JSS MAHAVIDYAPEETHA

Mini Project / Internship Assessment

Subject Name: Mini project / Internship Assessment
Subject Code : KCS-354

COURSE: B.Tech
SEMESTER: IIIrd

Project Name: ROUTE FINDER

by

Priya Raj (22DLAIML001)

Department of Computer Science and Engineering
JSS ACADEMY OF TECHNICAL EDUCATION
C-20/1, SECTOR-62, NOIDA

VISION AND MISSION

VISION OF THE INSTITUTE

JSS Academy of Technical Education Noida aims to become an Institution of excellence in imparting quality Outcome Based Education that empowers the young generation with Knowledge, Skills, Research, Aptitude and Ethical values to solve Contemporary Challenging Problems.

MISSION OF THE INSTITUTE

Develop a platform for achieving globally acceptable level of intellectual acumen and technological competence
Create an inspiring ambience that raises the motivation level for conducting quality research
Provide an environment for acquiring ethical values and positive attitude.

VISION OF THE DEPARTMENT

"To spark the imagination of the Computer Science Engineers with values, skills and creativity to solve the real-world problems."

MISSION OF THE DEPARTMENT

To inculcate creative thinking and problem-solving skills through effective teaching, learning and research.
To empower professionals with core competency in the field of Computer Science and Engineering.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM EDUCATIONAL OUTCOMES (PEOs)

PEO1: To apply computational skills necessary to analyze, formulate and solve engineering problems.

PEO2: To establish as entrepreneurs, and work in interdisciplinary research and development organizations as an individual or in a team.

PEO3: To inculcate ethical values and leadership qualities in students to have a successful career.

PEO4: To develop analytical thinking that helps them to comprehend and solve real-world problems and inherit the attitude of lifelong learning for pursuing higher education.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Acquiring in depth knowledge of theoretical foundations and issues in Computer Science to induce learning abilities for developing computational skills.

PSO2: Ability to analyse, design, develop, test and manage complex software system and applications using advanced tools and techniques.

COURSE OUTCOMES (COs)

C224.1	Undertake problem identification, formulation and design a solution
C224.2	Solve the real-world problems effectively and adapt with real life working environment.
C224.3	Acquire skills and knowledge on latest tools and technologies
C224.4	Develop effective communication skills for presentation of project related activities
C224.5	Effectively communicate solution to problems through technical reports

CO-PO-PSO MAPPING

	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO 1**	**PSO 2**
C224.1	3	3	3	3	2	3	3	3	3	3	2	3	3	3
C224.2	3	3	3	3	3	3	3	3	3	2	3	3	3	3
C224.3	2	2	3	3	3	2	3	3	3	1	2	3	3	3
C224.4	2	2	2	2	2	2	2	2	2	3	2	3	2	2
C224.5	2	2	2	2	2	2	2	2	2	3	2	3	2	2
C224	2.40	2.40	2.60	2.60	2.40	2.40	2.60	2.60	2.60	2.40	2.20	3.00	2.60	2.60

CERTIFICATE

*This is to certify that Mini Project/Internship Assessment Report entitled "**ROUTE FINDER**" which is submitted by Priya Raj in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science and Engineering of Dr. APJ Abdul Kalam Technical University, Uttar Pradesh, Lucknow is a record of the candidate's own work carried out by her under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.*

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:
Name: Priya Raj
Roll No.: 22DLAIML001
Date: 13/03/2023

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to teachers who gave me the golden opportunity to do this wonderful project on "ROUTE FINDER" which also helped me in doing a lot of Research & I came to know about so many things I am really thankful to them.

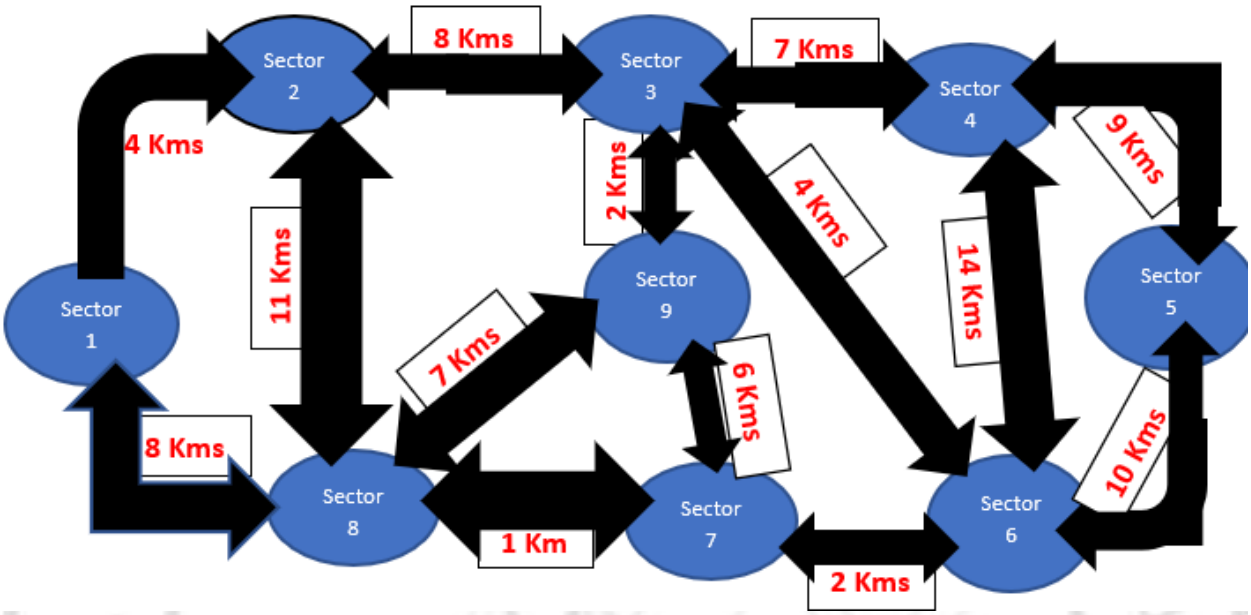
Signature:
Name: Priya Raj
Roll No.: 22DLAIML001
Date: 13/03/2023

