```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <fstream>

using namespace std;

class Employee {
private:
    string emp_id;
    string name;
    string designation;
    double salary;

public:
    Employee(string id = "", string n = "", string desig = "", double sal = 0.0) :
emp_id(id), name(n), designation(desig), salary(sal) {}

    string getID() const { return emp_id; }
    string getName() const { return name; }
    string getDesignation() const { return designation; }
    double getSalary() const { return salary; }

    friend ostream& operator<<(ostream& os, const Employee& emp) {
        os << "Employee ID: " << emp.emp_id << endl;
        os << "Name: " << emp.name << endl;
        os << "Designation: " << emp.designation << endl;
        os << "Salary: " << emp.salary << endl;
        return os;
    }
};
```

```
class EmployeeDatabase {
private:
    vector<Employee> employees;
    vector<pair<string, int>> index;
    string filename = "employee_data.txt";

public:
    EmployeeDatabase() {
        loadIndex();
    }

    ~EmployeeDatabase() {
        saveIndex();
    }

    void addEmployee(const Employee& employee) {
        employees.push_back(employee);
        updateIndex(employee.getID(), employees.size() - 1);
        cout << "Employee added successfully." << endl;
        writeToExternalFile(employee);
    }

    void removeEmployee(const string& emp_id) {
        auto it = find_if(employees.begin(), employees.end(), [&emp_id](const
Employee& emp) { return emp.getID() == emp_id; });
        if (it != employees.end()) {
            int position = it - employees.begin();
            employees.erase(it);
            removeFromIndex(emp_id, position);
            cout << "Employee with ID " << emp_id << " removed successfully." <<
endl;
            saveIndex();
```

```
        } else {
            cout << "Employee with ID " << emp_id << " not found." << endl;
        }
    }

    void displayEmployeeList() {
        if (employees.empty()) {
            cout << "No employees in the database." << endl;
        } else {
            cout << "\nEmployee List:" << endl;
            for (const auto& emp : employees) {
                cout << emp << endl;
            }
        }
    }

    Employee* searchEmployee(const string& emp_id) {
        auto it = find_if(employees.begin(), employees.end(), [&emp_id](const
Employee& emp) { return emp.getID() == emp_id; });
        if (it != employees.end()) {
            return &(*it);
        } else {
            return nullptr;
        }
    }

private:
    void loadIndex() {
        ifstream indexFile(filename);
        if (indexFile.is_open()) {
            string emp_id;
            int position;
```

```
        while (indexFile >> emp_id >> position) {
            index.push_back(make_pair(emp_id, position));
        }
        indexFile.close();
    }
}

void saveIndex() {
    ofstream indexFile(filename);
    if (indexFile.is_open()) {
        for (const auto& entry : index) {
            indexFile << entry.first << " " << entry.second << endl;
        }
        indexFile.close();
    }
}

void updateIndex(const string& emp_id, int position) {
    index.push_back(make_pair(emp_id, position));
}

void removeFromIndex(const string& emp_id, int position) {
    index.erase(remove_if(index.begin(), index.end(), [&emp_id, position](const
pair<string, int>& entry) {
        return entry.first == emp_id && entry.second == position;
    }), index.end());
}

void writeToExternalFile(const Employee& emp) {
    ofstream dataFile(filename, ios::app);
    if (dataFile.is_open()) {
        dataFile << emp.getID() << " " << emp.getName() << " " <<
emp.getDesignation() << " " << emp.getSalary() << endl;
```

```
            dataFile.close();
        }
    }
};

int main() {
    EmployeeDatabase db;

    int choice;
    do {
        cout << "\nMENU" << endl;
        cout << "\n1) Insert Employee Data." << endl;
        cout << "2) Display Employee List." << endl;
        cout << "3) Remove Employee." << endl;
        cout << "4) Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                string emp_id, name, designation;
                double salary;

                cout << "\nEnter Employee ID: ";
                cin >> emp_id;
                cout << "Enter Name: ";
                cin.ignore();
                getline(cin, name);
                cout << "Enter Designation: ";
                getline(cin, designation);
                cout << "Enter Salary: ";
                cin >> salary;
```

```
            Employee emp(emp_id, name, designation, salary);
            db.addEmployee(emp);
            break;
        }
        case 2:
            db.displayEmployeeList();
            break;
        case 3: {
            string delete_id;
            cout << "\nEnter Employee ID to remove: ";
            cin >> delete_id;
            db.removeEmployee(delete_id);
            break;
        }
        case 4:
            cout << "\nExiting program..." << endl;
            break;
        default:
            cout << "\nInvalid choice. Please try again." << endl;
      }
   } while (choice != 4);

   return 0;
}
```