***CASHFLOW MINIMIZATION***

***dynamic programming***

*An*

*Assignment Report*

*Submitted in partial fulfilment of the Requirements for the award of the Degree of*

***BACHELOR OF ENGINEERING***

***IN***

**INFORMATION TECHNOLOGY**

By

**G. HARSHITHA<1602-22-737-080>**

**Under the Guidance of**

**SIREESHA PANYAM**



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2023-24**

**CASHFLOW MINIMIZER**

**ABSTRACT**

The **Cashflow Minimizer** project is an interactive web application that generates a random directed graph representing a cashflow network between multiple individuals or entities. The main objective of the project is to minimize the cashflow between these entities by solving a flow problem using the Ford-Fulkerson algorithm.

The application consists of two main parts: the original graph and the solved graph. The original graph is generated randomly with nodes labeled as "Person 1", "Person 2", etc., and edges with random weights representing the cashflow between them. Users can generate a new graph by clicking the "Get New Problem" button.

Once the original graph is generated, users can click the "Solve" button to solve the cashflow minimization problem. The solved graph is then displayed, showing the optimized cashflow between the entities. The solved graph is centered within its container using flexbox properties, ensuring proper alignment and presentation.

The project's CSS code provides a consistent and visually appealing layout for the application. The HTML, CSS, and JavaScript code work together to create an interactive and engaging user experience. The Cashflow Minimizer project is an excellent example of using graph theory and algorithms to solve real-world problems in a user-friendly and accessible way.

**RELEVANCE OF THE PROJECT**

1. Real-world applications: The project addresses a common problem in finance and operations management, where minimizing cashflow between entities is essential for efficient resource allocation and cost reduction.
2. Graph theory and algorithms: The project demonstrates the practical application of graph theory and algorithms, specifically the Ford-Fulkerson algorithm, to solve real-world problems. This showcases the importance of understanding and implementing algorithms to optimize processes and make informed decisions.
3. Interactive and engaging user experience: The project provides an interactive and engaging user experience, allowing users to generate random graphs and visualize the solution to the cashflow minimization problem. This makes the project an excellent educational tool for teaching graph theory and algorithms.
4. Web development skills: The project involves HTML, CSS, and JavaScript, which are essential skills for web development. By creating this project, developers can demonstrate their proficiency in these technologies and showcase their ability to build interactive and visually appealing web applications.
5. Problem-solving and critical thinking: The project encourages problem-solving and critical thinking skills by challenging users to understand the cashflow minimization problem and interpret the results of the Ford-Fulkerson algorithm.

**PROBLEM STATEMENT**

Design and develop a web application that allows users to generate a random directed graph representing a cashflow network, solves the cashflow minimization problem using the Ford-Fulkerson algorithm, and displays the optimized graph with the minimized total cashflow.

**OBJECTIVES OF THE PROJECT**

1. Random Directed Graph Generation: Develop a feature for generating a random directed graph representing a cashflow network. Uses a JavaScript function to generate a random graph based on the number of nodes and edges provided by the user.
2. Ford-Fulkerson Algorithm Implementation: Implement the Ford-Fulkerson algorithm for solving the cashflow minimization problem. Displays the optimized graph with the minimized total cashflow.
3. Optimized Graph Visualization: Display the optimized graph with the minimized total cashflow in a clear and interactive manner.Uses the vis.js library to display the optimized graph.
4. User-Friendly Interface: Provide an engaging and user-friendly interface for generating and visualizing the cashflow network and its optimization. Uses HTML and CSS to create a visually appealing and user-friendly interface.
5. Responsiveness: Ensure that the web application is responsive and accessible across various devices and screen sizes, providing a seamless user experience on desktops, laptops, tablets, and smartphones.
6. Documentation: Provide clear and concise documentation for the web application, including a user manual, technical specifications, and implementation details. Includes screenshots and examples to illustrate the use of the application.

