```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>  // For exit(), remove(), rename()
#include <cstdio>   // Backup for remove() and rename()
#include <algorithm> // For std::transform
#include <cctype>    // For std::tolower
#include <limits>    // For numeric_limits

using namespace std;

// Struct to store property details
struct Property {
    string id;
    string name;    // House, Apartment, Land
    string location;
    string price;
    string status;  // Available, Sold, Rented
};

// Function to convert string to lowercase
string toLower(const string& str) {
    string lowerStr = str;
    transform(lowerStr.begin(), lowerStr.end(), lowerStr.begin(), ::tolower);
    return lowerStr;
}

// Function to validate property name (case-insensitive)
bool isValidPropertyName(const string& name) {
    string lowerName = toLower(name);
    return (lowerName == "house" || lowerName == "land" || lowerName == "apartment");
}
```

```
// Function to validate property status (case-insensitive)
bool isValidStatus(const string& status) {
    string lowerStatus = toLower(status);
    return (lowerStatus == "available" || lowerStatus == "sold" || lowerStatus == "rented");
}

// Function to validate price as a positive number
bool isValidPrice(const string& priceStr) {
    try {
        double price = stod(priceStr);
        return price > 0;
    } catch (...) {
        return false;
    }
}

// Function prototypes
void adminMenu();
void customerMenu();
void addProperty();
void displayProperties();
void searchProperty();
void updateProperty();
void deleteProperty();
bool login();

int main() {
    int choice;
    while (true) {
        cout << "\nWelcome to the Real Estate Property Listing System\n";
        cout << "1. Admin\n2. Customer\n3. Exit\nEnter your choice: ";
        cin >> choice;

        if (choice == 1) {
            if (login()) {
                adminMenu();
            } else {
                cout << "Invalid username or password.\n";
            }
        } else if (choice == 2) {
            customerMenu();
        } else if (choice == 3) {
            break;
        } else {
            cout << "Invalid choice. Try again.\n";
        }
    }
    return 0;
}

// Admin menu function
void adminMenu() {
    int choice;
    while (true) {
        cout << "\nAdmin Menu\n";
        cout << "1. Add Property\n2. Display Properties\n3. Search Property by ID or Location\n";
        cout << "4. Update Property\n5. Delete Property\n6. Back to Main Menu\n7. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: addProperty(); break;
            case 2: displayProperties(); break;
            case 3: searchProperty(); break;
            case 4: updateProperty(); break;
            case 5: deleteProperty(); break;
            case 6: return;
            case 7: exit(0);
            default: cout << "Invalid choice. Try again.\n";
        }
    }
}

// Customer menu function
// Note: Customers can add/update as per original, but in a real system, restrict these
void customerMenu() {
    int choice;
    while (true) {
        cout << "\nCustomer Menu\n";
        cout << "1. Display Properties\n2. Search Property by ID or Location\n";  // Restricted
add/update for customers
        cout << "3. Back to Main Menu\n4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: displayProperties(); break;
            case 2: searchProperty(); break;
            case 3: return;
            case 4: exit(0);
            default: cout << "Invalid choice. Try again.\n";
        }
    }
}

// Function to handle login for admin
```

```
bool login() {
    string username, password;
    cout << "Enter username: ";
    cin >> username;
    cout << "Enter password: ";
    cin >> password;
    return (username == "admin" && password == "1234");
}

// Function to add property details
void addProperty() {
    ofstream outfile("properties.txt", ios::app); // Relative path for portability
    if (!outfile) {
        cout << "Error opening file.\n";
        return;
    }

    Property newProperty;
    cout << "Enter Property ID: ";
    cin >> newProperty.id;
    if (newProperty.id.empty()) {
        cout << "ID cannot be empty.\n";
        outfile.close();
        return;
    }

    // Check for duplicate ID
    ifstream infile("properties.txt");
    string id, name, location, price, status;
    while (infile >> id >> name >> location >> price >> status) {
        if (id == newProperty.id) {
            cout << "Error: Property ID already exists! Please enter a unique ID.\n";
            infile.close();
            outfile.close();
            return;
        }
    }
    infile.close();

    // Input with validation
    cout << "Enter Property Name (House, Land, Apartment): ";
    cin >> newProperty.name;
    if (!isValidPropertyName(newProperty.name)) {
        cout << "Invalid property name. Must be House, Land, or Apartment.\n";
        outfile.close();
        return;
    }

    cout << "Enter Location: ";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');  // Clear buffer
    getline(cin, newProperty.location);
    if (newProperty.location.empty()) {
        cout << "Location cannot be empty.\n";
        outfile.close();
        return;
    }

    cout << "Enter Price: ";
    cin >> newProperty.price;
    if (!isValidPrice(newProperty.price)) {
        cout << "Invalid price. Must be a positive number.\n";
