Synapse

A Graph-Based Path Optimization System for Multi-Company Skill Preparation

By

Mansi Anand, Nidhish *Suvarna*, Rishit Laddha

PROBLEM STATEMENT

Preparing for placements across multiple companies often involves learning numerous skill sets, each with partially overlapping requirements. Students end up enrolling in several courses that teach similar or identical concepts, leading to **redundant learning**, **time inefficiency**, and **disorganized preparation**.

Traditional learning platforms or recommendation systems treat each goal independently. They suggest paths for *one* company or *one* skill domain at a time. As a result, when a student targets multiple companies, there's **no unified system** to optimize the journey across shared skills or minimize duplicate effort.

Synapse redefines this challenge as a **graph optimization problem**, where:

- Each **course** is a node.
- Each **connection** (edge) represents a learning dependency or skill transition.
- The **distance** between nodes represents time, difficulty, or effort.
- The **goal** is to find the *minimum set of paths* that collectively cover the skill requirements of multiple target companies.

By applying and comparing **shortest path algorithms** (Dijkstra, Bellman–Ford, SSSP) with a **multi-goal optimization model (Steiner Tree)**, *Synapse* aims to construct a **single, efficient learning roadmap** that ensures complete readiness with minimal redundancy.

OBJECTIVES

1. To identify the most efficient course paths covering skills required by multiple companies.
2. To reduce redundancy by merging overlapping courses.
3. To compare classical algorithms with advanced optimization methods.
4. To create a unified learning graph from zero knowledge to job readiness.
5. To build a scalable system adaptable to diverse learning datasets.

ALGORITHMS USED

The Synapse system integrates both traditional and advanced graph algorithms to construct optimal learning paths. Each algorithm plays a specific role in identifying and refining routes between courses (nodes) that collectively fulfil company-specific skill requirements.

1. DIJKSTRA'S ALGORITHM

Dijkstra's algorithm is a classic shortest-path algorithm that determines the minimum cost path from a single source node to all other nodes in a weighted graph with non-negative edge weights.

Use Case In Synapse: Dijkstra's algorithm models the most direct learning route for a single company identifying the shortest progression from a learner's starting point (zero knowledge) to a company's full skill requirement.

Limitations: While efficient for isolated company paths, Dijkstra's approach becomes computationally expensive when repeated for multiple companies, and it does not account for overlapping courses between companies.

2. BELLMAN FORD ALGORITHM

The Bellman–Ford algorithm extends pathfinding to graphs with negative edge weights and supports more flexible edge relaxation.

Use Case In Synapse: It represents a more generalized approach to learning path computation useful in cases where certain courses (edges) may have "negative weights," such as overlapping content or redundant topics, representing reduced learning effort.

Limitations: Although it accommodates broader cases, Bellman–Ford is slower than Dijkstra's algorithm and scales poorly when multiple companies are considered simultaneously.