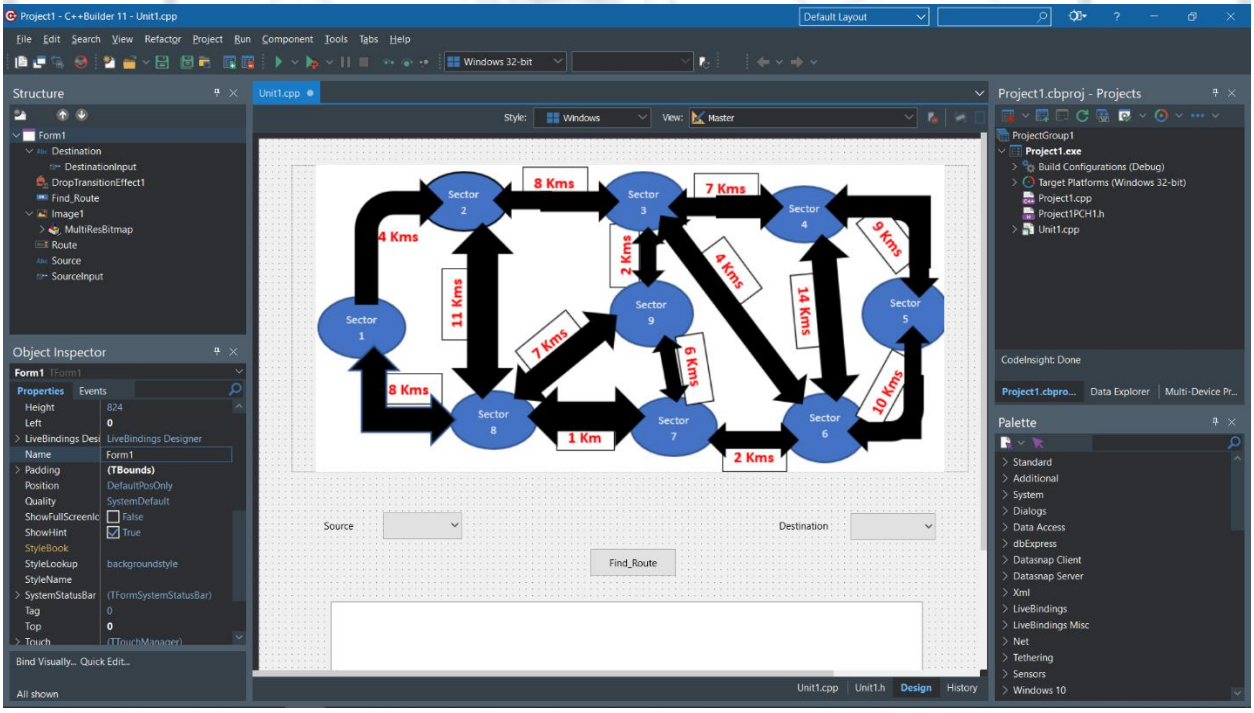
TABLE OF CONTENTS

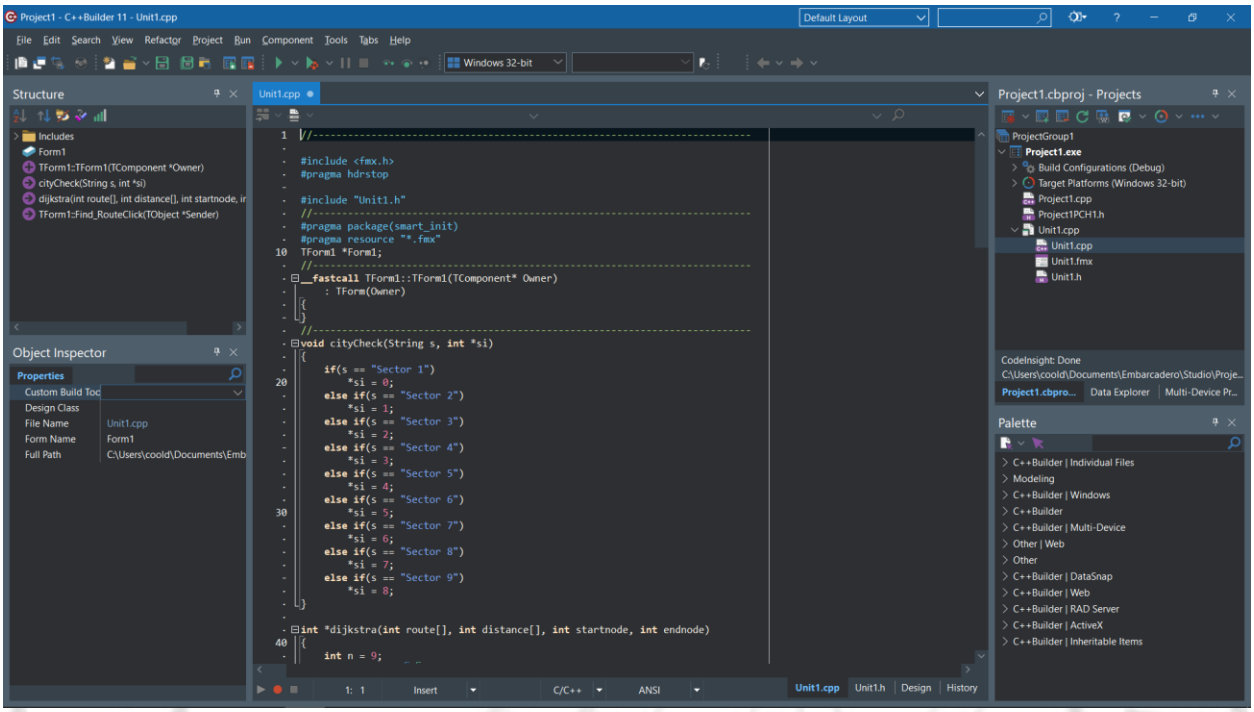
List of tables:

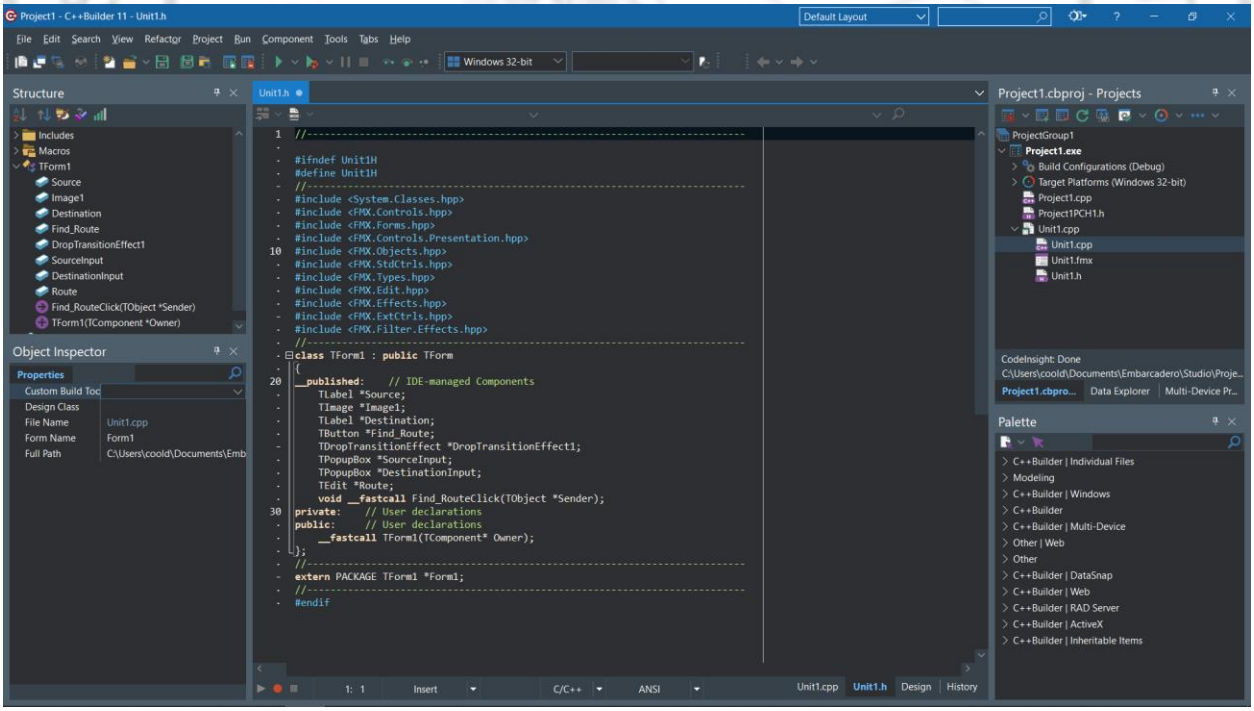
Nodes	*Connected Nodes*
Sector-1	Sector-2, Sector-8
Sector-2	Sector-3, Sector-8
Sector-3	Sector-2, Sector-4, Sector-6, Sector-9
Sector-4	Sector-3, Sector-5, Sector-6
Sector-5	Sector-4, Sector-6
Sector-6	Sector-3, Sector-4, Sector-5, Sector-7
Sector-7	Sector-6, Sector-8, Sector-9
Sector-8	Sector-1, Sector-2, Sector-7, Sector-9
Sector-9	Sector-3, Sector-7, Sector-8

