CITY NAVIGATOR

P-864

Submitted in partial fulfillment of requirements

For the award of degree of

Bachelor of Engineering
(Computer Science and Engineering)
of
Bharathiar University

is the report of original work done by

Archana Nandakumar	**9927K0116**
N.Kavitha Bharathi	**9927K0133**
A.Ramya	**9927K0155**
M.Shashikala	**9927K0162**

under the guidance of

Mr.DINESH RANGANATHAN, B.Tech.,M.S.(Computer.Science,Wisconsin)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE
MARCH 2003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
Coimbatore – 641006

PROJECT REPORT 2002-2003

This is to certify that the project report entitled

CITY NAVIGATOR

is the bonafide work of

ARCHANA NANDAKUMAR	9927K0116
KAVITHA BHARATHI N	9927K0133
RAMYA A	9927K0155
SHASHIKALA M	9927K0162

in partial fulfillment of the requirement for the award of

Bachelors Degree in Computer Science and Engineering

of Bharathiar University, Coimbatore – 641042

Project Guide [illegible] (13/3/2003)
Mr.DINESH RANGANATHAN.B.Tech.,M.S.
(Comp.Science,Wisconsin)

Head of Department
Prof. S. Thangaswamy

Place: COIMBATORE

Date: 17-3-'03

Submitted for the university exam held on 17-3-'03

[illegible]

Internal Examiner

External Examiner

Dedicated to our beloved
parents

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Dreams need foundations, but to realize them in reality, we need support.

We express our thanks to our Principal Dr.K.K.Padmanabhan Bsc.,(Engg).,M.Tech.,PhD., for the patronage and facilities provided to carry out the project.

We would like to express our gratitude to Prof.Dr.S.Thangasamy,BE.,(Hons).,PhD., Head of the Department, of Computer Science and Engineering for his valuable guidance.

We are immensely thankful for the wholehearted support, guidance and encouragement provided by our guide Mr.Dinesh Ranganathan, B.Tech, M.S. (Computer Science, Wisconsin), ASSISTANT ~~Senior~~ PROFESSOR ~~Lecturer~~, Department of Computer Science and Engineering.

We would like to thank Mr.M.N.Guptha, B.E., our class advisor, and Miss.Rajani, B.E., Senior Lecturer, Department of Computer Science and Engineering, who gave us constant encouragement and helped us in many ways for our successful completion of this project.

We would like to express our gratitude and special thanks to the staffs of Computer Science and Engineering Department, our lab technicians and all our friends without whom we couldn't have completed this endeavor.

Above all, We would like to thank our parents and all our well wishers who provided us with unwavering support during the course of the project.

SYNOPSIS

SYNOPSIS

In today's mechanical world where the traffic is erratic with the streets being horded with vehicles as well as people, and the time being so precious, it is becoming increasingly important to make quick and accurate decisions regarding the shortest and most efficient way to reach a destination. The mass commercialization has also added to the imbroglio. Thus it becomes imperative to have a system that will aid people to make the above decisions in an easy way. This project aims at making this possible.

The main objective of this project "**CITY NAVIGATOR**" is to determine the optimal path in terms of distance between any two places. In addition to this it will also provide various facilities such as directory service, timings of various transport facilities, log of recently visited paths and recently searched organizations.

This can be implemented by initially preparing the map of the area under consideration. An algorithm has to be formulated in order to find all the possible routes for a given source and destination. This has to be done taking into account the 1 way and the 2 way roads. The algorithm has to be then implemented on the map. Four types of routes will be provided namely shortest path, longest path, specified path and all possible paths.

The directory service that includes the important commercial centers and organizations is to be implemented with the help of a database and ASP. The timings for various modes of transport such as railways, airways and bus will also be implemented in a similar fashion. A bookmark facility is to be developed to view the most recently accessed routes and also the recently searched organizations. A feedback form is to be provided in order to obtain the comments from users. A help option which will guide the user in the operation of the software is also to be included.

This project is to be done using FLASH MX, ASP and MS- Access.

CONTENTS

CONTENTS

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

This project was done using Flash MX as front end, MS Access as the back end with Asp as an interface. The coding is done using Flash Action Script and VB script.

The project has the following modules

- ⇒ PATH MODULE
- ⇒ TIMING MODULE
- ⇒ DIRECTORY MODULE
- ⇒ BOOKMARK MODULE
- ⇒ FEEDBACK MODULE

Path Module

This module includes the following functions

- Calculation of all possible paths
- Calculation of shortest path
- Calculation of longest path
- Calculation of specific paths with two options as through and not through a specified place

This module provides the user with the path details from a source to destination. The user has to provide the source and destination, between which he needs the path details and also any one of the options that we provide such as all possible paths, shortest path, longest path or specific paths. Then the user will be provided with the required details. The input is highly user friendly and can be given in a graphical form by clicking the place on the map. The output for this module will be both in graphical and textual form.

All possible paths option provides all the possible road lines between the source and destination selected by the user. Shortest path and longest path option will provide the possible shortest and longest road lines respectively. Specific path is the option that is used to provide routes between places including or excluding some specific intermediate places.

Timing Module

Timings for various modes of transport will be provided in this module. The modes of transport considered here are bus, trains and airlines.

As far as the bus timing is concerned we provide the local bus timing and routes for the buses that passes through the covered area. The user can request for bus details either by choosing the bus numbers or by choosing the two terminals of the bus. The timing details provided for each bus is the time at which it starts from Gandhipuram or Ukkadam Bus Stands.

Train timings will be provided for the trains that start from or reaches or passes through Coimbatore. The trains are categorized as Express and Passenger trains. The user can request train details by selecting either train name or source or destination, from which he wishes to start. He will then be provided with details such as the train category, arrival and departure time of the train from Coimbatore and the frequency of days during which it passes through Coimbatore.

Similarly timings for airlines will also be provided for the planes that either arrive at or depart from or pass through Coimbatore.

Directory Module

Providing information regarding the various important landmarks in the covered area is dealt with in this module. User can search for an organization by its street name or its name or its organization type.

Details provided for an organization are its name, its type, street name on which it is located, its located area name, its e-mail address, its website address, its phone number and any other extra information.

Bookmark Module

This module, as the name suggests performs the same task as a normal bookmark does. It provides details regarding the recent path searched or a recent organization searched in the directory within a period of three days.

Feedback Module

Users feedback regarding the application can be sent online to the e-mail id provided. Users can also notify if there is any recent updations regarding any information provided in this application.

1.2 NEED FOR COMPUTERISATION

Computerizing any system has its own advantages. It enables the user to do the process very interactively. The benefits can be explained as follows

Flexibility

Flexibility is one of the main characteristics of the computes .The system is flexible enough to work with and the interactiveness is highly promising. The recent changes in the city can be reflected easily and quickly by updations through computers. Output can be obtained in whatever form the users needs.

Accuracy

Another main characteristics of the computers is the accuracy in producing the result. The user can program to produce the result in a suitable fashion. Practical experience has already shown that these machines are achieving the great degree of accuracy even in complex calculations that passes through many number of stages.

Reliability

The computers are reliable than manual system .The results can be relied upon because the computers are error free.

User Interactiveness

The user input and output are highly interactive in computerized system. Anyone who knows to operate mouse and keyboard can handle our application easily.

LITERATURE SURVERY

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEM

The current existing software that is similar to our proposed project doesn't exist so far. But there are some websites that provide similar information as our proposed system. Some of such web sites are www.navigateindia.com,www.coimbatore.com etc..,where navigateindia.com covers the city of Chennai whereas coimbatore.com covers the city of Coimbatore.

These websites provide only a static view, that is the users can't interact directly with the map. They don't provide a user-friendly interface, in the sense streets are not highlighted once a required path is obtained as our project does. Apart from this, the various options provided by our project such as finding all the possible routes and finding a specific route is not offered by any of the existing systems.

Also no software gives the local bus timings that go through the area.

2.2 FEASIBILITY STUDY

Feasibility study is essential for each and every system that is to be developed according to future usage and user requirements. It gives a deep knowledge of the existing system and the system to be developed. It is the developer's duty to find maximum details from the existing drawbacks, as what are the modifications needed, the level up to which the system can be divided into sub function etc., onducting a feasibility study on the above existing systems yielded in the following hortcomings

- Less user interactivity.
- Results were not clearly understandable.
- No information as how the directions are taken in a displayed route to the path
- Inaccuracy
- Less Reliability

Necessary steps are have been taken in the proposed system to overcome the above drawbacks and to improve user convenience.

SOFTWARE REQUIREMENTS

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 SOFTWARE ENVIRONMENT

This application has been developed using Flash MX and MS Access with ASP as interface for connecting Flash MX with the database.

SOFTWARE REQUIREMENTS

FRONT END

❖ Flash MX

BACK END

❖ MS Access

INTERFACE

❖ ASP (Active Server Page)

3.2 HARDWARE ENVIRONMENT

Pentium III 998Mhz

128 MB RAM

4.3 GB HardDisk

48x CD Drive

3.3 SOFTWARE FEATURES

FLASH MX

INTRODUCTION

- ▶The professional standard for producing high- impact web experiences
- ▶Provides enhanced capabilities
  - for creating artwork
  - streamlining workflow
- ▶Include expanded capabilities for creating interactivity with Action Script.

Creating artwork

Enhanced color controls, including the Mixer panel, Fill and Stroke panels, Swatches panel, and Fill and Stroke toolbox controls, provide expanded capabilities for painting artwork.

New selection highlights make it easy to identify selected lines, fills, and groups as well as the color of selected object.

Draggable guides aid you in arranging objects on the Stage.

The Pen tool lets you create precise paths; it works like the Pen tool in Macromedia Free Hand or Macromedia Fireworks.

Workflow

New panels for working with color, type, actions, frames, instances, and entire movies make it easy to access options for modifying elements in Flash movies..

Shared libraries let you link to library items as external assets. You can create font symbols to include in shared libraries, as well as buttons, graphics, movie clips, and sounds.

The Macromedia Dashboard provides a way for you to easily keep up with the latest information on using Flash.

Custom shortcut keys allow you to create your own shortcuts for Flash commands and functions to customize your workflow.

Support for importing MP3 sound files lets you import sounds into Flash that are already compressed. This reduces the time required for publishing and exporting a movie with sound, since you don't have to compress the sounds during export. Using compressed sounds reduces the file size of completed movies and reduces memory requirements during authoring.

Interactivity

Expanded ActionScript provides greatly enhanced capabilities for creating interactivity in Flash using ActionScript.

The Movie Explorer lets you easily view the complete contents of the current movie and view the Properties panel for a selected item to modify it..

The Print action lets you assign actions for printing Flash movie frames from the Flash Player as vector or bitmap graphics.

FEATURES OF FLASH

- Change Flash movie properties
- Import, create, and modify media that appear in your movie
- Add sound to a button
- Use the Stage and Timeline to assemble the movie
- Create motion-tweening and shape-tweening animations
- Use actions to add interactivity and streamline authoring
- Test the movie for download performance
- Publish the movie for Web playback

CREATING ANIMATION OVERVIEW

Animation, in general sense means making an object to move across the Stage, increase or decrease its size, rotate, change color, fade in or out, or change shape. Changes can occur independently of, or in concert with, other changes. For example, an object can be made to rotate and fade in as it moves across the Stage.

There are three important components for creating animation. They are:

- Frames
- Layers
- Scenes

FRAMES

Animation is created by changing the content of successive frames.

There are **two methods for creating an animation sequence** in Flash:

1. Frame-by-frame animation
2. Tweened animation.

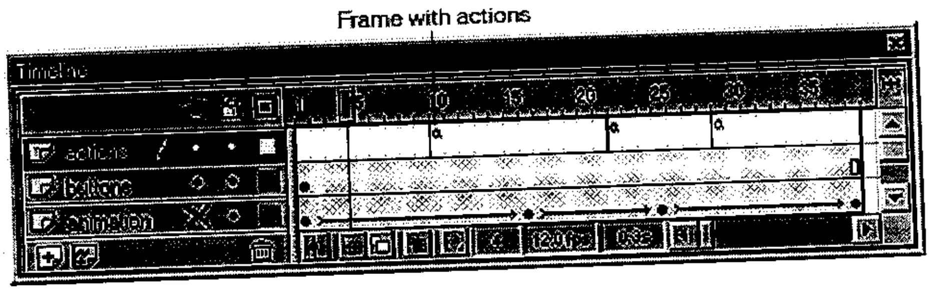
In **frame-by-frame animation,** we create the image in every frame.

In **tweened animation**, we create starting and ending frames and let Flash create the frames in between. Flash varies the object's size, rotation, color, or other attributes evenly between the starting and ending frames to create the appearance of movement.

Tweened animation is an effective way to create movement and changes over time while minimizing file size. In tweened animation, Flash stores only the values for the changes between frames. In frame-by-frame animation, Flash stores the values for each complete frame.

A **keyframe** is a frame where you define changes in the animation. When you create frame-by-frame animation, every frame is a keyframe.
In keyframe (tweened) animation, you define keyframes at important points in the animation and let Flash create the content of frames in between.

The following diagram shows the frames written with actions:

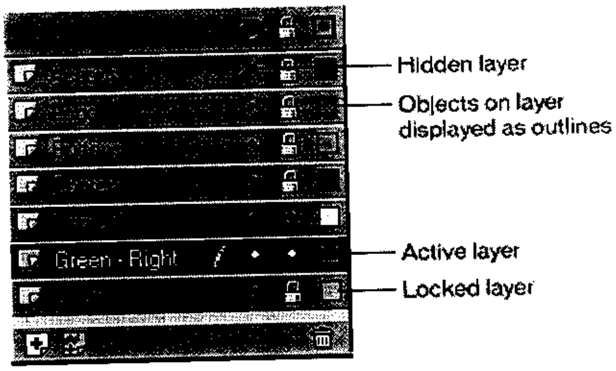


LAYERS

Layers are like transparent sheets of acetate stacked on top of each other. When you create a new Flash movie, it contains one layer. More layers can be added to organize the artwork, animation, and other elements in your movie. Objects can be drawn and edited on one layer without affecting objects on another layer. Where there is nothing on a layer, you can see through it to the layers below.

