## A

## PROJECT REPORT

## ON

**“SCHOOL BUS MANAGEMENT SYSTEM”**

* **SUBMITTED BY:-**

**Sarthak Arun Jadhav**

**PRN:- 2124UCEM1073**

* **SUBJECT:-**

**Programming and**

**problem solving using c++**

# **Under the guidance of**

**Miss. Ishwari Tirse**



**Department of Computer Science and Engineering**

**Sanjivani Rural Education Society’s**

**SANJIVANI UNIVERSITY**

## KOPARGAON – 423603, DIST : AHMEDNAGAR

## 2024-2025

|  |  |  |
| --- | --- | --- |
| **SR.**  **NO** | **CONTENT** | **PAGE NO.** |
| **1.** | **INTRODUCTION** | **3 -4** |
| **2.** | **CODE** | **5-10** |
| **3.** | **OUTPUT** | **11-12** |
| **4.** | **CONCLUSION** | **13** |

**INDEX**

**INTRODUCTION**

**Here's a C++ code introduction**

**for a School Bus Management System:-**

This C++ program is designed to efficiently manage a school bus system. It offers a comprehensive solution to streamline various operations, **Including:**

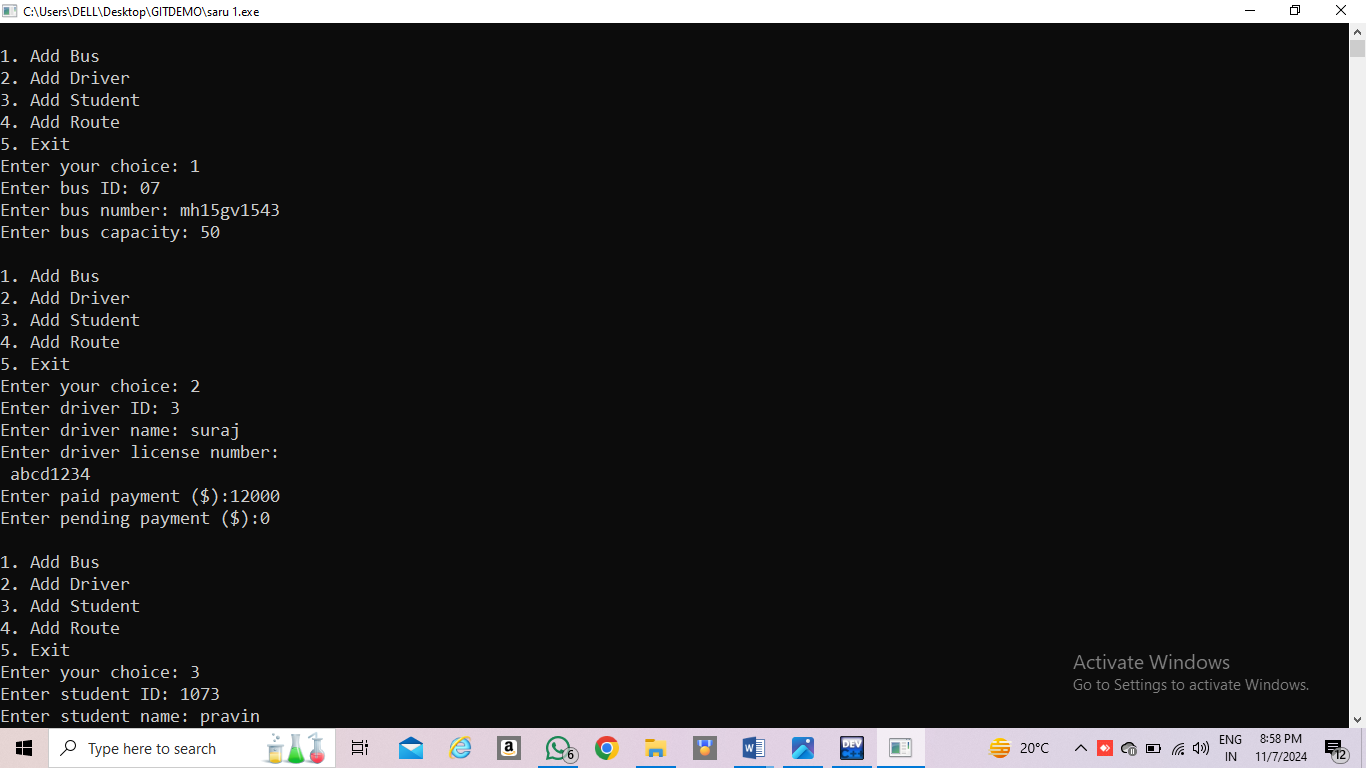
* **Vehicle Management:-**
* Add, modify, and delete bus details