**METHODOLOGY**

The methodology involves researching and planning, designing the algorithm and user interface, implementing the solution in JavaScript, testing thoroughly, documenting comprehensively, deploying, collecting feedback, and iteratively improving the application for ongoing maintenance and support.

**ALGORITHM**

1. **Input:**

* A directed graph G = (V, E) representing the cashflow network, where V is the set of vertices representing the debtors and creditors, and E is the set of edges representing the cashflow between them. Each edge (u, v) has a capacity c(u, v) representing the maximum amount of cash flow that can be transferred from u to v.

1. **Initialize:**

* Set the flow f(u, v) to 0 for all edges (u, v) in E.
* Set the residual graph G\_f = (V, E\_f) to G, where E\_f is the set of edges in G with non-zero capacity.

1. **Find an augmenting path:**

* Find a path P in G\_f from the source vertex s to the sink vertex t such that the capacity of each edge in P is greater than the flow on that edge.
* If no such path exists, go to step 5.

1. **Update the flow:**

* Find the minimum capacity c\_min of the edges in P.
* Add c\_min to the flow on each edge in P that goes from s to t, and subtract c\_min from the flow on each edge in P that goes from t to s.
* Update the residual graph G\_f by removing edges with zero capacity and adding reverse edges with the updated capacity.
* Go back to step 3.

1. **Output:**

* The flow f is the minimum cash flow required to settle all debts.
* Display the optimized graph with the minimized total cashflow.

**IMPLEMENTATION**

**Index.html**

<!DOCTYPE html>

<html>

<head>

    <title>Cashflow Minimizer

    </title>

    <link rel="stylesheet" type="text/css" href="style1.css">

</head>

<body>

    <header>

        <div class="main">

            <div class="logo">

                <img src="cover.png">

            </div>

                <ul>

                    <li class="active"><a href="#">Home</a></li>

                    <li><a href="index3.html">About</a></li>

                </ul>

        </div>

        <div class="title">

            <h2>DAA Project</h2>

            <br>

            <h1>Cashflow Minimizer</h1>

        </div>

        <div class="button">

            <a href="index2.html" class="btn">Solve problems</a>

        </div>

        <div class="footer">

            <p>Made by Harshitha Reddy</p>

        </div>

    </header>

</body>

</html>

**Index2.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Cashflow Minimizer</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"

        integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

    <script src="https://kit.fontawesome.com/5548f5ed00.js" crossorigin="anonymous"></script>

    <script type="text/javascript" src="https://unpkg.com/vis-network/standalone/umd/vis-network.min.js"></script>

    <link rel="stylesheet" href="style.css">

    <script src="script.js"></script>

</head>

<body>

    <nav class="navbar navbar-light " style="font-size: 25px; font-family: sans-serif; background-color: whitesmoke;">

        <class="d-inline-block align-top" alt="">

            Cashflow Minimization 💵

    </nav>

    <div id="container">

        <div id="mynetwork"></div>

        <div id="container2">

            <span id="temptext" style="width: 100%; text-align: center; font-size: x-large">

                Click on solve to get the best solution!!

            </span>

            <div id="mynetwork2" style="display: none"></div>

        </div>

    </div>

    <div>

        <button type="button" class="btn btn-danger" id="generate-graph">Get New Problem</button>

        <button type="button" class="btn btn-success" id="solve">Solve</button>

    </div>

    <div class="footer">

        <p>Made by Harshitha Reddy</p>

    </div>

</body>

</html>

**Index3.html**

<!DOCTYPE html>

<html>

    <head>

        <link rel="stylesheet" href="style3.css">

    </head>

<body>

    <header>

        <div class="logo">

            <img src="cover.png">

        </div>

        <p style="font-size:20px;font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif; margin:60px;">

            Cashflow minimization refers to the act of dividing money between friends, roommates, colleagues, relatives and so on. It allows each party to be reimbursed for the expenses they are owed. To make this process efficient our website has been created to ease this experience. When someone dines with their friends or shares expenses with their roommates or loved ones it is always a challenge to figure out who owes what to whom. It would be stressful to ask their friends to pay back the money they owe them. It would be time-consuming to determine who ordered what and how much they have to pay, incorporating taxes and tip amount in the estimate. And it would seem embarrassing to calculate all this in a restaurant or café where a lot of people are gazing or noticing.