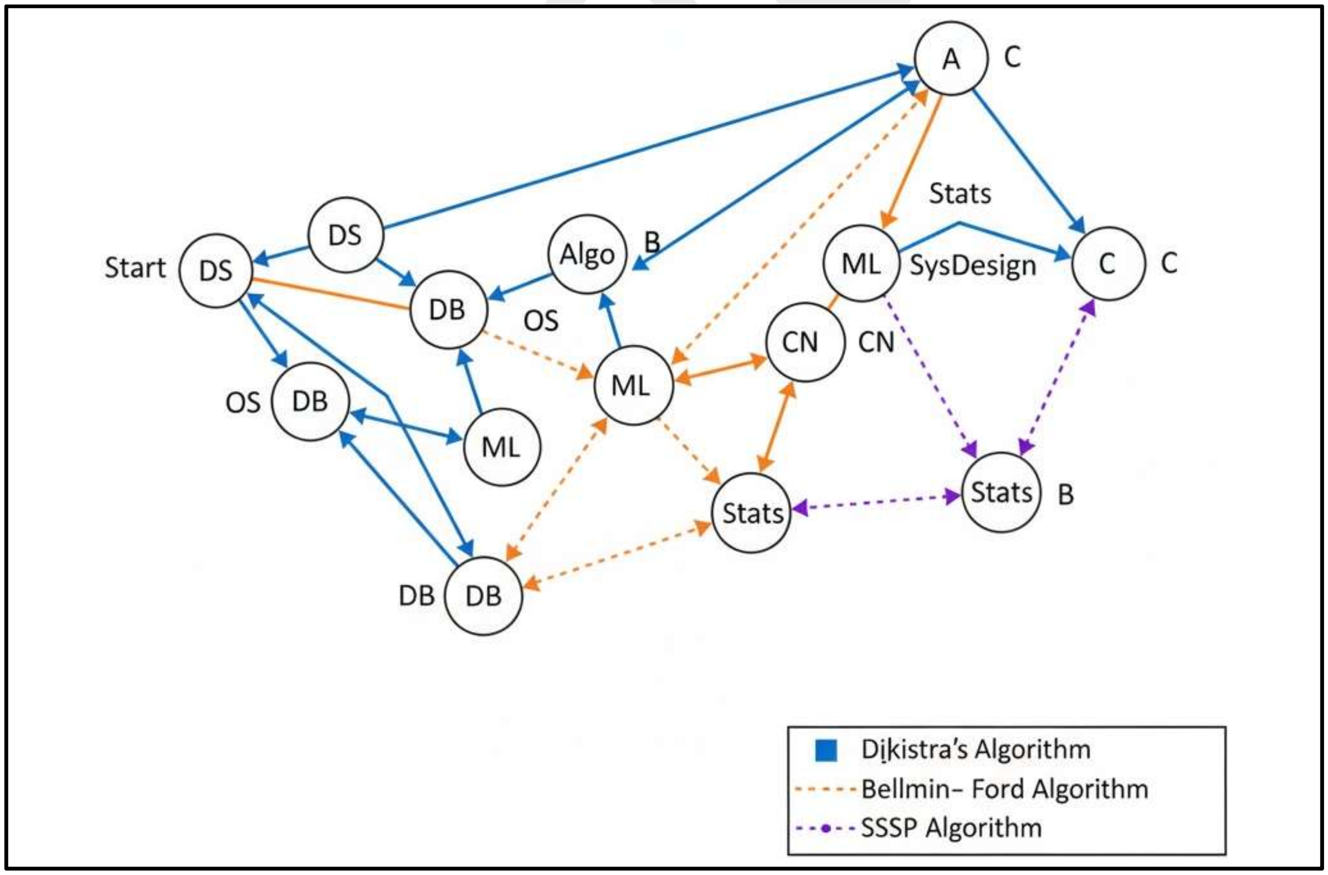
3. Single Source Shortest Path (SSSP)

The Single Source Shortest Path (SSSP) framework underpins both Dijkstra's and Bellman–Ford algorithms. It computes optimal paths from a single starting point to multiple destinations.

Use Case In Synapse: SSSP serves as the foundational layer generating optimal course sequences individually for each company. These sequences are then used as input for the global optimization stage (Steiner Tree).

This step ensures:

- A consistent baseline across all companies.
- Reusability when new companies are added.
- Comparability of algorithm performance (speed and path efficiency).