In addition, you can use special guide layers to make drawing and editing easier, and mask layers to help you create sophisticated effects.

The following figure shows the use of different layer objects:



The layer containing the logo has red outlines.

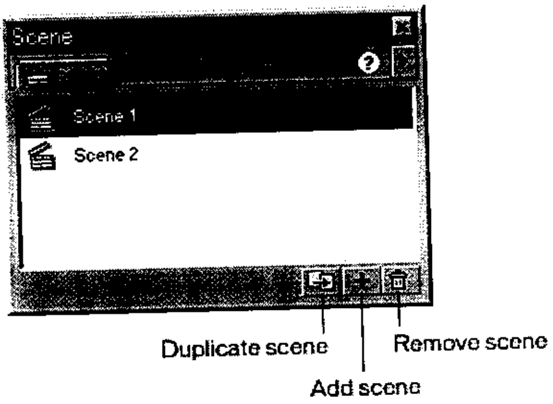
SCENES

Using scenes

Scenes are used to organize a movie thematically. For example, we might use separate scenes for an introduction, a loading message, and credits.

When a Flash movie is published that contains more than one scene, the scenes in the SWF file play back in one sequence in the order they are listed in the Scene panel in the FLA file. Frames in the SWF file are numbered consecutively through scenes. For example, if a movie contains two scenes with ten frames each, the frames in Scene 2 are numbered 11-20.

Scene panel is as follows



Types of symbol behavior

Each symbol has a unique Timeline and Stage, complete with layers. When you create a symbol, you choose how the symbol will behave, depending on how you want to use it in the movie.

Graphic symbols are used for static images and to create reusable pieces of animation that are tied to the Timeline of the main movie.

Button symbols are used to create interactive buttons in the movie that respond to mouse clicks or rollovers or other actions.

Movie clip symbols are used to create reusable pieces of animation. Movie clips have their own multiframe Timeline that plays independent of the main movie's Timeline—think of them as mini-movies inside a main movie that can contain interactive controls, sounds, and even other movie clip instances.

ACTION SCRIPT

Flash uses the ActionScript scripting language to add interactivity to a movie. Similar to JavaScript, ActionScript is an object-oriented programming language. In object-oriented scripting, information is organized by arranging it into groups called classes. Multiple nstances of a class, called objects can be created, to use in your scripts. ActionScript's predefined classes can be used to create our own instances.

Objects in ActionScript can contain data or they can be graphically represented on the Stage as movie clips.

CHARACTERISTICS OF SOFTWARE

What is Internet?

The **Internet** or **"internetwork"** is a collection of co-operating computer networks.
The goal of internetworking is to let many networks inter-operate, enabling computers to communicate with each other. Furthermore, the networks are owned by thousands of individual organizations, governments, universities, individuals, corporations and other groups. getting such diverse and disparate systems and organizations to work together is the primary objective of the internet.

Gateway, router and protocol

Internet requires connecting together different and frequently incompatible networks, sometimes by using machines called gateways to make connection and provide
the necessary translation, both in terms of hardware and software. Thus gateway is a combination of hardware, software and network connections that provide a link between one architecture to another.
A **router** is one kind of gateway. Examples of some gateways are application gateways, e-mail gateways, protocol gateways etc.
A **protocol** is a set of conventions or rules for how to communicate. Every computer that correctly follows the conventions will be able to work with all the other computers those that use the protocol.

TCP/IP protocol

Several network protocols are responsible for transmitting data across the Internet. The key protocol for internet is **TCP/IP** (Transmission Control Protocol/Internet Protocol).IP is the major protocol and TCP is one of the several that work on top of IP to make the internet work. The IP protocol manages the transfer of data across physically diverse networks.

Internet Services

Internet is built out of layers, with each layer depending on the ones below it. Some of the Internet services are Electronic Mail, Telnet, FTP, etc. Each service is a protocol that relies on the lower layers : DNS,TCP and IP.

Web

The web (World Wide Web), in one sense ,is another internet service. The Hyper Text Transfer Protocol (HTTP) is a set of rules for making and fulfilling web requests and any program that implements HTTP can use the web. But the web distinguishes itself from other Internet protocols because it is designed to encapsulate other protocols including FTP, gopher, WAIS, NNTP and Telnet. This makes the web a still higher layer application, running on top of other Internet applications.

Website

A website is a collection of web pages at the same location. web pages are created using HTML.

HTML

HTML stands for Hyper Text Markup Language, which is an application of Standard Generalized Markup Language (SGML). It is a simple language used to define and describe the layout of a web page. HTML also supports multimedia and document links. It consists of special codes which when embedded in text or the left and right brackets

(< & >). These brackets contain instructions known as tags that are not case sensitive.

ASP

The **Microsoft Active Server Pages** is a server side-scripting environment that you can use to create and run dynamic, interactive, high performance web server applications. when your script runs on the server rather than on the client your web server does all the work involved in generating HTML pages that you sent to browsers.

ASP is the feature of and can be used with the following web servers

- Microsoft Internet Information Server 3.0 and 4.0 on windows NT server
- Microsoft peer web servers version 3.0 and 4.0 on windows NT Workstation
- Microsoft Personal webserver on Windows 95

ASP is an ISAPI extension which builds on top of the ISAPI INFRASTRUCTURE to provide a server side application frame work making it even easier to build dynamic web applications. An ASP document can contain both HTML syntax and server side scripting logic. When a webserver receives an HTTP request for the ASP document, a 'Virtual' output HTML is generated for the response using a combination of both HTML static information plus any HTML that is generated by scripting.

ASP enables the scripting logic to interface with a number of ASP institute objects, which automatically handles many of the menial task, and so simplifies the amount of development required. These objects provide ASP with the concept of a user session and allow variables to persist across web pages, until either the session is programmatically abandoned or the web browser is closed. Prior to ASP, the handling of user context across multiple HTTP remarks was a complex process.

PROPOSED SYSTEM

CHAPTER 4

PROPOSED SYSTEM

The proposed system could overcome the above problems by implementing necessary constraints at appropriate junctions. As against the existing systems, the developed system provides a very user-interactive kind of map, where the user can directly go and click his required source and destination on the map. This is quite handy only if the users have some knowledge about the city. If he is entirely new to the city he has to go and search his required location in the map. Our system alleviates this kind of inconvenience by allowing the user to select his required source and destination by choosing from a combo box, instead of navigating through the map.

Our system also provides the local bus timings, which also will be an useful information for the user who wants to go from a particular source to destination. Our proposed system is flexible enough to handle any kind of situation.

DETAILS OF THE DESIGN

CHAPTER 5

DETAILS OF THE DESIGN

5.1 SYSTEM DESIGN

System analysis and design comprises of the input design, db design and output design phases. all these phases are related to one another in some manner.

The input phase helps to enter data into the database for later usage of calculations and to produce the result. Database phase is to store necessary data in a well-defined format, which will be accessed for further calculations. Output phase is for producing the desired output in a suitable fashion.

System design will be done in an integrated way so that the requirements specified by the user can be achieved.

5.2 INPUT DESIGN

Input design to the path module is in a graphical form. In graphical input user can navigate through the map and enter his input interactively by clicking on the landmarks for his source or destination. Inputs for choosing various other options according to user's choice are given in a clear and user understandable form. If a user navigates and clicks on a landmark it the map will be zoomed for user's clear visibility.

Input for other modules of this application are given in a list form which the user can scroll through and click his interested query.

5.3 DATABASE DESIGN

Database is designed to store all necessary information. Different types of information are stored in the database. MS Access is used for this purpose.

The tables used for the various modules are described in the following section

Tables used for the **Timings** module are

1.Table Name : **BusTimings**

Design

S.NO	**FIELD NAME**	**DATA TYPE**	**DESCRIPTION**
1	BusNo	Text	Tells about the bus number
2	Terminal1	Text	Tells about the place from where it starts
3	Terminal2	Text	Tells about the place from where it ends
4	Departure Time	Text	Tells about the departure time from Gandhipuram or Ukkadam Bus Stands
5	Route1	Text	Tells about a place in between source and destination
6	Route2	Text	Tells about a place in between source and destination
7	Route3	Text	Tells about a place in between source and destination
8	Route4	Text	Tells about a place in between source and destination
9	Route5	Text	Tells about a place in between source and destination

2.Table Name : **Train**

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	ExpPass	Text	Tells whether Express/Passenger tarin
2	TrainNo	Text	Tells Train Number
3	TrainName	Text	Tells Train Name
4	Source	Text	Tells from which station it starts
5	Destination	Text	Tells to which station it reaches
6	Arrival	Text	Tells the time it arrives at Coimbatore Station
7	Departure	Text	Tells the time it departs at Coimbatore Station
8	Days	Text	Tells the days of the week the train travels

3.Table Name : **AirTimings**

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	Airline	Text	Tells about Airline type
2	FlightNo	Text	Tells about Flight No
3	From	Text	Tells about the source of the plane
4	To	Text	Tells about the destination of the plane
5	Arrival	Text	Tells about the arrival time of the plane to Coimbatore
6	Departure	Text	Tells about the departure time of the plane from Coimbatore
7	Days	Text	Tells about the days of the week the plane travels

Tables used for the **Directory** module are

1.Table Name : TypeDetails

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	Type Code	Number	Tells about the code of an organization type [Primary Key]
2	Type Name	Text	Describes the type code

2.Table Name : **OrganizationDetails**

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	Name	Text	Tells about the name of the organization
2	Type Code	Number	Tells about the code of an organization type
3	Number and Complex	Text	Tells about the door no. & the complex (if any), the organization is there
4	Street Name	Text	Tells about the street name on which the organization is there
5	Area	Text	Tells about the area in which the organization is there
6	City	Text	Tells about the city in which the organization is located
7	Pin Code	Text	Tells about the pin code of the area the organization is located
8	Phone Number	Text	Tells about the phone numbers of the organization
9	Email	Text	Email of the organization (if any)

10	Web Address	Text	Website of the organization (if any)
11	Fax	Text	Fax of the organization
12	Other Info	Text	Other details of the organization

Tables used for **Bookmark** module are

1.Table Name: **PathTrans**

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	DateOfTrans	Text	The date the transaction was made
2	PathType	Text	Type of path option chosen
3	Source	Text	Source entered for path calculation
4	Destination	Text	Destination entered for path calculation

2.Table Name: **Directorytrans**

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	DateOfTrans	Text	The date the transaction was made
2	SearchMethod	Text	Tells about the search method as whether by name or type or street name
3	OrgType	Text	Type of the organization
4	OrgName	Text	Name of the organization
5	OrgStreet	Text	Street where the organizatiom is there

3.Table Name: **Placelist**

Design

S.NO	FIELD NAME	TYPE	DESCRIPTION
1	Node	Text	The node number of a place
2	location	Text	The places in the covered area

5.4 OUTPUT DESIGN

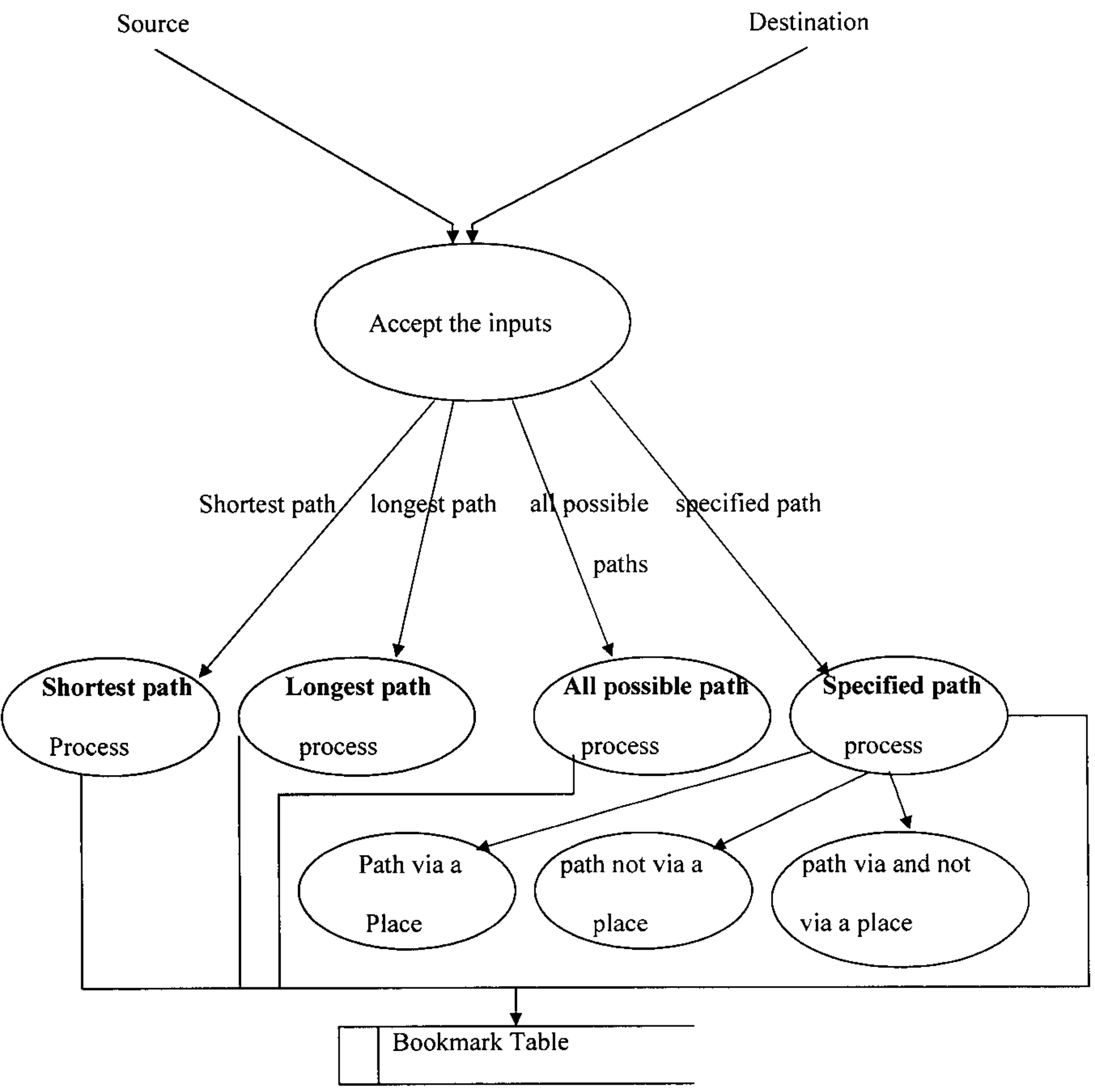
Output design is very important since the accuracy and ease of understanding the output is important. The output should be in a suitable format so that the user is fully satisfied with the result. The output should be able to convey a clear message about the status of the system.