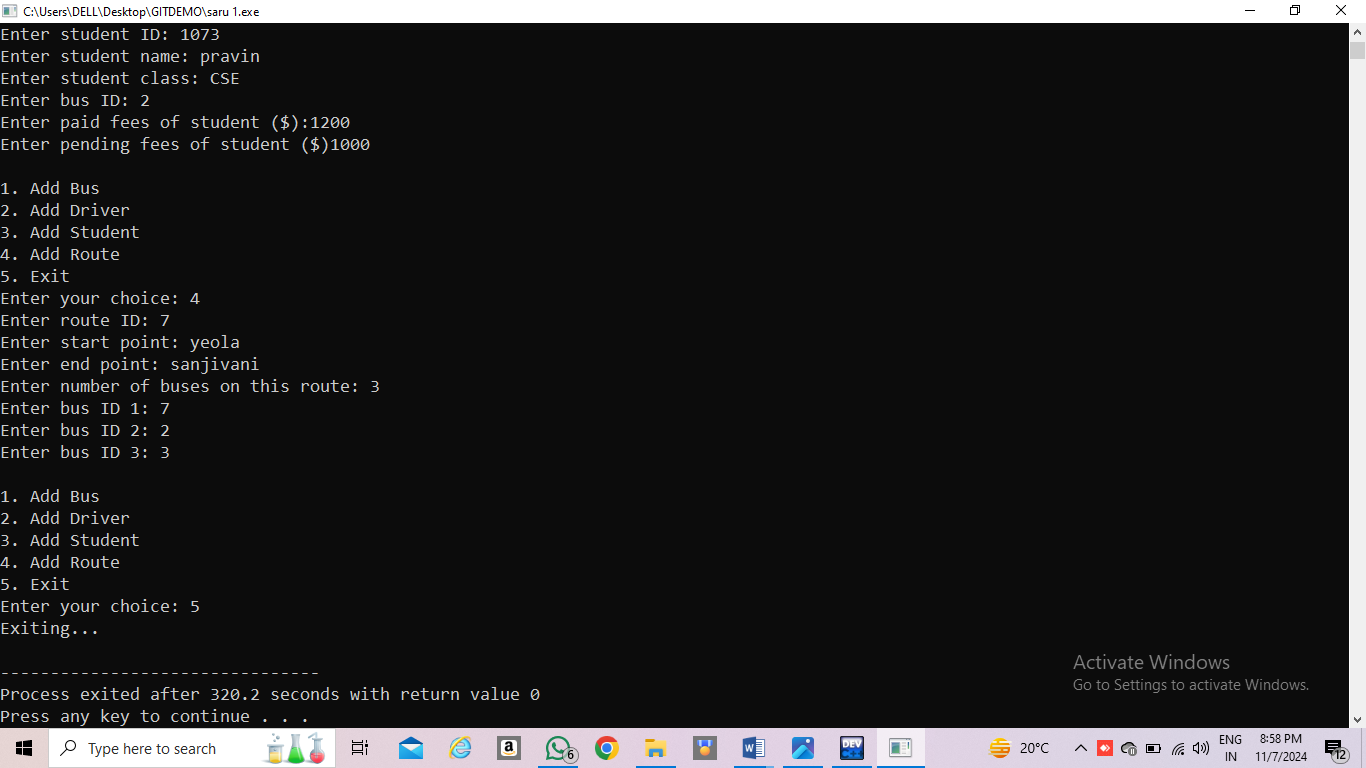
(e.g., bus number, capacity,route).

