 18: Modeling the network: Arcs and nodes.

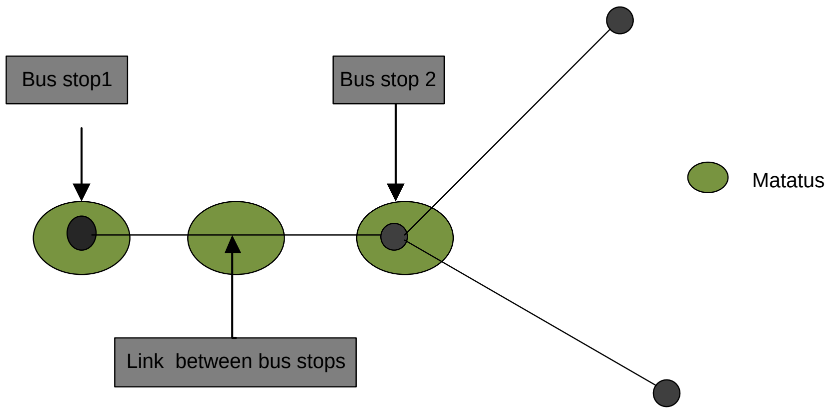


Figure 19: Relating Arcs and nodes to Matatus Numbers Information.

In the modeling of the network, it was maintained that every station (node) and link between stations (arc) was related to the bus numbers that pass through it. Hence it was possible to query database for all stations that any given bus number passes through. Similarly, it made it possible to query for the links that same bus number plies through. Eventually, all the stations and links that the bus number passes through can be plotted on a map and visualized (see Figure 19). Every arc was made of two nodes one at each end.

Every arc carried a weight factor or a cost value. Distance between any two nodes in a link became the weight value of that link. Since the network model consisted of directed graphs, the weight of a link in one direction was not always equal to the weight of the same link in the opposite direction. For instance, the city of Nairobi has one way streets and two way streets. Some links were passable from one direction only. The weight of the same link in opposite direction was assigned to infinity or a very large value. The weight of the link denotes the cost of crossing that link. This was done for links and directions and an adjacency matrix was formulated. This was simply a list of arcs defined by two nodes and the cost factor. Stations were numbered from 1 to 173. For example, a link connecting stations 1 and 2 with a cost value of 5 was denoted as (1,2,5) and if it is one way link, then the opposite direction took the form (2,1,∞) in the adjacency matrix (see appendix 3). However, it is worth noting that the notations (2, 1, ∞) and (1, 2, 5) point to specific locations (row and column) in the matrix and the value at that location is the cost value. For example, position (1, 2) is assigned 5 and (2, 1) is assigned ∞. Similarly, this was done for all available links until the whole adjacency matrix was formulated. The station numbers were adopted as the station IDs in the database. The weights between any two stations were taken to be the real measured distances between the two on the ground. One way streets had ∞ weight for prohibited directions and actual distance on the 'legal' direction.

The routing algorithm required this adjacency matrix for computations of the optimum routes. The routing algorithm computed the optimum path from any of the given station to every other station based on the cost denoted in the adjacency matrix.

Once a user inputs origin and destination, an optimum path is computed and the result of the path is given as a list of stations IDs from the origin to the destination. An SQL query was run in the database to obtain station names since the output from the algorithm were just numbers or station IDs which form the primary keys for the tables. Hence, relevant information about any station could be obtained and displayed to the user based on the IDs e.g. matatu numbers and station names. In order to achieve the above, a database that incorporated bus route information was necessary.

3.6.4 Database Modelling

The database was modeled based on the network model adopted. The most important task was to make it implement relational database management system requirements. This meant that relationships between various database entities had to be expressed. To maintain database integrity constraints, use of join tables to express relationships between tables was used. The following were the tables created in the database (See Figure 21).

a) **Bus stop table** - This was created to store station information for example, station name and coordinates (latitude and longitude).This was used to store station data as point data or nodes.

b) **Matatus numbers table** - This contains information about matatus i.e. Matatus number, origin, destination and route description.

c) **Link between stations table** - This was used to store arc or line data defining the links between stations. Every link was defined by two nodes (one at each end), link name and a string of coordinates that make up that link.

Join tables were used to link bus stops to matatus numbers and to link the arcs or links to matatus numbers. Matatu numbers are the bus route numbers plying a certain route through certain stations and along a given link. Every route around the city of Nairobi and its environs has a matatus number assigned to it and every route is made up of several stations or bus stops located along that route. These stations are locations where passengers could board matatus.

3.6.5 Entity – Relations Model

The formulated database model is hereby given below. PK represents primary key while FK represents foreign key. Foreign keys were used to enforce constraints in the database, where one entity in one table depends on or is related to another entity in another table. For example, the foreign keys here ensure that no single link has no associated information with two bus stops since two bus stops reference or define the end points of any given link. With relationships established across the tables, it was possible to run join queries and relate matatus numbers to bus stops and to the links. The database model was later mapped onto MYSQL database and populated with data. The entity relations model used is shown in Figure 19.Join tables were also made to make sure that a fully relational database model was achieved and to eliminate unnecessary redundancy and maintain integrity of the database.

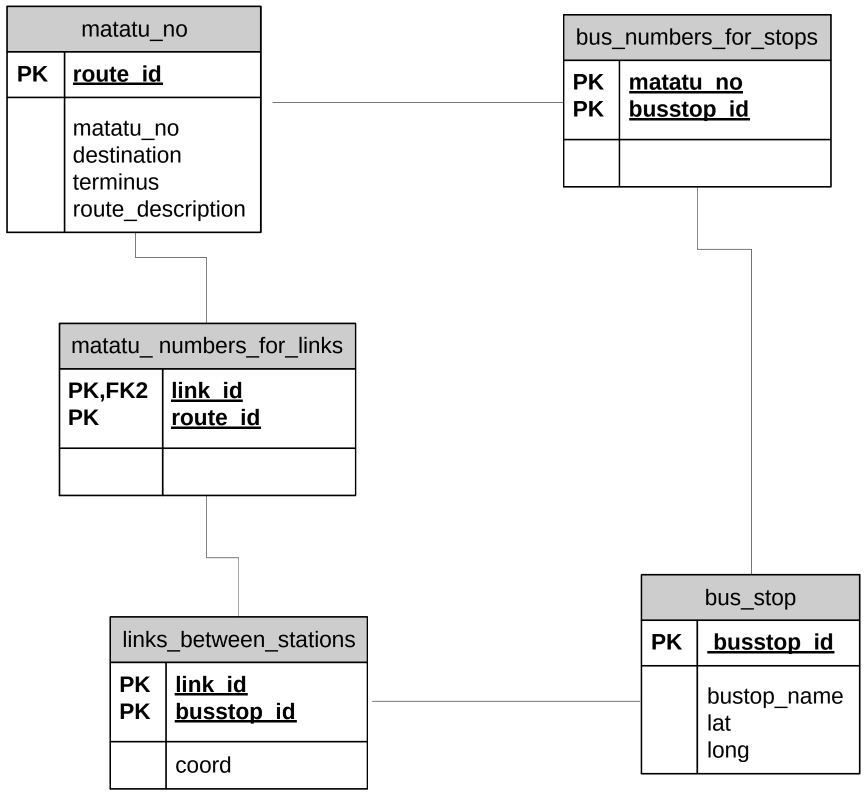


Figure 19: Entity relations model.

3.7 Final Application

The final application consisted of two parts, the online and the mobile version. The web version was developed using HTML web pages that were linked by navigation bars. The graphical user interface consisted of HTML forms where a user could select an origin and destination station and search for matatus routes information. The routing algorithm was done in PHP. The choice for PHP was because of its versatility and low learning curve involved in comparison to other languages e.g. Java Enterprise Edition, Adobe Flex, Action Script and ASP. However, Java Enterprise Edition required extra arrangement for an application server e.g. Tomcat in addition to web server and this came at very high cost for the final web hosting arrangements. Figure 21 shows the final system architecture adopted and implemented in this project.

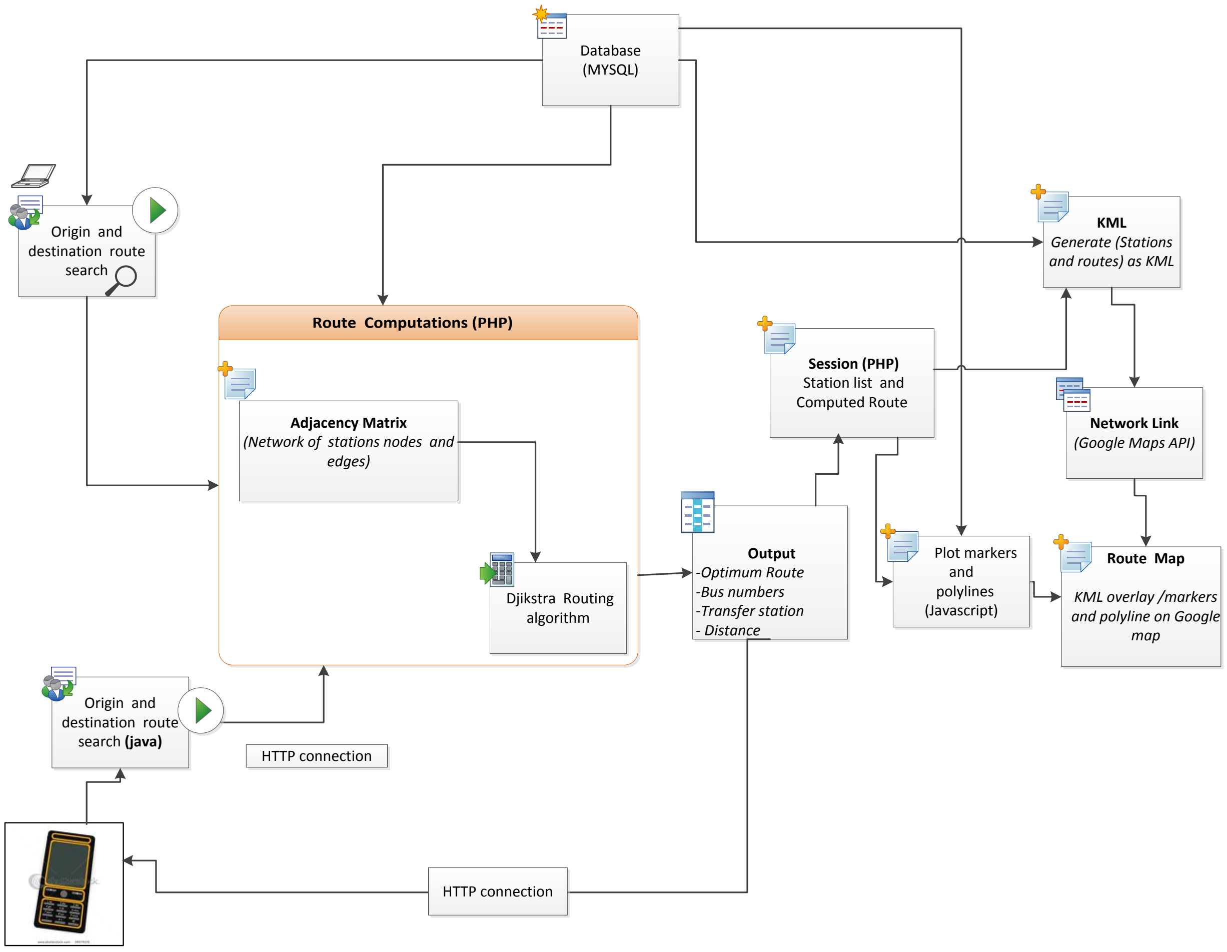

Database
(MYSQL)
Origin and destination route search
Route Computations (PHP)
Adjacency Matrix
(Network of stations nodes and edges)
Djikstra Routing algorithm
Output
-Optimum Route
-Bus numbers
-Transfer station
- Distance
Session (PHP)
Station list and Computed Route
KML
Generate (Stations and routes) as KML
Network Link
(Google Maps API)
Plot markers and polylines (Javascript)
Route Map
KML overlay /markers and polyline on Google map
Origin and destination route search (java)
HTTP connection
HTTP connection

Figure 20: Final System Architecture (for online and mobile clients).