Output design for this application, similar to input design can be given both textually and graphically.

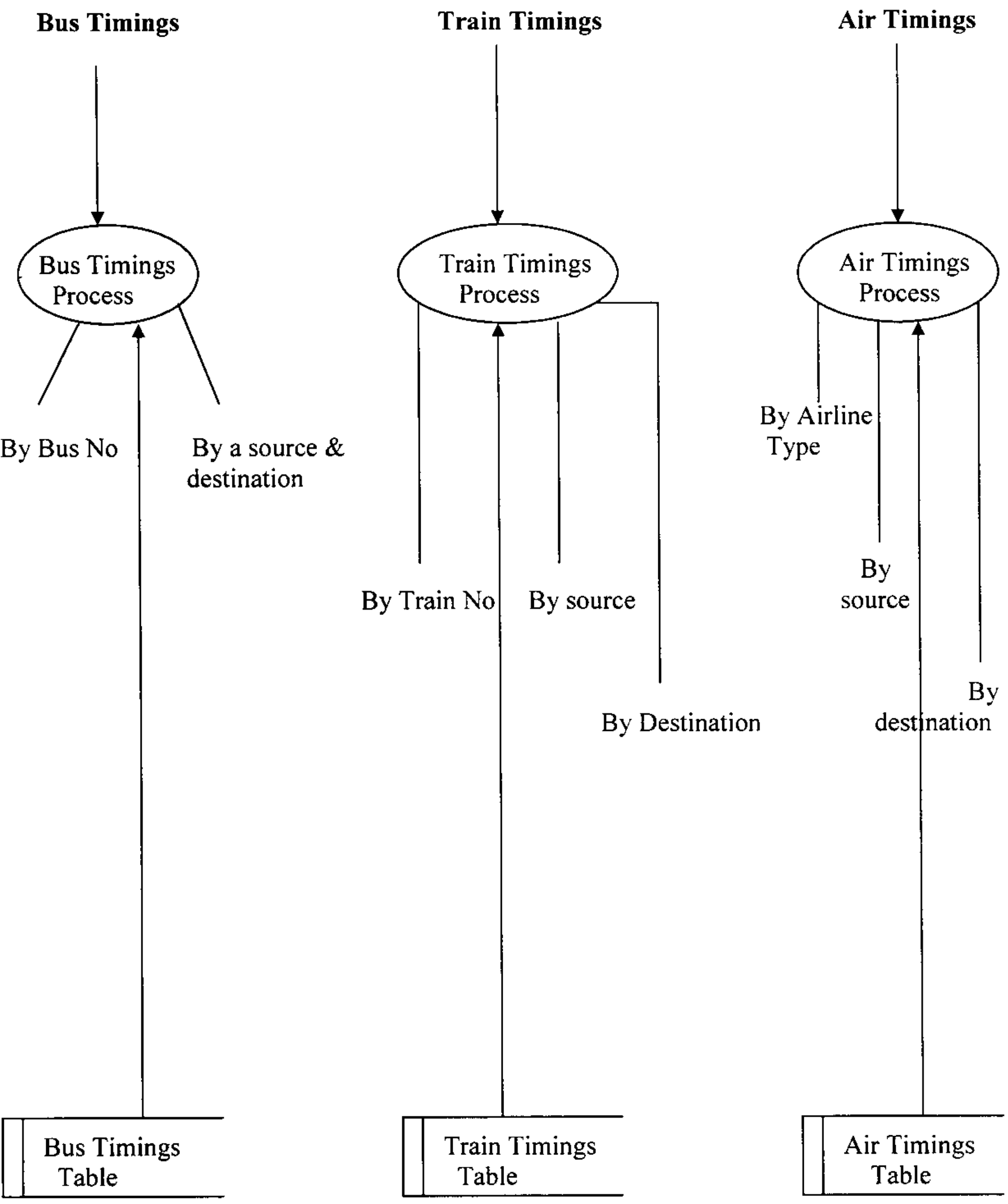
For example,in Path module the routes for a source and destination can be displayed in text according to textual mode.According to graphical mode the exact route can be highlighted on the map between a source and destination which provides an easy way of understanding the output.

5.5 DATA FLOW DIAGRAM

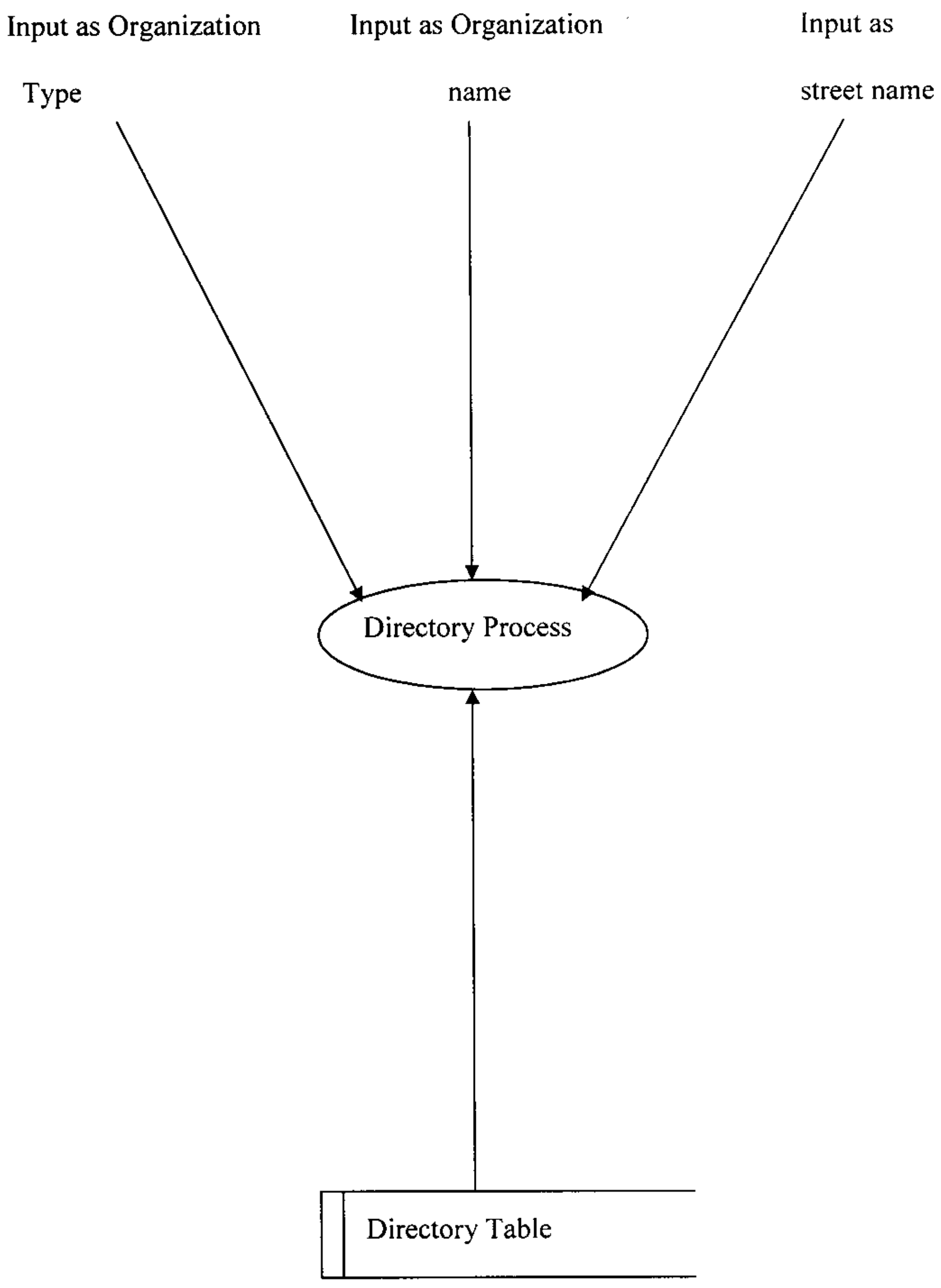
Path Module



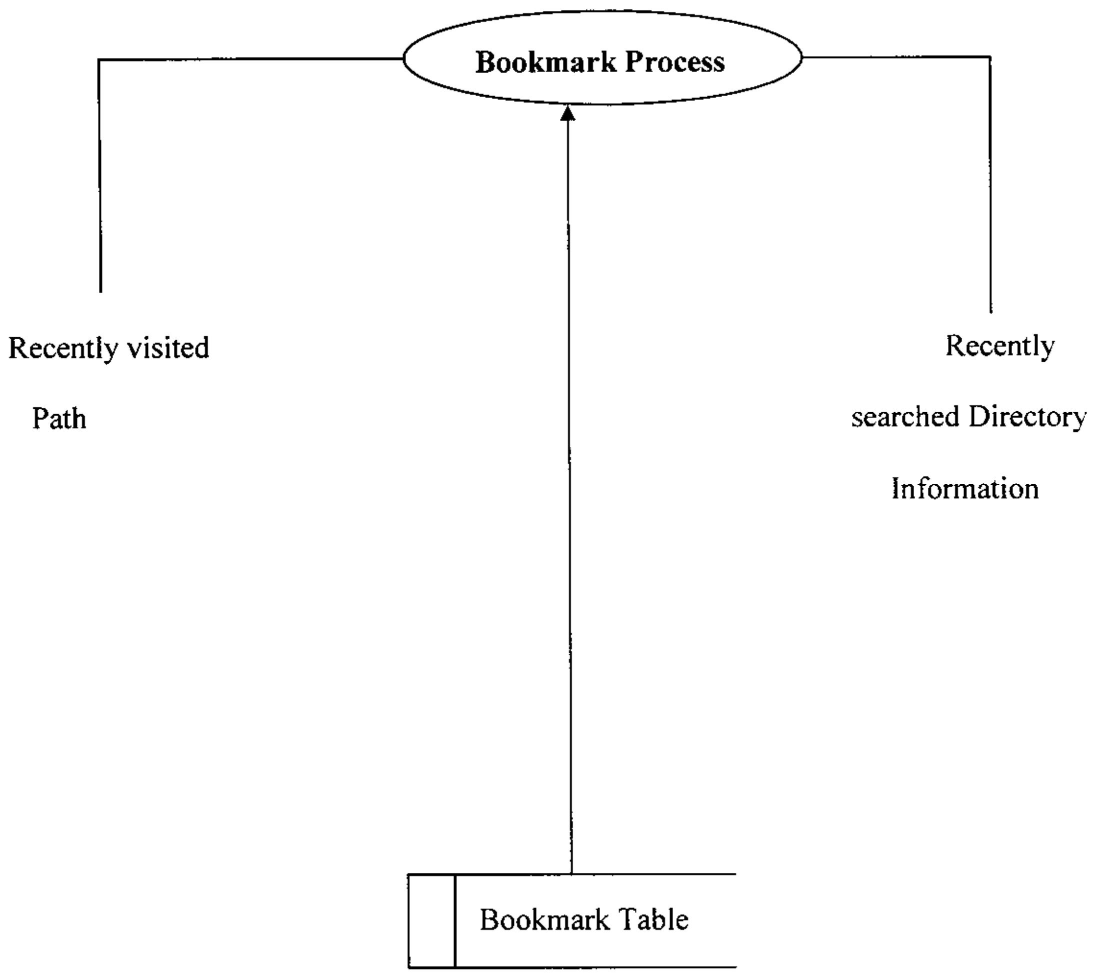
Timings Module



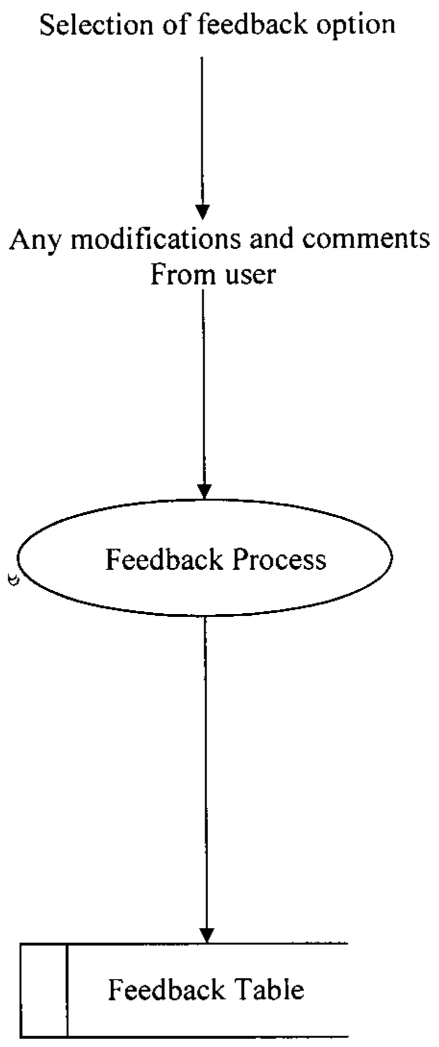
Directory Module



Bookmark Module



Feedback Module


Selection of feedback option
Any modifications and comments From user
Feedback Process
Feedback Table

IMPLEMENTATION DETAILS

CHAPTER 6

IMPLEMENTATION DETAILS

SYSTEM CODING

Coding for this software has been done using Flash's Action scripting, MS Access's database using ASP as an interface between the two.

