Employee Management System

Employee Information and Security

Session 2025 - 2029

Submitted by:

Uswa Nawaz 2025-CS-312

Supervised by:

Miss Hafsa

Course:

CSC-102 Programming Fundamentals

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Short Description

This project is a console-based Employee Management & security Interface designed to replace manual record-keeping using C++. The purpose of this system is to efficiently manage employee records by allowing administrator to add, search, update, and delete employee information. Manual record-keeping is slow and prone to errors, so this system is a simple computerized solution. This project applies basic programming concepts such as functions, arrays, loops, and file handling. At the end of the project, a functional system is produced that can store and manage employee data securely.

- ## Users of the Application

The application has the following users:

1. **Administrator (Admin)**
   The administrator is responsible for managing the entire system. The admin has full control over the application. The admin can add a new employee, update employee information, delete employee records, and view employee details. The admin also manages login and access control to ensure that only authorized users can access the system.
2. **Employee**
   Employees are registered users whose information is stored in the system. They can log in using their credentials to view their own details, such as employee ID, name, department, and salary. Employees have limited access and are not allowed to modify or delete records, which helps maintain data integrity.

- ## Functional Requirements

Functional requirements according to types of users:

User Type	Functionalities	Results
Admin	Add Employee	Add a new employee
	View Employee	View all employees
	Search Employee	Search employees detail
	Delete Employee	Remove employee
	Update Employee	Update employee data
	Logout	Securely exit the system

User Type	Functionalities	Results
Employee	View ID Card	Access official identification details digitally
	View Company Roster	See the names and departments of other employees
	View salary Slip	Check Salary details and payment information

	Update security password	Update the account security password
	Logout	Safely exit the system after completing my work

- **Wireframes**

  The wireframes represent the main console screens of the Employee Management System. Since the application is console-based, the wireframes illustrate the flow of menus and user interactions. Each wireframe shows how users navigate through the system, including login, dashboards, and core functionalities for both admin and employee roles.