The route computation was mainly done using Dijkstra routing algorithm (Adapted from GIS wiki [60], (see appendices 2 and 3)). The routing algorithm required the construction of the adjacency matrix for which the algorithm would use to determine the shortest path between any two stations. An excerpt from the adjacency matrix used is given in appendix 3. The algorithm found in [60] had to be customized to only show shortest path from one station to another and not from one station to all. The shortest path for any two stations consisted of only station IDs in an array. The algorithm also gave total distance of the computed path. The trickiest part was to query for station names based on results of computed paths. This was done using SQL thus names of stations could be obtained and displayed to the user.

Another challenging task was to plot stations and lines on the map. String manipulation was done to get link information from stations in the path. An example would be, say from station 1 to 5, the shortest path was given as 1-3-4-5. This means that we have stations 1,3,4,5 describing the shortest path from 1 to 5. It was possible to query from the database for station names given the IDs. The station numbers are the station IDs. There needed to be constructed a line (to be drawn on the map as the route).The primary key for line was in the form of start node-end node notation. For example, the line describing the above path would be in the form 1-3,3-4,4-5. As it can be seen, there is a relationship between the stations and the lines in terms of their IDs. To get the line information from station lists in the path, string manipulation was done. Therefore, the final result for the above example for route computation would yield stations 1,3,4,5 and link path as 1-3,3-4,4-5. It could then be queried for coordinate information for both a station and a line and plot the lines and stations on a map. A path was made of one or several lines joined end to end. Since every link and station had a matatu number associated with it, it was possible to give bus information based on origin and destination stations and buses departing from the two stations. Total distance for the paths was given by the algorithm directly. Eventually, it was possible to display a route map, bus numbers, station list and total distance to the user. This was the core of the project.

JavaScript was used to draw, load map and plot lines and points on it. Route information as a KML file was generated using *PHP DOM* functions with connections to *MYSQL* database. Styling was done in *CSS*.A network link file was made to link PHP script that generated the KML file and the JavaScript code that loaded Google maps and overlaid KML data. The network link bridges the KML generating script with Google's *GeoXML* server that retrieves the KML, parses it and overlays it on the Google map in the final application. The mobile version required a different approach due to limited computation power of phones. No map of the route was implemented for the mobile version. The application aimed to reduce the size of the data traffic between the phones and the web server. The cost for all the data transfer between the application and the server would be passed to the users. The resulting mobile journey planner application architecture is shown in Figure 22.

Users could use a text box to key in origin and destination station as well as select from station list. *Arraylist* java class served to enable the use of *arraylist* in the application to hold all stations since Java ME does not support *Java SE arraylists* directly. ‘Help’ and ‘About’ forms give orientation information about the application use and terms. The arrows depict the interaction or commands used to link between the various components in the application. Initially, there were opinions to implement the routing algorithm on the phone by pure java coding. It was suggested that since the application targeted low end phones with low computation power then, for large network computation would render the idea to be inefficient and unpractical.

It was decided to develop a small mobile client in Java ME (see appendix 5). Java ME was not the only language available in making mobile applications. Flashlite, X Code were some of the alternatives. Java ME was chosen because the researcher was well versed with it. It takes considerable time and resources to learn new languages. Since the online client was done in PHP, it was not very hard to interface the mobile client with this script by HTTP connection, hence, Java ME-PHP mobile client (see Figure 21). A java ME Midlet is given in appendix 5.

Figure 21 shows that the mobile client did not include the map but only displayed all other information available on the online client. Recall that the client sends a request by HTTP GET (see appendix 5) only sending origin and destination station IDs to the routing algorithm script in the server. The response from the server includes lists of station IDs making the path, total distance and matatu numbers. No station names were returned by the server since the station names were in the mobile phone array list. Running array search by indexing on the mobile client produced names for the stations. Eventually all the route information is displayed to the user. It was suggested not to include station names in the server response to save on the amount of data traffic in the mobile- server transactions and also to save cost for the end user.

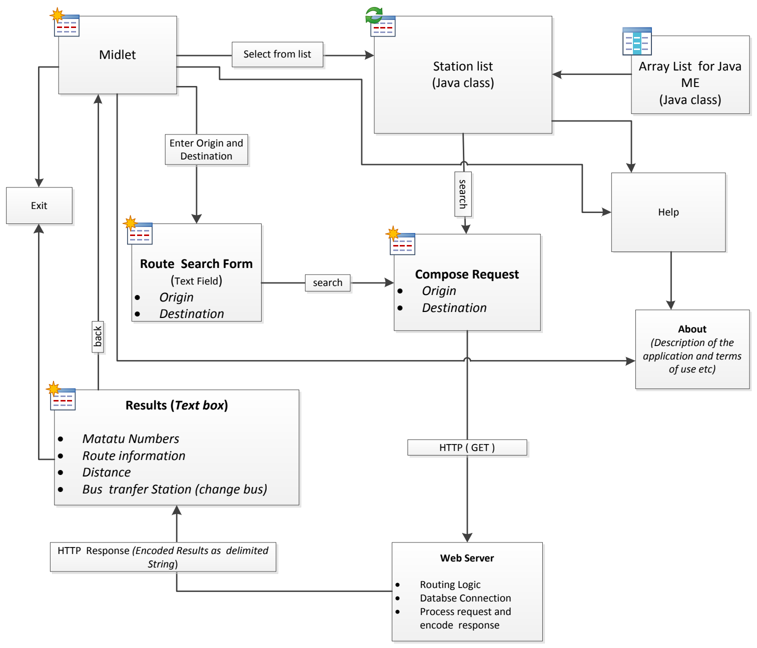


Figure 21: Final mobile journey planner architecture.

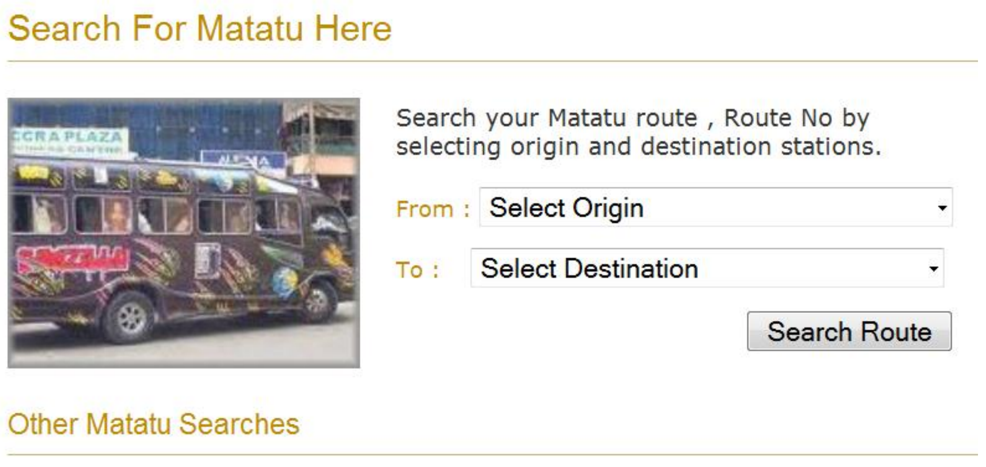
ANALYSIS AND DISCUSSION

This section covers information concerning the resulting application that was developed. These include development of a demo version, application testing and user feedback.

4.1 The Demo version

The working demo for the online application could be found on this link: http://www.matatuonline.com/index.php.

It is website that offered the online journey planner services to users for free. On the website, the user could choose origin and destination station and search a route based on that. The route search was performed using a form that had its select elements populated by station lists as shown in Figure 22.The graphical user interface was made as simple as possible to make it easier for users to use. Too much cluttering of information could be distracting to the user. Through simple navigation links, users could move to and from different pages with ease. However, some pages were hidden from direct links e.g. route search results and route map since they remained dependent on the user choices on the index page. Accordingly, these pages contained navigation links to related pages on the website.


Search For Matatu Here

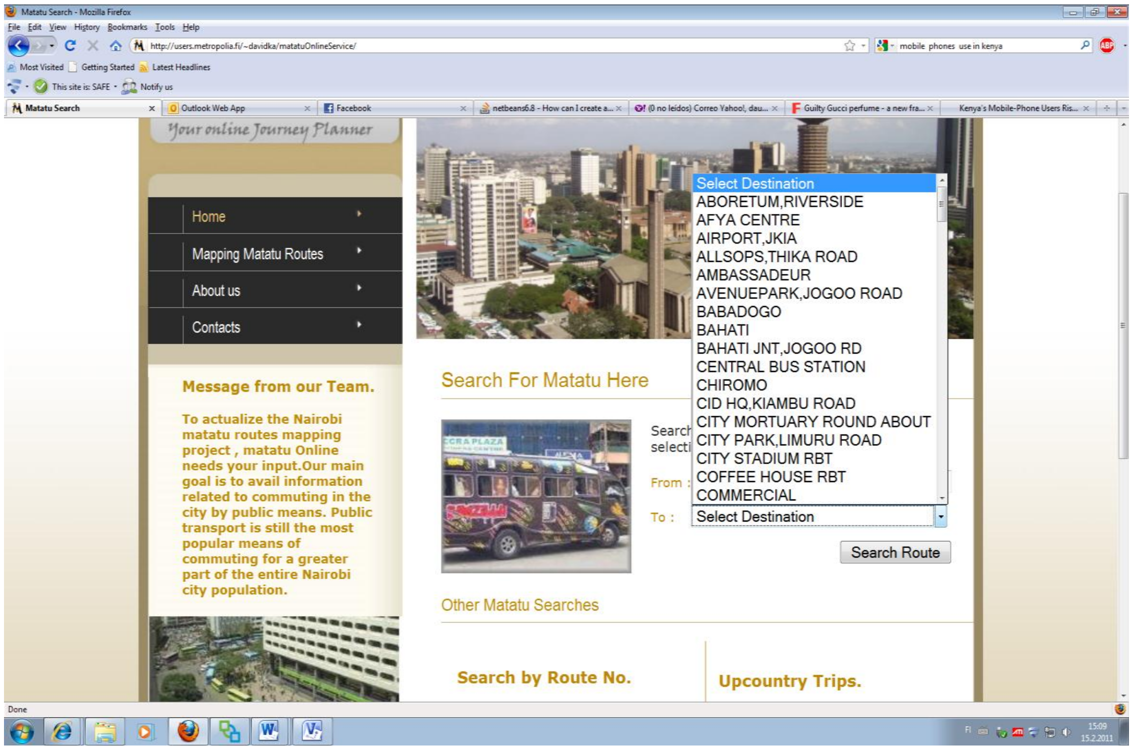
Search your Matatu route , Route No by selecting origin and destination stations.

From : Select Origin

To : Select Destination

Figure 22: Route Search form.

Since the stations were listed on the form, the user could just choose the origin and destination stations and search the route information; see Figure 23.The default browser used was *Mozilla Firefox.* Other browsers like *Google Chrome, Internet Explorer, safari and Opera* also ran the website in a similar way.