        </p>

        <p style="font-size:20px;font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;margin:60px;">

            Using our website you can track bills and other shared expenses among your friends and settle up stuff in the easiest manner possible with the least cashflow. This is the implementation of classical cashflow minimization using max-heaps made using Javascript.</p>

        <p style="font-size:20px;font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;margin:60px;"> Given a large group of friends who have to give or take some amount of money from one another, we intend to design an algorithm by which the total cash flow among all the friends is minimized. So, a particular friend need not worry about underpaying or overpaying money and pay exactly the net amount that has to be paid. Using this algorithm, we can save a lot of time which would've been spent on calculating it verbally through discussion. </p>

            <div class="footer">

                <p>Made by Harshitha Reddy</p>

            </div>

    </header></body>

</html>

**Style.css**

html,

body {

    height: 100%;

}

#mynetwork {

    width: 50%;

    height: 100%;

    border: 1px solid lightgray;

}

#container2 {

    width: 50%;

    height: 100%;

    border: 1px solid lightgray;

    display: flex;

    flex-wrap: wrap;

    align-content: center;

}

#mynetwork2{

    width: 100%;

    height: 100%;

}

#container{

    width: 100%;

    height: 70%;

    background-color: white;

    display: flex;

    margin: 0 auto;

}

#generate-graph{

    margin:auto;

    display:block;

    margin-top: 18px;

}

#solve{

    margin:auto;

    display:block;

    margin-top: 13px;

}

.footer {

    position: fixed;

    left: 0;

    bottom: 0;

    width: 100%;

    background-color: rgb(90, 88, 88);

    color: white;

    text-align: center;

}

**Style.1css**

\*{

    margin:0;

    padding:0;

    font-family: Century Gothic;

    }

    header{

        background-image:linear-gradient(rgba(0,0,0,0.5),rgba(0,0,0,0.5)), url(./color-money-bg.jpg);

        height:100vh;

        background-size: cover;

        background-position: center;

    }

    ul{

        float:right;

        list-style-type: none;

        margin-top: 25px;

    }

    ul li{

        display:inline-block;

    }

    ul li a{

        text-decoration: none;

        color: #fff;

        padding: 5px 20px;

        border:1px solid transparent ;

        transition: 0.6s ease;

    }

    ul li a:hover{

        background-color: #fff;

        color: #000;

    }

    ul li.active a{

        background-color: #fff;

        color: #000;

    }

    .logo img{

        float: left;

        width: 270px;

        height: auto;

        margin-top: 10px;

    }

    .main{

        max-width: 1200px;

        margin:auto;

    }

    .title{

        position: absolute;

        top: 50%;

        left: 50%;

        transform: translate(-50%,-50%);

    }

    .title h1{

        color: #fff;

        font-size: 70px;

    }

    .title h2{

        color: #fff;

        font-size: 30px;

    }

    .title h3{

        color: #fff;

        font-size: 20px;

    }

    .button{

        position: absolute;

        top: 70%;

        left: 50%;

        transform: translate(-50%,-50%);

    }

    .btn{

        border:1px solid #fff ;

        padding: 10px 30px;

        color:#fff;

        text-decoration: none;

        transition: 0.6s ease;

    }

    .btn:hover{

        background-color: #fff;

        color: #000;

    }

    .footer {

        position: fixed;

        left: 0;

        bottom: 0;

        width: 100%;

        height: 40px;

        background-color: rgb(90, 88, 88);

        color: white;

        text-align: center;

    }

**Style3.css**

\*{

    margin:0;

    padding:0;

