```
import numpy as np
from datetime import datetime

class UserAuthentication:
    """Class for managing user authentication (login and signup)."""

    def __init__(self):
        self.users = {}
        self.default_password = "456"

    def signup(self):
        user_id = input("Enter User ID: ")
        if user_id in self.users:
            print("User ID already exists. Please log in or choose a different User ID.")
            return

        password = input("Enter Password: ")
        confirm_password = input("Confirm Password: ")

        if password != confirm_password:
            print("Passwords do not match. Please try again.")
            return

        self.users[user_id] = password
        print("Signup successful! You can now log in.")

    def login(self):
        user_id = input("Enter User ID: ")
        password = input("Enter Password: ")

        if user_id in self.users and self.users[user_id] == password:
            print("Login successful! Welcome!")
            return True

        print("Invalid User ID or Password. Please try again.")
        return False


class Employee:
    def __init__(self, emp_id, name, age, position, salary):
        self.emp_id = emp_id
        self.name = name
        self.age = age
        self.position = position
```

```
        self.salary = salary
        self.start_year = datetime.now().year

    def calculate_bonus(self):
        """Base method to calculate bonus, overridden in subclasses."""
        return self.salary * 0.1

    def display_bonus(self):
        """Display the bonus for an employee."""
        bonus = self.calculate_bonus()
        print(f"Bonus for {self.name} (Position: {self.position}): {bonus}")
        return bonus

    def yearly_increment(self):
        """Increases the salary by 10%."""
        self.salary *= 1.10

    def salary_deduction_for_absence(self, absent_days, working_days=30):
        """Deducts salary based on the number of absent days."""
        if absent_days < 0:
            print("Absent days cannot be negative.")
            return
        if absent_days > working_days:
            print("Absent days cannot exceed total working days in a month.")
            return
        per_day_salary = self.salary / working_days
        deduction = per_day_salary * absent_days
        self.salary -= deduction
        print(f"Salary deducted: {deduction}. New salary: {self.salary}")

    def display(self):
        return f"ID: {self.emp_id}, Name: {self.name}, Position: {self.position}, Salary: {self.salary}"

    def working_years(self):
        """Calculates the number of years the employee has been working."""
        current_year = datetime.now().year
        return current_year - self.start_year

    def salary_after_increment(self):
        """Returns the salary after applying the yearly increment."""
        return self.salary * 1.10


class Manager(Employee):
```

```
    def __init__(self, emp_id, name, age):
        super().__init__(emp_id, name, age, "Manager", 50000)

    def calculate_bonus(self):
        return self.salary * 0.2


class Engineer(Employee):
    def __init__(self, emp_id, name, age):
        super().__init__(emp_id, name, age, "Engineer", 30000)

    def calculate_bonus(self):
        return self.salary * 0.15


class Intern(Employee):
    def __init__(self, emp_id, name, age):
        super().__init__(emp_id, name, age, "Intern", 15000)

    def calculate_bonus(self):
        return 500

class EmployeeManagementSystem:
    def __init__(self):
        self.employees = {}
        self.position_salary = {
            "Manager": 50000,
            "Engineer": 30000,
            "Intern": 15000
        }

    def add_employee(self, employee):
        if employee.emp_id in self.employees:
            raise ValueError("Employee ID already exists.")
        self.employees[employee.emp_id] = employee

    def remove_employee(self, emp_id):
        if emp_id not in self.employees:
            raise KeyError("Employee ID not found.")
        del self.employees[emp_id]

    def search_employee(self, emp_id):
        """Searches for an employee by ID and displays details including working years, future
salary, and bonus."""
```

```
        if emp_id in self.employees:
            emp = self.employees[emp_id]
            working_years = emp.working_years()
            future_salary = emp.salary_after_increment()
            bonus = emp.display_bonus()
            print(f"Employee Details:\n{emp.display()}")
            print(f"Working Years: {working_years} years")
            print(f"Salary Next Year (After Increment): {future_salary}")
            print(f"Bonus for Next Year: {bonus}")
        else:
            print("Employee not found.")

    def search_by_position(self, position):
        """Search employees by their position and show the salary for that position."""
        employees_in_position = [emp for emp in self.employees.values() if emp.position.lower()
== position.lower()]
        if employees_in_position:
            print(f"Employees in {position} position (Salary: {self.position_salary.get(position,
'N/A')}):")
            for emp in employees_in_position:
                print(f"{emp.name} (ID: {emp.emp_id})")
        else:
            print(f"No employees found in {position} position.")

    def apply_yearly_increment(self):
        """Applies a 10% salary increment to all employees."""
        for emp in self.employees.values():
            emp.yearly_increment()
        print("Yearly increment applied to all employees.")

    def apply_salary_deduction_for_absence(self):
        """Applies salary deduction based on employee's absence."""
        emp_id = int(input("Enter employee ID for deduction: "))
        if emp_id in self.employees:
            employee = self.employees[emp_id]
            absent_days = int(input(f"Enter the number of absent days for {employee.name}: "))
            working_days = int(input("Enter total working days in the month (default is 30): ") or 30)
            employee.salary_deduction_for_absence(absent_days, working_days)
        else:
            print("Employee not found.")

    def view_all_employees(self):
        if not self.employees:
            print("No employees found.")
```

```
        else:
            for emp in self.employees.values():
                print(emp.display())

    def calculate_payroll(self):
        salaries = [emp.salary for emp in self.employees.values()]
        total_payroll = np.sum(salaries)
        avg_salary = np.mean(salaries) if salaries else 0
        return total_payroll, avg_salary

    def recent_employees(self, count=3):
        ids = list(self.employees.keys())[-count:]
        return [self.employees[i] for i in ids]


def main():
    auth_system = UserAuthentication()
    emp_system = EmployeeManagementSystem()

    print("Welcome to the Employee Management System!")

    while True:
        print("\n1. Signup")
        print("2. Login")
        print("3. Exit")

        try:
            choice = int(input("Choose an option: "))

            if choice == 1:
                auth_system.signup()

            elif choice == 2:
                if auth_system.login():
                    while True:
                        print("\n1. Add Employee")
                        print("2. Remove Employee")
                        print("3. Search Employee by ID")
                        print("4. Search Employees by Position")
                        print("5. View All Employees")
                        print("6. Calculate Payroll")
                        print("7. Apply Yearly Increment")
                        print("8. Apply Salary Deduction for Absence")
                        print("9. View Recent Employees")
```

```
print("10. Logout")

sub_choice = int(input("Choose an option: "))

if sub_choice == 1:
    emp_id = int(input("Enter ID: "))
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    position = input("Enter position (Manager/Engineer/Intern): ")

    if position.lower() == "manager":
        employee = Manager(emp_id, name, age)
    elif position.lower() == "engineer":
        employee = Engineer(emp_id, name, age)
    elif position.lower() == "intern":
        employee = Intern(emp_id, name, age)
    else:
        raise ValueError("Invalid position.")

    emp_system.add_employee(employee)
    print("Employee added successfully.")

elif sub_choice == 2:
    emp_id = int(input("Enter employee ID to remove: "))
    emp_system.remove_employee(emp_id)
    print("Employee removed successfully.")

elif sub_choice == 3:
    emp_id = int(input("Enter employee ID to search: "))
    emp_system.search_employee(emp_id)

elif sub_choice == 4:
    position = input("Enter position to search (Manager/Engineer/Intern): ")
    emp_system.search_by_position(position)

elif sub_choice == 5:
    print("All Employees:")
    emp_system.view_all_employees()

elif sub_choice == 6:
    total_payroll, avg_salary = emp_system.calculate_payroll()
    print(f"Total Payroll: {total_payroll}, Average Salary: {avg_salary}")

elif sub_choice == 7:
```

```
                    emp_system.apply_yearly_increment()

                elif sub_choice == 8:
                    emp_system.apply_salary_deduction_for_absence()

                elif sub_choice == 9:
                    count = int(input("Enter number of recent employees to view: "))
                    recent_emps = emp_system.recent_employees(count)
                    print("Recent Employees:")
                    for emp in recent_emps:
                        print(emp.display())

                elif sub_choice == 10:
                    print("Logged out successfully!")
                    break

                else:
                    print("Invalid choice. Please try again.")
            else:
                print("Login failed. Please try again.")

        elif choice == 3:
            print("Exiting the system. Goodbye!")
            break

        else:
            print("Invalid choice. Please try again.")

    except ValueError as ve:
        print(f"Value Error: {ve}")
    except KeyError as ke:
        print(f"Key Error: {ke}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")


if __name__ == "__main__":
    main()
```

```
Welcome to the Employee Management System!

1. Signup
2. Login
3. Exit
```