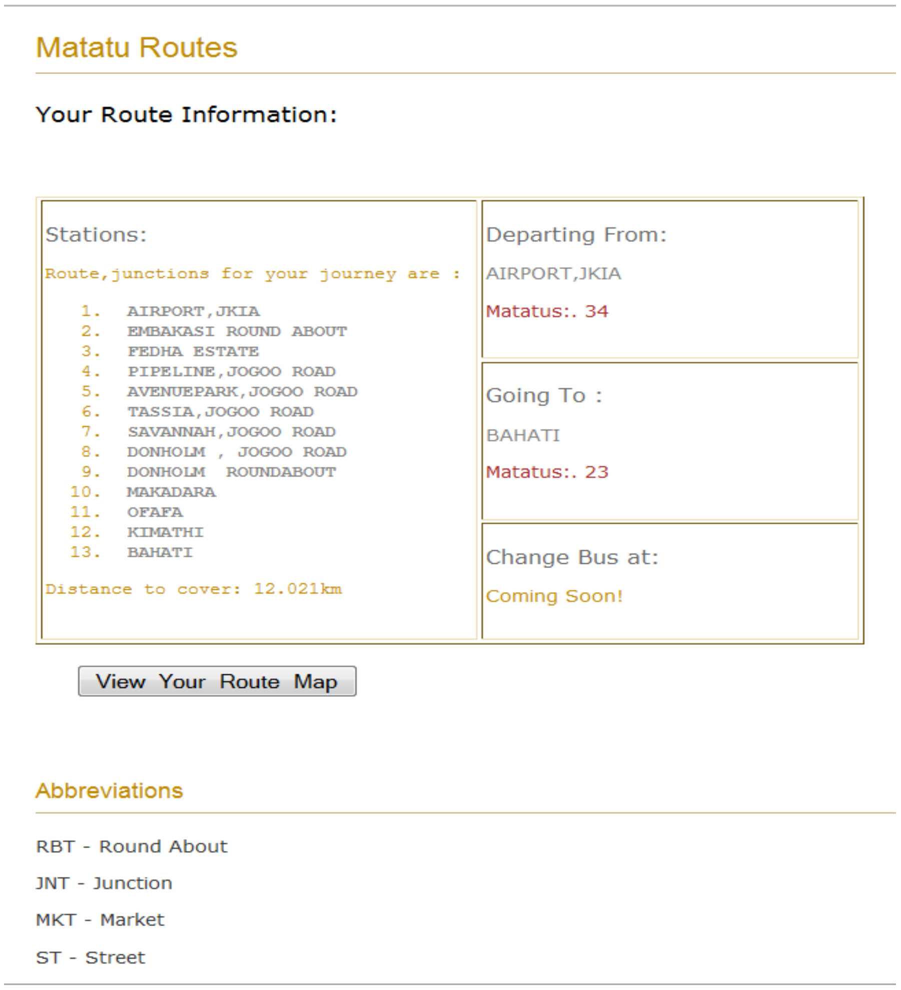
Message from our Team.

To actualize the Nairobi matatu routes mapping project , matatu Online needs your input.Our main goal is to avail information related to commuting in the city by public means. Public transport is still the most popular means of commuting for a greater part of the entire Nairobi city population.

Figure 23: Selecting origin and destination stations from the drop down menu.

Mobile phone browsers were not used in testing this site. Once the user selected the route to search, only the station ID were sent to route computation page by HTTP POST. The resulting route information, matatus numbers, transfer stations and total distance for the route was displayed to the user, see Figure 24. At the time this test was done, transfer station(s) for the routes had not been implemented. Transfer stations are the stations where the user needed to alight from one bus and board another to continue the journey, in order to reach the desired destination. Most of these transfer stations were road intersections where certain routes shared similar nodes or bus termini where various buses to different routes could be boarded.

The result printed out a list of stations in an ordered fashion right from the origin to the destination. The first and last stations were the origin and destination stations respectively. Information about buses or bus numbers plying through the origin station and to destination was given in the result. The user could then use this information to plan the journey by knowing what bus number to wait for at the departure station and where to change to another until the trip was complete. Explanations to the abbreviations were also provided in the key below the results. Link to the route map was also provided in case a user needed to visualize the route.



Matatu Routes

Your Route Information:

Stations:

Route,junctions for your journey are :

```
   1.   AIRPORT,JKIA
   2.   EMBAKASI ROUND ABOUT
   3.   FEDHA ESTATE
   4.   PIPELINE,JOGOO ROAD
   5.   AVENUEPARK,JOGOO ROAD
   6.   TASSIA,JOGOO ROAD
   7.   SAVANNAH,JOGOO ROAD
   8.   DONHOLM , JOGOO ROAD
   9.   DONHOLM  ROUNDABOUT
  10.   MAKADARA
  11.   OFAFA
  12.   KIMATHI
  13.   BAHATI
```

Distance to cover: 12.021km

Departing From:

AIRPORT,JKIA

Matatus:. 34

Going To :

BAHATI

Matatus:. 23

Change Bus at:

Coming Soon!

Abbreviations

RBT - Round About

JNT - Junction

MKT - Market

ST - Street

Figure 24: Route Search Results.

4.2 Route Map

The route map was generated from spatial information (coordinate data) stored in the database with respect to the user defined search and the route information derived from the routing algorithm. There were two ways of plotting points and lines on a map. One method was by use of JavaScript (see Figure 25) to plot markers and polylines on the map. The other method was by KML (see Figure 26).

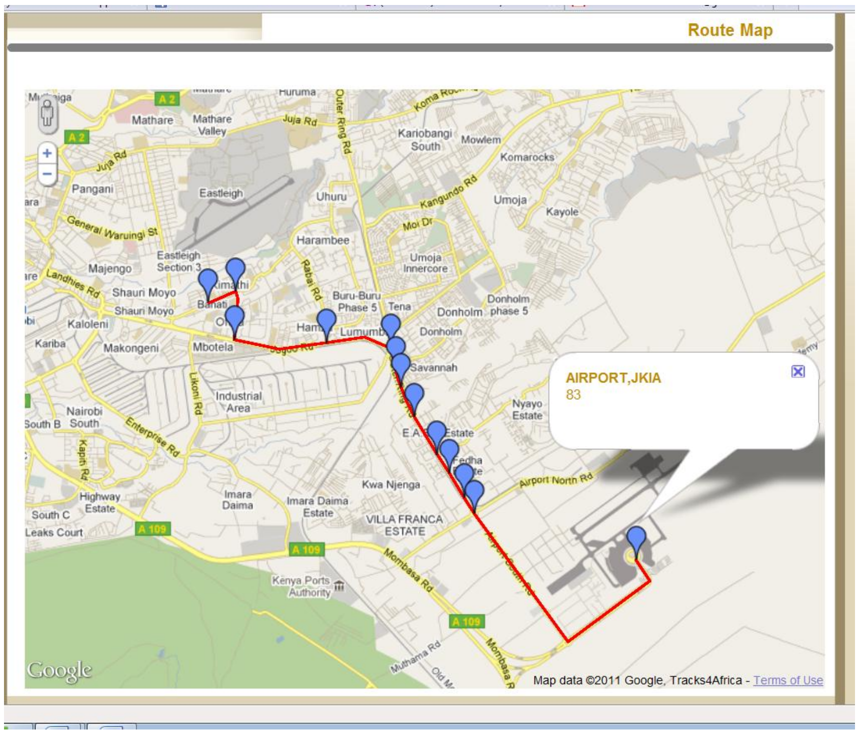


Figure 25: Route map generated by JavaScript.

The KML was output from a PHP script and overlaid on a map. Overlaying KML file on Google maps produced the following output in Fig.26 below.

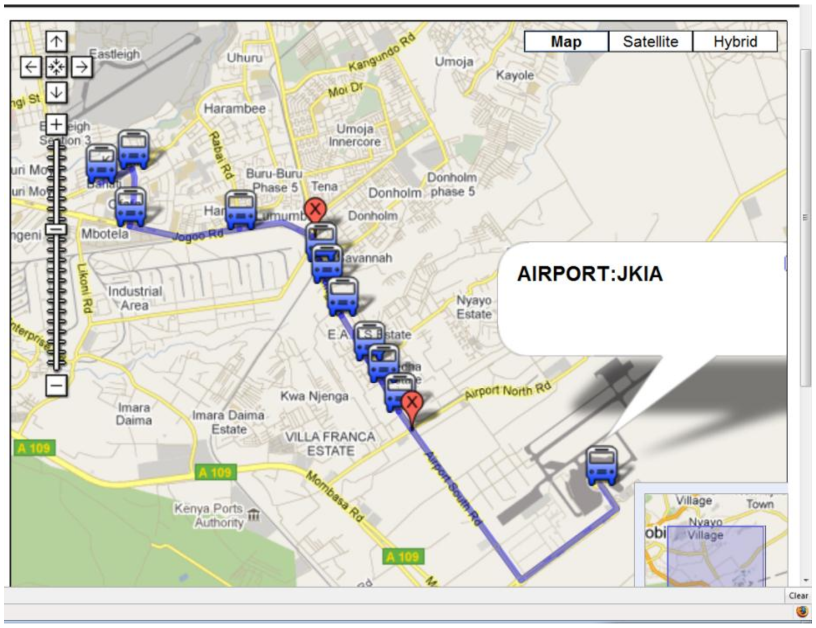


Figure 26: Overlaying KML on Google maps to show the route and stations.

From the two methods above, it was found that KML worked well with static data. There was a problem in overlaying dynamic KML on the map. Dynamic KML meant that for every route search, a KML file was generated. Only static KML was successfully overlaid on the map. To overcome this, JavaScript in combination with PHP was used to draw markers and polylines on the map. This has an effect of forcing the JavaScript to wait for PHP to ‘echo’ what to draw. Since JavaScript runs on the client browser, there was a significant delay in loading and drawing of the map. This was a serious problem especially for users with slow internet connections.

However, the KML approach was still maintained for future use and developments and it was possible to download the KML file and view the route on Google Earth client. This required Google earth client to be pre-installed by the user. Attempts were made to contact Google KML community to address that anomaly but no response had been obtained by the time this report was written. On a lighter note, the bus icons on Figure 26 appear to be bigger in size obstructing map reading to some extent. Figure 26 was just for demo purposes only. The final application was implemented according to Figure 26 given above. Table 4 shows the supported features and functionalities in the final application developed in this project.

Table 4: Supported features and functionality of the final application.

Features	Online journey planner	Mobile journey planner
Bus information	Yes	Yes
Trip distance	Yes	Yes
Route map	Yes	No
Route description	Yes	Yes
Platform	Browser based	Mobile phone
Access	Free	Free/Downloadable
Stand-alone/web based	Web-based	Standalone
Internet connection needed	Yes	Yes
Technology/programming	PHP, JavaScript, CSS	Java ME
Target devices	Smart phones and other browser enabled devices	Low end phones

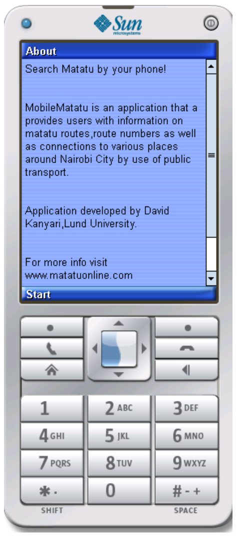
4.3 The Mobile Client

A mobile client was developed having very simple graphical user interfaces to enhance usability. Developing a mobile client was a complicated task since it required a different approach from the online version. The platform was no longer PHP but java ME. Selecting stations from a long list embedded on a mobile application could be a daunting task for users. Spelling mistakes for station names when a user entered a station name in the text fields would also be a drawback in using the application. It was proposed that a canvas that allows the use of game keys would be used in filtering of station names. This meant that if a user started to type some text on a text field, a list of possible station names matching that text would be displayed. As the user continued to type the list would become shorter and shorter until the desired station was obtained. This looked similar to the common way of searching for a contact from phone book in many phones.

The mobile client was built entirely as a MIDP application containing the following pages: About, Route search form, results form (text box) and choice group (stations list).The description of the application and what it does was given on the ‘About’ page. Help and instructions on using the application were given on the ‘Help’ page. Route search form and an option to search for routes by choosing stations from a list or by direct keying of station name using the text fields was provided in the application. Command interface was used to enable users to navigate between various pages in the application. An extract of the mobile client is given in appendix 5.

4.3.1 Using the Mobile Journey Planner

The application started by displaying the About page (see Fig.28) and the results (see Fig.29). To search for a route by direct keying in the station name, route search form was used (see Fig.29 and Fig. 30).

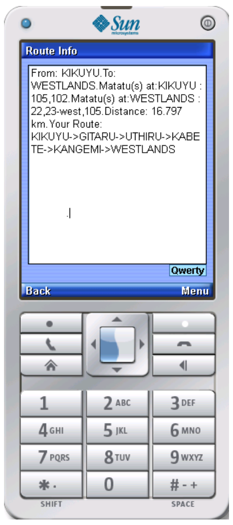


Search Matatu by your phone!