    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

.footer {

    position: fixed;

    left: 0;

    bottom: 0;

    width: 100%;

    height: 40px;

    background-color: rgb(90, 88, 88);

    color: white;

    text-align: center;

}

.logo img{

    float:none;

    width: 270px;

    height: auto;

    margin-top: 10px;

    margin-left: 60px;

}

**Script.js**

onload = function () {

    // create the network

    var curr\_data;

    var container = document.getElementById('mynetwork');

    var container2 = document.getElementById('mynetwork2');

    var genNew = document.getElementById('generate-graph');

    var solve = document.getElementById('solve');

    var temptext = document.getElementById('temptext');

    // initialise graph options

    var options = {

        edges: {

            arrows: {

                to: true

            },

            labelHighlightBold: true,

            font: {

                size: 20

            }

        },

        nodes: {

            font: '12px arial red',

            scaling: {

                label: true

            },

            shape: 'icon',

            icon: {

                face: 'FontAwesome',

                code: '\uf183',

                size: 50,  //50,

                color: '#3b916e',

            }

        }

    };

    // initialize the network

    var network = new vis.Network(container);

    network.setOptions(options);

    var network2 = new vis.Network(container2);

    network2.setOptions(options);

    function createData() {

        sz = Math.floor(Math.random() \* 8) + 3;

        nodes = [];

        for (i = 1; i <= sz; i++) {

            nodes.push({ id: i, label: "Person " + i })

        }

        nodes = new vis.DataSet(nodes);

        edges = [];

        for (i = 1; i <= sz; i++) {

            for (j = i + 1; j <= sz; j++) {

                if (Math.random() > 0.5) {

                    if (Math.random() > 0.5)

                        edges.push({ from: i, to: j, label: String(Math.floor(Math.random() \* 100) + 1) });

                    else

                        edges.push({ from: j, to: i, label: String(Math.floor(Math.random() \* 100) + 1) });

                }

            }

        }

        data = {

            nodes: nodes,

            edges: edges

        };

        curr\_data = data;

    }

    genNew.onclick = function () {

        createData();

        network.setData(curr\_data);

        temptext.style.display = "inline";

        container2.style.display = "none";

    };

    solve.onclick = function () {

        temptext.style.display = "none";

        container2.style.display = "inline";

        solvedData = solveData();

        network2.setData(solveData());

    };

    function solveData() {

        data = curr\_data;

        sz = data['nodes'].length;

        vals = Array(sz).fill(0);

        for (i = 0; i < data['edges'].length; i++) {

            edge = data['edges'][i];

            vals[edge['to'] - 1] += parseInt(edge['label']);

            vals[edge['from'] - 1] -= parseInt(edge['label']);

        }

        for (i = 0; i < sz; i++)

            console.log(vals[i]);

        console.log('\n');

        new\_edges = [];

        for (i = 0; i < sz; i++) {

            if (vals[i] > 0) {

                for (j = 0; j < sz && vals[i] > 0; j++) {

                    if (vals[j] < 0) {

                        if (vals[j] + vals[i] >= 0) {

                            new\_edges.push({ from: j + 1, to: i + 1, label: String(Math.abs(vals[j])) });

                            vals[i] += vals[j];

                            vals[j] = 0;

                        } else {

                            new\_edges.push({ from: j + 1, to: i + 1, label: String(vals[i]) });

                            vals[j] += vals[i];

                            vals[i] = 0;

                        }

                    }

                }

            }

        }

        data = {

            nodes: data['nodes'],

            edges: new\_edges

        };

        return data;