Algorithmic Path Comparison – Dijkstra, Bellman–Ford, and SSSP

STEINER TREE ALGORITHM

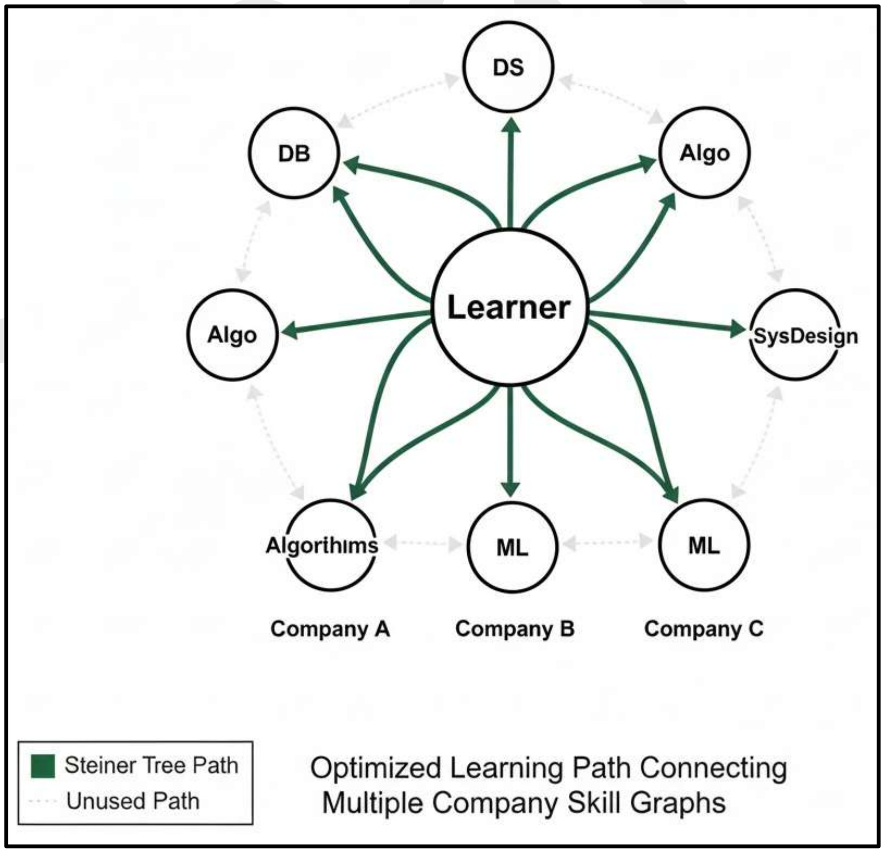
The **Steiner Tree Algorithm** is used to find the minimum-cost network connecting multiple target nodes. In **Synapse**, these nodes represent different companies' skill requirements, and edges represent the learning effort between courses.

Why We're Using It: Traditional algorithms like Dijkstra's or Bellman–Ford handle only single-source paths. Steiner Tree extends this by connecting multiple company paths simultaneously — reducing overlap and optimizing the total learning path.

Variants Used in Synapse:

- **KMB Algorithm (Kou–Markowsky–Berman):** Builds an approximate minimum network by connecting all company nodes through shared courses.
- **Takahashi–Matsuyama Algorithm:** Greedy approach that incrementally links each company to the network via the shortest available route.

Relevance to Synapse: Both algorithms help Synapse merge individual company paths (from SSSP) into one efficient learning roadmap that minimizes total course load while covering all required skills.



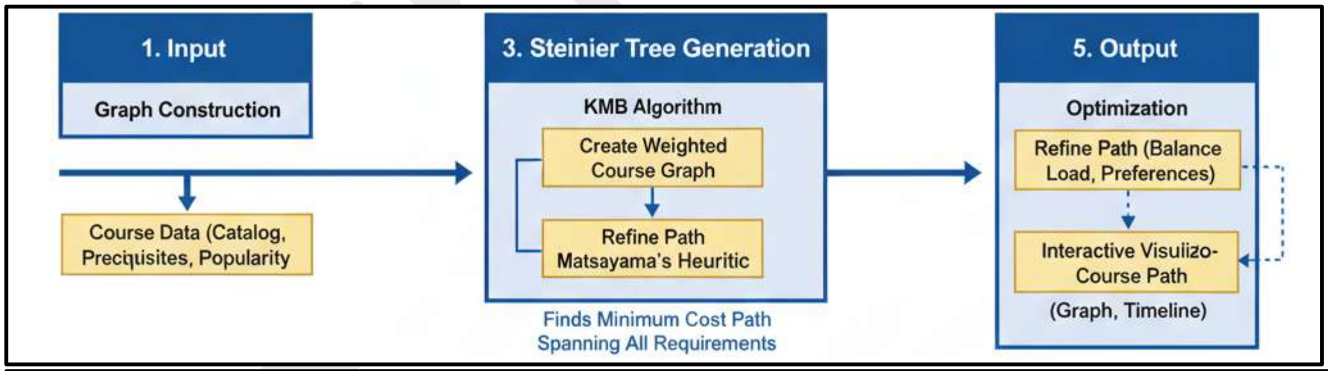
Optimized Learning Path Connecting Multiple Company Skill Graphs

