ARTIFICIAL INTELLIGENCE

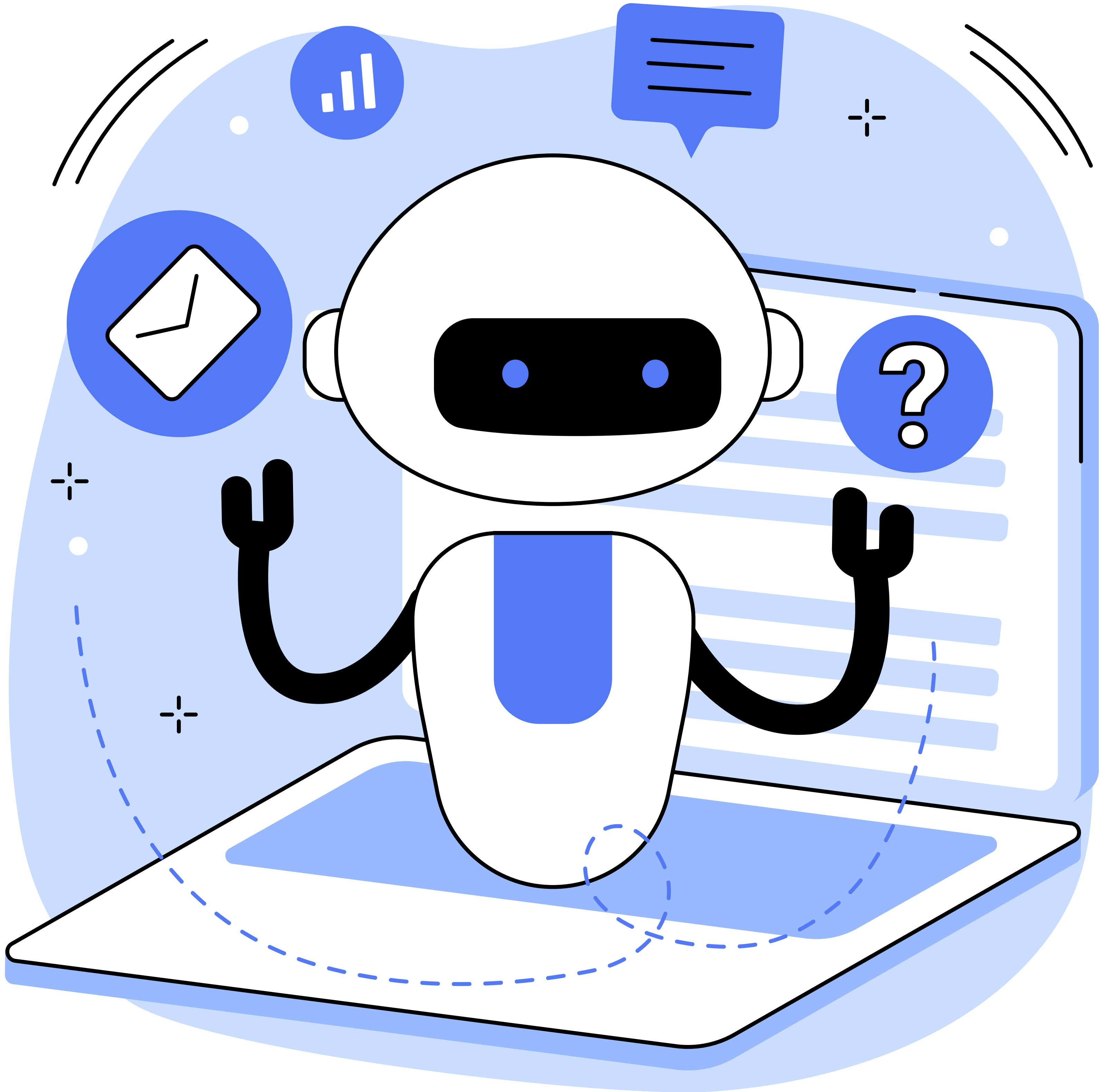
Practical uses in EFL

CONTENTS

CONTENTS

INTRODUCTION

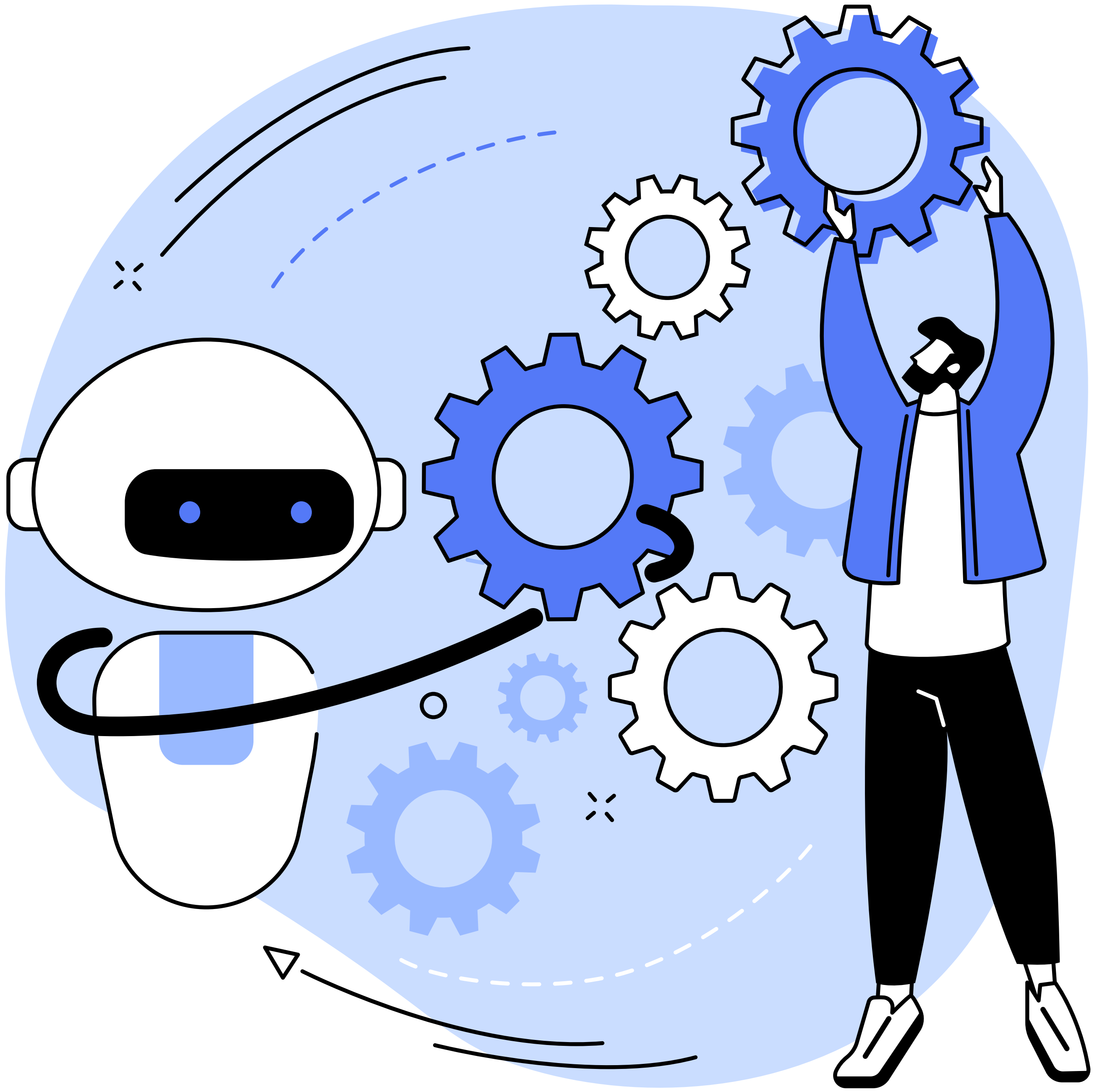
Timetabling issues can occur in a range of industries, including healthcare, sports, transportation, and education. This project focuses on the university course scheduling issue, which is often experienced in colleges around the world. Manual scheduling results in conflicts and flaws, but computer scheduling can be used to remedy these issues.