MobileMatatu is an application that a provides users with information on matatu routes,route numbers as well as connections to various places around Nairobi City by use of public transport.

Application developed by David Kanyari,Lund University.

For more info visit www.matatuonline.com



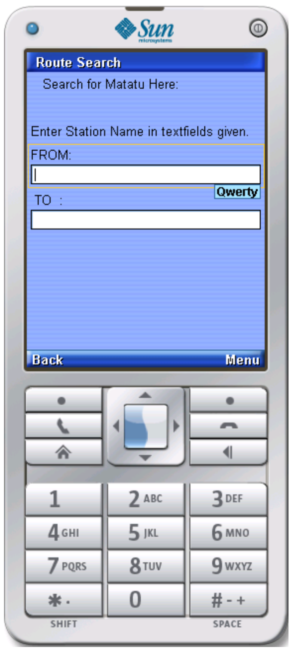
From: KIKUYU.To: WESTLANDS.Matatu(s) at:KIKUYU : 105,102.Matatu(s) at:WESTLANDS : 22,23-west,105.Distance: 16.797 km.Your Route: KIKUYU->GITARU->UTHIRU->KABETE->KANGEMI->WESTLANDS

Figure 27: About page for the mobile client

Figure 28: Route information.

If a user decided to choose origin and destination stations from a list then, station list (see Fig. 31 and Fig.32) would be used as an alternative option. These two options are discussed below.

a) Direct Keying in of station names. By using the text fields provided on the user interface, the user inputs the station names by manually keying the right spelling. Wrong spelling of a station name or blank fields in the search field was not allowed. The application handled this by use of alerts and warnings.


Route Search
Search for Matatu Here:
Enter Station Name in textfields given.
FROM:
TO :

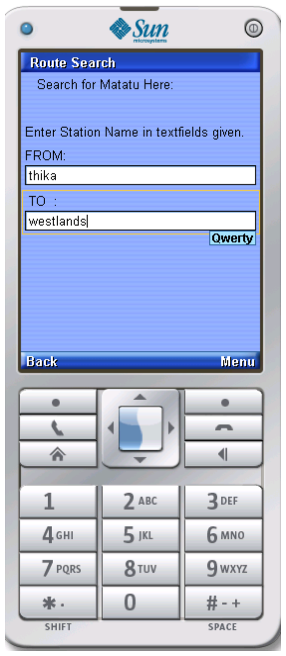
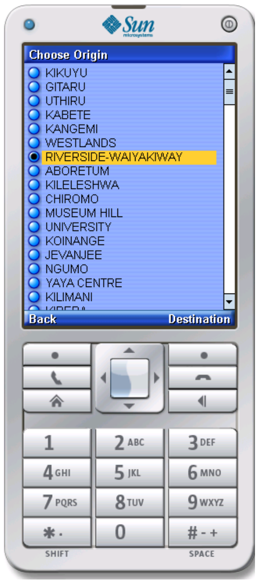

Route Search
Search for Matatu Here:
Enter Station Name in textfields given.
FROM:
thika
TO :
westlands

Figure 29: Route search form (empty)   Figure 30: Route search form (filled)

b) Choosing from a list: This was implemented to acquaint the user with the available bus tops to avoid spelling mistakes and warnings in case of wrong user inputs in the text field option (see Fig. 31 and Fig. 32).



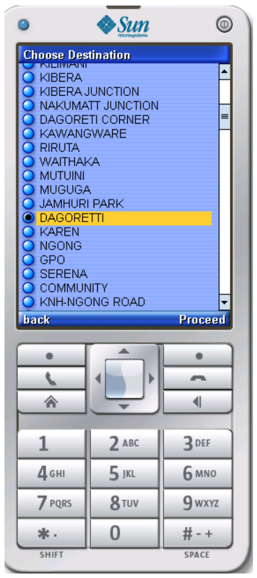


Figure 31: Choosing Origin from a list. Figure 32: Choosing Destination from a list.

So far, the user interface for the results output contained results as text only. Arranging and styling text in a mobile phone on Java ME format was not trivial. The main goal at this stage was to have the application running in readiness for user testing. It was planned that the feedback from users would play a vital role in the overall redesign of the user interface. Testing would be done on real devices.

4.3.2 Choice of Data Interchange Format

The choice of data interchange format to use in communication between the mobile client and the server was determined by several factors. These factors include cost and processing issues. Huge data traffic translates to higher costs passed to the user in using the application. In light of these, the following data exchange formats were considered:

a) JSON–JSON data format denotes data as key and value separated by full colon. An example of JSON data format used is hereby given as:

```
{"Results": {
  "from": "station A",
  "to": "station B",
  "distance": "7,52km",
  "Matatus": {
    "matatus number": {
" station A ": "4,7,33,34,29,30",
```

" station B ": "2,102,110”,
},
"transfer station":” station Z”,
"List of stations": “station A, station C,station D, stationE,station B”,
}}

Total size = 364 bytes

To use JSON, it would require JSON library [17] and additional code to be written on the client to read JSON data. This was attempted but there were data encoding and decoding problems and that would take too much time. This prompted the pursuit for alternative options like XML.

b) XML

The same text expressed in a simplified and basic XML format as follows:

<Results>
<from> station A </from>
<to> station B </to>
<distance> 7.52 km </distance>
<matatus>
<matatus number>
<station A> 4,7,33,34,29,30</station A>
<station B> 2,102,110 </station B>
</matatus number>
</matatus>
<transfer station >station Z</transfer station >
<List of stations > station A, station C, station D, station E, station B</List of stations>
</Results>

Total size = 452 bytes

However, XML required parsers that sometimes were heavy and slow as mentioned in the literature. There are many XML parsers available that were used in the project in a bid to determine the most suitable data interchange format to be used. This was attempted but the parsing made the operations of the service too slow and it added to inclusion of parsers in the client and more data overheads.

c) Custom method by use of string delimiters

The same information could be encapsulated into PHP variables and output by PHP 'echo' function and the required data held by variables.

echo " $from #$to #$distance#$matatus_A#$matatus_B#$tranfer #$route"*(see appendix 1)*

Total size = 114 bytes

Where PHP variables: *$from,$to ,$distance* denote the ' from', 'to' stations and distance respectively. *$matatus_A,$matatus_B,$tranfer,$route* denote the bus numbers at station A, B and transfer station and a list of stations in the route respectively.# represents the string delimiter used. See complete code in the appendix 1 where PHP script output one delimited string as the output to be sent to the mobile user.

This proved to be the best option since it required less coding and no inclusion of parsers or more libraries like JSON or XML. It proved to be more efficient and fast. This was the method that was used in the implementation, hence relieving the researcher of being tied to proprietary libraries. The goal was to use a protocol that worked with minimum coding to save data sizes.

Since station names could be obtained in the array list in the phone, only stations IDs were used. Parsing this string by using string tokenizers in the MIDP application proved superior in terms of data size and faster data processing. JSON and XML data formats required parsers and constituted larger data overheads compared to the custom method. From the example above, it was shown that using the custom protocol took roughly a quarter and a third of XML, JSON data sizes respectively. Mobile phone operators usually charge the costs for data transfer in relation to data size. The objective of the project was to make an application that offers minimum operation costs.

4.4 Application Testing

Software testing is a rigorous task to undertake especially in testing web based applications. This is due to the fact that a diversity of tests and approaches may be used in performing the tests. In this application, the testing was done by using use case tests namely:

a) **Functionality tests**

This focused on the overall coherence of the site's objectives and the output or if the application met its objectives and purposes with reference to user requirements and user stories stated in the methodology. This was conducted by cross referencing of user expected results with the output results from the service. This was similar to benchmarking the application by comparing its results to those on the ground. Input forms were also validated to check for invalid inputs. Orphaned pages, navigational links and buttons were also tested.

b) **Usability testing**

This involved testing of the product with real users in Kenya. The testing was more interested in observing and analyzing how the users completed tasks e.g. searching for routes, checking bus numbers, termini and destinations. This information later helped to analyze or determine user satisfaction of the application. This test was meant to test the design and flow of design logic and if the displayed results were satisfactory to a user. The main aim was to determine the design flaws, missing parts and to gather user experience. To enhance usability and ease of use of the site, images and brief instructions were included in the application. Every page contained a menu and navigational links.

c) **Content checking**

Spelling errors for station names and overall content were checked in this context. Furthermore, comprehensiveness and meaningfulness of the content and whether there were any missing and important information or parts were also to be considered. Legibility of the content, presentation and choice of site theme color were checked too.

d) **Interface checking**

This checked for any loopholes in the software design for example how invalid queries were handled and the overall look and feel of the application. Executional inconsistencies were also looked at. Web server-database interfaces and interactions were also examined and error checked.

e) **Compatibility testing**

This proved to be the most difficult task to do given a range of various web browsers available in the market. The application was to be made to be compatible and run on most of the browsers available. Various issues to be considered were:

- Available browsers: Mozilla Firefox, Internet Explorer, Safari, Opera etc.
- Operating systems: Windows, Linux, Apple Macintosh
- Mobile browsing and java script support issues

f) **Performance testing**

Since internet and data connection speeds in Kenya were low as witnessed by the researcher, there was a need to examine the rate of performance of the application in both high and low speed internet connection environments and also on mobile browsing. The issues to be addressed were the page loading rates and if the route maps on mobile phone browser and on computers had any significant differences in content and functionalities.

4.4.1 Testing methods

The following were the testing methods used in the user test:

a) Interviews

Live user interviews were conducted during the testing sessions. Oral interviews were conducted. Some users preferred this method to questionnaires since most of them had no internet access to air their feedbacks online and some complained of having no time to try the questionnaires. However, most of them preferred the mobile client as they always carry their mobile phones. During these sessions, the users were asked to test on their mobile phone the mobile client by searching for matatu routes and later their user experience and feedback recorded. This method proved to be inefficient, time consuming and inflexible and so alternative methods like questionnaires were adopted instead.

b) Questionnaires

This was carried out by posting an online and offline questionnaire where users would do specific tasks on the application as outlined. The questions spanned across the various application tests mentioned in the application testing namely; functionality, usability, performance, compatibility and interface tests. The main aim was to capture feedback from various cross sections of users with varying income levels, educational, hobbies, computer or mobile literacy and so on from the general population. Initially, it was purported that the most probable target user group was composed of young and middle aged people commonly referred to as 'dot-com group.' These were people who often have access to internet and computer and in most cases use mobile phones and computers more often.

For the mobile client testing an offline questionnaire was preferred as the users would fill in their feedback without the need for internet. The mobile users were informed that the application would cost little to use due to connection to web server for route computations. An example of the questionnaire used can be found in the appendix 6. Social media e.g. Facebook was also used (see appendix 8).

4.4.2 Results of user feedback

According to the user feedback obtained from 200 users whose ages range from 12-60 years old (see Table 5) and from different professional, educational back grounds and income levels the results have been presented below in the tables. The results have been presented in percentages (Out of the total number of participants who took part in the test). Questionnaires and brief oral interviews were used to capture the user feedback.

About 65% of the users preferred the mobile journey planner to the online version. According to them, they use their mobile phones more often and can even browse the internet on the phone. The online version possessed more features than the mobile version. Less than 10 percent of all the users owned a smart phone (see Table 7). Most of the phones available were low end phones with poor web browsing capabilities. Some phone brands had problems with the installation of the application. A big section of the users (70 %) complained of slow map loading due to slow internet connections. Internet connections in Kenya are mainly through USB – modem and normal wired internet, all of which give very slow connection speeds. Nokia and Samsung phones posed least installation and application running problems. Most of the users in the application testing recommended the extension of the service to cover more routes and if possible the whole of Nairobi city. Table 6 gives an illustration of the overall test results.