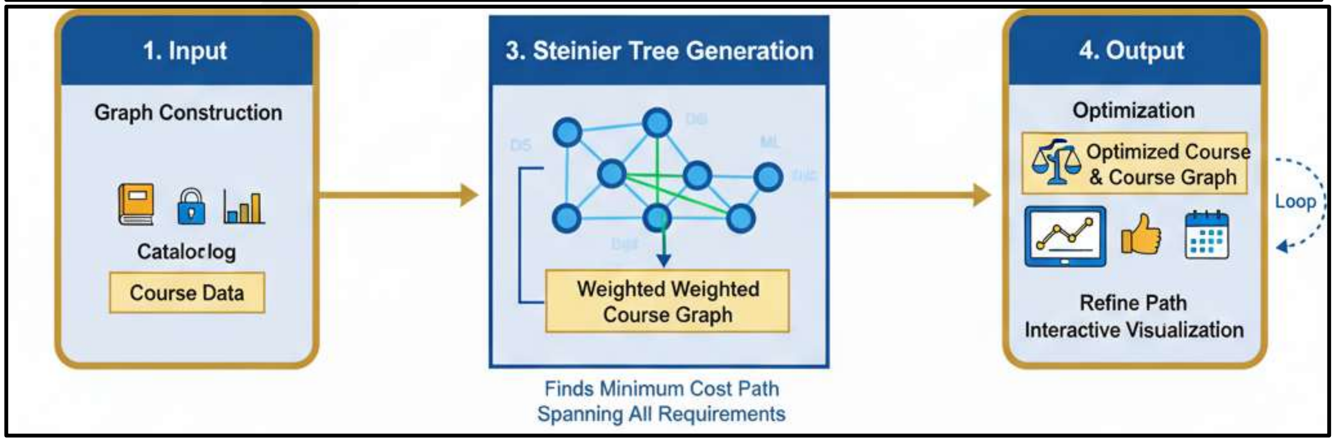
PROPOSED SYSTEM

The proposed system, Synapse, applies a Steiner Tree–based graph optimization approach to recommend the most efficient set of courses for students. Courses are modeled as nodes, and their relationships (prerequisites, dependencies, or