* Track vehicle maintenance schedules and history.
* **Driver Management**:-
* Add, modify, and delete driver information (e.g., name, license number, contact details).
* Assign drivers to specific routes and buses.
* **Student Management:-**
* Add, modify, and delete student details (e.g., name, class, parent contact information).
* Assign students to specific routes and buses.
* **Route Management:-**
  + Create, modify, and delete routes (e.g., pickup and drop-off points, timings).
  + Optimize routes to minimize travel time and fuel consumption.
  + **Fee Management:-**
  + Track and manage bus fees for each student.
  + Generate fee receipts and reports.
* **Key Features:**
* User-Friendly Interface: A simple and intuitive interface for easy navigation.
* Data Validation: Robust input validation to ensure data integrity.
* Error Handling: Efficient error handling to prevent unexpected program termination.
* Data Persistence: Store and retrieve data using files or a database.
* Security: Implement basic security measures to protect sensitive information.
* Reporting: Generate various reports (e.g., student attendance, fee collection, vehicle maintenance).

|  |  |
| --- | --- |
| **CODE**  #include <iostream>  #include <vector>  #include <string>  using namespace std;  struct Bus {  int bus\_id;  string bus\_no;  int capacity;  };  struct Driver {  int driver\_id;  string name;  string license\_no;  double payment\_paid,payment\_pending;  };  struct Student {  int student\_id;  string name;  string class\_name;    int bus\_id;  double bus\_fees\_paid,bus\_fees\_pending;  };  struct Route {  int route\_id;  string start\_point;  string end\_point;  vector<int> bus\_ids;  };  void addBus(vector<Bus>& buses) {  Bus bus;  cout << "Enter bus ID: ";  cin >> bus.bus\_id;  cout << "Enter bus number: ";  cin >> bus.bus\_no;  cout << "Enter bus capacity: ";  cin >> bus.capacity;  buses.push\_back(bus);  }  void addDriver(vector<Driver>& drivers) {  Driver driver;  cout << "Enter driver ID: ";  cin >> driver.driver\_id;  cout << "Enter driver name: ";  cin >> driver.name;  cout << "Enter driver license number: ";  cin >> driver.license\_no;  cout<<"Enter paid payment ($):";  cin>>driver.payment\_paid;  cout<<"Enter pending payment ($):";  cin>>driver.payment\_pending;  drivers.push\_back(driver);  }  void addStudent(vector<Student>& students) {  Student student;  cout << "Enter student ID: ";  cin >> student.student\_id;  cout << "Enter student name: ";  cin >> student.name;  cout << "Enter student class: ";  cin >> student.class\_name;  cout << "Enter bus ID: ";  cin >> student.bus\_id;  cout<<"Enter paid fees of student ($):";    cin>>student.bus\_fees\_paid;  cout<<"Enter pending fees of student ($)";  cin>>student.bus\_fees\_pending;  students.push\_back(student);  }  void addRoute(vector<Route>& routes) {  Route route;  cout << "Enter route ID: ";  cin >> route.route\_id;  cout << "Enter start point: ";  cin >> route.start\_point;  cout << "Enter end point: ";  cin >> route.end\_point;  int num\_buses;  cout << "Enter number of buses on this route: ";  cin >> num\_buses;  for (int i = 0; i < num\_buses; ++i) {  int bus\_id;  cout << "Enter bus ID " << i + 1 << ": ";  cin >> bus\_id;  route.bus\_ids.push\_back(bus\_id);  }  routes.push\_back(route);  }  int main() {  vector<Bus> buses;  vector<Driver> drivers;  vector<Student> students;  vector<Route> routes;  int choice;  do {  cout << "\n1. Add Bus\n2. Add Driver\n3. Add Student\n4. Add Route\n5. Exit\n";  cout << "Enter your choice: ";  cin >> choice;  switch (choice) {  case 1:  addBus(buses);  break;  case 2:  addDriver(drivers);  break;  case 3:  addStudent(students);    break;  case 4:  addRoute(routes);  break;  case 5:  cout << "Exiting...\n";  break;  default:  cout << "Invalid choice.\n";  }  } while (choice != 5);  return 0;  } | 4 |

**OUTPUT**



****

**CONCLUSION**

This C++ program provides a comprehensive solution for managing school bus operations. By automating various tasks, such as vehicle maintenance, driver scheduling, student allocation, and route optimization, it significantly improves efficiency and reduces administrative overhead.

This system offers a user-friendly interface for easy navigation and data input. Robust data validation and error handling mechanisms ensure data integrity and prevent system failures. The implementation of security measures safeguards sensitive information.

By effectively managing school bus operations, this system contributes to a safer and more reliable transportation system for students. It empowers school administrators to make informed decisions and optimize resource allocation.

**Thank**

**You !!**