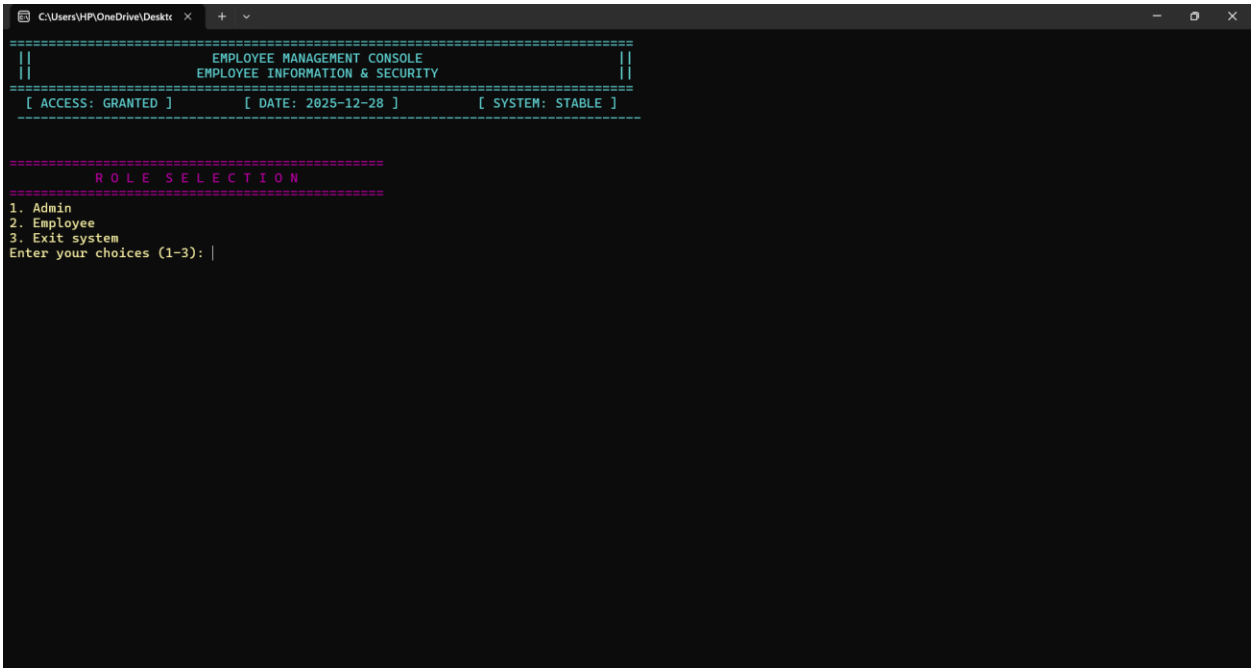


Figure 1: Start menu

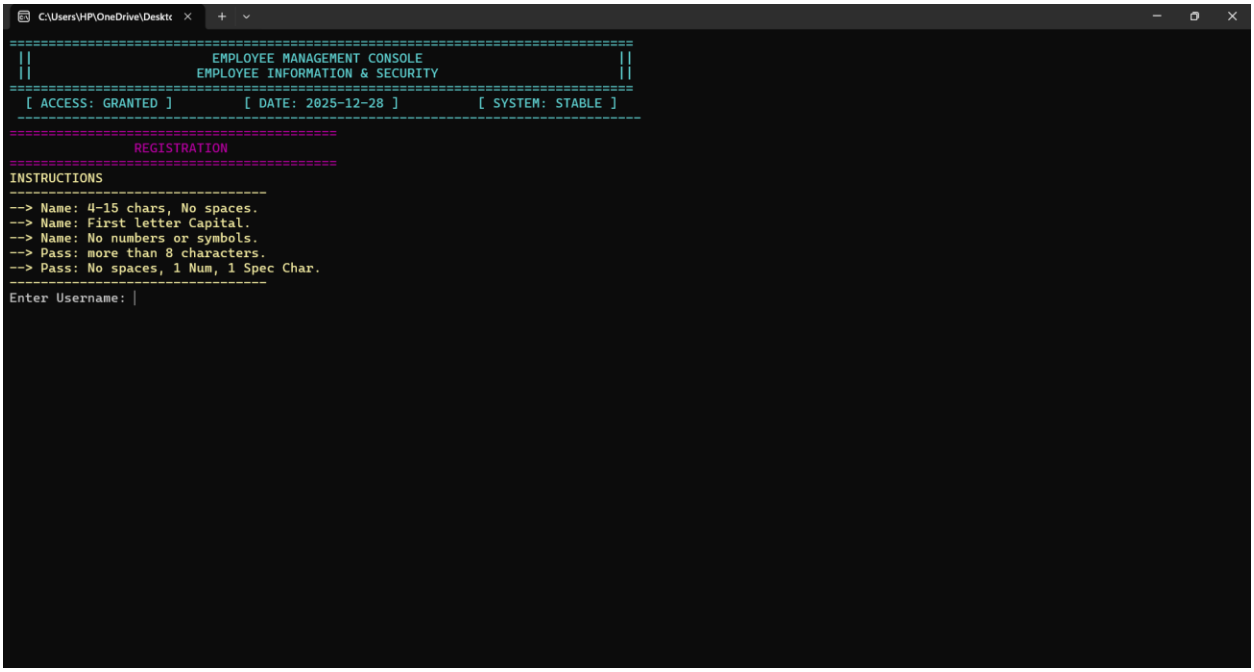


Figure 2: Admin Main Menu Screen

