# GIKab – Smart On-Campus Ride Management System using Data Structures in C++

**Submitted by:**

**Omer Imran Siddiqui (2024523)**

**Syed Muhamad Ali Akbar Shah (2024615)**

**Muhammad Danish (2024372)**

**1. Problem Statement**

At GIKI, students often face difficulties when **finding rides within the campus** — whether it’s traveling between hostels, academic blocks, or cafeterias.  
Currently, there is **no digital or organized system** for handling short-distance ride requests. Most coordination happens through word-of-mouth or informal group chats, leading to confusion, lack of accountability, and inefficiency.To address this, **GIKab** aims to **digitalize the campus ride process** by simulating a ride management system built entirely using **Data Structures and Algorithms (DSA)** in C++.  
The system models a simplified version of a ride-booking platform, allowing users to register as riders or drivers, store their ride history, and manage booking requests using a **First-In-First-Out (FIFO)** queue mechanism.Beyond solving a real student-life problem, the project also provides a **practical application of DSA concepts** taught during the course — including arrays, pointers, linked lists, queues, sorting, and file handling — all integrated into one cohesive system.

**2. Objectives**

* To design a **functional prototype** of a digital ride management system for the GIKI campus.
* To apply and demonstrate **fundamental DSA concepts** (arrays, pointers, linked lists, queues, searching, sorting, file handling).
* To implement **persistent data storage** using C++ file handling.
* To develop a **console-based application** that reflects the workflow of a real ride-booking system.
* To establish a foundation for further enhancements in the final deliverable, including graphs, trees, hashing, and GUI integration.

**3. Scope**

This deliverable represents the **initial phase** of the complete project (Deliverable 2).  
It focuses on the **core logic and data structure implementation**, not the full feature set.

**Current Scope Includes:**

* User registration and storage using arrays and pointers.
* Ride history management via linked lists.
* Ride request queuing using a FIFO linked list structure.
* File I/O for saving and reloading data persistently.

**Future Scope (Final Deliverable):**

* Route optimization using **graph algorithms (Dijkstra/BFS/DFS)**.
* Ride prioritization using **heaps/priority queues**.
* User login system with **hashing-based authentication**.
* GUI-based application using **Qt Framework**.

**4. Data Structures Used**

| **Concept** | **Data Structure** | **Application** |
| --- | --- | --- |
| **Arrays + Pointers** | User\* users[MAX\_USERS] | Stores all registered users (drivers and riders). |
| **Linear & Binary Search** | Iterative algorithms | Efficiently locate users by ID. |
| **Insertion Sort** | Sorting algorithm | Sorts users by ID before binary search. |
| **File Handling** | Text file (users.txt) | Persistent storage of user data. |
| **Singly & Doubly Linked List** | Ride structure | Manage and traverse each rider’s ride history. |
| **Circular Linked List** | Tail connects to head | Demonstrates continuous traversal of ride history. |
| **Queue (Linked List)** | FIFO structure | Handles ride requests in order received. |

**5. System Architecture**

**Module 1 – User Management System**

* Implements arrays, pointers, and searching.
* Supports adding, viewing, and sorting users.
* Uses **file handling** for persistent storage.

**Module 2 – Ride History Management**

* Uses **linked lists** to store ride details per user.
* Allows forward and backward traversal of rides.
* Circular traversal is demonstrated to show looped connections.

**Module 3 – Ride Request Queue**

* Implements **linked list–based FIFO queue**.
* New ride requests are added (enqueue).
* Assigned rides are removed (dequeue).

**6. Implementation Details**

* **Language:** C++
* **Program Type:** Console-based application
* **File I/O:** ifstream / ofstream
* **Data File:** users.txt
* **Structure:** Single modular .cpp file
* **Persistence:** Data auto-loads on startup and auto-saves on exit

**Main Menu Options:**

1. Add User

2. View All Users

3. Search User (Linear Search)

4. Sort + Binary Search

5. Add Ride (History)

6. Show Ride History (Forward)

7. Show Ride History (Backward)

8. Show Ride History (Circular)

9. Add Ride Request (Enqueue)

10. Assign Next Ride (Dequeue)

11. View Pending Ride Requests

0. Exit

**7. Expected Outcomes**

* A **working console prototype** that models a real-world ride system.
* Demonstration of DSA principles in a practical use case.
* Persistent user data storage.
* Smooth menu-driven interaction between modules.
* A clear foundation for implementing advanced algorithms in the final version.

**8. Limitations**

* No GUI (text-based interface only).
* No real authentication or password system.
* Route optimization and nearest-driver allocation not yet implemented.
* User data stored in a simple text file (not a database).

**9. Future Enhancements (for Final Deliverable)**

| **Week** | **Feature** | **DSA Concept** |
| --- | --- | --- |
| 8–9 | Advanced Sorting (Quick, Merge, Heap) | Improved user management |
| 10–11 | Trees & AVL Trees | Ride categorization and storage optimization |
| 12–13 | Graphs | Campus map routes and shortest path finding |
| 13–14 | Hashing | Secure login & ID mapping |
| 15 | Priority Queues (Heaps) | Driver allocation based on proximity |
| Final | GUI + CSV Export | Complete interactive system |

**10. Conclusion**

The **GIKab** project successfully demonstrates the integration of multiple DSA concepts into a **realistic, problem-solving system**.  
It provides a **structured, academic, and practical solution** to the lack of an on-campus ride coordination system while solidifying core C++ DSA concepts.

This deliverable (version 1) lays the **foundation** for the final system, which will incorporate advanced data structures, algorithms, and graphical interfaces for a complete, real-world ride management experience.