PROBLEM DESCRIPTION

Time conflicts

Prerequisites



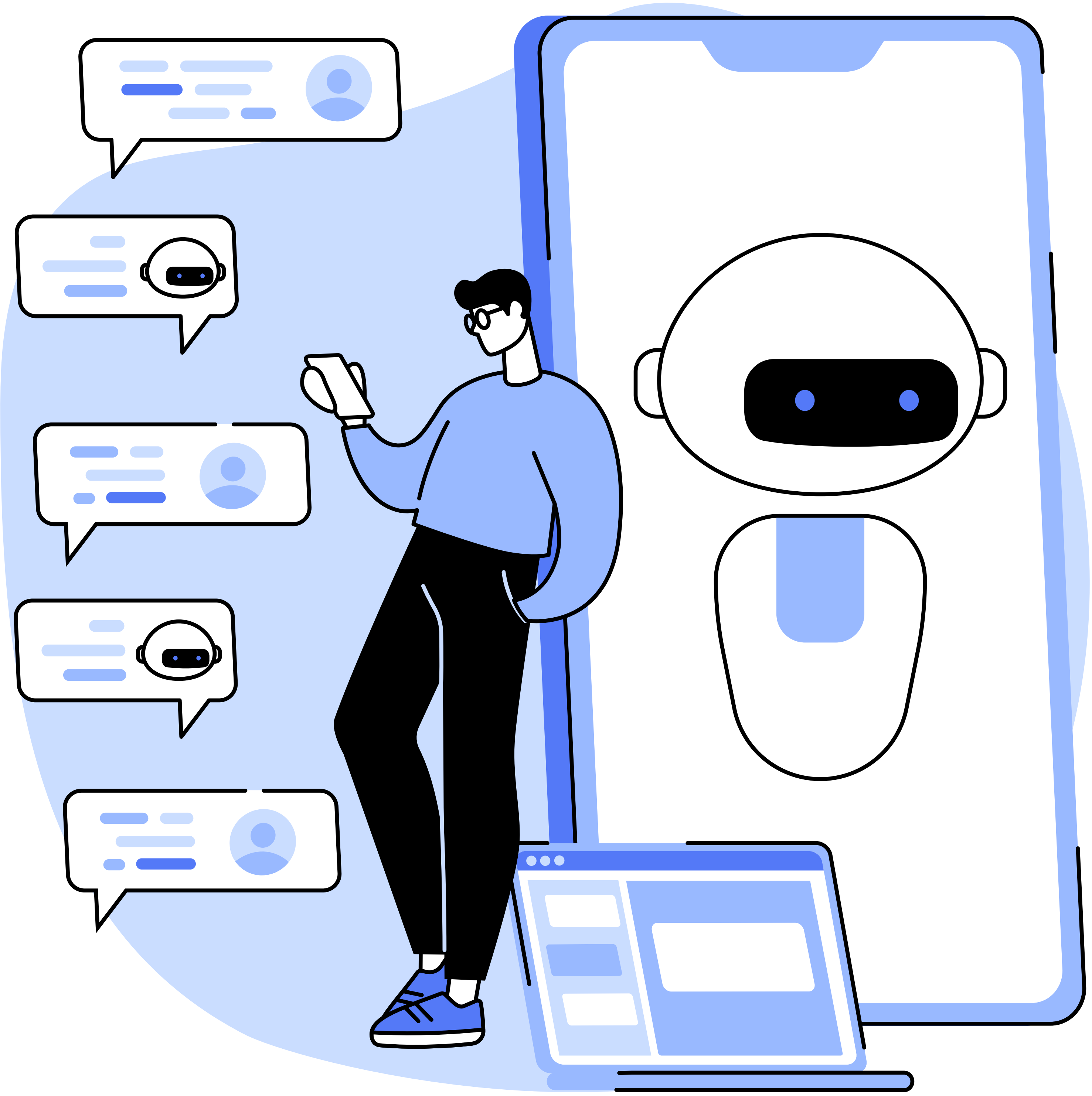
Course availibility

Credit Constraints

Student Preferences

ALGORITHM SELECTION

We agree to choose the **graph algorithms**, specifically **topological sorting**, as the solution paradigm for our Course Scheduling problem. Topological sorting is well-suited for this problem because it effectively handles prerequisites and credit constraints.



IMPLEMENTATION OVERVIEW

```python
    def create_top_sort(self):
        # Perform topological sorting using Kahn's algorithm
        in_degree = {node: 0 for node in self.course_graph.nodes}
        queue = []

        # Calculate in-degree for each node
        for node in self.course_graph.nodes:
            for neighbor in self.course_graph.neighbors(node):
                in_degree[neighbor] += 1

        # Enqueue nodes with in-degree 0
        for node in in_degree:
            if in_degree[node] == 0:
                queue.append(node)

        # Perform topological sorting
        sorted_nodes = []
        while queue:
            node = queue.pop(0)
            sorted_nodes.append(node)

            for neighbor in self.course_graph.neighbors(node):
                in_degree[neighbor] -= 1
                if in_degree[neighbor] == 0:
                    queue.append(neighbor)

        # Create a new graph with sorted nodes and edges
        new_graph = nx.DiGraph()
        new_graph.add_nodes_from(sorted_nodes)
        new_graph.add_edges_from(self.course_graph.edges())

        # Copy node data from the original graph to the sorted graph