co-requirements) are represented as edges. The system identifies a minimum-cost subgraph (Steiner Tree) connecting all essential courses while optionally including intermediate nodes that reduce the total cost.

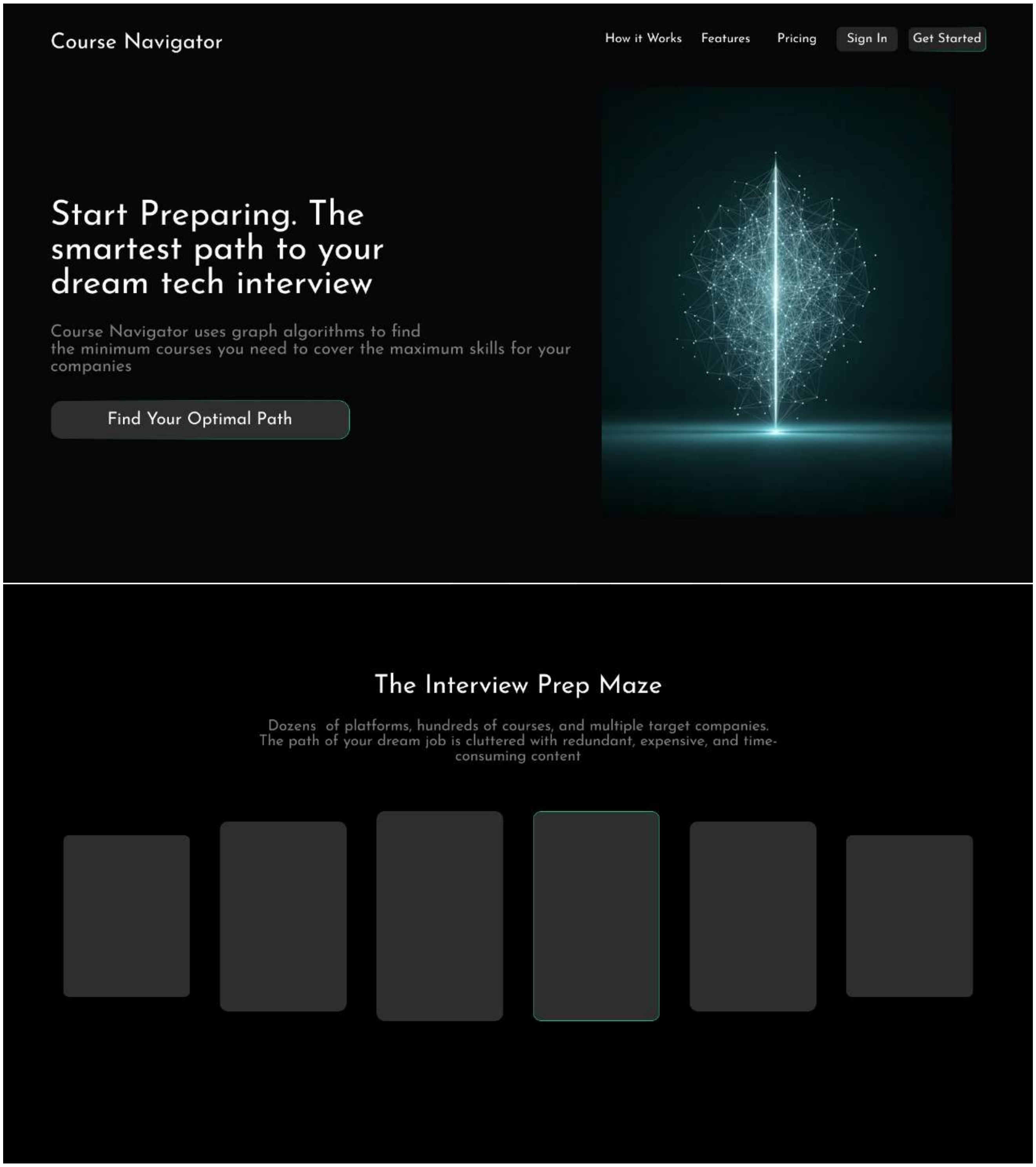
The workflow includes:

1. Input Processing: Collects course data (IDs, prerequisites, difficulty levels, credits).
2. Graph Generation: Constructs a weighted graph based on course interconnections.
3. Steiner Tree Formation: Uses KMB and Takayuki Matsuyama's methods to find the minimal connecting subgraph that covers all key courses.
4. Optimization: Evaluates alternate Steiner nodes to reduce total path cost or completion time.
5. Output: Displays the optimal course sequence with total weight reduction and a visual graph representation.


1. Input
Graph Construction
Course Data (Catalog, Preciquisites, Popularity
3. Steinier Tree Generation
KMB Algorithm
Create Weighted Course Graph
Refine Path Matsayama's Heuritic
Finds Minimum Cost Path Spanning All Requirements
5. Output
Optimization
Refine Path (Balance Load, Preferences)
Interactive Visuiizo- Course Path
(Graph, Timeline)


1. Input
Graph Construction
Catalorlog
Course Data
3. Steinier Tree Generation
Weighted Weighted Course Graph
Finds Minimum Cost Path Spanning All Requirements
4. Output
Optimization
Optimized Course & Course Graph
Loop
Refine Path Interactive Visualization

SAMPLE UI PROTOTYPE



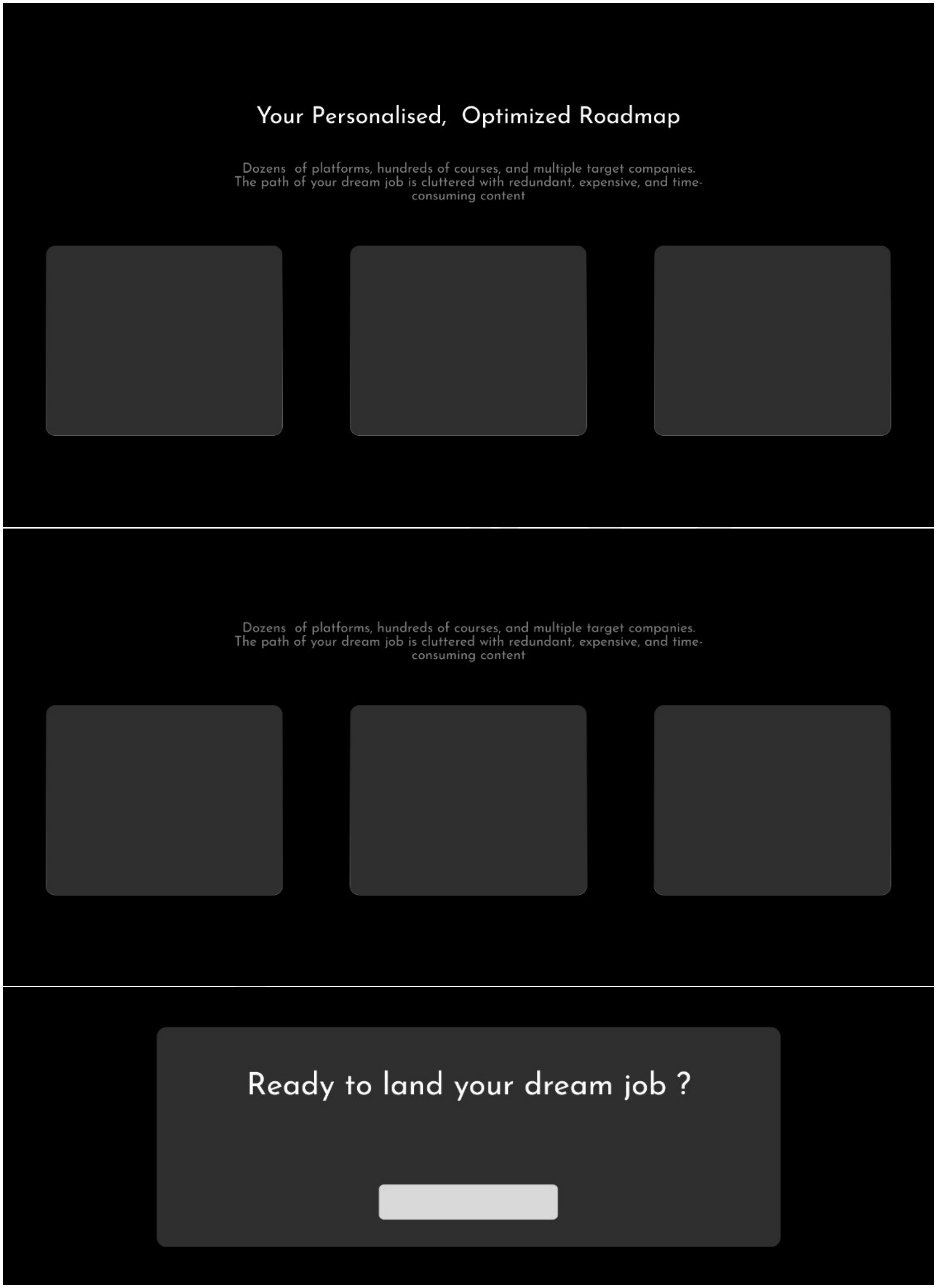
Course Navigator

Start Preparing. The smartest path to your dream tech interview

Course Navigator uses graph algorithms to find
the minimum courses you need to cover the maximum skills for your companies

The Interview Prep Maze

Dozens of platforms, hundreds of courses, and multiple target companies.
The path of your dream job is cluttered with redundant, expensive, and time-consuming content