Flash has the facility of creating high interactive web applications using a simple language called Action Script.

The software City Navigator is divided into a number of modules for handling the correctness, to reduce the complexity and to improve the system performance. Each module has specific functions and they are related to each other in some manner.

The system follows a well structured form and codes are optimized as much as possible. All codes are well documented so that it is understandable to others. Error messages are displayed when the user tries to enter wrong data Coding is done with flexibility as a main feature in mind.

The output of the software is made in different formats so that the user can understand the status of the system.

CONCEPT

The algorithm which we have developed to find the paths is as follows

Algorithm All(src,des). Given a source and destination this algorithm computes all the possible paths to reach the destination from the specified source.

S is the stack that stores the nodes being traversed. **Possible** is an array which stores all the links between the nodes in the network **Dest** is an array containing the flag status of the links. **Path** is an array used to store the paths from the source to the destination

STEP 1

Obtain the source and destination as the input.

STEP 2

Push the source into the stack (S).

STEP 3

```
While S not empty do

    ❖ Find a link for the topmost element of the stack from the possible array . This link
      (node) should satisfy the following conditions:
        ➢ The dest flag of the particular link should not be set.
        ➢ The node should not be the destination.
        ➢ The node should not be present already in the stack.
        If all the above conditions are satisfied then
            Push the node into the stack
        Else
            If the node is the destination then
                Copy the contents of the stack along with the destination and place it in
                the Path array.
            End If
            If the dest  flag is set then
                Find the next link in the possible array for the node.
            End If

            If there is no link found in the search then
                Pop the top element of the stack.
                Set the dest flag of all the links of the element that is popped to zero.
          End if.
    End if.
```

The functions used here are

(i).findlink(n,ptr).

n refers to the number of nodes in the network and ptr is the pointer to the stack.

This function is used to find the link of a node in the possible array.

(ii).already(a,ptr)

a refers to the node which has a link to the top element of the stack and ptr is the pointer to the stack.

This function is used to determine whether the node is already present in the stack.

(iii).makedest(b,n)

b refers to the top element of the stack and n is the number of nodes in the network.

This function is used to set the dest flag of all the links of the top element of the stack to zero.

(iv).stackdisplay(ptr,des,xindex)

ptr refers to the pointer to the stack, des to the destination node and xindex to the pointer of the path array.

With this as a basic function the longest and shortest paths are determined. This is done by comparing the distances and finding the maximum and minimum values respectively.
For the specific path, all the possible paths obtained from the basic function is analyzed and the desired path either through, not through or a combination of both (i.e. through a particular place and not through another place) is found.

USING FLASH FOR DISPLAY

Flash is used to interactively highlight the path between a start and an end. This is achieved by converting all the nodes, which in our case represents a place or a landmark, as buttons for easy operation on these nodes. The arcs connecting nodes, which in our project are the roads connecting the places are drawn as movie clips for achieving the interactive display of the path. A program written in ActionScript controls all the path calculations and the display.

TESTING

CHAPTER 7

TESTING

SYSTEM TESTING

System testing in the style of implementation, which is aimed at ensuring that the system works at all levels and is effective to deal any situation. The system test should be a definite confirmation that all are correct and an opportunity to show the users that the system works properly.

This software has been tested under different circumstances with various kinds of data. Verification and validation are performed on the system to show the specified security and reliability.

At each and every stage of the software, necessary procedures are carried out to avoid complexity due to errors in later stages. The displayed outputs are also stored for future reference. It also accepts modifications for further enhancements.

INTEGRATION TESTING

Strategies for integrating software components into a functioning product include the bottom up strategy, the top down strategy and the sandwich strategy. Careful planning and scheduling are required to ensure that modules will be available for integration into the evolving software product when needed. The integration strategy dictates the order in which the modules must be available and thus exerts a strong influence on the order in which modules are written, debugged and unit tested. The project was tested once the modules were integrated to ensure the proper working of the software.

ACCEPTANCE TESTING

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests to verify that the implemented system satisfies its requirements. Acceptance tests are performed by the quality assurance and / or the customer organizations. Depending on local circumstances the development group may or may not be involved in acceptance testing.

CONCLUSION

CHAPTER 8

CONCLUSION

The project "**City Navigator**" thus serves as a better online guide to the visitors to the city. It gives instructions on the basis of routes, timings of different modes of transport, and contact details and addresses of some important commercial centers and organizations. Since the interface is very user friendly it needs very less effort for the user to get any information. There is an option for the user to store his recently visited pages using the bookmark option.

The user is also provided with feedback facility, which enables him to pass any comments and feedback about the software or about the information provided. This enables the system to provide completely updated information. The results for the path module are displayed in graphical manner which helps the new visitor to the city to identify places with the help of the landmarks displayed in the map. Also since all the types of paths as shortest, longest, and specific paths are shown the user can choose a path that contains the places he/she wants to visit in between the source and destination. Also in future when all the enhancements planned such as considering the traffic constraint, railway crossings etc. are incorporated the user will be benefited the most.

FUTURE OUTLOOK

CHAPTER 9

FUTURE OUTLOOK

The system can be modified with any technology that has both the Flash and ASP supports. The system does not use any complex mathematical calculations but ordinary programming techniques.

The developed system covers only a small part of the city. It can be expanded to cover the entire city to include details from all spheres.

The present system is static, in the sense it doesn't guide the people as exactly where he is located in the calculated path. It can be made dynamic by incorporating the system with GPS(Global Positioning System) to guide the user through travel by displaying the present location where he is and how to get to his desired destination.

Also the present system can be used while traveling only if the user has access to Internet. As a future enhancement the developed system can be embedded in a portable device by incorporating the program in the chip of the device.

Future modifications to the project can be in terms of the path considering the traffic strength at different times of a day and stating that whether a path is feasible considering a railway blockade in the calculated path. If there is any railway crossing in the path, depending on the time of occurrence of the blockade the path can be calculated

If all the above-mentioned facilities are incorporated the software will be of utmost use to travelers and to those who are new to the city.

REFERENCE

CHAPTER 10

REFERENCE

1. Bill Sanders, Flash5 ActionScript, dreamtech Press.
2. A.Russell Jones,Active Server Pages 3,BPB Publications.
3. www.macromedia.com
4. Flash Tutorials.