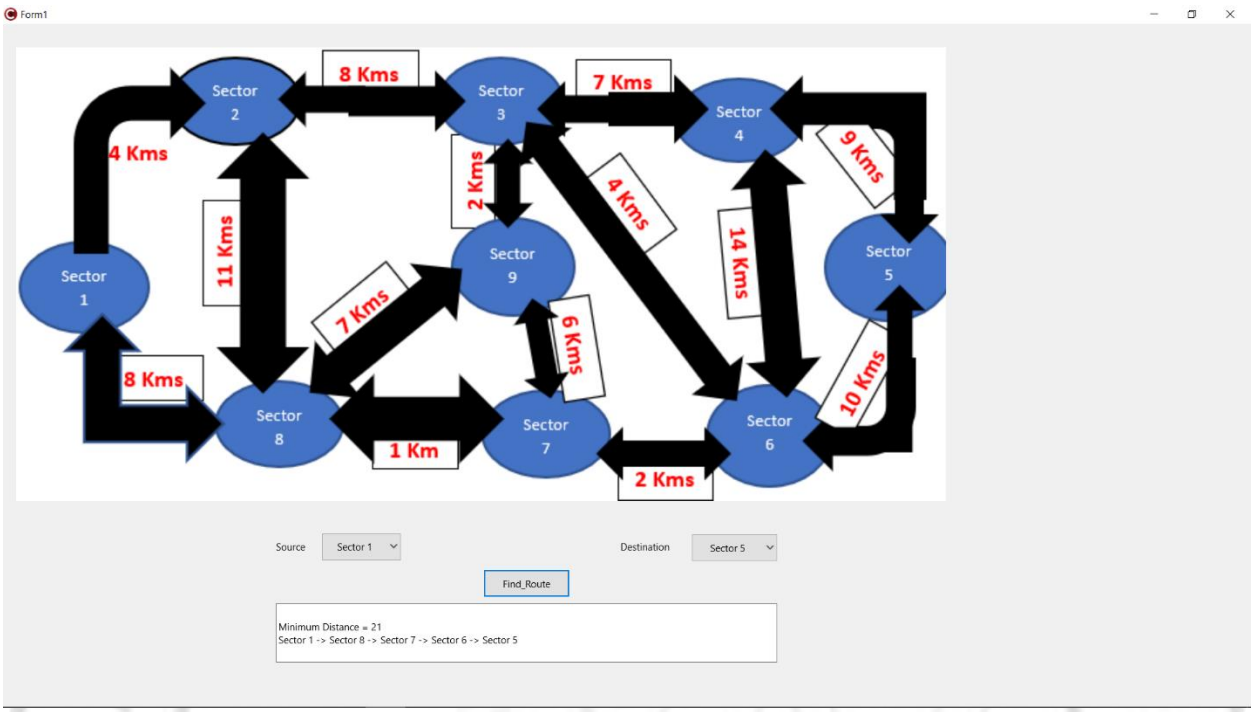
List of figures:






Project1 - C++Builder 11 - Unit1.cpp
File Edit Search View Refactor Project Run Component Tools Tabs Help
Structure
Includes
Form1
TForm1::TForm1(TComponent *Owner)
cityCheck(String s, int *si)
dijkstra(int route[], int distance[], int startnode, ir
TForm1::Find_RouteClick(TObject *Sender)
Object Inspector
Properties
Custom Build Too
Design Class
File Name Unit1.cpp
Form Name Form1
Full Path C:\Users\coold\Documents\Emb
Unit1.cpp
1 //---
#include <fmx.h>
#pragma hdrstop
#include "Unit1.h"
//---
#pragma package(smart_init)
#pragma resource "*.fmx"
TForm1 *Form1;
//---
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
//---
void cityCheck(String s, int *si)
{
if(s == "Sector 1")
*si = 0;
else if(s == "Sector 2")
*si = 1;
else if(s == "Sector 3")
*si = 2;
else if(s == "Sector 4")
*si = 3;
else if(s == "Sector 5")
*si = 4;
else if(s == "Sector 6")
*si = 5;
else if(s == "Sector 7")
*si = 6;
else if(s == "Sector 8")
*si = 7;
else if(s == "Sector 9")
*si = 8;
}
int *dijkstra(int route[], int distance[], int startnode, int endnode)
{
int n = 9;
1: 1 Insert C/C++ ANSI
Unit1.cpp Unit1.h Design History
Project1.cbproj - Projects
ProjectGroup1
Project1.exe
Build Configurations (Debug)
Target Platforms (Windows 32-bit)
Project1.cpp
Project1PCH1.h
Unit1.cpp
Unit1.cpp
Unit1.fmx
Unit1.h
CodeInsight: Done
C:\Users\coold\Documents\Embarcadero\Studio\Proje...
Project1.cbpro... Data Explorer Multi-Device Pr...
Palette
C++Builder | Individual Files
Modeling
C++Builder | Windows
C++Builder
C++Builder | Multi-Device
Other | Web
Other
C++Builder | DataSnap
C++Builder | Web
C++Builder | RAD Server
C++Builder | ActiveX
C++Builder | Inheritable Items