Choose an option: 1
Enter User ID: 222
Enter Password: jj
Confirm Password: jj
Signup successful! You can now log in.

1. Signup
2. Login
3. Exit
Choose an option: 2
Enter User ID: 222
Enter Password: jj
Login successful! Welcome!

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 1
Enter ID: 1
Enter name: omar
Enter age: 22
Enter position (Manager/Engineer/Intern): Engineer
Employee added successfully.

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 1
Enter ID: 2
Enter name: Khayyam

Enter age: 22
Enter position (Manager/Engineer/Intern): Manager
Employee added successfully.

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 1
Enter ID: 3
Enter name: Tamim
Enter age: 22
Enter position (Manager/Engineer/Intern): Intern
Employee added successfully.

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 2
Enter employee ID to remove: 3
Employee removed successfully.

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees

10. Logout
Choose an option: 3
Enter employee ID to search: 1
Bonus for omar(Position: Engineer): 4500.0
Employee Details:
ID: 1, Name: Omar, Position: Engineer, Salary: 30000
Working Years: 0 years
Salary Next Year (After Increment): 33000.0
Bonus for Next Year: 4500.0

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 4
Enter position to search (Manager/Engineer/Intern): Manager
Employees in Manager position (Salary: 50000):
Khayyam (ID: 2)

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 5
All Employees:
ID: 1, Name: omar, Position: Engineer, Salary: 30000
ID: 2, Name: khayyam, Position: Manager, Salary: 50000

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position

5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 6
Total Payroll: 80000, Average Salary: 40000.0

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 7
Yearly increment applied to all employees.

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 8
Enter employee ID for deduction: 1
Enter the number of absent days for Omar: 3
Enter total working days in the month (default is 30): 27
Salary deducted: 3666.6666666666665. New salary: 29333.333333333332

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll

7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 9
Enter number of recent employees to view: 2
Recent Employees:
ID: 1, Name: omar, Position: Engineer, Salary: 29333.333333333332
ID: 2, Name: Khayyam, Position: Manager, Salary: 55000.00000000001

1. Add Employee
2. Remove Employee
3. Search Employee by ID
4. Search Employees by Position
5. View All Employees
6. Calculate Payroll
7. Apply Yearly Increment
8. Apply Salary Deduction for Absence
9. View Recent Employees
10. Logout
Choose an option: 10
Logged out successfully!

1. Signup
2. Login
3. Exit
Choose an option: 3
Exiting the system. Goodbye!