NOIDA INSTITUTE OF ENGG & TECHNOLOGY

DEPARTMENT OF CYBERSECURITY

PROJECT REPORT

Competitive programming progress hub

G11

NOIDA INSTITUTE OF ENGG & TECHNOLOGY

DEPARTMENT OF CYBERSECURITY

BONAFIDE CERTIFICATE

This is to certify that **Avnit Verma ,Ankit Kumar, Sant kr. singh** of class **cyber security** has done a project work on **Competitive Programming Progress Hub** during the year 2024-2025 at the **NOIDA INSTITUTE OF ENGG & TECHNOLOGY**

Guide-In- Charge

HOD

CYBERSECURITY

Dated: ……………..

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this AI Chatbot project. I am particularly grateful to my mentors and professors for their invaluable guidance and support throughout the project. Their expertise and encouragement have been instrumental in shaping the project's direction and ensuring its success. I would also like to thank my friends and colleagues for their helpful suggestions and feedback, which have significantly improved the quality of the project. Finally, I would like to acknowledge the contributions of the various online resources and tools that have been utilized in the development of this chatbot. Their availability has greatly facilitated the learning process and the implementation of various functionalities.

Avnit Verma
Ankit Kumar
Sant Kumar Singh

Theory of project

This theory posits that a structured online platform, specifically designed to track progress, provide targeted resources, and foster a sense of community, can significantly enhance self-efficacy and facilitate the development of mastery in competitive programming. This is achieved through the following interconnected mechanisms

key components:

1. **Enhanced Self-Awareness and Goal Setting:**
   - **Mechanism:** The hub provides tools for users to log their practice, track problem-solving performance (e.g., success rate, time taken), and visualize their progress over time. This data-driven approach fosters greater self-awareness of their strengths and weaknesses.
   - **Theoretical Basis:** This aligns with **Social Cognitive Theory**, particularly the concept of **self-monitoring**. By observing their own performance, users can more accurately assess their current skill level and set realistic, achievable goals, a crucial element for building self-efficacy. Furthermore, the ability to see tangible progress reinforces their belief in their capacity to improve.
2. **Targeted Resource Allocation and Personalized Learning:**
   - **Mechanism:** The hub curates and recommends relevant learning materials (e.g., tutorials, problem sets, algorithm explanations) based on a user's identified skill gaps and progress patterns. This personalized approach ensures learners focus on areas where they need the most improvement.
   - **Theoretical Basis:** This resonates with principles of **Cognitive Load Theory**. By providing targeted resources, the hub minimizes cognitive overload by directing learners to information directly relevant to their needs, optimizing their learning efficiency. Additionally, it supports the idea of **scaffolding**, gradually introducing more complex concepts as the user's understanding develops.

3. Cultivating a Supportive Community and Social Learning:

- **Mechanism:** Features like forums, discussion boards, and potentially collaborative practice sessions foster a sense of community among learners. This allows for peer-to-peer support, knowledge sharing, and the exchange of problem-solving strategies.
- **Theoretical Basis:** This draws upon **Social Learning Theory**. Observing others succeed, receiving constructive feedback, and engaging in discussions can enhance motivation, build confidence, and provide diverse perspectives on problem-solving approaches. The social interaction can also mitigate feelings of isolation often associated with independent learning.

4. Promoting Deliberate Practice and Feedback Loops:

- **Mechanism:** The hub encourages consistent practice and provides mechanisms for users to analyze their solutions, compare them with others, and receive feedback (either automated or peer-based). This iterative process of practice and feedback is essential for skill refinement.
- **Theoretical Basis:** This aligns with the concept of **deliberate practice**, which emphasizes focused, systematic practice with the goal of improvement. The feedback mechanisms provide crucial information for identifying errors, understanding underlying concepts, and adjusting future practice strategies.