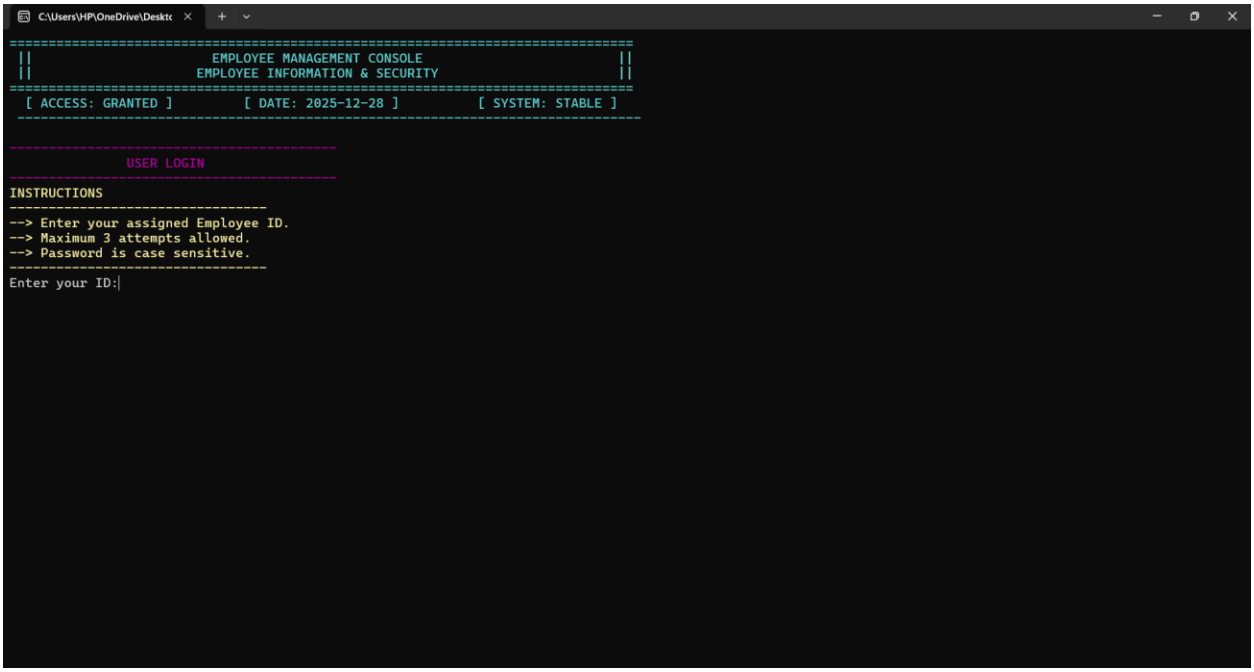


Figure 3: Employee Main Menu Screen

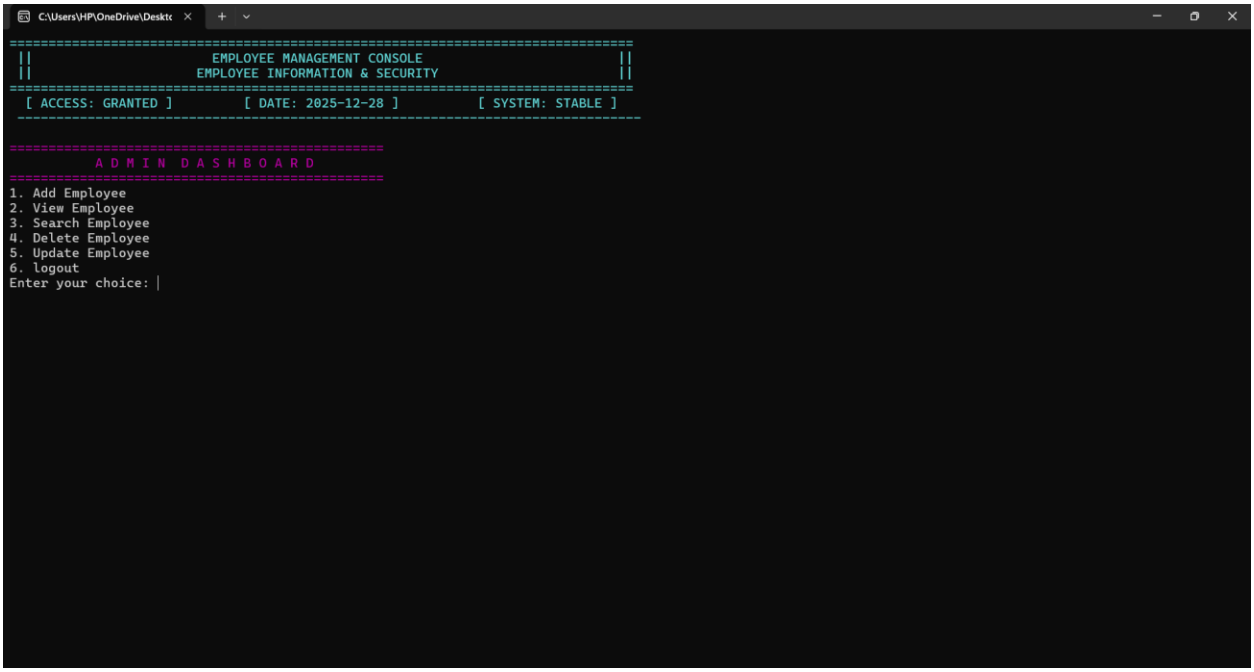


Figure 4: Admin Dashboard Menu Screen

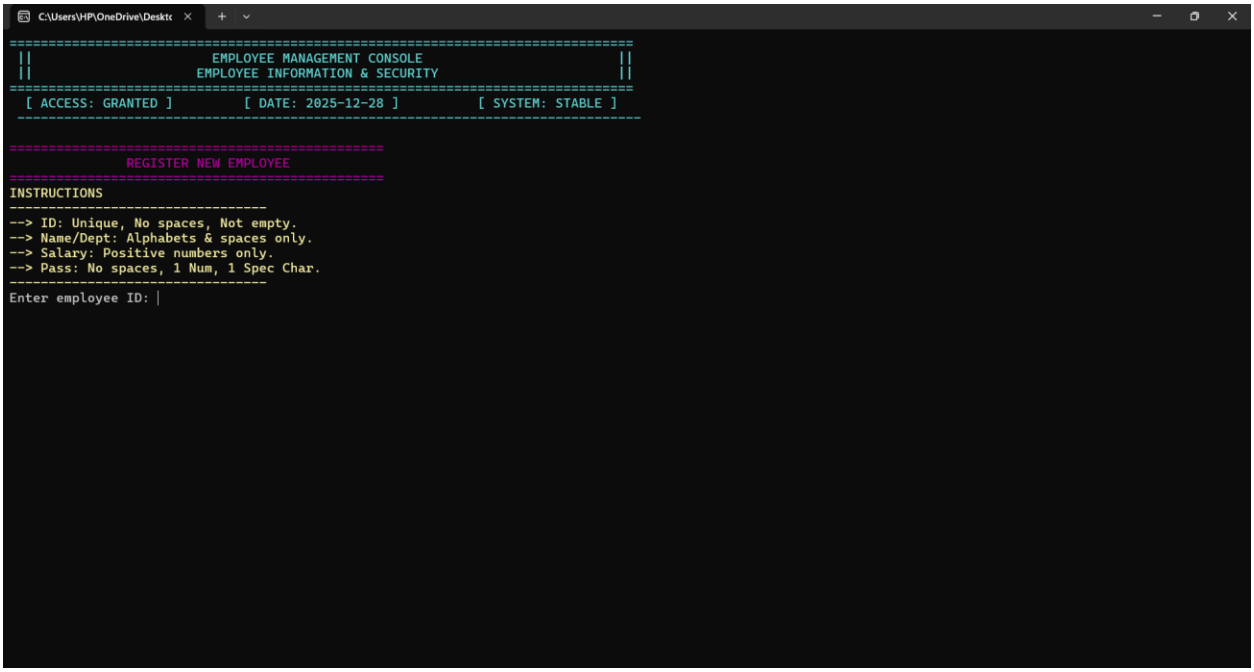