APPENDICES

CHAPTER 11

APPENDICES

APPENDIX A- SAMPLE CODING

```
//Introduction-Path option

on (press) {
	gotoAndPlay("map", 1);
}
on (press) {
	option = "";
	source = "";
	destination = "";
	warning = "";
	fscommand("fullscreen", "true");
	var tempo = -1, opt = -1;
	var dsource, ddestination, dpath, dvia = "--not applicable--", dnotvia = "--not applicable--";
	function find(but) {
		tempo = tempo+1;
		if (tempo<1) {
			src = but;
		} else {
			des = but;
		}
		if (src>0 and des>0) {
			warning = "";
		}
		return tempo;
	}
}

//Map-Shortest option

on (press)
{
 option=" Type Of Path : Shortest Path"
  opt=1;
  dpath="shortest"
 warning="";
}
```

```
//Map-Navigate option

on (press) {
        if (opt<0) {
                warning = "Choose Ur Option";
        }
else {
                if (opt>0 and source != null and destination != null and src>=0 and dest>=0 and src
!= dest) {
                        loadVariables("Pathtrans.asp", "", "POST");
                        gotoAndStop("output scene", 1);
                }
        }
}

//Map-Reset option

on (press)
 {

        serial="";
   source="";
   destination="";
   option="";
        warning="";
        src=-1;
        dest=-1;
        tempo=-1;
        opt=-1;
}

//Map-Specific path option

on (press) {
        if (source == null and destination == null) {
                warning = "Enter proper source and destination";
        } else if (source == null) {
                warning = "Enter proper source";
        } else if (destination == null) {
                warning = "Enter proper destination";
        } else if (source != null and destination != null) {
                option = " Type Of Path : Specific Path";
                opt = 4;
                warning = "";
                gotoAndStop("SpecificPath", 1);
        }
        dpath="specific"
```

```
}

//Output Scene- Frame

/* Declarations */
path = new Array();
par=new Array();
distance_arr=new Array();
s=new Array();
alldisp=new Array();
alldist=new Array();
specdisp=new Array();
specdist=new Array();
lonarray=new Array();
shortarray=new Array();
spec=new Array();
notspec=new Array();
combspec=new Array();
var cnt_all=0;
var ncnt = 0, len, temp1 = 0, combspecar=0,specar=0,notspecar=0,i, j, f = 0, x = 0, x1, y, z = 0, z1
= 0, k, cnt = 0, temp = 0, b1, ser = 0;
var n, temp2 = 0, temp3 = 0, a = 0, b = 0, tcnt = 0, ncmp = 0, h = 0;
var opt,svopt;
var tmp2 = 0;
var stptr,loc,temp,a,tempvar,des,k=21,src,ptr,n,k,i,j,xindex=0;
possible=[[0,14],[1,0],[2,1],[3,2],[4,3],[4,17],[5,4],[5,19],[6,5],[6,21],[7,6],[8,7],[8,9],[8,10],[8,11],
[8,13],[9,8],[10,8],[10,11],[11,10],

[12,13],[12,20],[13,8],[13,12],[13,21],[14,15],[14,16],[15,1],[15,17],[16,14],[16,17],[16,18],[17,4],
[17,15],[17,16],[17,19],[18,16],[18,19],

[18,20],[19,5],[19,17],[19,18],[19,21],[20,12],[20,18],[20,21],[21,6],[21,13],[21,19],[21,20]];
dest=[0,0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0];
var k=possible.length;

dist=[[0,      0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0.52,  0,     0,     0,     0,     0,     0,     0],
[0.13,  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0.24,  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0.1,   0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0,     0.15,  0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0.11,  0,     0,     0,     0],
[0,     0,     0,     0,     0.05,  0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0.12,  0,     0],
[0,     0,     0,     0,     0,     0.11,  0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0.19],
[0,     0,     0,     0,     0,     0,     0.22,  0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0,     0,     0,     0,     0,     0.12,  0,     0.26,  0.4,   0,     0,
        0.44,  0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0,     0,     0,     0,     0,     0,     0.26,  0,     0,     0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0,     0,     0,     0,     0,     0,     0.4,   0,     0,     0.5,   0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.5,   0,     0,
        0,     0,     0,     0,     0,     0,     0,     0,     0],
[0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0.1,   0,     0,     0,     0,     0,     0,     0.24,  0],
[0,     0,     0,     0,     0,     0,     0,     0,     0.44,  0,     0,     0,     0.1,
        0,     0,     0,     0,     0,     0,     0,     0,     0.59],
[0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0.1,   0.46,  0,     0,     0,     0,     0],
[0,     0.12,  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0.5,   0,     0,     0,     0],
[0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0.46,  0,     0,     0.14,  0.06,  0,     0,     0],
[0,     0,     0,     0,     0.11,  0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0.5,   0.14,  0,     0,     0.06,  0,     0],
[0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0.06,  0,     0,     0.14,  0.07,  0],
[0,     0,     0,     0,     0,     0.12,  0,     0,     0,     0,     0,     0,     0,
        0,     0,     0,     0,     0.06,  0.14,  0,     0,     0.07],
[0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.24,
        0,     0,     0,     0,     0,     0.07,  0,     0,     0.1],
[0,     0,     0,     0,     0,     0,     0.19,  0,     0,     0,     0,     0,     0,
        0.59,  0,     0,     0,     0,     0,     0.07,  0.1,   0]];

rout=[[ "", "" ,"",    "",  "","" ,"",   "r07", "" ,""  ,""]
[ "r10","" ,"",    "",  "","" ,"r16",   "", "" ,""  ,""]
[ "", "r21" ,"",   "",  "","" ,"r26",   "", "" ,""  ,""]
[ "", "" ,"r32",   "",  "r34","" ,"",   "",  "" ,"r39"  ,""]
[ "", "" ,"",    "r34","","r45" ,"",   "", "r48" ,"r49"  ,""]
[ "", "" ,"",    "",  "r45","" ,"r56",   "", "" ,""  ,""]
[ "", "r16" ,"r26","",  "","r56" ,"",   "r67", "" ,""  ,""]
[ "r70","","",    "",  "","" ,"r67",   "", "" ,""  ,""]
[ "",  "" ,"",     "",  "r48","" ,"",   "",  "" ,"" ,""]
[ "",  "" ,"",    "r39","r49","" ,"",   "",  "" ,"" ,"r910"]
[ "",  "" ,"",     "",  "","" ,"",   "",  "" ,"r910"  ,""]];
place=["Pykara Power House","Shobika","Singapore Plaza","Kannan Departmental
Stores","Zimson Showroom","Lakshmi Complex","Sri Krishna Sweets","Mahaveers","Town Bus
Stand","CAG Pride","Central Bus Stand","Heritage Inn","Arul Jothi Lodge","GP Hospital","Arya
Vaidya Sala","Kongunadu Hospital","Karpagam Complex","10th Street","Extn 9","9th
Street","Extn 7","7th Street"];

n = dist.length;
heading="";
tempdisplay="";
testspec = "";
tempspec = "";
tempnotspec = "";
testnotspec = "";
serial1=1;
serial2="";
dispdistance="";
dtxt1="";
dtxt2="";
display="";
txt1="";
txt2="";
if (des != -1 and src != -1) {

        if (opt == 1) {
                shortest(src,des);
                                option=" Type: Shortest Path";
        } else if (opt == 2) {
                                longest(src,des);
                                option=" Type: Longest Path ";
        } else if (opt == 3) {
                                alldistancedisplay(src, des);
                                option=" Type: AllPossible Paths ";
        }else if (opt == 4 and displayvia!="" and displaynotvia!="") {
                     specificcomb(src,des,via,notvia);
                                option=" Type: Specific Path ";
                                svopt=1;
        }else if (opt == 4 and displayvia=="" and displaynotvia!=""){
                    specificnotthrough(src,des,notvia);
                          option=" Type: Specific Path ";
                          svopt=2;
        }else if (opt == 4 and displayvia!="" and displaynotvia == ""){
                    specificthrough(src,des,via);
                          option=" Type: Specific Path ";
                          svopt=3;
        }
}
//Basic Function to calculate all possible paths
function all(src,des)
{
stptr=0;
s[stptr]=src;

while(stptr>=0)
        {

          loc=findlink(k,stptr);
          if(loc>=0)
          {
          a=possible[loc][1];
           if(a!=des)
                 {
                  stptr=stptr+1;
                  s[stptr]=a;
                  dest[loc]=1;
                 }
                 if(a==des)
                  {
                    stackdisplay(stptr,des,xindex);
                          cnt_all=cnt_all+1;
                          xindex=xindex+1;
                   dest[loc]=1;
                  }
                 }
                 else
                 {
                 if(loc==-1)
                  {
                   tempvar=s[stptr];
                   stptr=stptr-1;
                   makedest(tempvar,k);
                  }
                 }
}

// To display all possible paths between specified  source and destination
function alldistancedisplay (src,des)
{
        all(src,des);
        allpossible = "";
        counter = "No. Of Paths: "+cnt_all;
        dis="";
        if(cnt_all>1)
        {
                serial2=2;
        }
        heading =" Possible Paths To Reach "+place[des]+"  from "+ place[src] ;
        for (i=0; i<cnt_all; i++)
    {
                tempdisplay="";
                path="";
                path=par[i].split(":",length(par[i]));
                for(j=0;j<path.length;j++)
                {
                t1 = path[j];
                t2 = path[j+1];
                tempdisplay=tempdisplay+place[t1]+" ";
                }
                alldisp[i]=tempdisplay;
                alldist[i]=distance_arr[i]+" km";
    }
        count=cnt_all;
        alldispcnt=alldisp.length;
        alldispi=alldispcnt;
        alldispj=0;

        txt1=alldisp[alldispj];
        txt2=alldisp[alldispj+1];

        display=alldisp[0];
        dispdistance=alldist[0];

        dtxt1=alldist[alldispj];
        dtxt2=alldist[alldispj+1];


}
//To Find Longest Path

function longest (src,des)
{
   serial2="";
        txt2="";
        dtxt2="";

        all(src,des);
        var longdis = 0;
        var longpath="";
```

```
for (i=0; i<cnt_all; i++)
{
        if (distance_arr[i]>longdis)
        {
                longdis = distance_arr[i];
                longpath = par[i];
        }
}
dis = "Longest Distance :"+longdis+"km";

heading = " Longest  Paths To Reach From "+place[src] +"  to "+place[des];
count = 0;
for (i=0; i<cnt_all; i++)
{
        if (distance_arr[i]== longdis)
        {
                count = count+1;
                lonarray[count]=par[i];
        }
}
counter = "No. Of Paths: "+count;

if(count>0)
{
        for(i=1;i<=count;i++)
        {
    lp="";
         lp=lonarray[i];
         tempdisplay="";
         path=lp.split(":",length(lp));
                for(j=0;j<path.length;j++)
                {
                t1 = path[j];
                t2 = path[j+1];
                tempdisplay=tempdisplay+place[t1]+" ";
                }
                alldisp[i]=tempdisplay;
        alldist[i]=longdis+" km";
        }

 }
        display=alldisp[1];
        dispdistance=alldist[1];
        alldispcnt=alldisp.length;
        alldispi=alldispcnt;
        alldispj=1;
```

```
            txt1=alldisp[alldispj];
            dtxt1=alldist[alldispj];

        if(count>1)
        {
                serial2=2;
                txt2=alldisp[alldispj+1];
                dtxt2=alldist[alldispj+1];
        }
        else
        {
                serial2="";
                txt2="";
                dtxt2="";
        }

}

// To find the shortest path between specified source and destination

function shortest (src,des)
{
   serial2="";
        txt2="";
        dtxt2="";

        all(src,des);
        var shortdis = 1000;
        var shortpath="";

        for (i=0; i<cnt_all; i++)
        {

                if (distance_arr[i] < shortdis)
                {
                        shortdis = distance_arr[i];
                        shortpath = par[i];
                }
        }


        dis = "Shortest Distance :"+shortdis+"km";
        heading = " Shortest  Paths To Reach From "+place[src] +"  to "+place[des];
        count = 0;

        for (i=0; i<=cnt_all; i++)
        {
```

```
        if (distance_arr[i] == shortdis)
        {
                count = count+1;
                shortarray[count]=par[i];

        }
}
counter = "No. Of Paths: "+count;

if(count>0)
{
        for(i=1;i<=count;i++)
        {
    lp="";
         lp=shortarray[i];
         tempdisplay="";
         path="";
 path=lp.split(":",length(lp));
                for(j=0;j<path.length;j++)
                {
                t1 = path[j];
                t2 = path[j+1];
                tempdisplay=tempdisplay+place[t1]+" ";
                }
                alldisp[i]=tempdisplay;
        alldist[i]=shortdis+" km";
        }

}

        display=alldisp[1];
        dispdistance=alldist[1];
        alldispcnt=alldisp.length;
        alldispi=alldispcnt;
        alldispj=1;

        txt1=alldisp[alldispj];
        dtxt1=alldist[alldispj];

 if(count>1)
 {
         serial2=2;
         txt2=alldisp[alldispj+1];
         dtxt2=alldist[alldispj+1];
 }
 else
 {
```

```
                serial2="";
                txt2="";
                dtxt2="";
        }

}
//To find specific path through a particular place
function specificthrough (src, des, via)
{
 specar=0;
 t1="";
 t2="";
 path="";
 speccnt = 0;
        all(src, des);
        for (i=0; i<cnt_all; i++)
        {
                testspec = par[i];
                path=testspec.split(":",length(testspec));
                for (j=0; j<path.length; j++)
                {
                        if (path[j] == via)
                        {
           spec[specar]=path;
                                alldist[specar]=distance_arr[i]+" km";
                                specar=specar+1;

                        }
                }
        }

   if (specar == 0)
        {
                trace ("No path found!!!Sorry!!!");
        }

        counter = "No. Of Paths: "+specar;
        heading = " Specific Paths To Reach From "+place[src] +"  to "+place[des]+" Via
"+place[via];

        if(specar>0)
        {
           if(specar>1)
                {
                        serial2=2;
                }
```

```
            for(i=0;i<specar;i++)
            {
        lp="";
             path=spec[i];
             tempdisplay="";

                    for(j=0;j<path.length;j++)
                    {
                    t1 = path[j];
                    t2 = path[j+1];
                    tempdisplay=tempdisplay+place[t1]+" ";
                    }
                    alldisp[i]=tempdisplay;
            }

    }

    alldispcnt=alldisp.length;
    alldispi=alldispcnt;
    alldispj=0;

    txt1=alldisp[alldispj];
    txt2=alldisp[alldispj+1];

    display=alldisp[0];
    dispdistance=alldist[0];
    dtxt1=alldist[alldispj];
    dtxt2=alldist[alldispj+1];

    }

//To find a specific path not through a particular place
function specificnotthrough (src,des, via) {
  t1="";
  t2="";
       notspecar=0;
  notspeccnt = 0;
       all(src, des);
       for (i=0; i<cnt_all; i++) {
              notspeccnt = 0;
              path=par[i].split(":",length(par[i]));
              for (j=0; j<path.length; j++)
              {
                     if (path[j] == via)
                     {
                            notspeccnt = notspeccnt+1;
                     }
            }
            if (notspeccnt == 0) {
                    trace ("Path is"+path);
               notspec[notspecar]=path;
                    alldist[notspecar]=distance_arr[i];
                    notspecar=notspecar+1;
            }
        }

counter = "No. Of Paths: "+notspecar;
heading = " Specific Paths To Reach From "+place[src] +"  to "+place[des]+" Notvia "+place[via];

if(notspecar>0)
        {
          if(notspecar>1)
              {
                      serial2=2;
              }

              for(i=0;i<notspecar;i++)
              {
          nsp="";
                path=notspec[i];
                ntempdisplay="";

                      for(j=0;j<path.length;j++)
                      {
                      t1 = path[j];
                      t2 = path[j+1];
                      ntempdisplay=ntempdisplay+place[t1]+" ";
                      }
                      alldisp[i]=ntempdisplay;
              }

        }



        alldispcnt=alldisp.length;
        alldispi=alldispcnt;
        alldispj=0;

        txt1=alldisp[alldispj];
        txt2=alldisp[alldispj+1];

        display=alldisp[0];
        dispdistance=alldist[0];
        dtxt1=alldist[alldispj];
        dtxt2=alldist[alldispj+1];
}

//Function to calculate combination of both
function specificcomb (src,des,via,notvia)
{
        combcnt=0;
        combspecar=0;
        specdispcnt=0;
        ctempdisplay="";

        specificthrough(src,des,via);
        for(i=0;i<specar;i++)
                {
      combcnt=0;
                path=spec[i];
                for(j=0;j<path.length;j++)
       {
        if(path[j] ==notvia)
                        {
         combcnt=combcnt+1;
                        }
       }
     if(combcnt==0)
       {
        combspec[combspecar]=spec[i];
                 specdist[combspecar]=distance_arr[i]+" km";
                 combspecar=combspecar+1;
                 }
         }
                counter = "No. Of Paths: "+combspecar;
                heading = " Specific Paths To Reach From "+place[src] +" to "+place[des]+"
 Notvia "+place[notvia] +" Via "+place[via];

        serial1="";
        serial2="";
   dtxt1="";
   dtxt2="";
        display="";
        dispdistance="";

        if(combspecar>0)
        {
          serial1=1;
                if(combspecar>1)
                {
                        serial2=2;
                }

                for(i=0;i<combspecar;i++)
                {
            nsp="";
                  path=combspec[i];
                  ntempdisplay="";

                        for(j=0;j<path.length;j++)
                        {
                        t1 = path[j];
                        t2 = path[j+1];
                        ntempdisplay=ntempdisplay+place[t1]+" ";
                        }
                        specdisp[i]=ntempdisplay;
                }

        }
        alldisp=specdisp;
        alldist=specdist;
        alldispcnt=alldisp.length;
        alldispi=alldispcnt;
        alldispj=0;

        txt1=alldisp[alldispj];
        txt2=alldisp[alldispj+1];

        display=alldisp[0];
        dispdistance=alldist[0];

        dtxt1=alldist[alldispj];
        dtxt2=alldist[alldispj+1];

}

//Output Scene-Combo Box Up Button
on(press)
{
        if(alldispj>=1)
        {
        alldispj=alldispj-1;
        txt2=txt1;
        txt1=alldisp[alldispj];
        dtxt2=dtxt1;
        dtxt1=alldist[alldispj];
        serial1=serial1-1;
        serial2=serial2-1;
        }

}
}

//Output Scene-Display option Button
on (press) {
        display = txt1;
        dispdistance = dtxt1;
        if (opt == 3) {
                lonp = par[serial1-1].split(":", length(par[serial1-1]));
        }
        if (opt == 1) {
                pathshort = shortarray[serial1].split(":", length(shortarray[serial1]));
                lonp = pathshort;
        }
        if (opt == 2) {
                pathlong = lonarray[serial1].split(":", length(lonarray[serial1]));
                lonp = pathlong;
        }
        if (opt == 4 and svopt == 3) {
                lonp = spec[serial1-1];
        }
        if (opt == 4 and svopt == 2) {
                lonp = notspec[serial1-1];
        }
        if (opt == 4 and svopt == 1) {
                lonp = combspec[serial1-1];
        }
        gotoAndStop("input scene1", 1);
}

//Input Scene-Pykara node Button
//This is similar for all the other nodes
on (press) {
        temp = find(0);
        if (temp<1) {
                source = "Source : Pykara Power House";
                dsource = "Pykara Power House";
        } else {
                destination = "Destination : Pykara Power House";
                ddestination = "Pykara Power House";
        }
}

//FrontPageDirectory-Organization type/Street
```

```
on (release)
{
        dirsearch="Organization Type/Street";
        gotoAndStop("Organization Type / Street","1");
}
//Air timings-Frame
pla = new String();
path = new Array();
path1= new Array();
SrcAir="Loading....";
DestAir="Loading....";
AirName="Loading....";
loadVariables("AirTimings.asp","","POST");
function ASinitpath()
{
pla=SrcAir
path1=pla.split("\r",pla.length);
j=0;
for(i=0;i<path1.length;i++)
{
          if(path1[i]!="")
          {
           path[j]=path1[i];
           j++;
          }
}
}

function ADinitpath()
{
pla=DestAir
path1=pla.split("\r",pla.length);
j=0;
for(i=0;i<path1.length;i++)
{
          if(path1[i]!="")
          {
           path[j]=path1[i];
           j++;
          }
}
}

function ANinitpath()
{
pla=AirName
path1=pla.split("\r",pla.length);
```

```
j=0;
for(i=0;i<path1.length;i++)
{
        if(path1[i]!="")
        {
         path[j]=path1[i];
         j++;
        }
}
}
function findscroll(c)
{
        count1 = path[c];
        gotoAndStop("air source", "1");
}

function findscroll1(c)
{
        count1 = path[c];
        gotoAndStop("air destination", "1");
}

function findscroll2(c)
{
        count1 = path[c];
        gotoAndStop("air name", "1");
}

//Train Timings

Etrainname="Loading...........";
SrcTrain="Loading...........";
DestTrain="Loading...........";
var but, Ecntup = 0, Ecntdown = 0, up = 0, down = 0, t = 0, Ech, Pch, Pcntup = 0, Pcntdown = 0,
Scntup = 0, Scntdown = 0, Dcntup = 0, Dcntdown = 0, Sch, Dch;
pla = new String();
path = new Array();
path1= new Array();
a = new Array();
b = new String();
loadVariables("TrainTimings.asp", "");
function Einitpath()
{
pla=Etrainname
path1=pla.split("\r",pla.length);
j=0;
for(i=0;i<path1.length;i++)
```

```
{
          if(path1[i]!="")
          {
           path[j]=path1[i];
           j++;
          }
}
}

function Pinitpath()
{
pla=Ptrainname
path1=pla.split("\r",pla.length);
j=0;
for(i=0;i<path1.length;i++)
{
          if(path1[i]!="")
          {
           path[j]=path1[i];
           j++;
          }
}
}

function Sinitpath()
{
pla=SrcTrain
path1=pla.split("\r",pla.length);
j=0;
for(i=0;i<path1.length;i++)
{
          if(path1[i]!="")
          {
           path[j]=path1[i];
           j++;
          }
}
}

function Dinitpath()
{
pla=DestTrain
path1=pla.split("\r",pla.length);
j=0;
for(i=0;i<path1.length;i++)
{
          if(path1[i]!="")
```

```
        {
         path[j]=path1[i];
         j++;
        }
}
}


function find(c) {
        but = c;
}
function findscroll(up, down,ch) {
        if (up>down) {
                t = up-down;
        } else {
                t = down-up;
        }
        c=t+ch;
        count = path[c];
        gotoAndStop("train name", "1");
}

function findscroll1(up, down,ch) {
        if (up>down) {
                t = up-down;
        } else {
                t = down-up;
        }
        c=t+ch;
        count = path[c];
        gotoAndStop("train source","1");
}

function findscroll2(up, down,ch) {
        if (up>down) {
                t = up-down;
        } else {
                t = down-up;
        }
        c=t+ch;
        count = path[c];
        gotoAndStop("train destination","1");
}

//Organization type/street-next button
on (release) {
if (orgtypetxt=="Default" and stNameTxt=="Default")
```

```
{
        disp="Click any type or street name to continue!!!";
}
else
{
        dirorgtype=orgtypeTxt;
        dirorgstreet=stNameTxt;
        loadVariables("Directorytrans.asp", "", "POST");
        gotoAndStop("Organization Type", 1);
}
}
//Displaying-next Button
on (press) {

        count1=substring(count, 0, (count.length-2));
        if (next==(count1-1))
        {
                next = 0;
        }
        else
        {
                next = next+1;
        }
        discnt=next;
        loadVariables("Pathbookmark.asp", "", "POST");
        }

//SpecificPath-Frame
//via
temp=["Pykara Power House","Shobika","Singapore Plaza","Kannan Departmental
Stores","Zimson Showroom","Lakshmi Complex","Sri Krishna Sweets","Mahaveers","Town Bus
Stand","CAG Pride","Central Bus Stand","Heritage Inn","Arul Jothi Lodge","GP Hospital","Arya
Vaidya Sala","Kongunadu Hospital","Karpagam Complex","10th Street","Extn 9","9th
Street","Extn 7","7th Street"];
i=temp.length;
j=0;


t1=temp[j];
t2=temp[j+1];

displayvia=""


//not via
```

```
tempnotvia=["Pykara Power House","Shobika","Singapore Plaza","Kannan Departmental
Stores","Zimson Showroom","Lakshmi Complex","Sri Krishna Sweets","Mahaveers","Town Bus
Stand","CAG Pride","Central Bus Stand","Heritage Inn","Arul Jothi Lodge","GP Hospital","Arya
Vaidya Sala","Kongunadu Hospital","Karpagam Complex","10th Street","Extn 9","9th
Street","Extn 7","7th Street"];
inotvia=temp.length;
jnotvia=0;


t1notvia=temp[jnotvia];
t2notvia=temp[jnotvia+1];

displaynotvia=""

stop();

//Specific Path-Navigate
on (press) {
place=["Pykara Power House","Shobika","Singapore Plaza","Kannan Departmental
Stores","Zimson Showroom","Lakshmi Complex","Sri Krishna Sweets","Mahaveers","Town Bus
Stand","CAG Pride","Central Bus Stand","Heritage Inn","Arul Jothi Lodge","GP Hospital","Arya
Vaidya Sala","Kongunadu Hospital","Karpagam Complex","10th Street","Extn 9","9th
Street","Extn 7","7th Street"];
	if (displayvia == displaynotvia) {
		warnspec = "Enter proper via and not via place";
	} else {
		for (i=0; i<length(place); i++) {
			if (displayvia == place[i]) {
				via = i;
				dvia = place[i];
			}
		}
		for (i=0; i<length(place); i++) {
			if (displaynotvia == place[i]) {
				notvia = i;
				dnotvia = place[i];
			}
		}
		gotoAndStop("output scene", 1);
	}
}
```