In essence, this Competitive Programming Progress Hub aims to create a positive feedback loop where structured tracking leads to better self-awareness, targeted resources facilitate efficient learning, community support enhances motivation and knowledge acquisition, and deliberate practice coupled with feedback drives skill mastery and strengthens self-efficacy in competitive programming.

How to Use This in Your Report:

- **Introduction:** Briefly introduce your project and state that it is theoretically grounded in the principles outlined above.
- **Literature Review (if applicable):** You can expand on the theoretical bases mentioned (Social Cognitive Theory,

Cognitive Load Theory, Social Learning Theory, Deliberate Practice) by citing relevant academic literature.

- **Project Design and Features:** Explicitly link the features of your hub to the mechanisms described in the theory. For example, explain how the progress tracking feature directly supports self-monitoring.
- **Expected Outcomes:** Frame your expected outcomes in terms of enhanced self-efficacy, improved problem-solving skills, increased engagement, and a stronger sense of community among users.
- **Evaluation Plan:** If you have an evaluation plan, explain how you will measure the impact of your hub on the factors outlined in the theory (e.g., surveys on self-efficacy, analysis of user activity and performance data, feedback on community features).

Hardware and software requirements

Hardware Requirements :

1. **CPU**:
   - Basic: Intel Core i5 or equivalent.
   - Advanced (for AI models): Intel Core i7/i9 or equivalent.
2. **GPU** (for deep learning):
   - Recommended: NVIDIA RTX 3060/3080 or similar.
   - Cloud services (AWS, Google Cloud) for heavy models.
3. **RAM**:
   - Basic: 8GB to 16GB.
   - Advanced (for large models): 32GB+.
4. **Storage**:
   - SSD: 256GB+ for development, 1TB+ for AI models.
5. **Internet**: Fast connection for cloud services and data transfer.

Software Requirements :

1. **Languages**:
   - Python (main for AI/ML).
   - JavaScript (for web integration).
2. **Libraries**:
   - NLP: spaCy, Hugging Face Transformers.
   - ML: TensorFlow, PyTorch, scikit-learn.
3. **Web Frameworks**:
   - Flask/Django (Python) for backend.
   - Node.js/Express.js (JavaScript) for real-time communication.
4. **Databases**:
   - SQL (MySQL/PostgreSQL) or NoSQL (MongoDB) for data storage.
5. **Cloud Services** (for deployment):
   - AWS, Google Cloud, or Azure for hosting and scalability.
6. **Version Control**:
   - Git (with GitHub/GitLab).

Source code

Index.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Competitive Programming Tracker</title>
   <link rel="stylesheet" href="index.css">
   <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
   <header>
      <h1>Competitive Programming Tracker</h1>
   </header>

   <section class="container">
      <div class="form-container">
         <h2>Add / Edit Problem</h2>
         <form id="problemForm">
            <input type="hidden" id="editIndex">

            <label for="problemName">Problem Name:</label>
            <input type="text" id="problemName" required>

            <label for="platform">Platform:</label>
            <input type="text" id="platform" required>

            <label for="difficulty">Difficulty:</label>
            <select id="difficulty">
               <option value="Easy">Easy</option>
               <option value="Medium">Medium</option>
               <option value="Hard">Hard</option>
            </select>

            <label for="category">Category:</label>
            <input type="text" id="category" required>

            <label for="date">Date Solved:</label>
            <input type="date" id="date" required>

            <label for="timeTaken">Time Taken (in minutes):</label>
            <input type="number" id="timeTaken" required>

            <label for="solutionCode">Solution Code:</label>
```

```
                <textarea id="solutionCode" rows="5" placeholder="Paste your solution
here..."></textarea>

                <button type="submit">Save Problem</button>
            </form>
        </div>

        <div class="tracker-container">
            <h2>Your Progress</h2>
            <table id="problemTable">
                <thead>
                    <tr>
                        <th>Problem Name</th>
                        <th>Platform</th>
                        <th>Difficulty</th>
                        <th>Category</th>
                        <th>Date Solved</th>
                        <th>Time Taken</th>
                        <th>Solution</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                </tbody>
            </table>
        </div>
    </section>