        nodes = new_graph.nodes
        nodes_old = self.course_graph.nodes
        for course in nodes:
            nodes[course]["data"] = nodes_old[course]["data"]

        self.top_graph = new_graph
```

ROLE OF TOPOLOGICAL SORTING IN COURSE SCHEDULING

Resolving dependencies	**Establishing a feasible schedule**	**Handling cyclic dependencies**	**Generating a sorted graph**

CODE DEMONSTRATION

Click this GitHub Repository link

CORRECTNESS

Creating the graph

The algorithm constructs a directed graph by iterating through the courses and their prerequisites. It accurately represents the dependencies between courses.

Topological sorting

Using Kahn's algorithm, the algorithm performs topological sorting on the course graph. It calculates the in-degree of each node, enqueues nodes with an in-degree of 0, and processes the queue to generate a valid order of courses to satisfy prerequisites.

Creating the schedule

The algorithm generates a course schedule based on the sorted graph. It considers the prerequisites and credits for each course, assigns courses to semesters, and creates a schedule representing years and semesters.

OVERALL, THIS ALGORITHM SUCCESSFULLY CREATES THE COURSE GRAPH, PERFORMS TOPOLOGICAL SORTING, AND GENERATES A VALID COURSE SCHEDULE CONSIDERING PREREQUISITES, DESIRED CREDITS, AND STARTING SEMESTER.

TIME COMPLEXITY

Creating the graph

This step takes O(V + E) time, where V is the number of courses and E is the number of prerequisites

Topological sorting

The topological sorting algorithm used is Kahn's algorithm, which has a time complexity of O(V + E)

