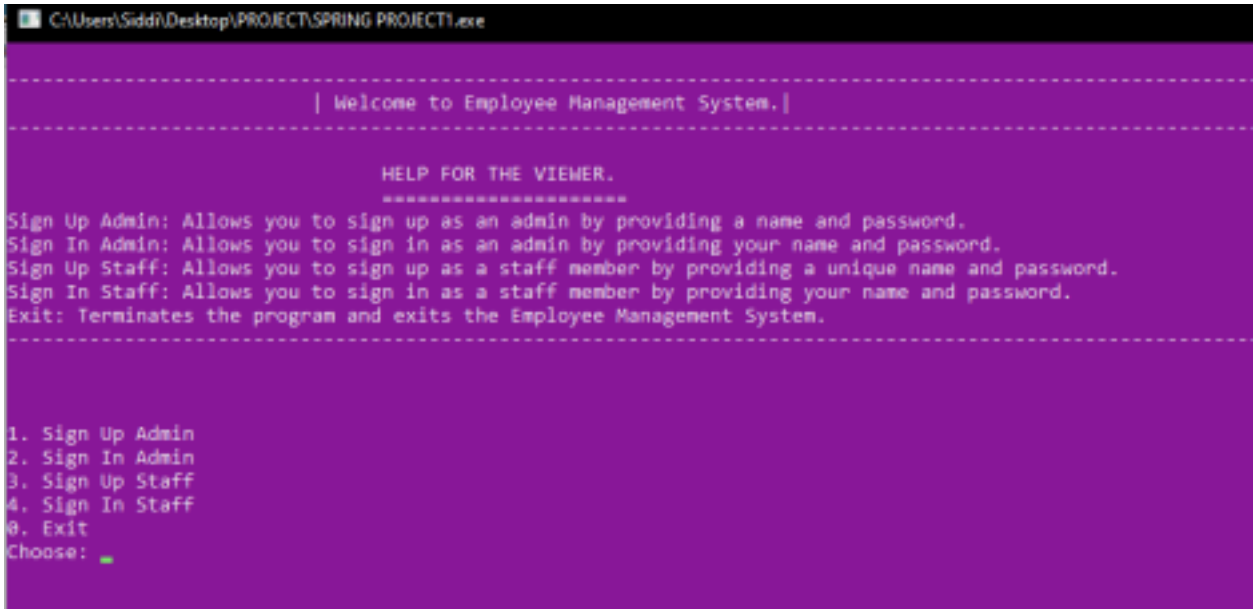
EMPLOYEE MANAGEMENT SYSTEM

OUTPUT:



PROJECT DESCRIPTION:

The Employee Management System in C++ aims to create a program that enables efficient management of employee records. The code aims to achieve the following objectives:

• **Record Management**: The system allows for the creation, storage, and management of employee records. It provides functionality to add new employee records, remove existing records, modify record details, search for specific records, and list all records.

• **Object-Oriented Approach:** The code demonstrates the use of object-oriented programming (OOP) principles to structure the system. It utilizes inheritance, polymorphism, and templates to create classes and objects that represent different entities and facilitate code reuse and extensibility.

• **User Authentication:** The program includes a user authentication system to ensure secure access to the system. It allows administrators and staff members to sign up and sign in with their respective accounts, providing appropriate levels of access and functionality based on the user's role.

• **Separation of Concerns:** The code separates the responsibilities of different entities within the system. It distinguishes between administrators and staff members, assigning them different privileges and functionalities. This segregation ensures that the system remains organized and manageable.

• **Flexibility and Extensibility:** The use of templates in the `Database` class allows for

the management of records of any type. This makes the system adaptable to different types of employee records or potentially other types of records in the future.

PRINCIPLES OF OOP USED:

In this Employee Management System, several principles of object-oriented programming (OOP) have been employed. Here is a justification of how these principles are utilized:

• **Encapsulation:** Encapsulation is demonstrated through the use of classes, which encapsulate data and methods related to specific entities. For example, the Person, Employee, Database, Account, Admin, Staff, and EmployeeManagementSystem classes encapsulate the relevant attributes and behaviors of persons, students, databases, accounts, and the system itself. The class members are appropriately defined as private, protected, or public to enforce encapsulation and ensure data integrity.

• **Inheritance:** The code exhibits inheritance by implementing class hierarchies. The Employee class inherits from the Person class, acquiring its attributes and methods. Inheritance allows for code reuse and enables specialization and differentiation of classes. The Admin and Staff classes inherit from the Account class, inheriting its common attributes and methods while providing additional functionality specific to administrators and staff members.

• **Polymorphism:** Polymorphism is utilized through method overriding in the Student class, where the inputData() and printData() methods are overridden to include student specific attributes. The polymorphic behavior allows the program to work with objects of different classes through a common interface (Person), invoking the appropriate methods based on the object's actual type.

• **Abstraction:** Abstraction is achieved by defining classes with abstracted attributes and behaviors that represent real-world entities. For instance, the Person class represents a generic person with essential attributes such as name, age, and contact information. The Database class abstracts the storage and management of records, while the Account class abstracts the concept of user accounts.