Table 5: Age distribution of the Users

Age in years	**Number of persons**
12 to 20	2
20 - 29	136
30 - 39	51
40 - 49	25
50 - 60	6
Total	**200**

Table 6: User feedback results in brief

Feedback	**Online Version**	**Mobile version**
Preference	35%	65%
Usability	78%	80%
Completeness	79%	83%
Performance	57%	59%
Map loading problem(slow internet connection)	70%	Not applicable

Table 7: Phone ownership information for the users in application testing

Model	**Number**
Nokia s40 series e.g. *Nokia 6500*	90
Samsung	40
Smart phones e.g. *Iphone,Nokia (N8,C3),Huawei etc.*	5
Others including *'Chinese cloned phones' etc.*	65
Total	**200**

However, the results might not be very analytic since the number of users in the testing was very small and also that the goal of the testing was mainly to capture users' look and feel of the application including acceptance and performance. It was proposed that the application needed to be further developed and more and more users to be incorporated in future testing. Appendices 7 and 8 contain a collection of photos taken during user testing in the streets of Nairobi City and some preliminary user feedback through use of social media (Facebook).

The site was easy to use with no broken links and had easy navigation between pages (see Table 6). However, minor issues like spelling mistakes for station names were noticed. Few compatibility issues were noted on the site especially in the browsers. Internet explorer and Mozilla Firefox showed negligible differences in compatibility. Roughly all the users had a clue about what the application was meant to do prior to testing it and even reading the information on the site. However, most users were dissatisfied with having to type a station name only to get warnings of spelling errors. They suggested that hints or suggestions be implemented to assist in typing the correct and available station names. They were also dissatisfied with choosing from a long list.

The users recommended the application to be used by tourists, tour operators, matatu organizations, business men and other transportation sectors. In the overall user feedback, the users were generally satisfied and excited by the application. Some regarded it as a vital tool in line with vision 2030 for Kenya as the country tries to embrace new technological advancements. In all the users feedback report, concerns were raised concerning bus transfer stations and also on the accuracy of the estimation in trip duration and other route information especially in extreme cases like traffic jams and alternative routes in case of road constructions. From the user feedback collected, there were several issues that needed to be fixed to improve the application in the future. These were:

- Addition of more stations and routes to cover the whole of Nairobi city.
- Implementation of route KML overlay as an alternative to JavaScript in plotting of routes as JavaScript contributed to overall slow loading of the map.
- To enable viewing of route map for each given matatu route numbers. This was fixed in the final application for a few selected route numbers.
- Improve look and feel for the mobile journey planner as it lacked style although it worked. Also, they requested *change bus* event information in case of bus transfers and to have the updated mobile journey planner on a web server to enable more users to access it and download the application.

CONCLUSION AND FUTURE WORK

Conclusion

As stated in the introduction and objectives, this thesis aimed at the development of both a browser based and a mobile based Nairobi city journey planner application. This application was conceptualized, designed, developed, implemented and user testing conducted in Nairobi, Kenya.
From the project's objectives, mapping of all existing matatu routes for the study area, including all bus stops and bus termini was done. However, it was not possible to cover the whole of the city due to lack of adequate data although this will be done in future developments.

A mobile and an online journey planner were developed, in which users could search for matatu routes based on origin and destination. The search results showed lists of stations along the journey, estimated trip duration, trip distance and a map of the route. However, the mobile version did not include a map due to small screens and poor displays for targeted phones.

From the feedback obtained from user testing, some issues concerning the application were raised. These included slow map loading, bus transfer event handling and addition of more stations and routes. These will be fixed in the future development or research.

GIS is a vital tool in analysis and presentation of spatial data especially in modeling of the real world. GIS in combination with computer programming aspects can produce powerful applications that can be used in making peoples' lives easier. With the mobile phone usage trend experiencing a major growth in Kenya, the need for more GIS driven mobile applications comes into play. In summary, the project was completed and implemented successfully and can be accessed online at: www.matatuonline.com

Recommendations and Future Development

In the recommendations from user feedback, it was proposed to have the project extended to cover the whole of the city. More research and collection of more data is required for future developments to address issues such as transfer stations, slow map loading and better estimation of trip durations. Since the project was the first one to be done for Nairobi, it was suggested that further work and improvement to be done. In addition to this, the project could be extended to cover other towns and also to be done in partnership with transportation sectors.

References

1. Sommer K, Fernando C, Tempra O. Nairobi Urban Sector Profile.UN Habitat Report 2004 p.6-10

2. King'ori Z. Decongesting Nairobi: Urban Transportation Challenges.2007; p. 3-19

3. Project Formulation Study on Nairobi Metropolitan Development Planning Project.Study Report March 2008, p. 2. (Online). Accessed 10.2.2011 http://www.ecfa.or.jp/japanese/act-pf_jka/H19/renkei/koei_nairobi.pdf

4. Ottichilo W. Spatial Integration Laboratory for Urban Systems (SILUS). Working Paper. Tracking Regional Growth and Development: The Nairobi Case.2007; (Online). Accessed 08.2.2011 http://gislab.wharton.upenn.edu/silus/Tracking%20Regional%20Growth.htm

5. Berg N. Traffic Costs Nairobi $746,000 per Day. (Online) Accessed 10.2.2011 http://www.planetizen.com/node/27264

6. Oketch O. Traffic Management in Nairobi Kenya. 2009; (Online).Accessed 10.2.2011 http://www.kenyaimagine.com/Politics-and-Governance/Traffic-Management- in-Nairobi.html

7. Ngirachu J. How poor planning is spelling doom for Nairobi .Daily Nation: Issue June 07, 2009 (Online). Accessed 11.2.2011 http://www.nation.co.ke/News/regional/-/1070/607158/-/7jar82/-/index.html

8. Trade invests Africa. Development opportunities for Nairobi city.(Online). Accessed 10.2.2011 http://www.tradeinvestafrica.com/investment_opportunities/165932.htm

9. Kumar A, Fanny B. Stuck in Traffic: Urban Transport in Africa. 2008; p.15

10. Ndong'a S. Kenya Smart Buses to ease traffic. 2009; (Online). Accessed 10.2.2011 http://www.skyscrapercity.com/showthread.php?t=950320

11. Korte G. The GIS Book: How to implement, Manage and Asses the value of Geographical Information Systems. 5th ed. 2001; p. 4-10

12. GIS.com. (Online). Accessed 20.3.2011 http://gis.com/content/what-gis

13. Hanson S. The Geography of Urban Transportation, 3rd ed. New York .2004; p. 160-170.

14. Economic and Social Research Institute (ESRI). Data Types and Models (Online).Accessed 09.2.2011
http://www.gis.com/content/data-types-and-models

15. Bernhardsen T. Geographic Information Systems: An Introduction.2nd Ed. 1999; p. 38.

16. Lakhan V, Chris. Introductory Geographical Information Systems.1996; p. 54.

17. Longley P. Geographical Information Systems Principles, Techniques, Management and applications.2nd ed;2005 p. 40-70

18. Demers, Michael. N. Fundamentals of Geographic Information Systems. p. 117.

19. Burrough. Principles of Geographical Information Systems, Spatial information Systems and Geostatitics; Oxford ;1998.2nd ed. p.30-50

20. Harvey J, Shih-Lung S. GIS-T Data Models. Excerpts from Geographic Information Systems for Transportation: Principles and Applications, Oxford University (Online). Accessed 09.2.2011
http://www.gisvisionmag.com/Book/miller_shaw.pdf

21. Hensher D. Handbook of Transport Geography and Spatial Systems. London, UK. 2004;Vol 5, p. 390-407.

22. Heywood I, Sarah C, Steve C. An Introduction to Geographical Information Systems. 2nd Ed. 2000; p. 60.

23. UnixSpace. Database Models (Online).Accessed 10.2.2011.
http://www.unixspace.com/context/databases.html

24. Kioskea. DBMS Models (Online). Accessed 10.2.2011
http://en.kioskea.net/contents/bdd/bddtypes.php3

25. Oracle. Sun Developer Network. Connected Limited Device Configuration (CLDC); JSR 139 (Online). Accessed 20.1.2011
http://java.sun.com/products/cldc/

26. Sams Publishing. MIDP Programming with J2ME,2002 (Online). Accessed 20.1.2011.
http://www.developer.com/java/j2me/article.php/10934_1561591_5/MIDP-Programming-with-J2ME.htm

27. W3 Schools. HTML Introduction, (Online). Accessed 10.2.2011
http://www.w3schools.com/html/html_intro.asp

28. Hemphil D.UsingHttpConnection in J2ME (Retrieve web content from a website to a phone), 2008 (Online). Accessed 10.1.2011
http://www.java-samples.com/showtutorial.php?tutorialid=739

29. Economic and Social Research Institute (ESRI). Data Types and Models (Online). Accessed 09.2.2011
http://www.gis.com/content/data-types-and-models

30. Ghosh S: Add XML parsing to your J2ME applications (Online). Accessed 01.2.2011
http://www.ibm.com/developerworks/library/wi-parsexml/

31. Ortiz C. Using JavaScript Object Notation (JSON) in Java ME for Data Interchange,2008 (online). Accessed 10.1.2011
http://java.sun.com/developer/technicalArticles/javame/json-me/

32. Jimmy. Java ME (J2ME) JSON Implementation Tutorial/Sample (Online). Accessed 10.2.2011
http://jimmod.com/blog/2010/03/java-me-j2me-json-implementation- tutorialsample/

33. Tavon Org. JSON- J2ME (Online). Accessed 08.2.2011
http://tavon.org/work/JSON-J2ME

34. Nicholas C, McPeak J, Fawcett J. Professional Ajax. Indianapolis, IN. 2006; pg 194-198.

35. JSON Org: The Fat-Free Alternative to XML (Online). Accessed 10.2.2011
http://json.org/xml.html

36. Oracle. ClassStringTokenizer (Online). Accessed 10.2.2011
http://download.oracle.com/javase/1.4.2/docs/api/java/util/StringTokenizer.html

37. Tim F. User /centered design.2006.(Online) .Accessed 10.2.2011
http://www.webcredible.co.uk/user-friendly-resources/web-usability/user-centered-design.shtml

38. Usability first. User –testing (Online). Accessed 10.2.2011
http://www.usabilityfirst.com/glossary/user-testing/

39. Buzzle.com .Manual Testing Interview Questions (Online). Accessed 10.2.2011
http://www.buzzle.com/articles/manual-testing-interview-questions.html

40. Sanjeev A, Arunadevi J, Mohan V. Intelligent Transport Route Planning Using Genetic Algorithms in Path Computation Algorithms. European Journal of Scientific Research . Vol.25 No.3 (2009), pp.463-468

41. Mainali M, Mabu Y, Eto S, Hirasawaa K. Dynamic Optimal Route Search Algorithm for Car Navigation Systems with Preferences by Dynamic Programming: IEEJ Transactions on Electrical and Electronic Engineering. 2011; 6: 14–22

42. Morris J. Dijkstra'sAlgorithm, Data Structures and Algorithms. (Online).Accessed 10.2.2011
http://www.cs.auckland.ac.nz/~jmor159/PLDS210/dijkstra.html

43. ALgoList.Dijkstra’sAlgorithm (Online). Accessed 10.2.2011
http://www.algolist.com/Dijkstra%27s_algorithm

44. Path Finding - A* Algorithm (Online). Accessed 10.2.2011
http://www.edenwaith.com/products/pige/tutorials/a-star.php

45. Gutman R. Reach-based Routing. A new Approach to Shortest Path Algorithms optimized for Road Networks.2004 pp. 1-12
http://www.siam.org/meetings/alenex04/abstacts/rgutman1.pdf

46. Rolf H, Heiko S. Partitioning Graphs to Speedup Dijkstra’s Algorithm. Journal of Experimental Algorithmics (JEA):vol 11:2006

47. Schulz F, Wagner D. Dijkstra’s Algorithm On-Line: An Empirical Case Study from Public Railroad Transport. pg 3-15

48. Juan C. Dijkstra's Algorithm as a Dynamic Programming strategy (Online). Accessed 10.2.2011
http://www.intag.org/downloads/ds_006.pdf

49. Bauzer C, Medeiros, Max J, Elisa. Advances in Spatial and Temporal Databases: 9th International Symposium, SSTD, Angra dos Reis, Brazil: 2005; p. 298-300.