//SAMPLE ASP CODING

//Pathtrans.asp

```
<%
db1 = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ="& Server.MapPath("CityNavigator.mdb") &";DefaultDir="& Server.MapPath(".") &";DriverId=25;FIL=MS Access;MaxBufferSize=512;PageTimeout=5"
Dim DBConn, strSECTION
Set DBConn = Server.CreateObject("ADODB.Connection")
DBConn.Open db1
Set CmdA = Server.CreateObject("ADODB.Recordset")
CmdA.Open "PathTrans", DBConn,1, 3

                CmdA.AddNew
                CmdA("Source") = replace(request("dsource"),"&","and")
                CmdA("Destination") = replace(request("ddestination"),"&","and")
                CmdA("Pathtype") = replace(request("dpath"),"&","and")
                CmdA("DateofTrans") =date
                CmdA("Via") = replace(request("dvia"),"&","and")
                CmdA("Notvia") = replace(request("dnotvia"),"&","and")
                CmdA.Update
                %>succes=1<%
CmdA.Close
Set CmdA = nothing
DBConn.Close
Set DBConn = nothing
%>
```

//Pathbookmark.asp

```
<%
dim a,b,c,nextasp
db1 = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ="&
Server.MapPath("CityNavigator.mdb") &";DefaultDir="& Server.MapPath(".")
&";DriverId=25;FIL=MS Access;MaxBufferSize=512;PageTimeout=5"
Dim DBConn, strSECTION
Set DBConn = Server.CreateObject("ADODB.Connection")
DBConn.Open db1
Set CmdA = Server.CreateObject("ADODB.Recordset")
Set CmdB = Server.CreateObject("ADODB.Recordset")
Set CmdC = Server.CreateObject("ADODB.Recordset")
Set CmdD = Server.CreateObject("ADODB.Recordset")
Set CmdE = Server.CreateObject("ADODB.Recordset")

a=date
b=date-1
c=date-2

if request("today")=1 then
CmdA.Open "select * from PathTrans where DateofTrans='" & a & "' " ,DBConn,1,3
nextasp=request("next")
CmdA.move(nextasp)
%>
&bmdd=<%=a%>
&bmsrc=<%=CmdA("Source")%>
&bmdest=<%=CmdA("Destination")%>
&bmpt=<%=CmdA("Pathtype")%>
&bmvia=<%=CmdA("Via")%>
&bmnvia=<%=CmdA("Notvia")%>
&count=<%=CmdA.recordCount%>
<%
CmdB.Open "select * from placelist where location='" & CmdA("Source") & "' " ,DBConn,1,3
CmdC.Open "select * from placelist where location='" & CmdA("Destination") & "' "
,DBConn,1,3
CmdD.Open "select * from placelist where location='" & CmdA("Via") & "' " ,DBConn,1,3
CmdE.Open "select * from placelist where location='" & CmdA("Notvia") & "' " ,DBConn,1,3
%>
&bmsh=<%=CmdB("node")%>
&bmdh=<%=CmdC("node")%>
&bmvh=<%=CmdD("node")%>
&bmnvh=<%=CmdE("node")%>

<%
else
if request("yesterday")=1 then
CmdA.Open "select * from PathTrans where DateofTrans='" & b & "' " ,DBConn,1,3
nextasp=request("next")
CmdA.move(nextasp)
%>
&bmdd=<%=b%>
&bmsrc=<%=CmdA("Source")%>
&bmdest=<%=CmdA("Destination")%>
&bmpt=<%=CmdA("Pathtype")%>
&bmvia=<%=CmdA("Via")%>
&bmnvia=<%=CmdA("Notvia")%>
<%
CmdB.Open "select * from placelist where location='" & CmdA("Source") & "' " ,DBConn,1,3
CmdC.Open "select * from placelist where location='" & CmdA("Destination") & "' "
,DBConn,1,3
```

```
CmdD.Open "select * from placelist where location='" & CmdA("Via") & "' " ,DBConn,1,3
CmdE.Open "select * from placelist where location='" & CmdA("Notvia") & "' " ,DBConn,1,3
%>
&bmsh=<%=CmdB("node")%>
&bmdh=<%=CmdC("node")%>
&bmvh=<%=CmdD("node")%>
&bmnvh=<%=CmdE("node")%>
&count=<%=CmdA.recordCount%>
<%
else
if request("daybefore")=1 then
CmdA.Open "select * from PathTrans where DateofTrans='" & c & "' " ,DBConn,1,3
nextasp=request("next")
CmdA.move(nextasp)
%>
&bmdd=<%=c%>
&bmsrc=<%=CmdA("Source")%>
&bmdest=<%=CmdA("Destination")%>
&bmpt=<%=CmdA("Pathtype")%>
&bmvia=<%=CmdA("Via")%>
&bmnvia=<%=CmdA("Notvia")%>
&count=<%=CmdA.recordCount%>
<%
CmdB.Open "select * from placelist where location='" & CmdA("Source") & "' " ,DBConn,1,3
CmdC.Open "select * from placelist where location='" & CmdA("Destination") & "' "
,DBConn,1,3
CmdD.Open "select * from placelist where location='" & CmdA("Via") & "' " ,DBConn,1,3
CmdE.Open "select * from placelist where location='" & CmdA("Notvia") & "' " ,DBConn,1,3
%>
&bmsh=<%=CmdB("node")%>
&bmdh=<%=CmdC("node")%>
&bmvh=<%=CmdD("node")%>
&bmnvh=<%=CmdE("node")%>
<%
end if
end if
end if
%>

<%
CmdA.Close
Set CmdA = nothing
DBConn.Close
Set DBConn = nothing
%>
```

```
//TrainDest.asp

<%
Dim acc
dim conn
dim rs
dim strsql
dim sql
dim ts,td,ta,z,tp,tt,ty
Dim arry(100),arry2(100),arry3(100),arry4(100),arry5(100),arry6(100)
dim i,j,k
'to get the variables from flash thro textbox
acc=request.form("count")
set conn=server.CreateObject("ADODB.Connection")
set rs=server.CreateObject("ADODB.Recordset")
strconn = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" &
Server.MapPath("TrainTimings.mdb")
conn.Open strconn

rs.open "select * from train where destination='"& request.form("count") & "'",conn,1,2

z=rs.recordCount
%>
&checktrain=<%=z%>
<%
ts=""
td=""
ta=""
tp=""
tt=""
ty=""
i=0
j=0
k=0
%>
<% if acc<>"Coimbatore" then%>

&train_det=Details of the trains reaching +<%=acc%>+ and passing through Coimbatore
<%else%>
&train_det=Details of the trains reaching +<%=acc%>
<%end if%>
<% if z=0 then %>
<%
'&TrainSrc=No source
```

```
'&TrainDest=No destination
'&TrainArrival=Nil
%>

<% else
        ' to store the records in an array
rs.Movefirst
Do while i<z
arry(i)=rs("TrainName")
arry2(i)=rs("source")
arry3(i)=rs("arrival")
arry4(i)=rs("days")
arry5(i)=rs("departure")
arry6(i)=rs("ExpPass")
i=i+1
rs.Movenext
loop
i=0
k=0
%>

<%    'loop to save the strings in the array arr2 in a single string
i=0
Do while i<z
ts=ts+arry(i)+vbCr
td=td+arry2(i)+vbCr
ta=ta+arry3(i)+vbCr
tt=tt+arry5(i)+vbCr
tp=tp+arry4(i)+vbCr
ty=ty+arry6(i)+vbCr
i=i+1
loop
%>

&TrainName=<%=ts%>
&TrainSrc=<%=td%>
&TrainArrival=<%=ta%>
&TrainDay=<%=tp%>
&TrainDep=<%=tt%>
&TrainType=<%=ty%>
<% end if %>


//orgTypeResultNext.asp

<%
```