    <section class="chart-container">
        <h2>Difficulty Breakdown</h2>
        <canvas id="progressChart"></canvas>
    </section>

    <footer>
        <p>Competitive Programming Tracker by Your Name</p>
    </footer>

    <script src="index.js"></script>
</body>
</html>
```

Script.JS :

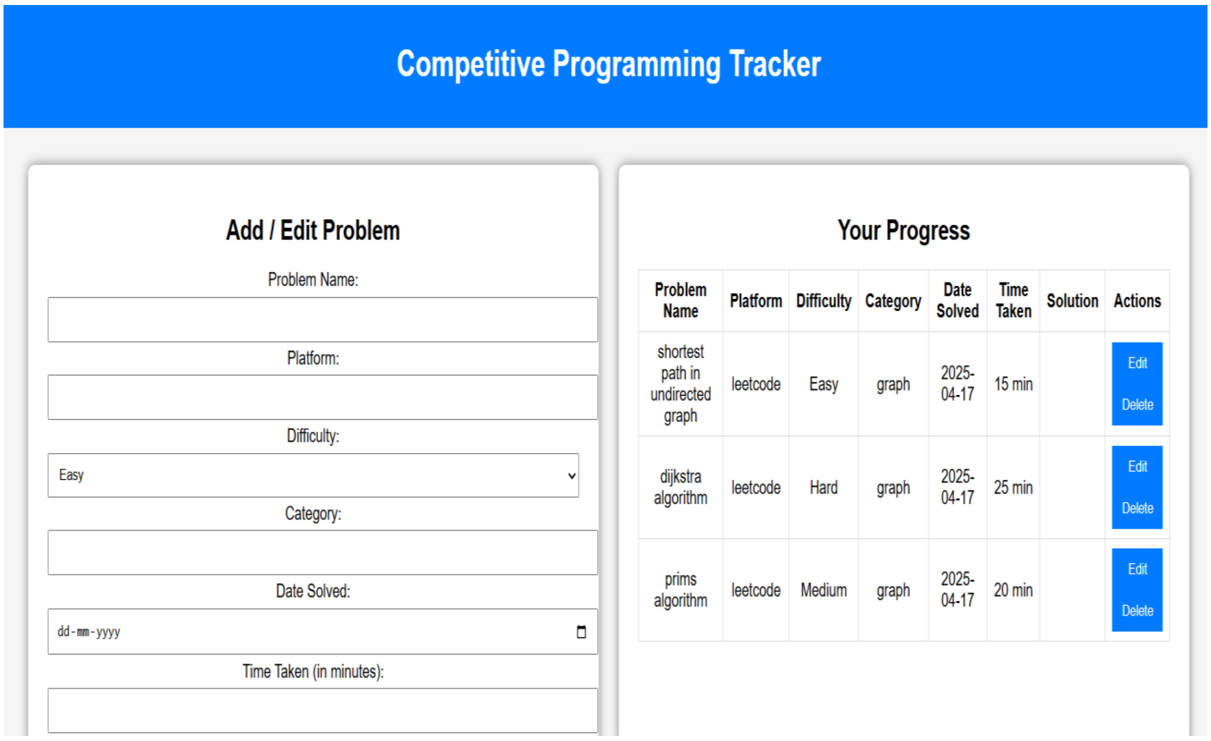
```
document.addEventListener("DOMContentLoaded", function () {
  const problemForm = document.getElementById("problemForm");
  const problemTable =
document.getElementById("problemTable").getElementsByTagName("tbody")[0];
  const editIndex = document.getElementById("editIndex");
  let problems = JSON.parse(localStorage.getItem("problems")) || [];

  function saveToLocalStorage() {
    localStorage.setItem("problems", JSON.stringify(problems));
    renderTable();
    updateChart();
  }