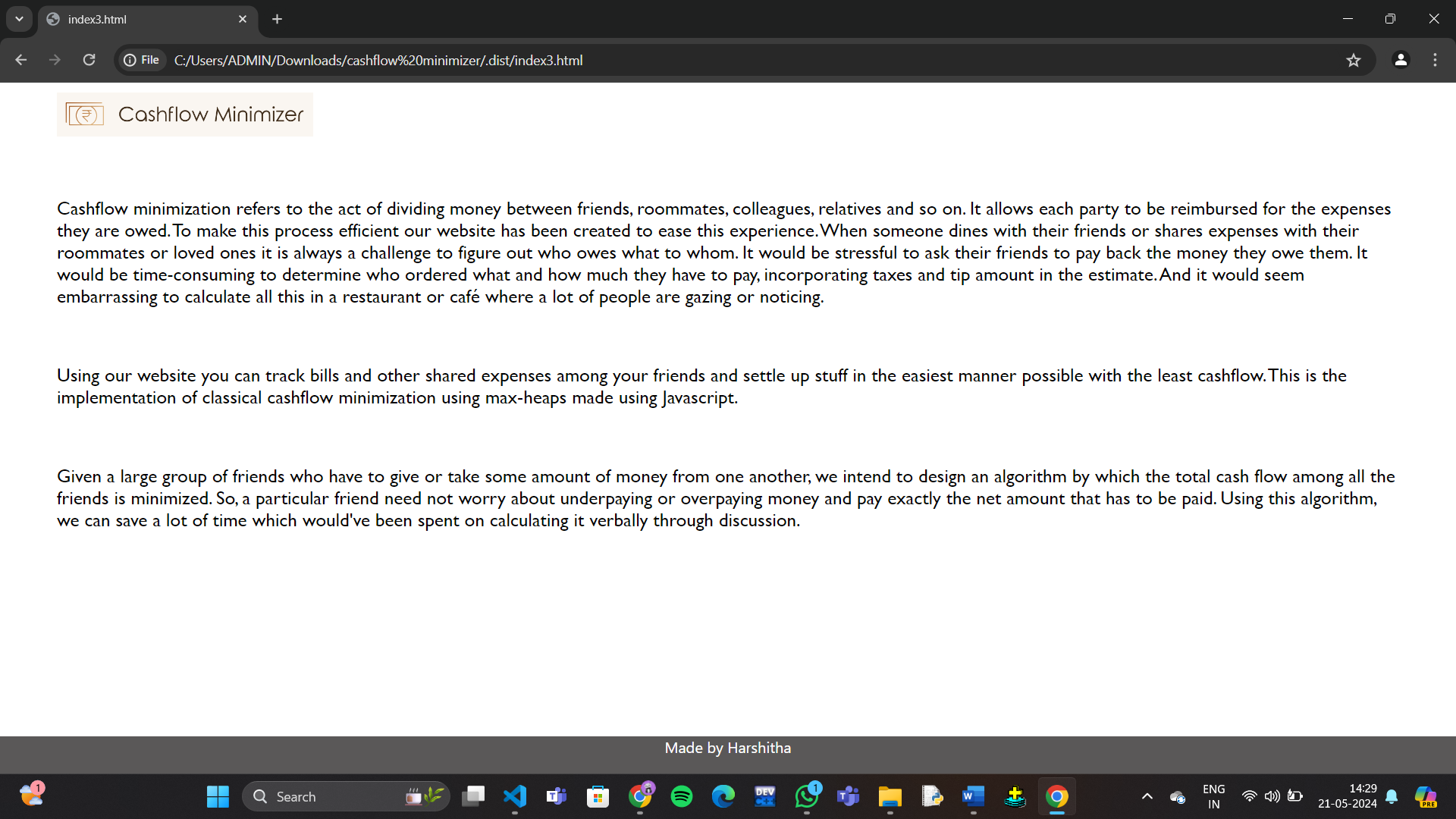
    }