50. Michael P. Commission on maps and the internet, Maps and the Internet (Online). Accessed 10.2.2011
http://maps.unomaha.edu/ica/

51. Google. Google Maps API.Welcome to Google Enterprise: Earth and Maps (Online). Accessed 10.2.2011
http://www.google.com/intl/en_uk/enterprise/earthmaps/maps.html#utm_campaign=en%26utm_source=en-ha-emea-no-sk%26utm_medium=ha%26utm_term=google%20maps%20API

52. Google. The Solution for Maps Applications for both the Desktop and Mobile Devices (Online). Accessed 11.2.2011
http://code.google.com/apis/maps/documentation/javascript/

53. Google. KML tutorial (Online). Accessed 10.2.2011

http://code.google.com/apis/kml/documentation/kml_tut.html

54. Caitlin D. KML Now an OGC Standard (Online). Accessed 11.2.2011 http://gislounge.com/kml-now-an-ogc-standard/

55. Google. KML Documentation (Online). Accessed 11.2.2011 http://code.google.com/apis/kml/documentation/whatiskml.html

56. Africa Information Technology Initiative (AITI). Strathmore 2009 SMS Final Project Descriptions: SMS Final Projects (online). Accessed 08.2.2011 http://aiti.mit.edu/wiki/index.php?title=SMS_Final_Projects

57. Google Kenya. Google SMS for Your Phone Directions (Online). Accessed 07.2.2011 http://www.google.co.ke/mobile/default/sms.html

58. Arena Kenya. ArenaSMS Service (Online). Accessed 07.2.2011 http://www.arenakenya.com/

59. Fleishman D, Jon E. Strategies for Improved Traveler information: A guide to Fundamental Change in Local Public Transportation Organizations, Transit (TCRP),Report 97.United States. 2003; p. 67-69

60. GIS wiki. (Online). Accessed 07.2.2011 http://en.giswiki.net/wiki/Dijkstra%27s_algorithm

Appendices

Appendix 1: Mobile client connecting to a web server

```
<?php
require  ('dbconnect.php'); // database details
require  ('adjacencies.php');//adjacency  matrix  for Dijkstra  routing
require  ('dijkstra.php'); // Dijkstra  routing Algorithm class

/*
*  Get  values  from  mobile   client as station IDs.
 */

if((isset($_GET))&isset($_GET["from"])&isset($_GET["to"])) {


$from =(int) $_GET['from'];// station id  of  FROM  station
$to = (int)$_GET['to'];// Station id  of  TO station

// connect  to  database
$connection = mysql_connect ($server, $username, $password);
if(!$connection)
{
die('Not connected : ' . mysql_error());
}
// Set the active MySQL database.
$db_selected= mysql_select_db($database, $connection);
if(!$db_selected)
{
die('Can not use db : ' . mysql_error());
}

/* Perform  sql  queries  on the matatus at destination  and  at  origin
 */

$query="SELECT description,matatus_no  FROM stations   WHERE station_id=$from ";
$result = mysql_query($query);
if((!$result) )
          {
die('Invalid query: ' . mysql_error());
          }
//iterate  through  the  results
while($row=mysql_fetch_array($result)) {
// echo "Departing  From: ";
$name=$row["description"];
$matatu=$row["matatus_no"];
         }

$query2="SELECT description ,matatus_no FROM stations   WHERE station_id=$to ";
$result2 = mysql_query($query2);
if((!$result2) )
          {
die('Invalid query: ' . mysql_error());
          }
//iterate  through  the  results
while($row2=mysql_fetch_array($result2)) {

$name2=$row2["description"];
$matatu2=$row2["matatus_no"];
          }

/*
  * Compute  the  routes  by  Dijkstra algorithm from 'FROM' to 'TO' station
  */
$matrixWidth= 143;// total  no  of  stations or  nodes
// I is the infinite distance.
define('I',1000);

$ourMap= array();
// Read in the points and push them into the map by  looping through all the points array
foreach($points as $point) {
        $x = $point[0];
        $y = $point[1];
        $c = $point[2];
        $ourMap[$x][$y] = $c;

}
// ensure that the distance from a node to itself is always zero
for($i=0; $i <$matrixWidth; $i++) {
        $ourMap[$i][$i] = 0;
}

// initialize the algorithm class with  the  appropriate parameters
$dijkstra= new Dijkstra($ourMap, I,$matrixWidth);
$dijkstra->findShortestPath($from,$to); //To find shortest path from stop 'FROM'  to stop 'To'
// Display the results

$stationString=implode(' OR station_id = ',$dijkstra->getResults($to));//  This   can  be  changed  to the  desired
destination
/*make  query for  list  of  stations   from start  to   destination and order  * them systematically
   */
$query3="SELECT description  FROM stations WHERE  station_id IN ('" . implode("','", $dijkstra->getResults($to)) . "')
ORDER BY FIELD(station_id,'" . implode("','", $dijkstra->getResults($to)) . "')";
$result3=mysql_query($query3);

$Stations=""; // a  string variable to  hold  the   ordered station names
// iterate  through  database  query  results
while($row=mysql_fetch_array($result3)) {
$name3=$row["description"];
$Stations.=$name3."->";
   }
//  remove  last  2 characters from the   result  string and replace with   nothing
$StationString= substr_replace($Stations,"",-2);
//get  total distance  for  the trip from start to end
$dist=$dijkstra->distance[$to];

//  final  delimited  string to be sent to   mobile   device.
echo$name."%".$name2."%".$matatu."%".$matatu2."%".$dist."%".$StationString;

//  if  no  variables  were  set  when mobile  client connected to this script
}else{

        echo"problems  FAIL";

  }
?>
```

Appendix 2: Dijkstra algorithm class

Source: Adapted from GIS wiki[60]

```
<?php
classDijkstra {
		var$visited = array();
		var$distance = array();
		var$previousNode= array();
		var$startnode=null;
		var$map = array();
		var$infiniteDistance= 0;
		var$numberOfNodes= 0;
		var$bestPath= 0;
		var$matrixWidth= 0;
var$mypaths= array();
		functionDijkstra($ourMap, $infiniteDistance) {
				$this ->infiniteDistance = $infiniteDistance;
				$this -> map = &$ourMap;
				$this ->numberOfNodes = count($ourMap);
				$this ->bestPath = 0;
		}

		functionfindShortestPath($start,$to) {
				$this ->startnode = $start;
				for($i=0;$i<$this ->numberOfNodes;$i++) {
						if($i == $this ->startnode) {
								$this ->visited[$i] = true;
								$this ->distance[$i] = 0;
						} else {
								$this ->visited[$i] = false;
								$this ->distance[$i] = isset($this -> map[$this ->startnode][$i])
								? $this ->map[$this ->startnode][$i]
								: $this ->infiniteDistance;
						}
						$this ->previousNode[$i] = $this ->startnode;
				}
				$maxTries= $this ->numberOfNodes;
				$tries = 0;
				while(in_array(false,$this->visited,true) &&$tries <= $maxTries) {
						$this ->bestPath = $this->findBestPath($this->distance,array_keys($this ->visited,false));
						if($to !== null &&$this ->bestPath === $to) {
								break;
						}
						$this ->updateDistanceAndPrevious($this ->bestPath);
						$this ->visited[$this ->bestPath] = true;
				$tries++;
				}
		}
		functionfindBestPath($ourDistance, $ourNodesLeft) {
				$bestPath= $this ->infiniteDistance;
				$bestNode= 0;
				for($i = 0,$m=count($ourNodesLeft); $i <$m; $i++) {
						if($ourDistance[$ourNodesLeft[$i]] <$bestPath) {
								$bestPath= $ourDistance[$ourNodesLeft[$i]];
						$bestNode= $ourNodesLeft[$i];
						}
				}
				return$bestNode;
```

```
            }

            functionupdateDistanceAndPrevious($obp) {
                        for($i=0;$i<$this ->numberOfNodes;$i++) {
                                    if( (isset($this->map[$obp][$i]))
                                    && (!($this->map[$obp][$i] == $this->infiniteDistance) || ($this->map[$obp][$i] == 0 ))
                                    && (($this->distance[$obp] + $this->map[$obp][$i]) <$this -> distance[$i])
                                    )
                                    {
                                                $this ->distance[$i] = $this -> distance[$obp] + $this -> map[$obp][$i];
                                                $this ->previousNode[$i] = $obp;
                                    }
                        }
            }
            functionprintMap(&$map) {
                        $placeholder = ' %' .strlen($this ->infiniteDistance) .'d';
                        $foo = '';
                        for($i=0,$im=count($map);$i<$im;$i++) {
                                    for($k=0,$m=$im;$k<$m;$k++) {
                                                $foo.=sprintf($placeholder, isset($map[$i][$k]) ? $map[$i][$k] : $this -
>infiniteDistance);
                                    }
                        $foo.="\n";
                        }
                        return$foo;
            }
            functiongetResults($to) {
                        $ourShortestPath= array();
                        $mypaths= array();// not sure
                        $foo = '';
                        for($i = 0; $i <$this ->numberOfNodes; $i++) {
                                    if($to !== null &&$to !== $i) {
                                                continue;
                                    }
                                    $ourShortestPath[$i] = array();
                                    $endNode= null;
                                    $currNode= $i;
                                    $ourShortestPath[$i][] = $i;
                                    while($endNode=== null || $endNode!= $this ->startnode) {
                                                $ourShortestPath[$i][] = $this ->previousNode[$currNode];
                                                $endNode= $this ->previousNode[$currNode];
                                                $currNode= $this ->previousNode[$currNode];                          }
                                    $ourShortestPath[$i] = array_reverse($ourShortestPath[$i]);

                                    if($to === null || $to === $i) {// reached  end point********
                                                if($this -> distance[$i] >= $this ->infiniteDistance) {
                                                            //$foo .=sprintf("no route from %d to %d. \n",$this ->startnode,$i);
                                                } else {

                                    }                       }
                        return$ourShortestPath[$i] ;
} // end class
?>
```

Appendix 3: Adjacency matrix

An extract from the Adjacency matrix showing only a few stations

```
<?php
//  this  hold  the  adjacencies   between  various  points  in the  network
//  load point as arrays in the  form array(startPoint, endPoint,Adjacency or Weight or Distance)
$points = array(
array(84,85,0.984), //  station Id, station Id, distance  between them.
array(85,86,0.466),
array(96,98,1.072),
array(98,99,1.516),
array(99,100,8.416),
array(97,95,4.975),
array(95,94,3.197),
array(94,90,3.216),
array(90,89,0.732),
array(89,88,0.937),
………………………Additional  stations
array(132,105,2.368),
array(110,123,1.120),
array(13,115,1000)


);//  end  of points
```

Appendix 4: Google maps API example loading map on a web page.

```
<?php
/*
 * To  load   Google map  on web page, we  need to  declare a  html div place  holder that  has  width and
height  as the   dimensions  of the map  to be loaded. Then to include  map  controls  like zoom, pan and
also  to have it  centered  at a location. To overlay KML we need to direct Google's GeoXml Server to point
to the path where  KML  file is  located(Google map server prefers  KML to be fetched from a public
accessible  server
 */


?><!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01
Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<metahttp-equiv="content-type"content="text/html; charset=utf-8"/>
<title>Google Maps JavaScript API Example</title>

<scriptsrc="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=true&amp;key=ABQIAAAA4O
5ez5KxrxHFaAXUgaeKihQngLTJFFoCVx_kIKRYGDn4lJP_fBQ7oilmjOTzwrC26ExJ3cMITxLy5A"type="text/jav
ascript"></script>


<scripttype="text/javascript">
//<![CDATA[

function load()
{
var map;
vargeoXml;

if (GBrowserIsCompatible())
  {
map = new GMap2(document.getElementById('map'));
map.addControl(newGSmallMapControl());
map.addControl(newGMapTypeControl());
geoXml = newGeoXml('http://users.metopolia.fi/~davidka/matatu/RoutesNetwork.kml');
map.addOverlay(geoXml);
map.setCenter(newGLatLng(-1.28170776,36.81500626), 13);
  }
}
//]]>
</script>
</head>
<bodyonload="load()"onunload="GUnload()">

<divid="map"style="width: 700px; height: 500px"></div>

</body></html>
```