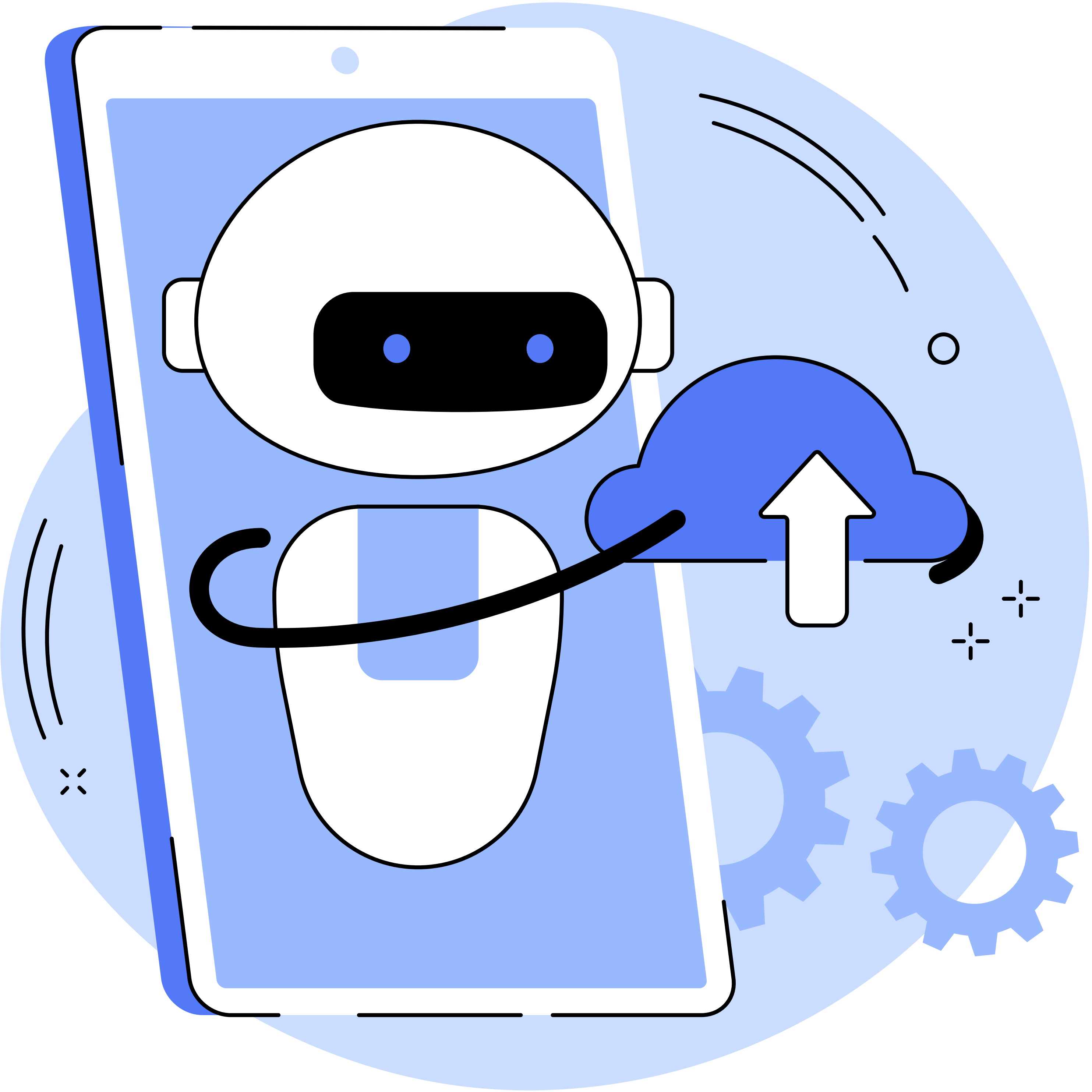
Creating the schedule

In the worst case, it may iterate over all the courses and perform checks for each prerequisite and semester. Thus, the time complexity is dependent on the number of courses and their attributes.

IN SUMMARY, THE ALGORITHM HAS A TIME COMPLEXITY OF O(V + E), WHERE V IS THE NUMBER OF COURSES AND E IS THE NUMBER OF PREREQUISITES

CONCLUSION

- The optimization of university course scheduling is a crucial task that requires careful consideration of various factors
- Manual scheduling processes are prone to errors
- Graph algorithms, specifically topological sorting, provide a suitable solution paradigm for the course scheduling problem
- Topological sorting ensures a valid order for course scheduling



THANK YOU FOR LISTENING!