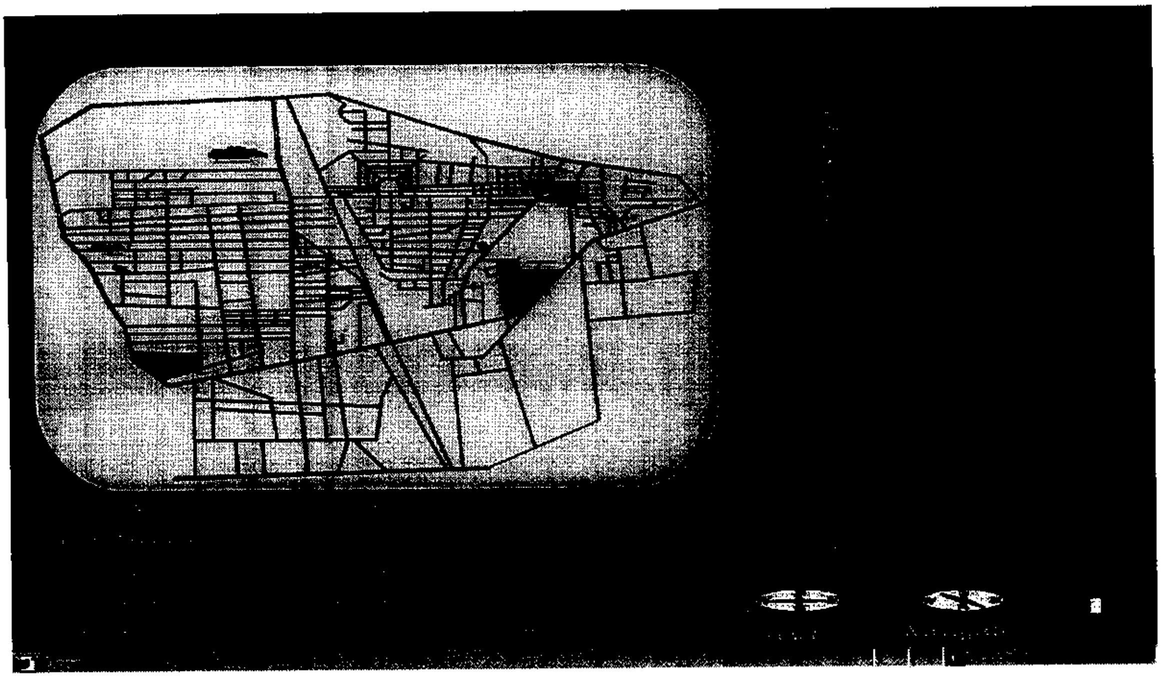
```
db1 = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ="&
Server.MapPath("Directory.mdb") &";DefaultDir="& Server.MapPath(".")
&";DriverId=25;FIL=MS Access;MaxBufferSize=512;PageTimeout=5"
Dim DBConn, strSECTION
Dim i,cnt,cnt1,col,acc,acc1,nc1,stn
Dim arr(1000),arr2(1000),sss
Set DBConn = Server.CreateObject("ADODB.Connection")
DBConn.Open db1
Set CmdA = Server.CreateObject("ADODB.Recordset")
Set CmdA1 = Server.CreateObject("ADODB.Recordset")
col=""
acc=request.form("orgtypetxt")
stn=request.form("stNameTxt")

%>

&org_det=Details of the organization type <%=request.form("orgtypetxt")%>
<%
if (acc<>"Default") then
        CmdA.open "select * from TypeDetails where TypeName='" & acc & "'",DBconn,1,2
        typcode=CmdA("TypeCode")
cnt1=CmdA.recordCount
if cnt1<>0 then
        acc1=CmdA("TypeCode")
else%>
        &org_det=No Records
<%
end if
CmdA.Close
end if
 %>

<%
if (acc<>"Default" and stn="Default") then

   if cnt1<>0 then
    CmdA.open "select * from OrganizationDetails where TypeCode='"& acc1 &"'",DBconn,1,2
   end if
 elseif (acc<>"Default" and stn<>"Default") then
  if cnt1<>0 then
    CmdA.open "select * from OrganizationDetails where TypeCode='"& acc1 &"' and
StreetName='" & stn & "'",DBconn,1,2
   end if
 elseif (acc="Default" and stn<>"Default") then
         CmdA.open "select * from OrganizationDetails where StreetName='" & stn &
"'",DBconn,1,2
 end if
```

```
cnt=CmdA.recordCount
%>
&prevcnt=<%=cnt%>
<%nc1=request("next")
CmdA.Move(nc1)
%>

&NoCo=<%=CmdA("NumberAndComplex")%>
&StreetName=<%=CmdA("StreetName")%>
&pinCode=<%=CmdA("PinCode")%>
&Phoneno=<%=CmdA("PhoneNumber")%>
&City=<%=CmdA("City")%>
&OrgsName=<%=CmdA("Name")%>
&OtherInfo=<%=CmdA("OtherInfo")%>

<% CmdA.Close
Set CmdA = nothing
DBConn.Close
Set DBConn = nothing
%>


//orgName.asp
<%
db1 =  "DRIVER={Microsoft Access Driver (*.mdb)};DBQ="&
Server.MapPath("Directory.mdb") &";DefaultDir="& Server.MapPath(".")
&";DriverId=25;FIL=MS Access;MaxBufferSize=512;PageTimeout=5"
Dim DBConn, strSECTION
Dim i,cnt,col,acc,acc1
Dim arr(1000),arr2(1000),sss
Set DBConn = Server.CreateObject("ADODB.Connection")
DBConn.Open db1
Set CmdA = Server.CreateObject("ADODB.Recordset")
col=""
acc=request.form("orgNametxt")


%>

&org_det=Details of the organization +<%=acc%>
<%
function initrs(col)

sss=""
for j=0 to 1000
        arr(j)=""
        arr2(j)=""
```

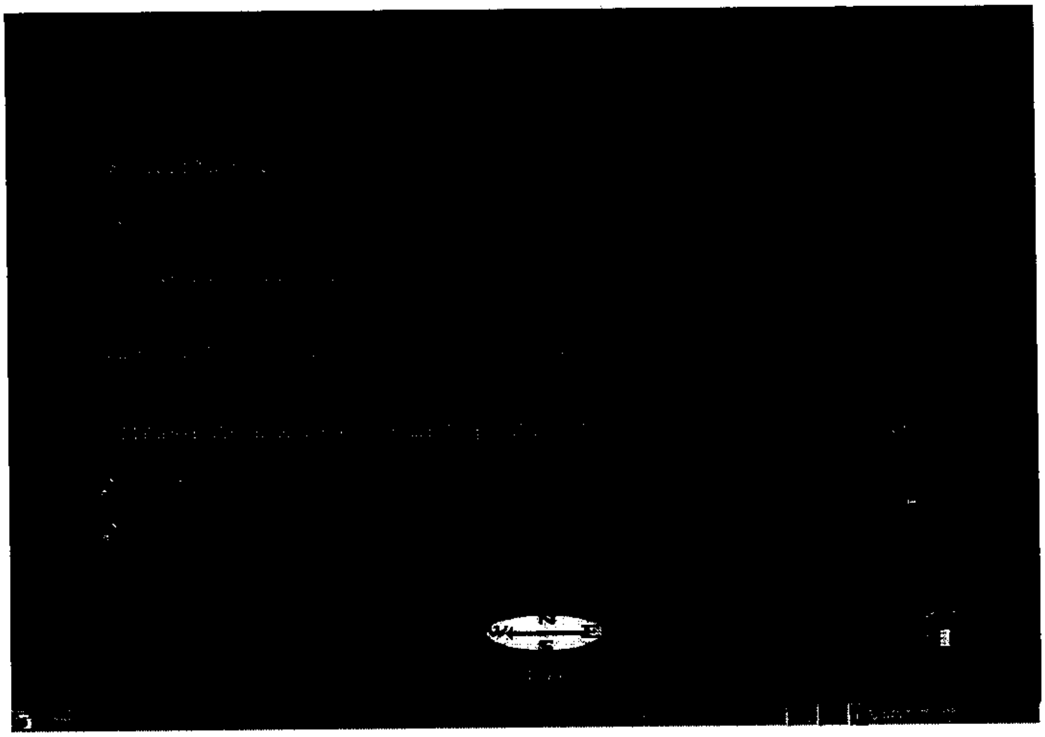
```
next
z=0
i=0
cnt=CmdA.recordCount
if cnt<>0 then
        ' loop to store sources into an array arr
         CmdA.Movefirst
         Do while i<cnt
         arr(i)=CmdA(col)
         i=i+1
         CmdA.Movenext
         loop
         i=0
         k=0
         z=0

end if
      'loop to get the unique values from th array arr to arr2

Do while i<=cnt
         for j=0 to 1000
                  if arr(i)=arr2(j) then
                           k=k+1
                  end if
         next
         if k=0 then
                  arr2(z)=arr(i)
                  z=z+1
         end if
         i=i+1
         k=0
loop

       'loop to save the strings in the array arr2 in a single string

i=0
if cnt<>0 then

         Do while i<=cnt
                  sss=sss+arr(i)+vbCr+vbCr
                  initrs = sss
                  i=i+1
         loop

else

         sss="No Records"
        initrs=sss

end if
end function
%>
<%
CmdA.open "select * from OrganizationDetails where Name='"& acc &"'",DBconn,1,2
%>
&NoCo=<%=initrs("NumberAndComplex")%>
&StreetName=<%=initrs("StreetName")%>
&pinCode=<%=initrs("PinCode")%>
&Phoneno=<%=initrs("PhoneNumber")%>
&City=<%=initrs("City")%>
&OtherInfo=<%=initrs("OtherInfo")%>
<% CmdA.Close%>
<%
CmdA.open "select * from OrganizationDetails where Name='"& acc &"'",DBconn,1,2
acc1=CmdA("TypeCode")
%>
<% CmdA.Close%>
<%
CmdA.open "select * from TypeDetails where TypeCode='"& acc1 &"'",DBconn,1,2
%>
&OrgType=<%=initrs("TypeName")%>

<% CmdA.Close
Set CmdA = nothing
DBConn.Close
Set DBConn = nothing
%>
```

APPENDIX A- SAMPLE SCREEN SHOTS

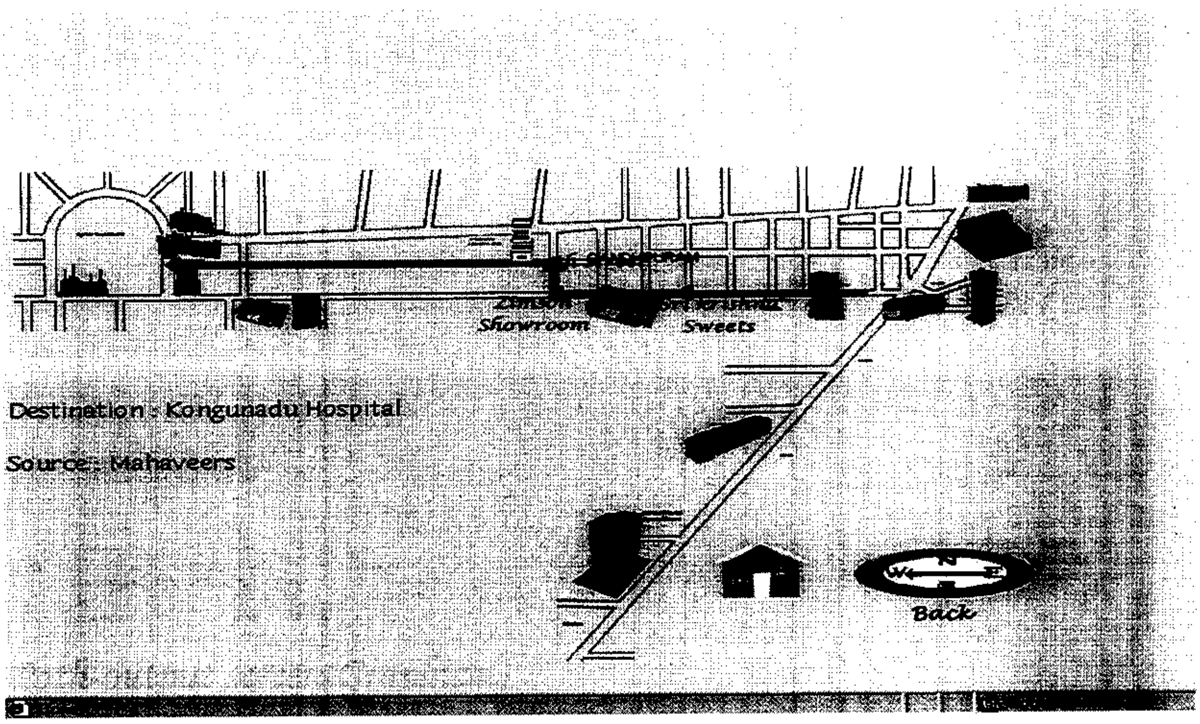
Path Module-Front Screen



Path-output screen(Textual)