Appendix 5: Java ME mobile client

(Only important code is given)

```
packagecom.matatuonline.mobile;
import java.io.*;
importjava.util.Stack;
import javax.microedition.io.*;
importjavax.microedition.lcdui.*;
importjavax.microedition.lcdui.TextField;
importjavax.microedition.midlet.*;
/**
 * @author david kanyari
 */
public class PangaSafari extends MIDlet implements CommandListener,Runnable{

private Display display;
privateint to;// destination
privateint from;  //origin
privateTextBoxtextBox = null;// hold  contents  for  data  from request
private Command back;
privateMatatuRoutesArraymatatuStations;
privateTextFieldFromField;// input text  field
private Form FormSearchRoutes;// matatu  serach form
privateTextFieldToField; input  text  field
private Command Search;  // route  search command
private Command back2;
   private String url;  //  URL  to  the  PHP  script  in the  server


// constructor
publicPangaSafari() {
display = Display.getDisplay(this);
   }


public void startApp() {

           //  Search route  Form
FormSearchRoutes= new Form(" Route Search");
FromField=   new TextField("FROM:","",20,TextField.ANY);

            String detail=  "   Search for Matatu Here:";

ToField=   new TextField(" TO  : ","",20,TextField.ANY);
              //  commands.
            Search = new Command("Search",Command.OK,0);
exit = new Command("Exit",Command.EXIT,0);
            //info = new Command("Info",Command.OK,1);
             back2= new Command("Back",Command.BACK,0);

FormSearchRoutes.append(detail);
FormSearchRoutes.append("                         ");
FormSearchRoutes.append("Enter Station Name in textfields given.");
            // FormSearchRoutes.append("                         ");
FormSearchRoutes.append(FromField);
            // FormSearchRoutes.append("                         ");
FormSearchRoutes.append(ToField);
FormSearchRoutes.addCommand(Search);
FormSearchRoutes.addCommand(back2);
FormSearchRoutes.setCommandListener(this);
select= new Command("Station Lists",Command.SCREEN,1);
FormSearchRoutes.addCommand(select);
matatuStations= new  MatatuRoutesArray();// load  matatu stations

        //  lists  for   selecting  stations

chooseOrigin= new List("Choose Origin",List.EXCLUSIVE);
proceed =new Command("Destination",Command.OK,1);
chooseOrigin.addCommand(proceed);

        // selecting  destination

        chooseOrigin2= new List("Choose Destination",List.EXCLUSIVE);

        //       origin destination set up form

formRequest=  new  Form("Route Request ");
formRequest.append("Confirming Your  Request as:");
compose =  new Command("Search",Command.OK,1);
        compose3 =  new Command("Proceed",Command.OK,0);
back5  =  new Command("Back",Command.BACK,1);
back6  =  new Command("Back",Command.BACK,1);

formRequest.addCommand(back5);
formRequest.addCommand(compose);
formRequest.setCommandListener(this);


        //info  form
Infoform = new Form("About");
start= new Command("Start",Command.OK,0);
Infoform.append("Search Matatu by your phone! ");
Infoform.append("                         ");
Infoform.append("MobileMatatu is an application that "
             + "provides users with information on matatu routes,route numbers as well as connections to various
               places around Nairobi City "
           + "by use of public transport.");
Infoform.append("                         ");
Infoform.append("Application developed by David Kanyari, Lund University.");
Infoform.append("                         ");

Infoform.append("For more info visit www.matatuonline.com");
Infoform.append("©Copyright 2011.matatuonline.com.All rights reserved.");
Infoform.addCommand(start);
Infoform.setCommandListener(this);

display.setCurrent(Infoform);
    }

public void pauseApp() {
   }

public void destroyApp(boolean unconditional) {

notifyDestroyed();
   }
```

```
  //HTTP connection     function .To communicate between web server and mobile client
voidtestGET(String url) throws IOException {
HttpConnection connection = null;
InputStream is = null;
OutputStreamos = null;
StringBufferstringBuffer = new StringBuffer();


try {
connection = (HttpConnection) Connector.open(url);
connection.setRequestMethod(HttpConnection.GET);
connection.setRequestProperty("IF-Modified-Since", "20 Jan 2001 16:19:14 GMT");
connection.setRequestProperty("User-Agent", "Profile/MIDP-2.0 Confirguration/CLDC-1.0");
connection.setRequestProperty("Content-Language", "en-CA");
connection.setRequestProperty("Content-Type", "//text plain");
os = connection.openOutputStream();
is = connection.openDataInputStream();
intch;
while ((ch = is.read()) != -1) {
stringBuffer.append((char) ch);
         }

         //  split the incoming   string from request
char delimiter = '%';

String[] result = split(stringBuffer.toString(), delimiter, false);

if (result != null) {

             //  make  string   buffer;
StringBufferRouteData = new StringBuffer();
RouteData.append("From: " + result[0] + ".");
RouteData.append("To: " + result[1] + ".");
RouteData.append("Matatu(s) at:" + result[0] + " : " + result[2] + ".");
RouteData.append("Matatu(s) at:" + result[1] + " : " + result[3] + ".");
RouteData.append("Distance: " + result[4] + " km.");
RouteData.append("Your Route: " + result[5] + ".");
textBox = new TextBox("Route Info", RouteData.toString(), 1024, 0);
back= new Command("Back",Command.BACK,1);
textBox.addCommand(exit);
textBox.addCommand(back);
textBox.setCommandListener(this);
         } else {
System.out.println("Sorry,Problem in  the request");
         }

      } finally {
if (is != null) {
is.close();
         }
if (os != null) {
os.close();
         }
if (connection != null) {
connection.close();
         }
      }
```

```
display.setCurrent(textBox);
   }


  //   commands  processing
public void commandAction(Command c, Displayable d) {

if( (c== select)|(c== back7))
         {
        //select  origin list

MakeSearchForm();
       }

if(c==Search){
        //  search routes   form

         //  read   from    FROM:  textfield.
String  stationName1 = FromField.getString().toUpperCase() ;
from  = (matatuStations.getMyArr().indexOf(stationName1))+1;
System.out.println("To " + from);

        //  read input  from   To  field
String  stationName2 = ToField.getString().toUpperCase() ;

to=(matatuStations.getMyArr().indexOf(stationName2))+1;

if ((!searchStation(stationName1)) | (from == 0)) {

              Alert alert = new Alert("Request Info","Your (From) station does not exist or (From) Field is empty!Please
check the spelling or use the station list command to view/select available stations",
null, AlertType.WARNING);
alert.setTimeout(Alert.FOREVER);
display.setCurrent(alert, FormSearchRoutes);

}else if ( (!searchStation(stationName2))|(to==0 )  ){

              Alert alert = new Alert("Request Info","Your (To) Station does not exist or(To)Field is empty!Please check
the spelling or use the station list command to view/select available stations",
null, AlertType.WARNING);
alert.setTimeout(Alert.FOREVER);
display.setCurrent(alert, FormSearchRoutes);
    // proceed  with request
}else{System.out.println("To " + to);
url = "http://users.metropolia.fi/~davidka/matatuMobileServices/matatuServices.php?from="+from+"&to="+to;
System.out.println("URL: " +url);
   //  testGET(url);  //  send  request   to  Server  and  process it.
     //   use  thread to  economize  on memory.
            Thread GetRequest= new Thread(this);
GetRequest.start();
     }    }
else if  (c==proceed){
        //  select  destination
fromX = chooseOrigin.getSelectedIndex();
fromlist= chooseOrigin.getString(fromX);
    // System.out.print( "You selected: " +fromlist);
MakeSearchFormDestination();
    }
```

```
else if (c==compose3){
        //  select  destination
toX = chooseOrigin2.getSelectedIndex();
    String tolist= chooseOrigin2.getString(toX);
   // System.out.print( "You selected: " +tolist);
formRequest.append("From:"+fromlist);
formRequest.append("To:"+tolist);

display.setCurrent(formRequest);
    }
else if  (c==compose){
System.out.print("from:"+fromX);
System.out.print("from:"+toX);
int X=fromX+1;int Y=toX+1;//  remember  station id  and array indexing.
  //   set up  form  read  values  from lists  and  append
url = "http://users.metropolia.fi/~davidka/matatuMobileServices/matatuServices.php?from="+X+"&to="+Y;
    Thread GetRequest= new Thread(this);
GetRequest.start();
}
else if (c==back5){

MakeSearchFormDestination();
}
else if (c==start){

display.setCurrent(FormSearchRoutes);
}
/*   Run the    URL   Request Thread    in    th  thread.
*/
public void run() {
try {
testGET(url);
      } catch (IOException ex) {
ex.printStackTrace();
      }    }
/*
Search  in  the  Station array  and  see if   station name  exists,
if  not  found   return  false
*/
publicbooleansearchStation( String stationName){
        //  check
booleanblnFound = matatuStations.getMyArr().contains(stationName);
returnblnFound;
  }
   /*
 * A   function to  compose the  origin  search list.User can choose any station
as he so wishes
 */
public void MakeSearchForm(){
     // loop through    Station array list
for (int i = 0; i <matatuStations.getMyArr().size(); i++) {
         String gg =matatuStations.getMyArr().get(i).toString();
chooseOrigin.append(gg, null);
}
chooseOrigin.addCommand( back6);
chooseOrigin.setCommandListener(this);
display.setCurrent(chooseOrigin);
  }
```

Appendix 6: Questionnaire

Nairobi City Journey Planner Application

Application Testing Questionnaire

Place of test: ________________________________ Date: __________________

Participant's Name: __________________________ Age : __________________

Email: _____________________________________ Phone No. : ______________

Participant's Occupation: ____

Place of Residence (Estate Name e.g. Westlands, Bahati) : _______________________

Background

This questionnaire is meant to collect user feedback concerning the functionality, usability, relevance, content and performance of the mobile Nairobi city journey planner applications for both online and mobile versions. The journey planner application is meant to assist users plan and to get information about moving from one place to another by public transport (matatu)

The testing exercise is sub divided into several test cases where the user carries out some specified tasks and reports his or her experience as feedback. Only matatu route search will be considered in this test.

Task 1: Look and Feel

The online journey planner service can be accessed by visiting the site www.matatuonline.com.Type the full address of the site to your mobile or computer browser. However internet connection is necessary to accomplish this task.

a) Please comment on your initial impressions about the layout of this page and what you think of the colors choices, graphics, images, etc.

b) What do you think is the purpose or goal of this site?

b) Who do you think this site is intended for (e.g target group, customers)?

Task 2: Functionality

Now that you have an idea about what the site offers, it is time to try out some real tasks on it.

i) Route search: Assume you are travelling from one place to another in the city and you wanted to know the names of stations , distance, duration of the journey, and what bus number to take, use the route search form to test that case. A result page is displayed for the matatu route search.

a) Please comment on your general impression about the results displayed and the information given concerning your route search. Also comment on the meaningfulness of the results.

b) What information do you think is missing from the results displayed apart from ‘change buses?

ii) Route map: By clicking ‘view route map’ a map of the route is shown in a different page. Comment on the usefulness and ability to view the route on the map. Station names are also displayed by clicking on the ‘balloon ‘icons. The map can be zoomed for viewability.

Task 3: Usability

Explore the site using navigational links and buttons. Try different pages!

a) How easy is it to navigate across the web pages of the website. Any difficulties encountered?

b) Did you find any missing or broken links? If so state which one and the name of the link?

c) In your opinion what would you recommend to be done to make the site more user friendly and easy to use?

Compatibility and Performance

a) Try visiting the website using different browsers. e.g Mozilla, Internet Explorer, Opera and mobile browser. Is there any significance differences noted? If so give comments.

b)How fast or slow does the page take to load under the slow internet connection.eg using USB modem or wired internet? Please give comments.

General Comments

a) What are your overall impressions of the Web site?

b) If you had to give the site a grade, from 1 to 5, where 5 was Excellent and 1 was poor, what grade would you give it, and why?

c) What are the two things you like best about the Web site?

d) What are the two things you like least about the Web site?

e) What target groups, organizations etc do you think would be interested in using the website?

f) Would you recommend the site to a colleague, friend, organization etc?