  function renderTable() {
    problemTable.innerHTML = "";
    problems.forEach((problem, index) => {
      let row = problemTable.insertRow();
      row.innerHTML = `
        <td>${problem.name}</td>
        <td>${problem.platform}</td>
        <td>${problem.difficulty}</td>
        <td>${problem.category}</td>
        <td>${problem.date}</td>
        <td>${problem.timeTaken} min</td>
        <td><pre>${problem.solution}</pre></td>
        <td>
          <button onclick="editProblem(${index})">Edit</button>
          <button onclick="deleteProblem(${index})">Delete</button>
        </td>
      `;
    });
  }

  function updateChart() {
    const counts = { Easy: 0, Medium: 0, Hard: 0 };
    problems.forEach(p => counts[p.difficulty]++);
    const ctx = document.getElementById("progressChart").getContext("2d");
    new Chart(ctx, {
      type: "pie",
      data: {
        labels: ["Easy", "Medium", "Hard"],
        datasets: [{
          label: "Problems Solved",
          data: [counts.Easy, counts.Medium, counts.Hard],
          backgroundColor: ["green", "orange", "red"]
        }]
```

```
            }
        });
    }

    problemForm.addEventListener("submit", function (e) {
        e.preventDefault();
        const problem = {
            name: document.getElementById("problemName").value,
            platform: document.getElementById("platform").value,
            difficulty: document.getElementById("difficulty").value,
            category: document.getElementById("category").value,
            date: document.getElementById("date").value,
            timeTaken: document.getElementById("timeTaken").value,
            solution: document.getElementById("solutionCode").value
        };

        if (editIndex.value !== "") {
            problems[editIndex.value] = problem;
            editIndex.value = "";
        } else {
            problems.push(problem);
        }

        saveToLocalStorage();
        problemForm.reset();
    });

    window.editProblem = function (index) {
        let p = problems[index];
        document.getElementById("problemName").value = p.name;
        document.getElementById("platform").value = p.platform;
        document.getElementById("difficulty").value = p.difficulty;
        document.getElementById("category").value = p.category;
        document.getElementById("date").value = p.date;
        document.getElementById("timeTaken").value = p.timeTaken;
        document.getElementById("solutionCode").value = p.solution;
        editIndex.value = index;
    };

    window.deleteProblem = function (index) {
        problems.splice(index, 1);
        saveToLocalStorage();
    };

    renderTable();
    updateChart();
}); https://github.com/avnit024/project
```

Style.css :

```
body {
   font-family: Arial, sans-serif;
   margin: 0;
   padding: 0;
   background-color: #f5f5f5;
   text-align: center;
}

header {
   background-color: #007bff;
   color: white;
   padding: 1rem;
}

.container {
   display: flex;
   justify-content: space-around;
   padding: 2rem;
}

.form-container, .tracker-container {
   background: white;
   padding: 1.5rem;
   border-radius: 8px;
   box-shadow: 0px 0px 10px gray;
   width: 45%;
}

input, select, textarea {
   width: 100%;
   margin: 5px 0;
   padding: 10px;
}

button {
   width: 100%;
   padding: 10px;
   background-color: #007bff;
   color: white;
   border: none;
   cursor: pointer;
}