Project1 - C++Builder 11 - Unit1.h
File Edit Search View Refactor Project Run Component Tools Tabs Help
Structure
Includes
Macros
TForm1
Source
Image1
Destination
Find_Route
DropTransitionEffect1
SourceInput
DestinationInput
Route
Find_RouteClick(TObject *Sender)
TForm1(TComponent *Owner)
Object Inspector
Properties
Custom Build Too
Design Class
File Name Unit1.cpp
Form Name Form1
Full Path C:\Users\coold\Documents\Emb
Unit1.h
1 //---
#ifndef Unit1H
#define Unit1H
//---
#include <System.Classes.hpp>
#include <FMX.Controls.hpp>
#include <FMX.Forms.hpp>
#include <FMX.Controls.Presentation.hpp>
#include <FMX.Objects.hpp>
#include <FMX.StdCtrls.hpp>
#include <FMX.Types.hpp>
#include <FMX.Edit.hpp>
#include <FMX.Effects.hpp>
#include <FMX.ExtCtrls.hpp>
#include <FMX.Filter.Effects.hpp>
//---
class TForm1 : public TForm
{
__published: // IDE-managed Components
TLabel *Source;
TImage *Image1;
TLabel *Destination;
TButton *Find_Route;
TDropTransitionEffect *DropTransitionEffect1;
TPopupBox *SourceInput;
TPopupBox *DestinationInput;
TEdit *Route;
void __fastcall Find_RouteClick(TObject *Sender);
private: // User declarations
public: // User declarations
__fastcall TForm1(TComponent* Owner);
};
//---
extern PACKAGE TForm1 *Form1;
//---
#endif
1: 1 Insert C/C++ ANSI
Unit1.cpp Unit1.h Design History
Project1.cbproj - Projects
ProjectGroup1
Project1.exe
Build Configurations (Debug)
Target Platforms (Windows 32-bit)
Project1.cpp
Project1PCH1.h
Unit1.cpp
Unit1.cpp
Unit1.fmx
Unit1.h
CodeInsight: Done
C:\Users\coold\Documents\Embarcadero\Studio\Proje...
Project1.cbpro... Data Explorer Multi-Device Pr...
Palette
C++Builder | Individual Files
Modeling
C++Builder | Windows
C++Builder
C++Builder | Multi-Device
Other | Web
Other
C++Builder | DataSnap
C++Builder | Web
C++Builder | RAD Server
C++Builder | ActiveX
C++Builder | Inheritable Items


Form1
Sector 2
8 Kms
Sector 3
7 Kms
Sector 4
4 Kms
2 Kms
4 Kms
9 Kms
11 Kms
Sector 9
14 Kms
Sector 1
Sector 5
7 Kms
6 Kms
8 Kms
Sector 8
Sector 7
Sector 6
10 Kms
1 Km
2 Kms
Source
Sector 1
Destination
Sector 5
Find_Route
Minimum Distance = 21
Sector 1 -> Sector 8 -> Sector 7 -> Sector 6 -> Sector 5

Chapter 1: Introduction

1.1: Introduction to the project

- The project is named as “Route Finder”.
- It finds the best optimal path (shortest distance) between two addresses.
- We have taken few addresses and their details by default. Dijkstra algorithm is used to find the minimum distance between two addresses.
- Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a weighted graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger Wybe Dijkstra in 1956 and published three years later.
- We have used programming language C++ for writing our code and RAD (Rapid Application Development) for making the GUI (Graphical User Interface) for our code.
- **C++:**
    - C++ is a cross-platform language that can be used to create high-performance applications.
    - C++ was developed by Bjarne Stroustrup, as an extension to the C language.
    - C++ gives programmers a high level of control over system resources and memory.
    - The language was updated 4 major times in 2011, 2014, 2017, and 2020 to C++11, C++14, C++17, C++20.

      We Use C++ because:
    - C++ is one of the world's most popular programming languages.
    - C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.
    - C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
    - C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
    - C++ is fun and easy to learn!
    - As C++ is close to C, C# and Java, it makes it easy for programmers to switch to C++ or vice versa.