        outfile.close();
        return;
    }
```

```
    cout << "Enter Status (Available, Sold, Rented): ";
    cin >> newProperty.status;
    if (!isValidStatus(newProperty.status)) {
        cout << "Invalid status. Must be Available, Sold, or Rented.\n";
        outfile.close();
        return;
    }

    outfile << newProperty.id << " " << newProperty.name << " " << newProperty.location
            << " " << newProperty.price << " " << newProperty.status << endl;

    outfile.close();
    cout << "Property added successfully!\n";
}

// Function to display all properties
void displayProperties() {
    ifstream file("properties.txt");
    if (!file) {
        cout << "Error opening file or no properties found.\n";
        return;
    }

    Property p;
    cout << "\nProperties List:\n";
    bool found = false;

    while (file >> p.id >> p.name >> p.location >> p.price >> p.status) {
        // Note: location may have spaces, but since saved without, display as-is
        cout << "ID: " << p.id << ", Name: " << p.name << ", Location: " << p.location
             << ", Price: " << p.price << ", Status: " << p.status << endl;
        found = true;
    }

    if (!found) {
        cout << "No properties available.\n";
    }
    file.close();
}

// Function to search property by ID or location (case-insensitive for location)
void searchProperty() {
    ifstream file("properties.txt");
    if (!file) {
        cout << "Error opening file.\n";
        return;
    }

    Property p;
    string key;
    cout << "Enter Property ID or Location to search: ";
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    getline(cin, key);  // Allow spaces in search key
    string lowerKey = toLower(key);
    bool found = false;

    while (file >> p.id >> p.name >> p.location >> p.price >> p.status) {
        if (p.id == key || toLower(p.location) == lowerKey) {
            cout << "ID: " << p.id << ", Name: " << p.name << ", Location: " << p.location
                 << ", Price: " << p.price << ", Status: " << p.status << endl;
            found = true;
        }
    }
    file.close();
    if (!found) cout << "Property not found.\n";
}

// Function to update property
void updateProperty() {
    ifstream file("properties.txt");
    if (!file) {
        cout << "Error opening file.\n";
        return;
    }

    ofstream temp("temp.txt");
    if (!temp) {
        cout << "Error creating temp file.\n";
        file.close();
        return;
    }

    Property p;
    string key;
    cout << "\nEnter Property ID to update: ";
    cin >> key;
    bool found = false;
    bool updateValid = true;

    while (file >> p.id >> p.name >> p.location >> p.price >> p.status) {
        if (p.id == key) {
            cout << "Current Property: Name=" << p.name << ", Location=" << p.location
                 << ", Price=" << p.price << ", Status=" << p.status << endl;
            cout << "Enter new Price (or press Enter to keep current): ";
            string newPrice;
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            getline(cin, newPrice);
            if (!newPrice.empty() && !isValidPrice(newPrice)) {
                cout << "Invalid price. Keeping current.\n";
                newPrice = p.price;
            } else if (!newPrice.empty()) {
                p.price = newPrice;
            }

            cout << "Enter new Status (Available, Sold, Rented) (or press Enter to keep current): ";
            string newStatus;
            getline(cin, newStatus);
            if (!newStatus.empty() && !isValidStatus(newStatus)) {
                cout << "Invalid status. Keeping current.\n";
                newStatus = p.status;
            } else if (!newStatus.empty()) {
                p.status = newStatus;
            }

            found = true;
        }
        temp << p.id << " " << p.name << " " << p.location << " " << p.price << " " << p.status << endl;
    }

    file.close();
    temp.close();

    if (found && updateValid) {
        remove("properties.txt");
        rename("temp.txt", "properties.txt");
        cout << "Property updated successfully!\n";
    } else {
        remove("temp.txt");
        if (!found) cout << "Property not found.\n";
    }
}

// Function to delete a property
void deleteProperty() {
    ifstream infile("properties.txt");
    if (!infile) {
        cout << "Error opening file.\n";
        return;
    }

    ofstream tempFile("temp.txt");
    if (!tempFile) {
        cout << "Error creating temp file.\n";
        infile.close();
        return;
    }

    string idToDelete;
    cout << "Enter Property ID to delete: ";
    cin >> idToDelete;

    string id, name, location, price, status;
    bool found = false;

    while (infile >> id >> name >> location >> price >> status) {
        if (id == idToDelete) {
            found = true;
            cout << "Deleting: " << name << " at " << location << endl;
            continue;
        }
        tempFile << id << " " << name << " " << location << " " << price << " " << status << endl;
    }

    infile.close();
    tempFile.close();

    if (found) {
        remove("properties.txt");
        rename("temp.txt", "properties.txt");
        cout << "Property deleted successfully!\n";
    } else {
        remove("temp.txt");
        cout << "Error: Property not found!\n";
    }
}
```