button:hover {
   background-color: #0056b3;
}

table {
   width: 100%;
   border-collapse: collapse;
   margin-top: 10px;
}

th, td {
   border: 1px solid #ddd;
   padding: 8px;
}

.chart-container {
   width: 60%;
   margin: 20px auto;
}
```

input and outputs



Competitive Programming Tracker

Add / Edit Problem

Problem Name:

Platform:

Difficulty:

Easy

Category:

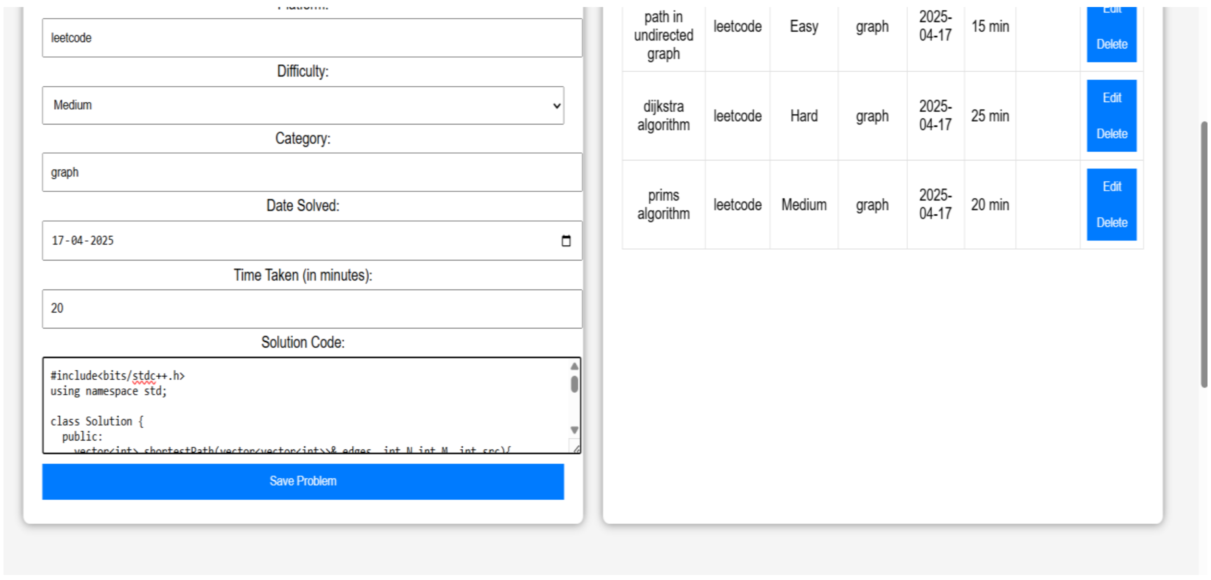
Date Solved:

dd-mm-yyyy

Time Taken (in minutes):

Your Progress

Problem Name	Platform	Difficulty	Category	Date Solved	Time Taken	Solution	Actions
shortest path in undirected graph	leetcode	Easy	graph	2025-04-17	15 min		Edit Delete
dijkstra algorithm	leetcode	Hard	graph	2025-04-17	25 min		Edit Delete
prims algorithm	leetcode	Medium	graph	2025-04-17	20 min		Edit Delete



leetcode

Difficulty:

Medium

Category:

graph

Date Solved:

17-04-2025

Time Taken (in minutes):

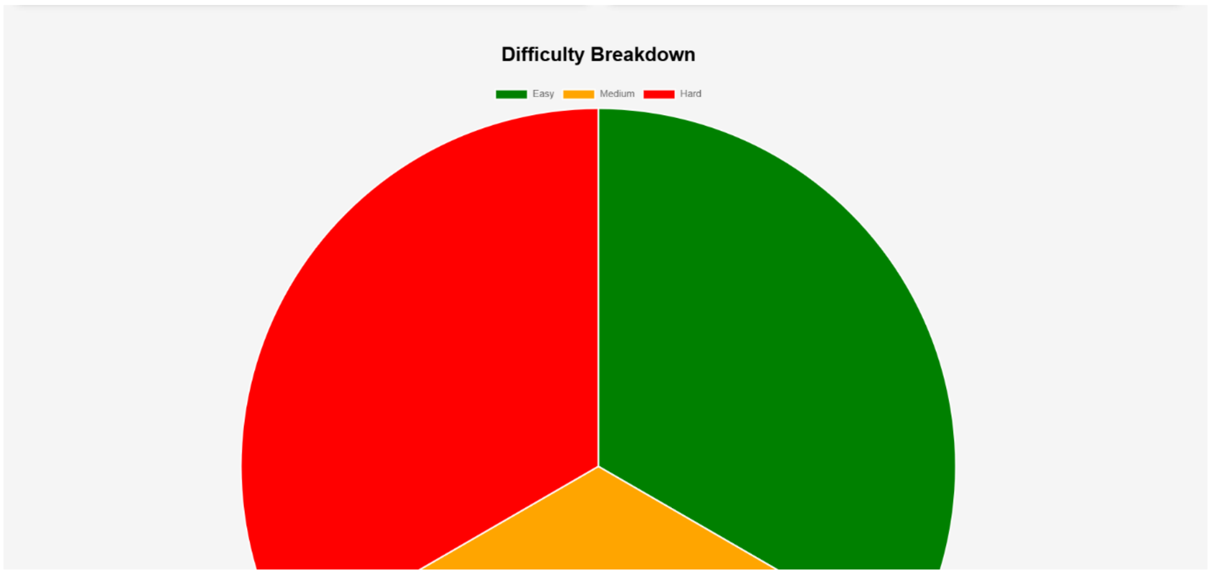
20

Solution Code:

```
#include<bits/stdc++.h>
using namespace std;

class Solution {
  public:
    vector<int> shortestPath(vector<vector<int>>& edges, int N,int M, int src){
```

Save Problem

path in undirected graph	leetcode	Easy	graph	2025-04-17	15 min		Delete
dijkstra algorithm	leetcode	Hard	graph	2025-04-17	25 min		Edit Delete
prims algorithm	leetcode	Medium	graph	2025-04-17	20 min		Edit Delete


Difficulty Breakdown
Easy
Medium
Hard

References

I. **Theoretical Foundations (Self-Efficacy, Learning Theories):**

- Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review, 84*(2), 191–215. (A foundational paper on self-efficacy)
- Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory*. Prentice-Hall. (A comprehensive exploration of Social Cognitive Theory)
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*(2), 257–285. (Key paper on Cognitive Load Theory)
- Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Harvard University Press. (Relevant for understanding social learning and the Zone of Proximal Development, which relates to scaffolding)
- Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance. *Psychological Review, 100*(3), 363–406. (Fundamental work on deliberate practice)

II. **Competitive Programming and Online Learning Platforms**:

- Halim, S., & Halim, F. N. (2013). *Competitive Programming 3: The New Lower Bound of Programming Contests*. Lulu.com. (A widely used textbook in the competitive programming community, providing insights into problem-solving strategies and algorithms)
- Laaksonen, A. (2017). *Competitive Programmer's Handbook*. (A comprehensive online resource and book covering essential topics in competitive programming)
- Websites of Popular Competitive Programming Platforms:
  - Codeforces (codeforces.com): A leading platform with contests, problems, and community features. You can reference it as an example of an existing platform.
  - CodeChef (codechef.com): Another popular platform with a focus on the Indian competitive programming

community.
    - LeetCode (leetcode.com): Widely used for interview preparation and competitive programming practice.
    - AtCoder (atcoder.jp): A well-regarded Japanese competitive programming platform.
    - TopCoder (topcoder.com): One of the original competitive programming platforms.
    - *(Mentioning these can provide context for the existing landscape and highlight the need for your specific hub)*
- Research Papers on Online Learning and Educational Technology: Search academic databases (like ACM Digital Library, IEEE Xplore, Google Scholar, ERIC) for studies on:
    - The effectiveness of online learning platforms.
    - The role of community and social interaction in online learning.
    - Personalized learning approaches in computer science education.
    - Gamification and motivation in online learning environments.
    - Progress tracking and visualization in educational tools.

III. **Software Development and Web Technologies (Depending on your stack):**

- References specific to your chosen programming languages, frameworks, and databases. For example:
    - Official documentation for Python, Django/Flask, JavaScript, React/Angular/Vue.js, Node.js, PostgreSQL, MySQL, etc.
    - Books and tutorials on the specific technologies you used.
- Books or articles on software engineering best practices:
    - Clean Code by Robert C. Martin.
    - Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, [1] and John Vlissides.