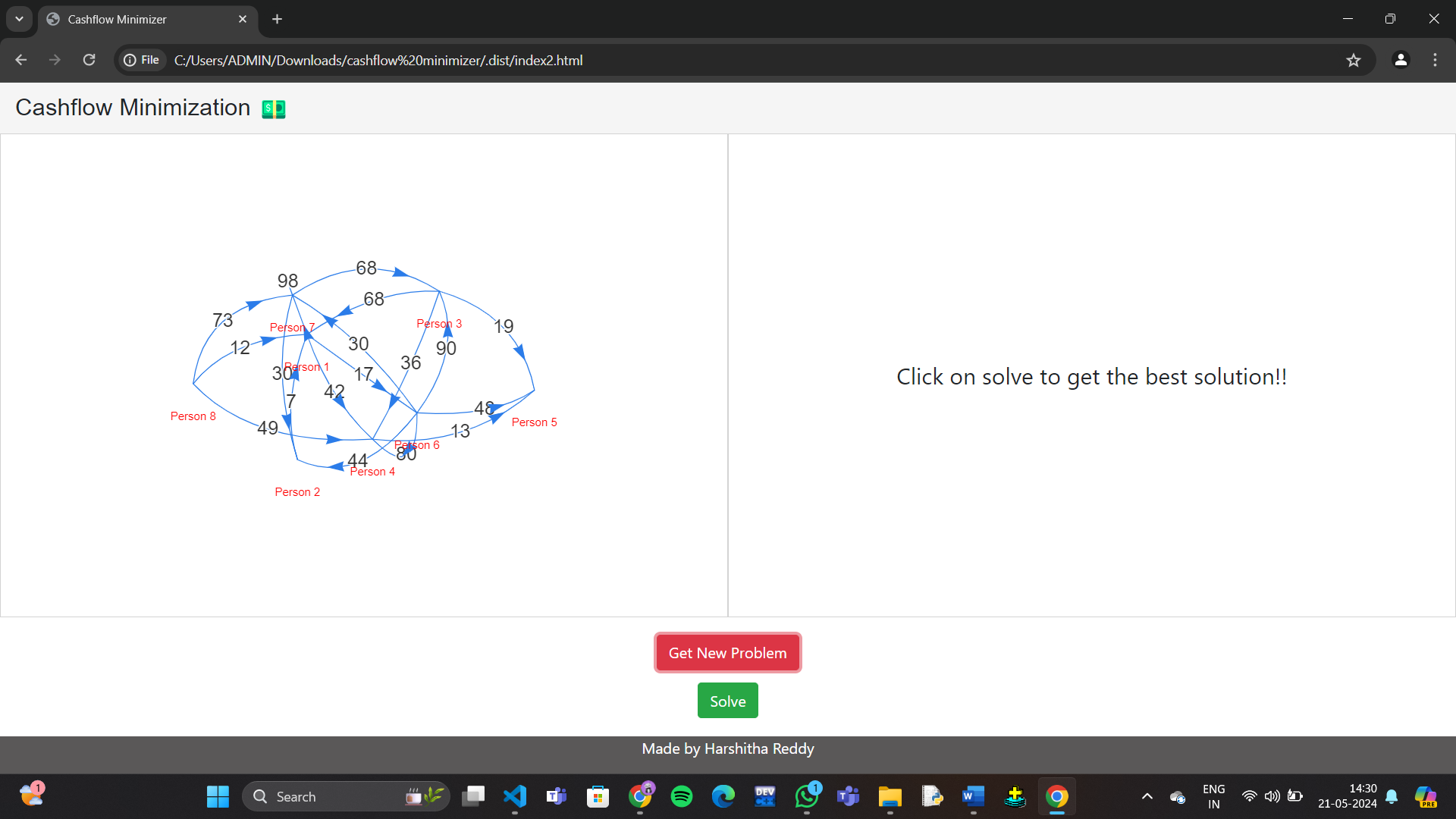
    genNew.click();