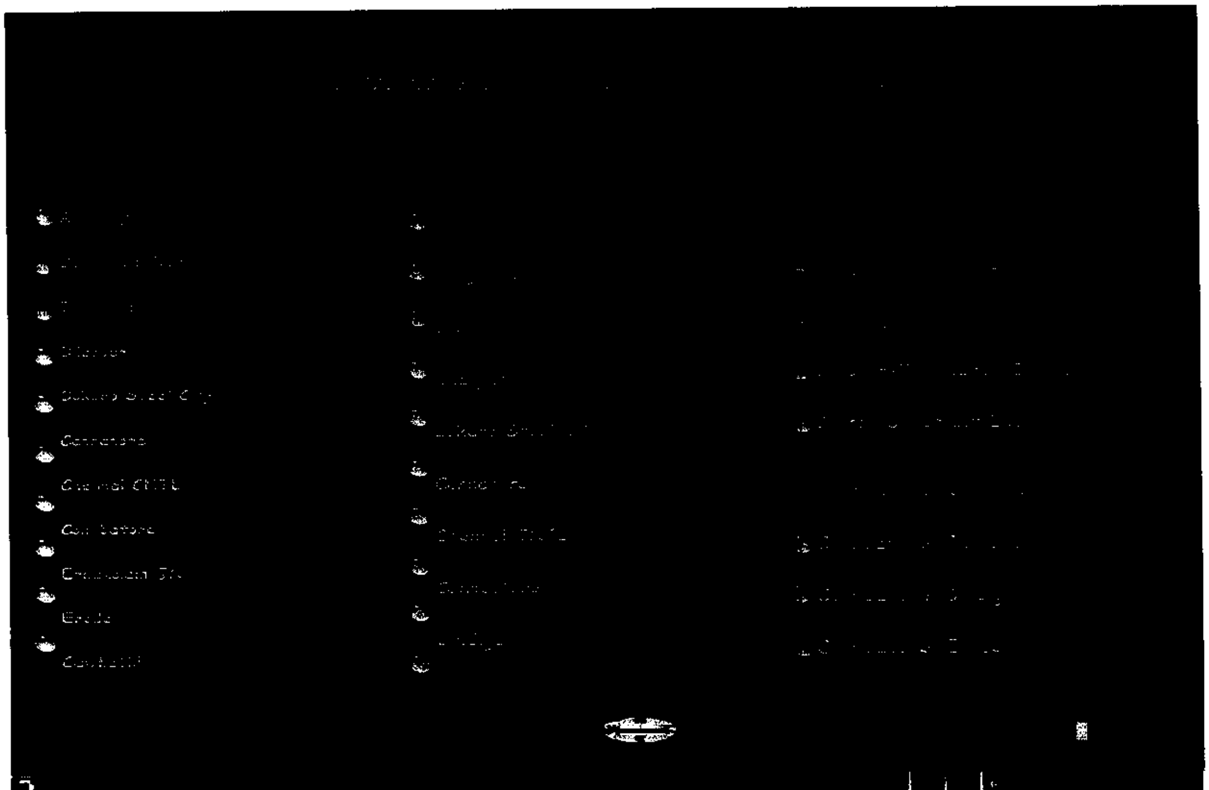
Path-output screen(Graphical)



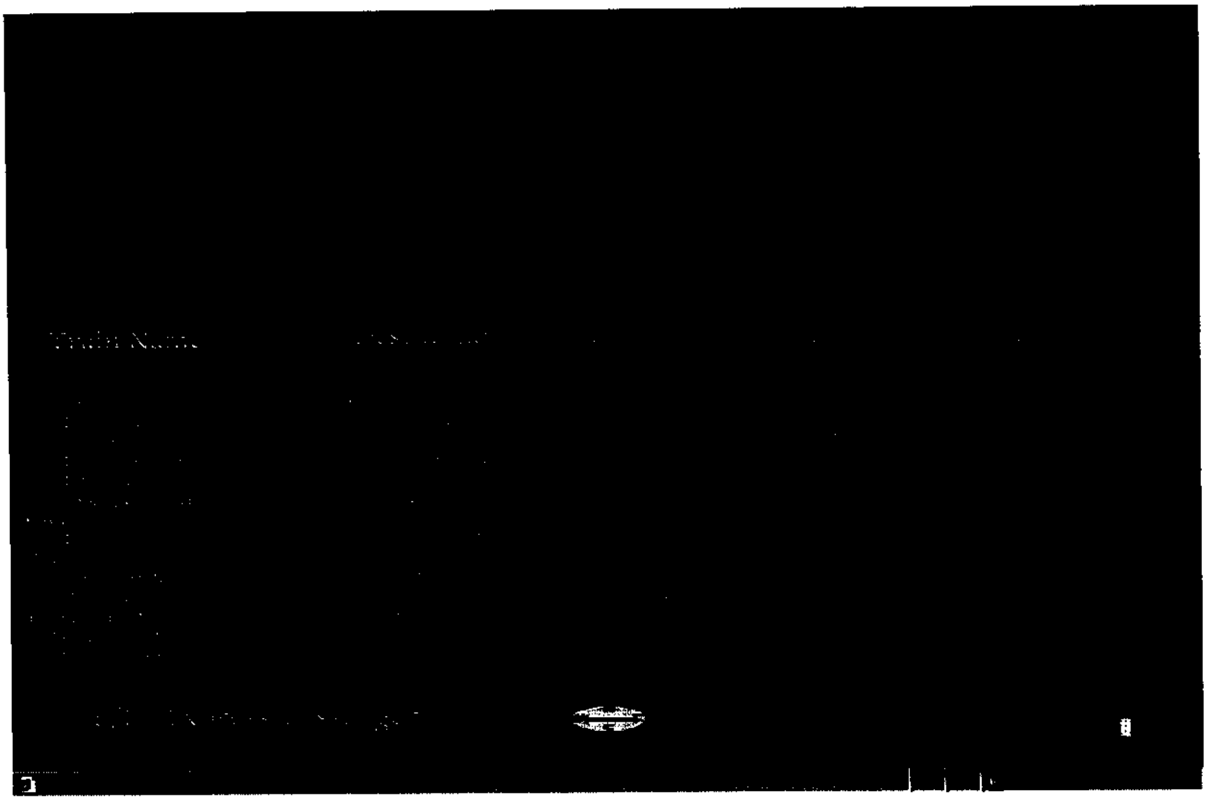
Destination : Kongunadu Hospital

Source : Mahaveers

Train Timing screen

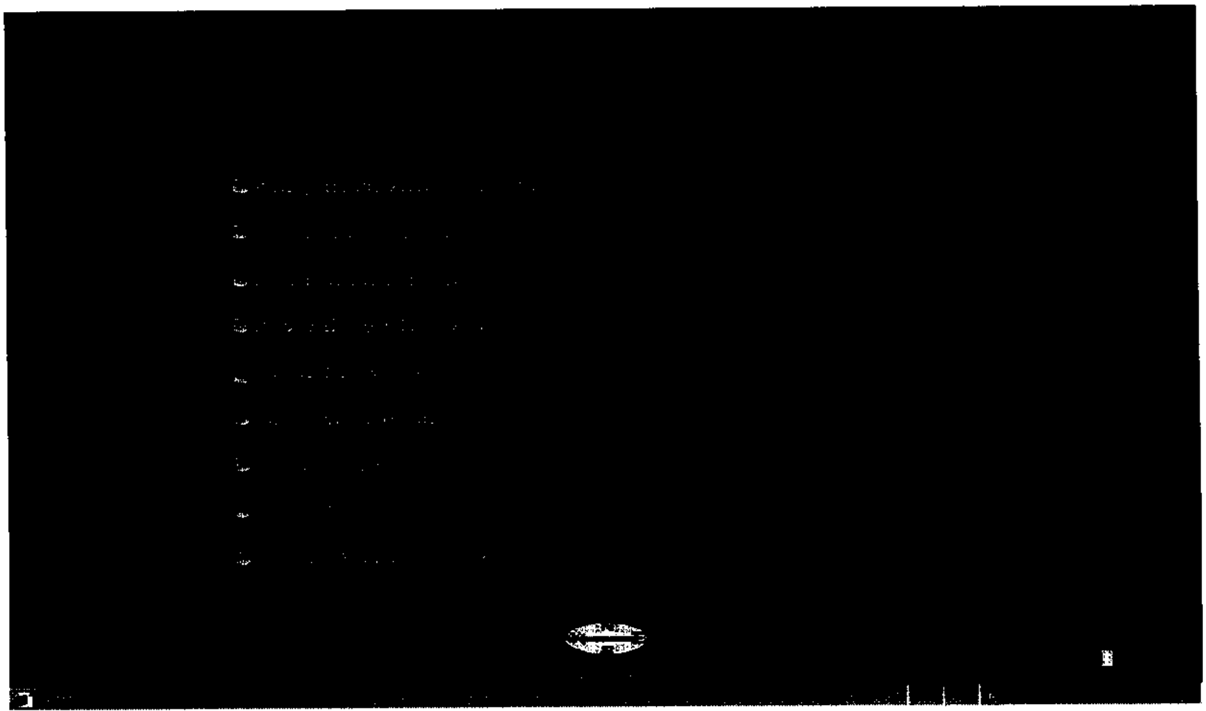


Train Timing output screen(After Clicking a source)

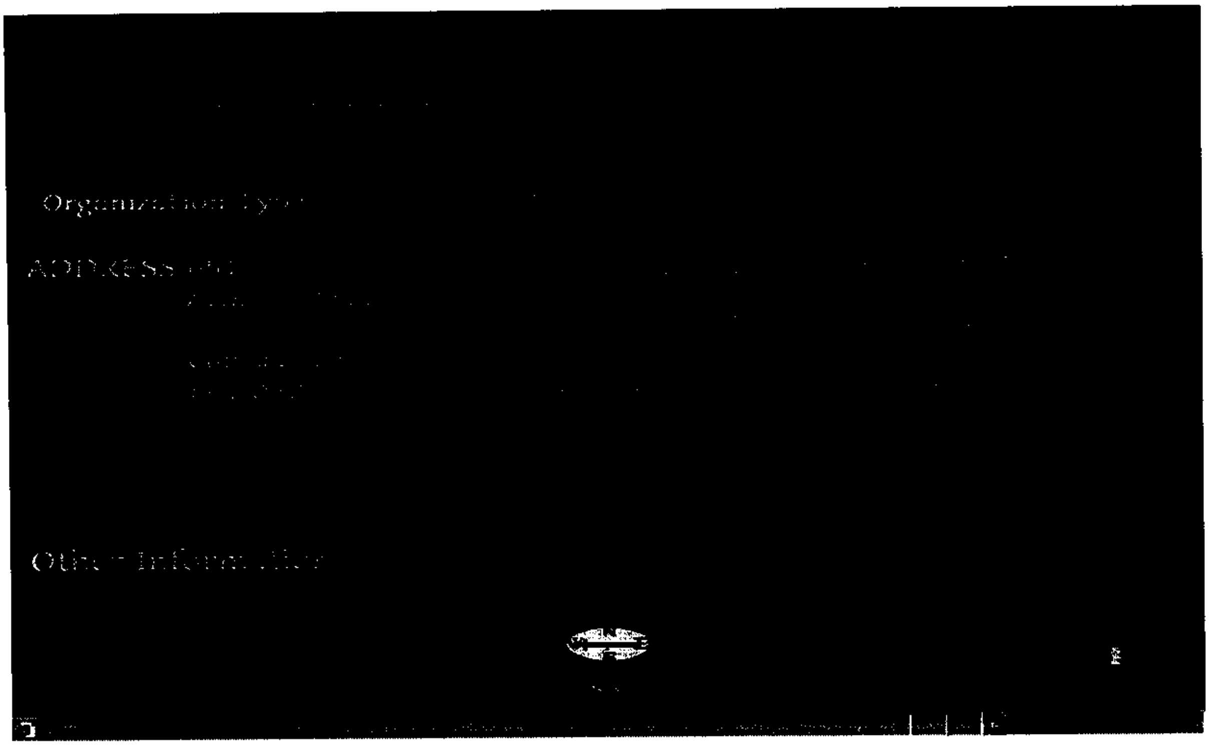


Train Name

Directory-Search by Organization name



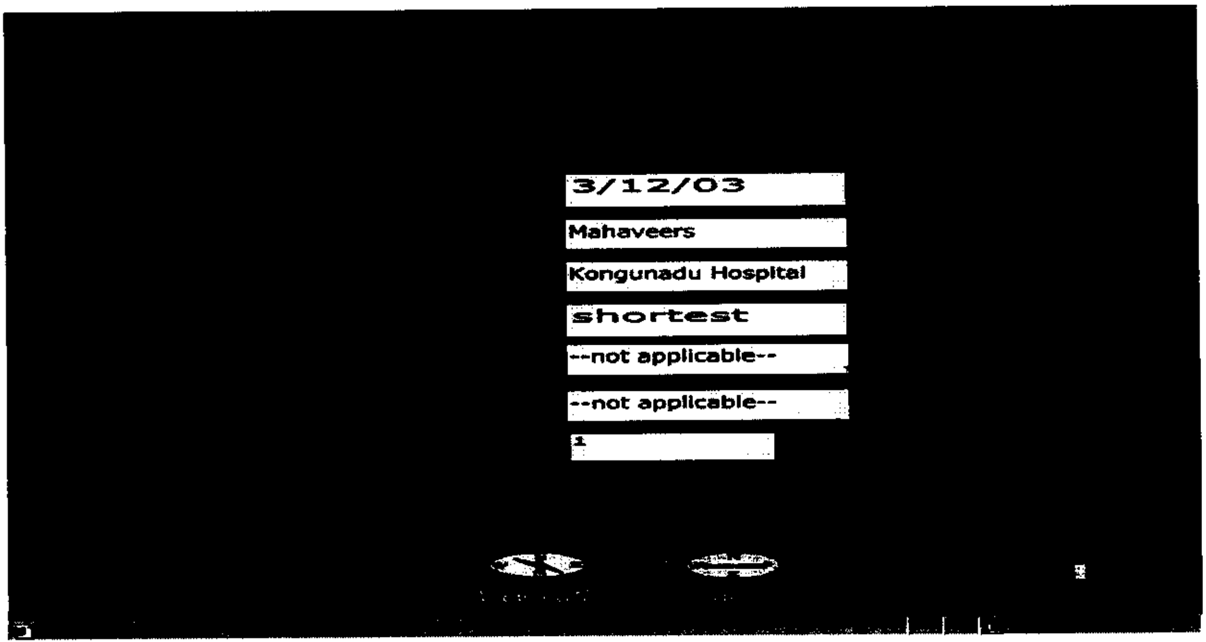
Directory-Output Screen(Search by Organization name)


Organization Type

ADDRESS etc

Other Information

Bookmark-Path (Today)


3/12/03

Mahaveers

Kongunadu Hospital

shortest

--not applicable--

--not applicable--

1