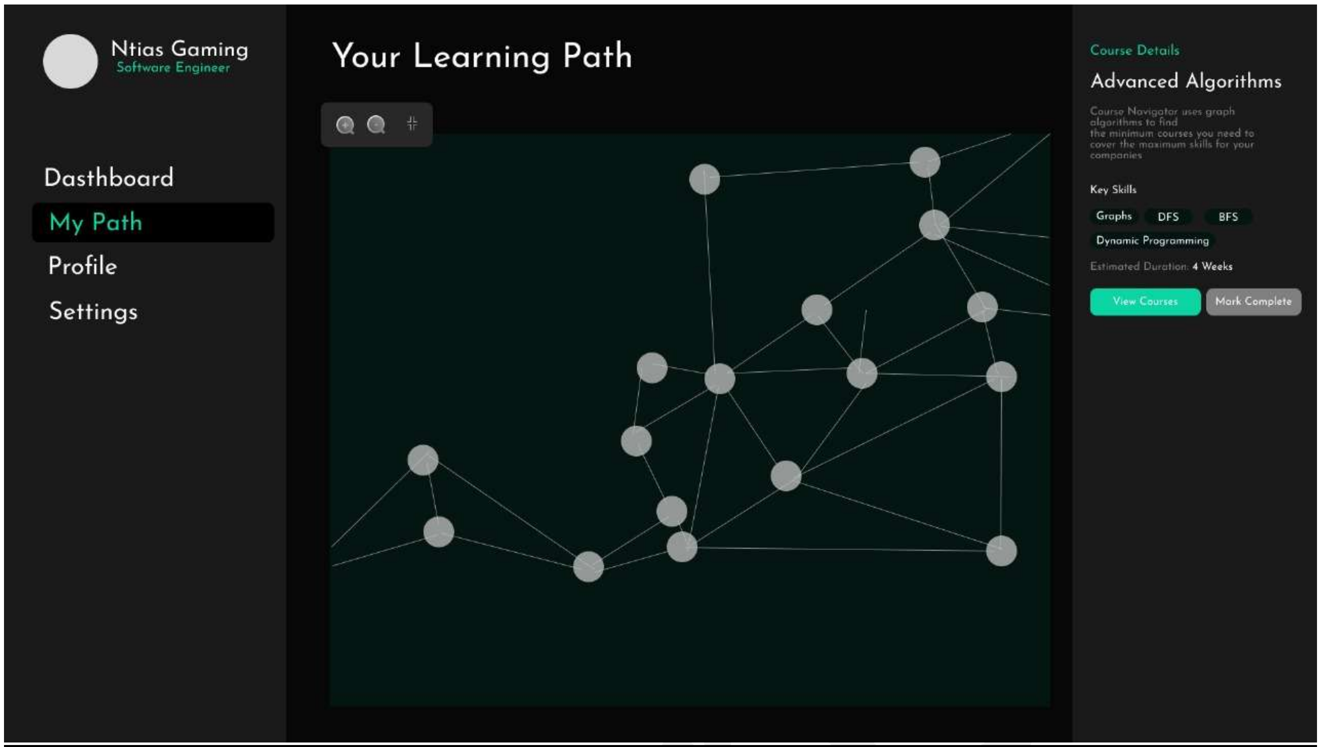


Your Personalised, Optimized Roadmap

Dozens of platforms, hundreds of courses, and multiple target companies.
The path of your dream job is cluttered with redundant, expensive, and time-consuming content

Dozens of platforms, hundreds of courses, and multiple target companies.
The path of your dream job is cluttered with redundant, expensive, and time-consuming content

Ready to land your dream job ?