- **Dijkstra Algorithm:**
  - With Dijkstra's Algorithm, you can find the shortest path between nodes in a graph. Particularly, you can find the shortest path from a node (called the "source node") to all other nodes in the graph, producing a shortest-path tree.
  - This algorithm is used in GPS devices to find the shortest path between the current location and the destination. It has broad applications in industry, especially in domains that require modeling networks.
  - Dijkstra's Algorithm basically starts at the node that you choose (the source node) and it analyzes the graph to find the shortest path between that node and all the other nodes in the graph.
  - The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path.
  - Once the algorithm has found the shortest path between the source node and another node, that node is marked as "visited" and added to the path.
  - The process continues until all the nodes in the graph have been added to the path. This way, we have a path that connects the source node to all other nodes following the shortest path possible to reach each node.
  - Dijkstra's Algorithm can only work with graphs that have positive weights. This is because, during the process, the weights of the edges have to be added to find the shortest path.
  - This algorithm was created and published by Dr. Edsger W. Dijkstra, a brilliant Dutch computer scientist and software engineer.
  - In 1959, he published a 3-page article titled "A note on two problems in connexon with graphs" where he explained his new algorithm.

- **RAD (Rapid Application Development) Studio:**
  - RAD Studio is an object-oriented, visual programming environment for rapid application development (RAD).

- Using RAD Studio, you can create highly efficient visual applications with a minimum of manual coding, using either the Delphi or C++ programming languages.
- RAD Studio provides all the tools you need to model applications, design user interfaces, automatically generate and edit code.
- It also gives you tools to compile, debug, and deploy applications
- The tools available in the IDE depend on the version of RAD Studio you are using.

1.2: Objective: The purpose of this project is to make a model that finds the shortest optimal path having least distance between two addresses, thus providing the user to choose the optimal path and save the time.

Chapter 2: Tools & Technology Used

2.1: Software Environment

- Microsoft Windows 7/8/10/11
- RAD Studio

2.2: Hardware Requirements

- Intel Processor 2.0 GHz and above.
- 2 GB RAM or more.
- 10 GB or more Hard Disk Drive or above

Chapter 3: History and features of the technology

3.1: History of C++

- C++ was developed by Bjarne Stroustrup at Bell Laboratories over a period starting in 1979.
- Since C++ is an attempt to add object-oriented features (plus other improvements) to C, earlier it was called as "C with Objects".
- As the language developed, Stroustrup named it as C++ in 1983.
- The name C++ suggests "C incremented".
- C++ was made available outside Bell Laboratories in 1985.
- The first commercial C++ compiler, Cfront, was released in 1985.

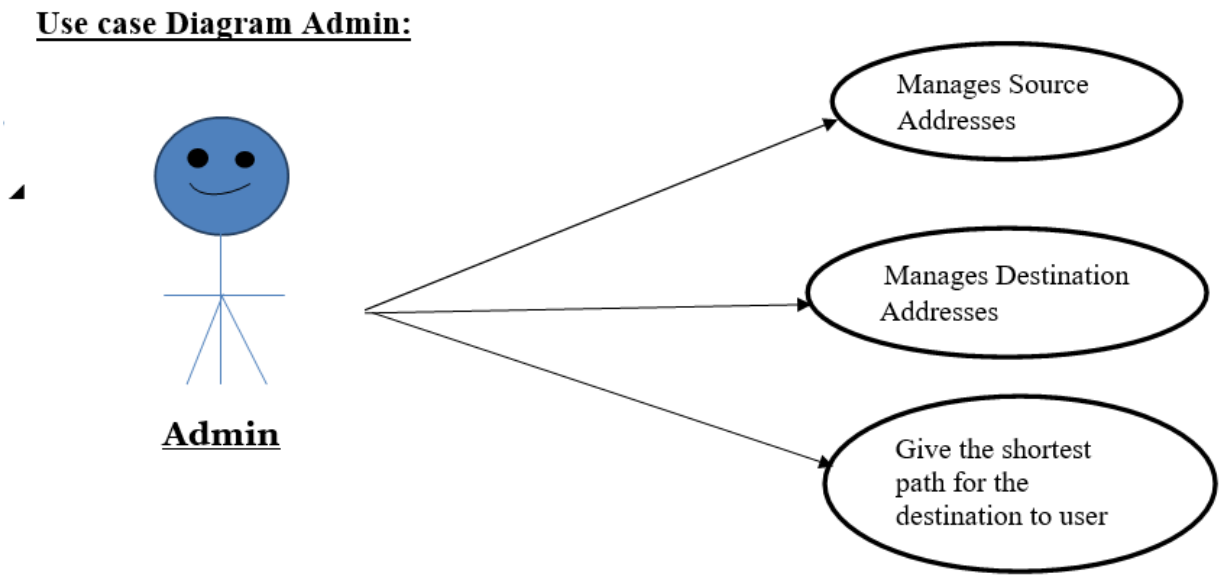
3.2: Features of C++

- Simple
- Abstract Data types
- Machine Independent or Portable
- Mid-level programming language
- Structured programming language
- Rich Library
- Memory Management
- Quicker Compilation
- Pointers
- Recursion
- Extensible
- Object-Oriented