Figure 5: Add employee Screen

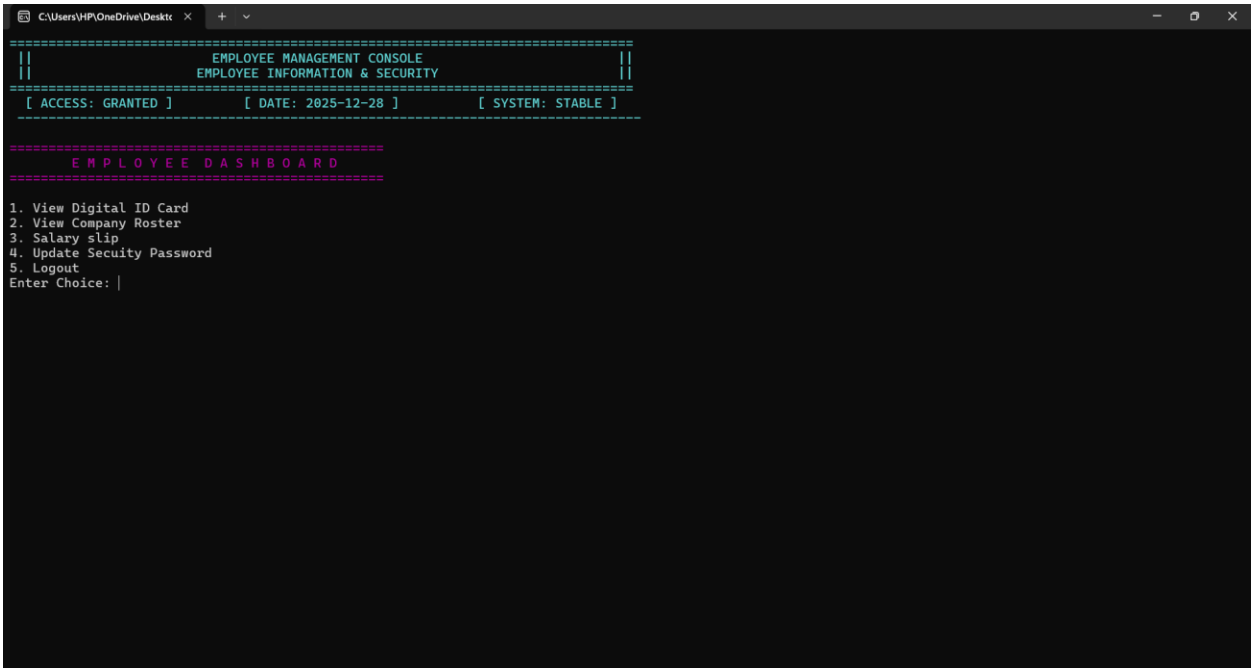


Figure 6: Employee Dashboard Screen

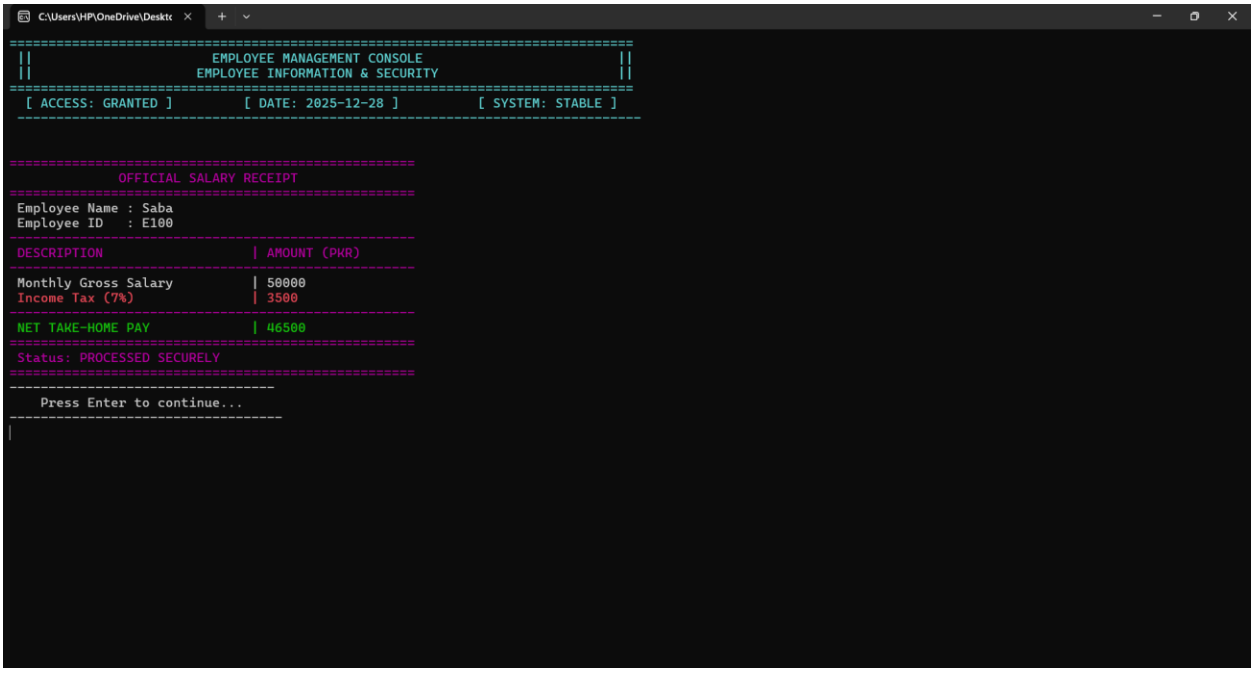


Figure 7: Employee Information Display

- **Data Structures**

```
const int MAX_EMP=50;

string empID[MAX_EMP];

string empName[MAX_EMP];

string empDepartment[MAX_EMP];

string empPassword[MAX_EMP];

int empsalary[MAX_EMP];

\int empCount=0;

string saved_Admin_UserName="";

string saved_Admin_Password="";
```