Mobile Journey Planner also known as 'panga Safari'

Place of test: ________________________________ Date: __________________
Participant's Name: __________________________ Age : __________________
Email: _____________________________________ Phone No. : ______________
Participant's Occupation: ____
Place of Residence (Estate Name e.g. Westlands, Bahati) : _______________________

Background

The mobile journey planner is a simplified version of the online application specifically developed to work in low end phones. Phones with small displays and limited computation power. To test the mobile journey planer application, open *Panga safari* application. The application opens a start page. Using command start takes a use to a simple search form.

Task 1: Route search form: Using the search form type the name of preferred origin and destination station and then click search.

a) Your phone model e.g. Nokia 2690: ___________________

b) What are your overall expectations of the application, look and feel? Is it difficult to use? What difficulties did you encounter while using the search form?

c) Did you receive warnings for misspelling or entering wrong station names?

d) How long did it take to wait the results to be displayed? Do you think it took longer than expected? However this may depend on the strength of network connection.

Task 2: Station list: Apart from search form, you can choose origin and destination stations from station lists by using command 'station lists'. Use commands 'Proceed 'to initiate route search. How does this compare in terms of ease of use to direct keying in of station names in search form? Which one do you prefer and why?

Task 3: Display of Results: Since results are displayed in a text box, is the information in the results meaningful, legible and comprehensive? Does it look cluttered somehow? Please comment and include your recommendations.

General comments

a) What are the two things you like best about the mobile journey planner?

b) What are the two things you like least about the mobile journey planner?

c) What target groups, organizations etc do you think would be interested in using the application?

d) What would you like to be improved on or added to the mobile application and why?

e) In a scale of 1 to 10. How would you rate this application? 10 is extremely high, 5 averages and 1 is very poor. Justify your rating!

f) Between both the online and mobile journey planner applications, which one do you prefer to the other and why?

We appreciate your feedback as it helps us improve our services in line with the user needs. However, all user information or details filled on this form will be held confidential and only used for the purpose of improving our service to the city residents and entire Nairobi community.

Thanks
Email your comments to: David.Kanyari@metropolia.fi or
david.kanyari@matatuonline.com

Appendix 7: A Collection of Nairobi Matatu Photos

Photo 1: Matatu no. 11A and 48 at Odeon Terminus.

Photo 2: A view of Odeon Bus Terminus.

Photo 3: Passengers boarding a matatu .

Photo 4: A stranded traveller asking for directions from a conductor.

Photo 5: A Typical scene at a matatu terminus.

Photo 6: Upcountry travellers wait for a matatu at Tea Room terminus.

Photo 7: A Rough matatu Time/Queue Schedule.

Photo 8: A conductor describing the matatu no.48 route map.

Photo 9: A long distance matatu at Tea Room terminus.

Photo 10: A commuter bus at Bus Station terminus.

Photo 11: A view of Nairobi Central Bus Station.

Photo 12: Matatu Route No. and Destination information board.

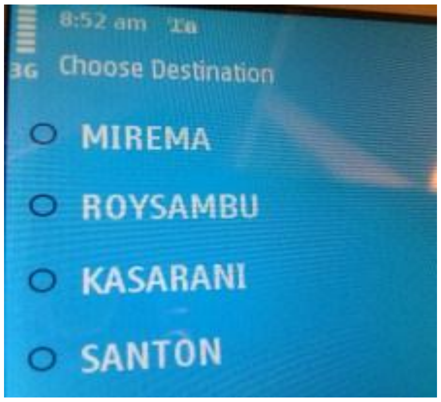
Photo 13: A commuter tests the mobile journey planner.

Photo 14: Another traveller tests the journey planner.

Photo 15: A group of users examine the journey planner application on the streets of Nairobi.

Photo 16: Excited Drivers and conductors test the journey planner at Odeon Bus terminus. .

Photo 17: A parting shot after user testing.

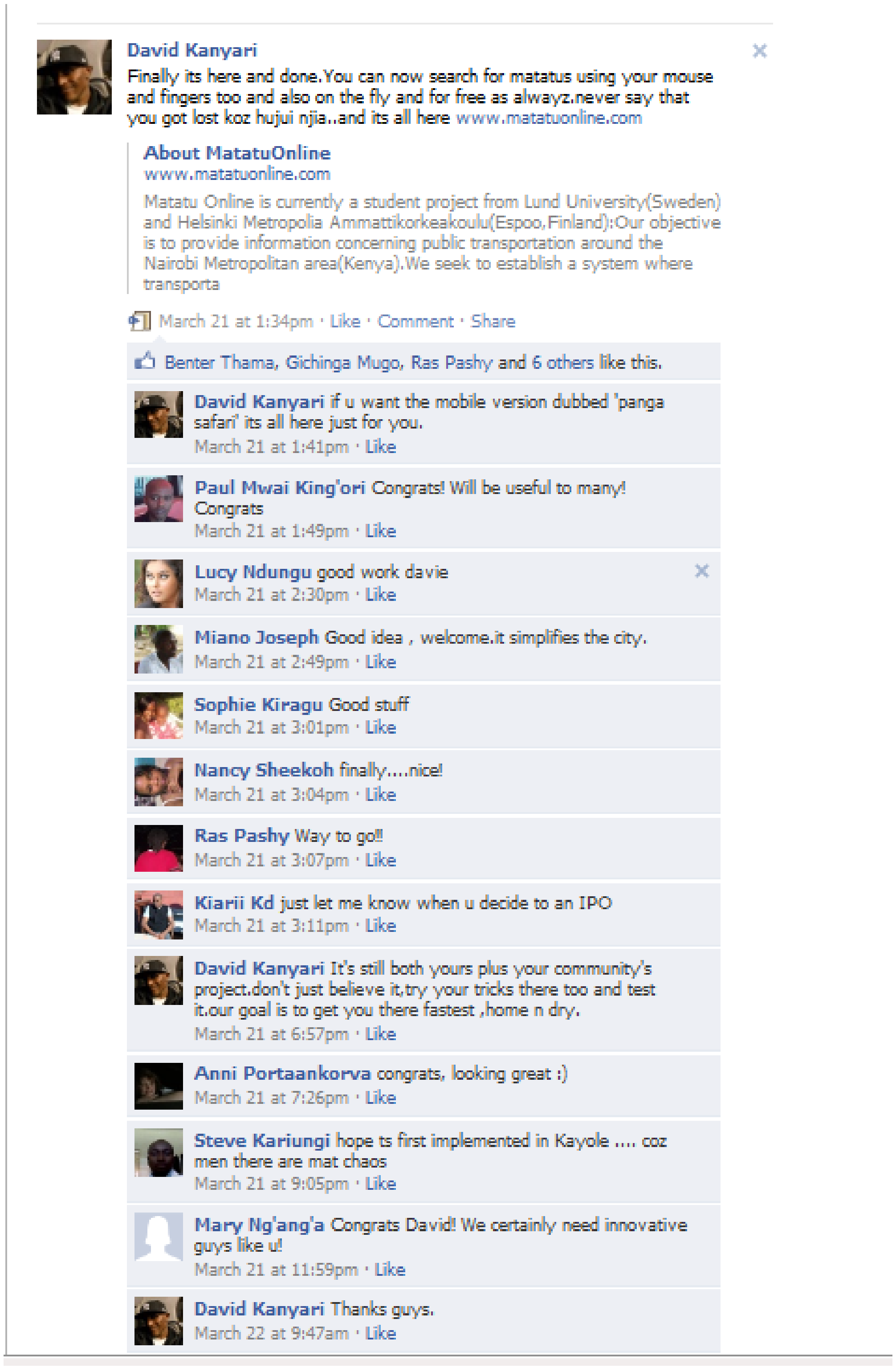

8:52 am

Choose Destination

- MIREMA
- ROYSAMBU
- KASARANI
- SANTON

Photo 18: A shot taken on a user's phone during testing.

Appendix 8: Using Social Media (Face book) for user feed back



David Kanyari
Finally its here and done.You can now search for matatus using your mouse and fingers too and also on the fly and for free as alwayz.never say that you got lost koz hujui njia..and its all here www.matatuonline.com

About MatatuOnline
www.matatuonline.com
Matatu Online is currently a student project from Lund University(Sweden) and Helsinki Metropolia Ammattikorkeakoulu(Espoo,Finland):Our objective is to provide information concerning public transportation around the Nairobi Metropolitan area(Kenya).We seek to establish a system where transporta

March 21 at 1:34pm · Like · Comment · Share

Benter Thama, Gichinga Mugo, Ras Pashy and 6 others like this.

David Kanyari if u want the mobile version dubbed 'panga safari' its all here just for you.
March 21 at 1:41pm · Like

Paul Mwai King'ori Congrats! Will be useful to many! Congrats
March 21 at 1:49pm · Like

Lucy Ndungu good work davie
March 21 at 2:30pm · Like

Miano Joseph Good idea , welcome.it simplifies the city.
March 21 at 2:49pm · Like

Sophie Kiragu Good stuff
March 21 at 3:01pm · Like

Nancy Sheekoh finally....nice!
March 21 at 3:04pm · Like

Ras Pashy Way to go!!
March 21 at 3:07pm · Like

Kiarii Kd just let me know when u decide to an IPO
March 21 at 3:11pm · Like

David Kanyari It's still both yours plus your community's project.don't just believe it,try your tricks there too and test it.our goal is to get you there fastest ,home n dry.
March 21 at 6:57pm · Like

Anni Portaankorva congrats, looking great :)
March 21 at 7:26pm · Like

Steve Kariungi hope ts first implemented in Kayole .... coz men there are mat chaos
March 21 at 9:05pm · Like

Mary Ng'ang'a Congrats David! We certainly need innovative guys like u!
March 21 at 11:59pm · Like

David Kanyari Thanks guys.
March 22 at 9:47am · Like

Series from Lund University
Department of Physical Geography and Ecosystem Science

Master Thesis in Geographical Information Science (LUMA-GIS)

1. *Anthony Lawther:* The application of GIS-based binary logistic regression for slope failure susceptibility mapping in the Western Grampian Mountains, Scotland. (2008).
2. *Rickard Hansen:* Daily mobility in Grenoble Metropolitan Region, France. Applied GIS methods in time geographical research. (2008).
3. *Emil Bayramov:* Environmental monitoring of bio-restoration activities using GIS and Remote Sensing. (2009).
4. *Rafael Villarreal Pacheco:* Applications of Geographic Information Systems as an analytical and visualization tool for mass real estate valuation: a case study of Fontibon District, Bogota, Columbia. (2009).
5. *Siri Oestreich Waage:* a case study of route solving for oversized transport: The use of GIS functionalities in transport of transformers, as part of maintaining a reliable power infrastructure (2010).
6. *Edgar Pimiento:* Shallow landslide susceptibility – Modelling and validation (2010).
7. *Martina Schäfer:* Near real-time mapping of floodwater mosquito breeding sites using aerial photographs (2010)
8. *August Pieter van Waarden-Nagel:* Land use evaluation to assess the outcome of the programme of rehabilitation measures for the river Rhine in the Netherlands (2010)
9. *Samira Muhammad:* Development and implementation of air quality data mart for Ontario, Canada: A case study of air quality in Ontario using OLAP tool. (2010)
10. *Fredros Oketch Okumu*: Using remotely sensed data to explore spatial and temporal relationships between photosynthetic productivity of vegetation and malaria transmission intensities in selected parts of Africa (2011)
11. *Svajunas Plunge:* Advanced decision support methods for solving diffuse water pollution problems (2011)
12. *Jonathan Higgins:* Monitoring urban growth in greater Lagos: A case study using GIS to monitor the urban growth of Lagos 1990 - 2008 and produce future growth prospects for the city (2011).
13. *Mårten Karlberg:* Mobile Map Client API: Design and Implementation for Android (2011).
14. *Jeanette McBride:* Mapping Chicago area urban tree canopy using color infrared imagery (2011)
15. *Andrew Farina:* Exploring the relationship between land surface temperature and vegetation abundance for urban heat island mitigation in Seville, Spain (2011)
16. *David Kanyari*: Nairobi City Journey Planner An online and a Mobile Application (2011)