Your Learning Path

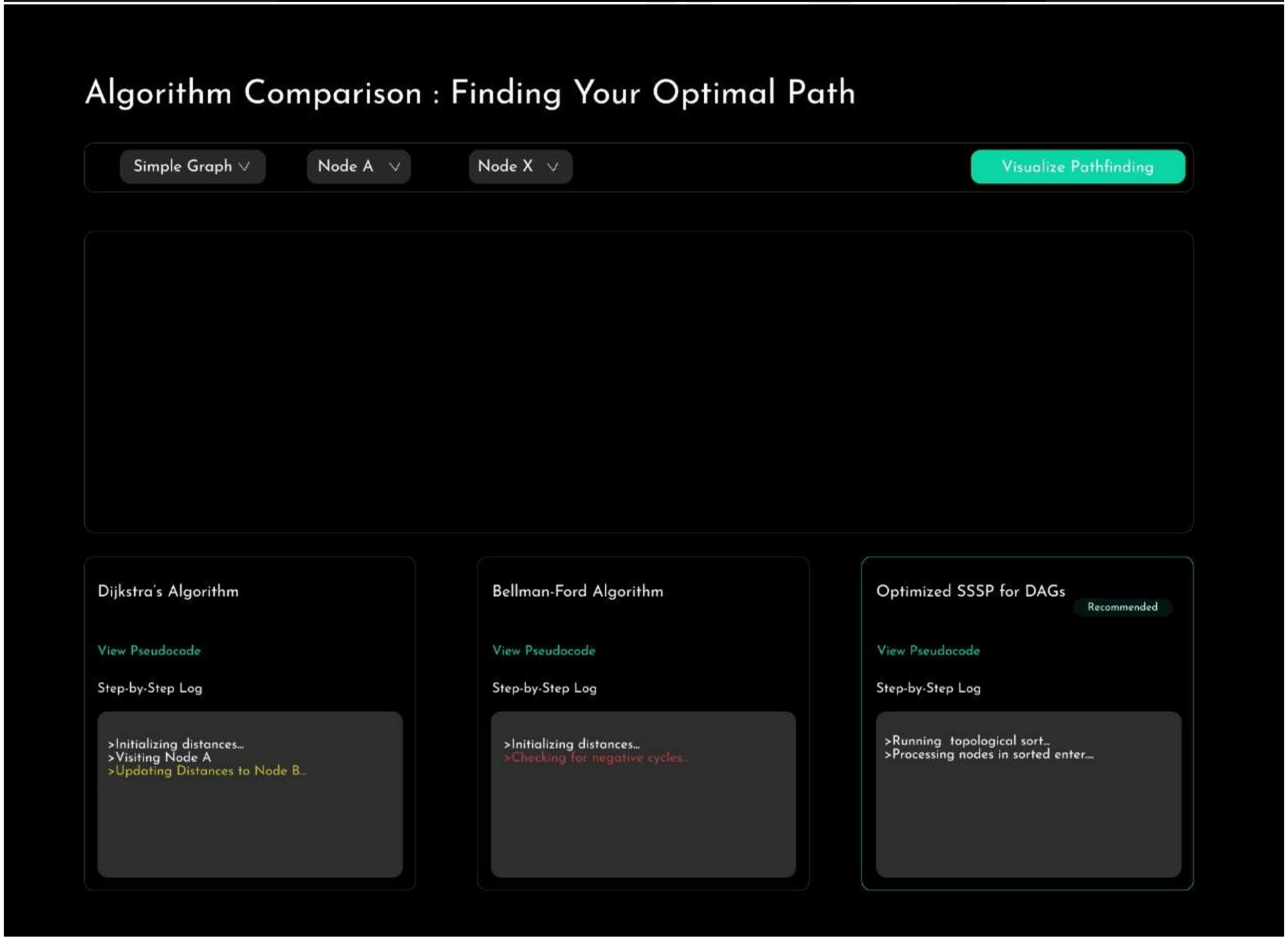
Course Details

Advanced Algorithms

Course Navigator uses graph algorithms to find
the minimum courses you need to cover the maximum skills for your companies

Key Skills

Estimated Duration: 4 Weeks



Algorithm Comparison : Finding Your Optimal Path

Dijkstra's Algorithm

Step-by-Step Log

```
>Initializing distances...
>Visiting Node A
>Updating Distances to Node B...
```

Bellman-Ford Algorithm

Step-by-Step Log

```
>Initializing distances...
>Checking for negative cycles...
```

Optimized SSSP for DAGs

Step-by-Step Log

```
>Running  topological sort...
>Processing nodes in sorted enter...
```

FUTURE SCOPE

- Synapse can be expanded into a full-fledged adaptive learning platform that dynamically updates course paths based on real-time hiring trends and user performance.
- Integration with APIs from platforms like Coursera, LinkedIn Learning, or company job boards can automate course recommendations.
- The system can evolve into a predictive engine, forecasting the optimal learning sequence for a desired role or company using AI-based clustering and recommendation models.
- Visualization modules can be added to display optimized learning routes and progress analytics in real time.
- Multi-user collaboration features can allow peer comparisons, team-based learning, and shared preparation paths.

CONCLUSION

Synapse simplifies and optimizes the skill-preparation journey for multiple companies using graph-based algorithms. By integrating concepts from traditional shortest path methods and the Steiner Tree approach, it ensures efficiency, scalability, and adaptability. The project demonstrates how algorithmic optimization can bridge the gap between complex data structures and real-world educational personalization, paving the way for intelligent, adaptive learning ecosystems.

GITHUB REPOSITORY LINK

https://github.com/Course-Navigator

TEAM MEMBERS

Mansi Anand - 2306261
Nidhish Suvarna - 2309713
Rishit Laddha - 2309575