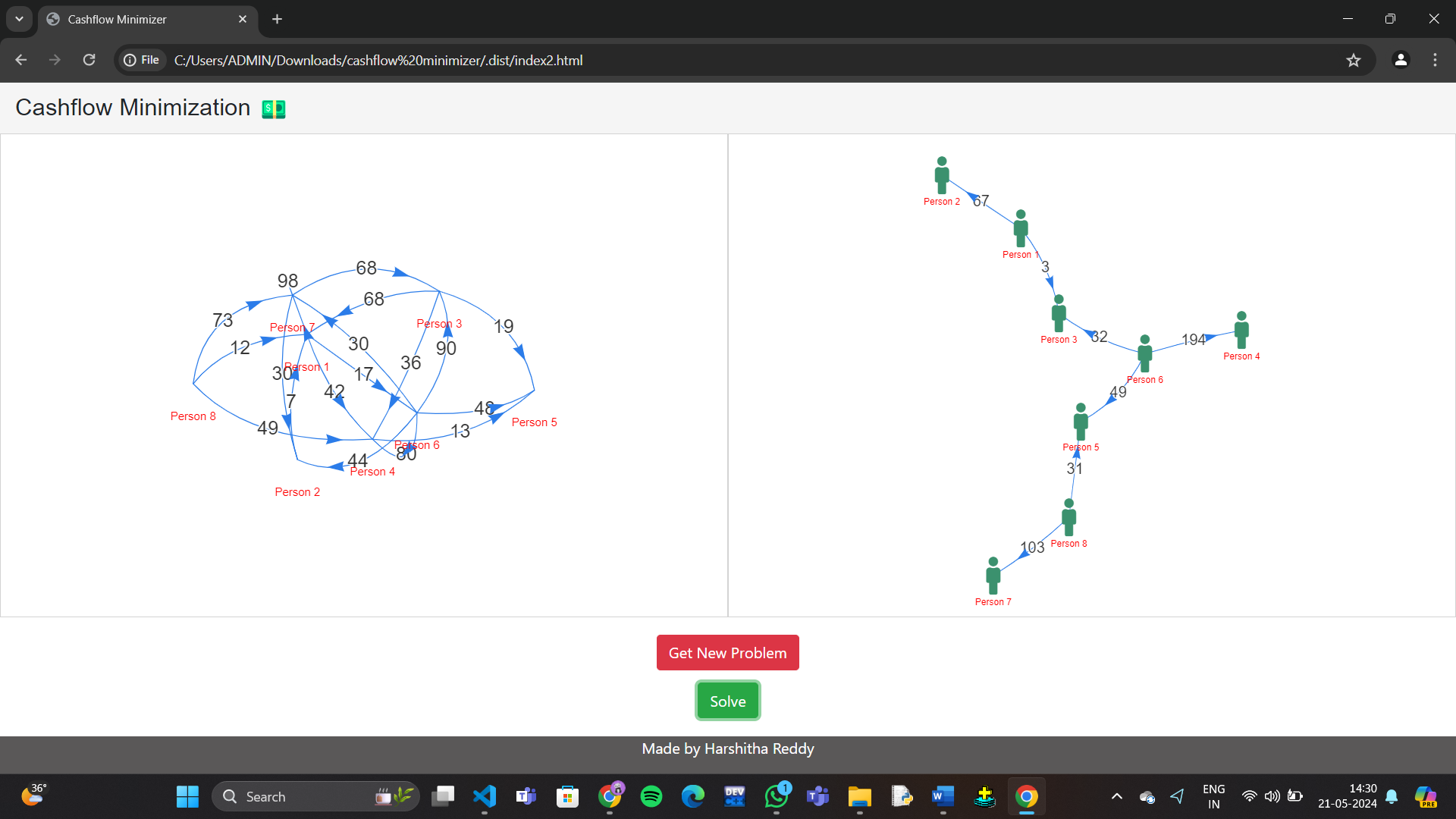
};

**OUTPUTS:**









**RESULTS**

The application for FINDING CASHFLOW MINIMIZATION was successfully created.

**REFERENCES**

* [**https://stackoverflow.com/**](https://stackoverflow.com/)
* **https://www.geeksforgeeks.org/introduction-to-graphs-data-structure-and-algorithm-tutorials/**