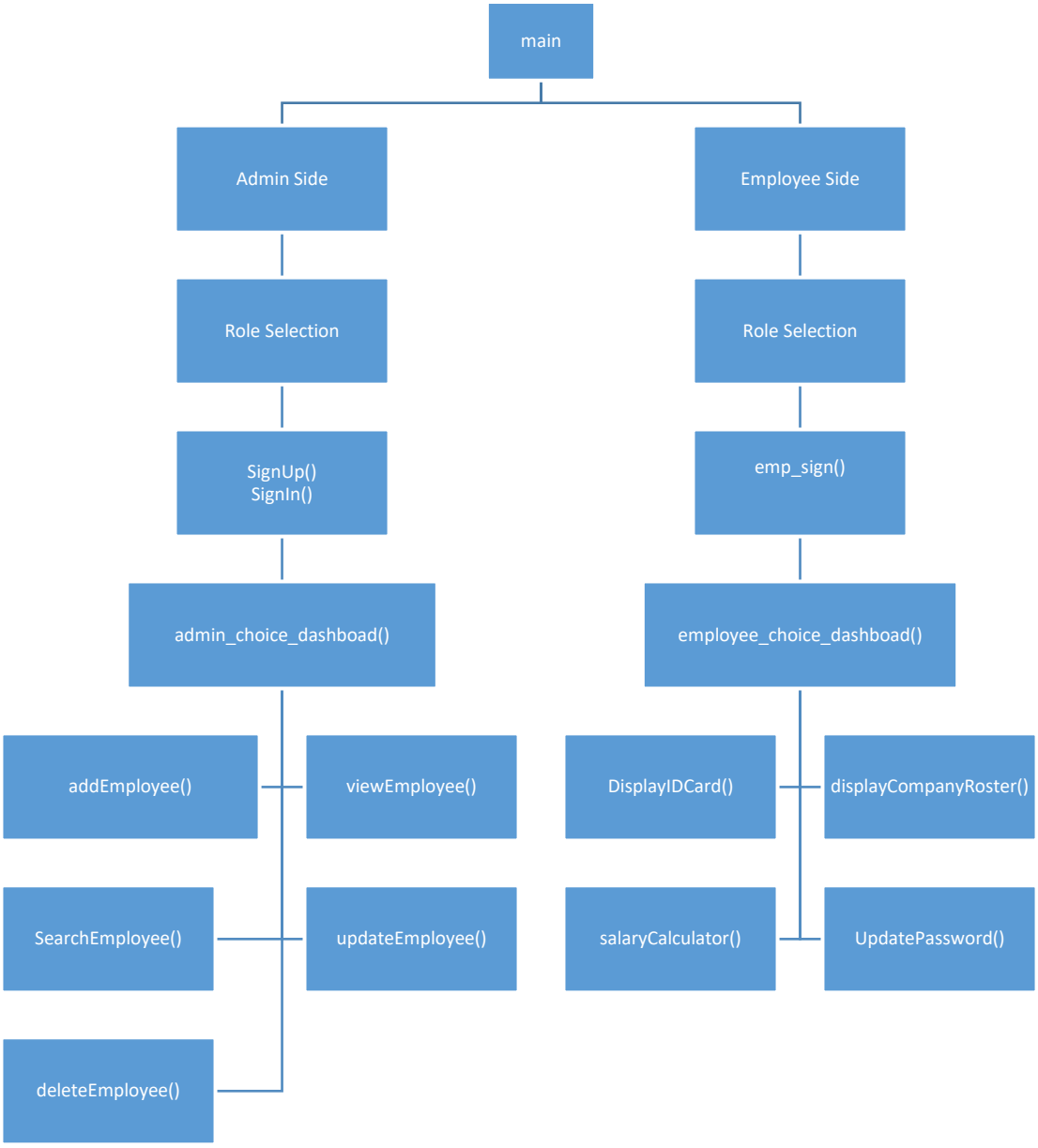
- **Function Prototypes**

```
void welcomesceen();

void signupheader();

void signup_ending_header();

void signinheader();

void admin_dashboard_header();

void employee_dashboard_header();

void emp_password_update();

void add_employee_header();

void search_employee_header();

void delete_employee_header();

void update_employee_header();

void globalHeader();
```

```
void loadingEffect(string message);

void successMessage(string message);

void errorMessage(string message);

void showInterface(string mode);  // instuctions

void setColor(int color);

void saveAdmin(string user, string pass);
void loadAdmin(string &user, string &pass);

void saveEmployee(string id[], string pass[], string name[], string dept[], int salary[], int count);

void loadEmployee(string id[], string pass[], string name[], string dept[], int salary[],int &count);

int roleSelection_and_validation();

int admin_choice_dashboard();

int employee_choice_dashboard();

bool isEmpty(string text);

bool hasSpace(string text); //for passwords

void pause();

int findEmployeeIndex(string empID[], int empCount, string targetID);

bool isStorageEmpty(int empCount);

bool userNameChecker(string u);

bool passwordChecker(string p);

void signUp(string &username,string &password);

bool signIn(string username,string password);

int emp_signIn(string empID[], string empPassword[], int empCount);
```

```
bool validempID(string id, string empID[], int empCount);

bool validName(string name);

bool validsalary(int empsalary[], int empCount);

void addEmployee(string empID[],string empPassword[], string empName[], string
empDepartment[], int empsalary[], int &empCount,int MAX_EMP);

void viewEmployee(string empID[], string empName[], string empDepartment[], int
empsalary[],int empCount);
void searchEmployee(string empID[], string empName[], string empDepartment[], int
empsalary[], int empCount);

void deleteEmployee(string empID[],string empPassword[], string empName[], string
empDepartment[], int empsalary[], int &empCount);

void updateEmployee(string empID[], string empName[], string empDepartment[], int
empsalary[], int empCount);

void displayIDCard(string empName[], string empID[], string empDepartment[], int index);

void displayCompanyRoster(string empName[], string empDepartment[], int empCount);

void salaryCalculator(string empName[], string empID[], int empsalary[], int index);

void UpdatePassword(string empID[], string empPassword[],string empName[],string
empDepartment[],int empsalary[],int empCount,int index);
```

- **Functions Working Flow**



- ## Weakness in the Business Application

  The code relies on a manual counter variable to track the number of employees. If this counter is not handled perfectly, the system could overwrite existing data or crash.

  The arrays have a pre-defined limit. Once the company reaches that number of employees, the code will fail to add more unless the source code is manually changed and recompiled.

  The "View Employee" function uses a linear search. While it works for a small list, it would be very slow and inefficient for a company with thousands of employees.

- ## Future Directions

  Advanced Data Validation: I will implement more sophisticated input validation using regular expressions to ensure that names, IDs, and salaries follow strict formats, preventing system crashes from invalid user inputs

  Graphical User Interface (GUI): I plan to move beyond the command-line interface and develop a GUI to provide a more modern and user-friendly experience

  Object-Oriented Programming (OOP): I will transition from using parallel arrays to using **Classes** and **Objects**. This will allow for better data encapsulation and make the code much more organized and professional