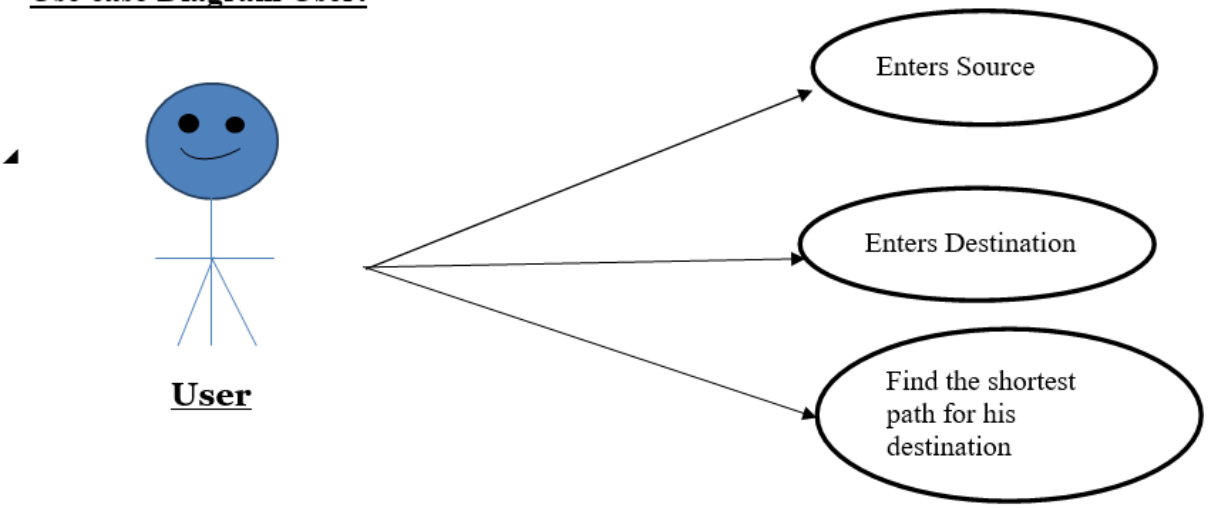
- Compiler based
- Reusability
- National Standards
- Errors are easily detected
- Power and Flexibility
- Strongly typed language
- Redefine Existing Operators
- Modeling Real-World Problems
- Clarity

Chapter 4: Work done

4.1: Use Case Diagram

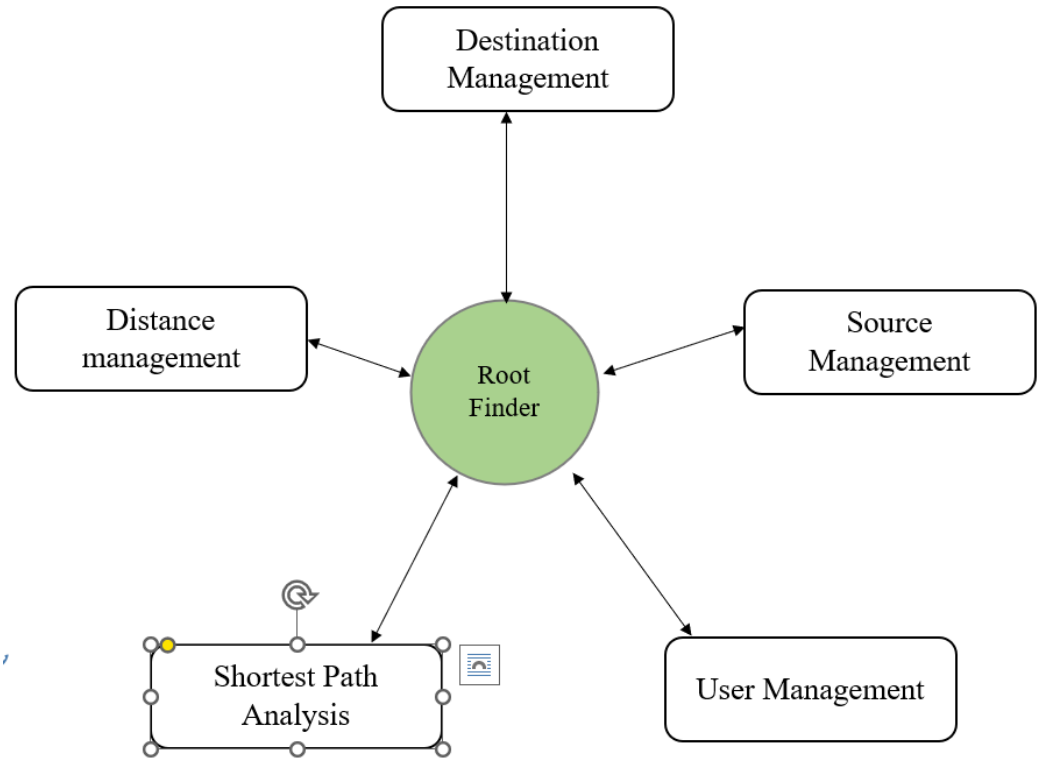


Use case Diagram User:

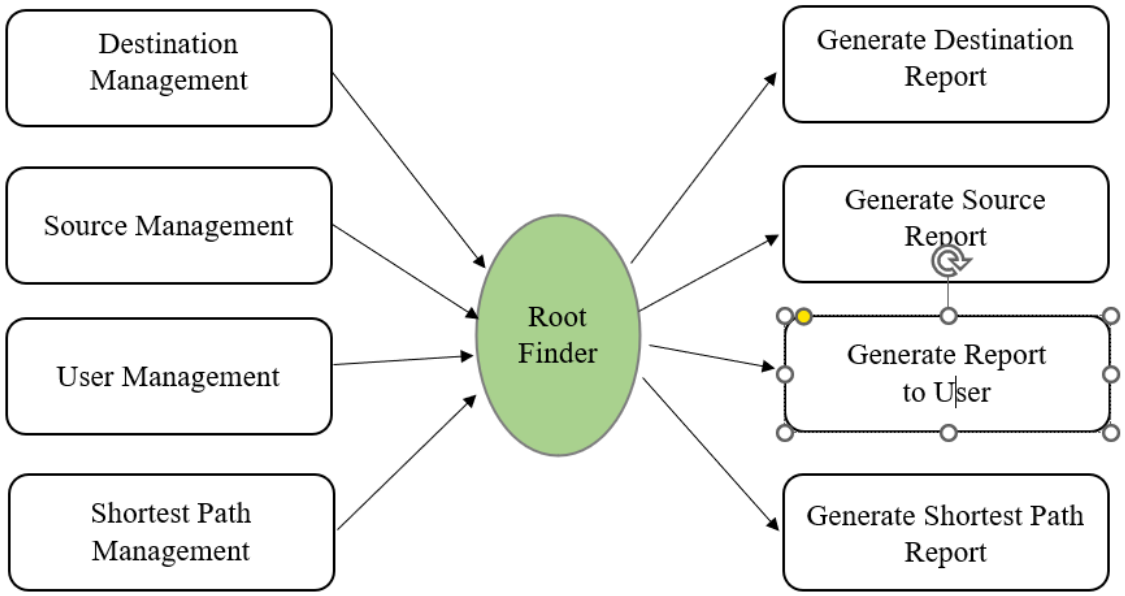


4.1: Data Flow Diagram

DFD Level 0:



DFD Level 1:



4.3: Module Information

dijkstra():

This module is taking start address and end address as parameter and then calculates the best optimal path (shortest) between the start and end address, together with the distance needs to be travelled.

dijkstra() module is a designed to find the shortest paths between nodes in a graph.

```
int *dijkstra(int route[], int distance[], int startnode, int endnode)
{
    int n = 9;
    int G[9][9] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                    { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                    { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                    { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                    { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                    { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                    { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                    { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                    { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };
    int cost[9][9], pred[9];
    int visited[9], count, mindistance, nextnode, i,j;
    for(i=0;i < n;i++)
        for(j=0;j < n;j++)
            if(G[i][j]==0)
                cost[i][j]=9999;
            else
                cost[i][j]=G[i][j];

    for(i=0;i< n;i++)
    {
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
    }
    distance[startnode]=0;
    visited[startnode]=1;
    count=1;
    while(count < n-1){
        mindistance=9999;
        for(i=0;i < n;i++)
            if(distance[i] < mindistance&&!visited[i])
            {
                mindistance=distance[i];
                nextnode=i;
            }
        visited[nextnode]=1;
        for(i=0;i < n;i++)
            if(!visited[i])
                if(mindistance+cost[nextnode][i] < distance[i])
                {
                    distance[i]=mindistance+cost[nextnode][i];
                    pred[i]=nextnode;
                }
            count++;
    }
    if(endnode!=startnode)
    {
        int k = 1;
        route[k] = endnode;
        j=endnode;
        do
        {
            j=pred[j];
            route[++k] = j;
        }
        while(j!=startnode);
        route[0] = k;
    }
    return route;
}
```

Chapter 5: Conclusion & Future Scope

5.1: Conclusion

- This model helped us in finding the best optimal path if someone has to travel from any of the sectors from 1 – 9 to any other sector 1 – 9.
- It can be said as a subset of Google maps.
- We can also implement route finder for any other addresses by just adding their address and details in the model.

5.2: Future Scope

This model is providing the best path between two addresses, thus helping them as they just need to find the route and they get the shortest path to their destination. It can be used in future by increasing its efficiency by adding more number of addresses and their details in the model.

We can further add up the feature of providing the path from the current location to the desired destination of the user by accessing his current location.