REAL ESTATE ADVISOR

INDEX

SERIAL NUMBER	TOPIC
• 1	• Introduction
• 2	• Project description
• 3	• Module explanation
• 4	• Data flow diagram
• 5	• Conclusion
• 6	• Project code
• 7	• Output

INTRODUCTION

The Real Estate Management System is a software application that aims to simplify the process of buying or renting properties by providing a platform for property owners to list their properties and for potential clients to search for properties and connect with the owners. The application allows property owners to upload property details along with their contact information and store it in the system. Clients can then browse the list of properties and contact the owners for further information.

PROJECT DESCRIPTION

The real estate industry has always been known for its complex processes and time-consuming procedures, which often results in frustration for both property owners and clients. The Real Estate Management System aims to address this problem by providing a centralized platform for property owners to list their properties and for clients to search for properties

MODULE EXPLANATION

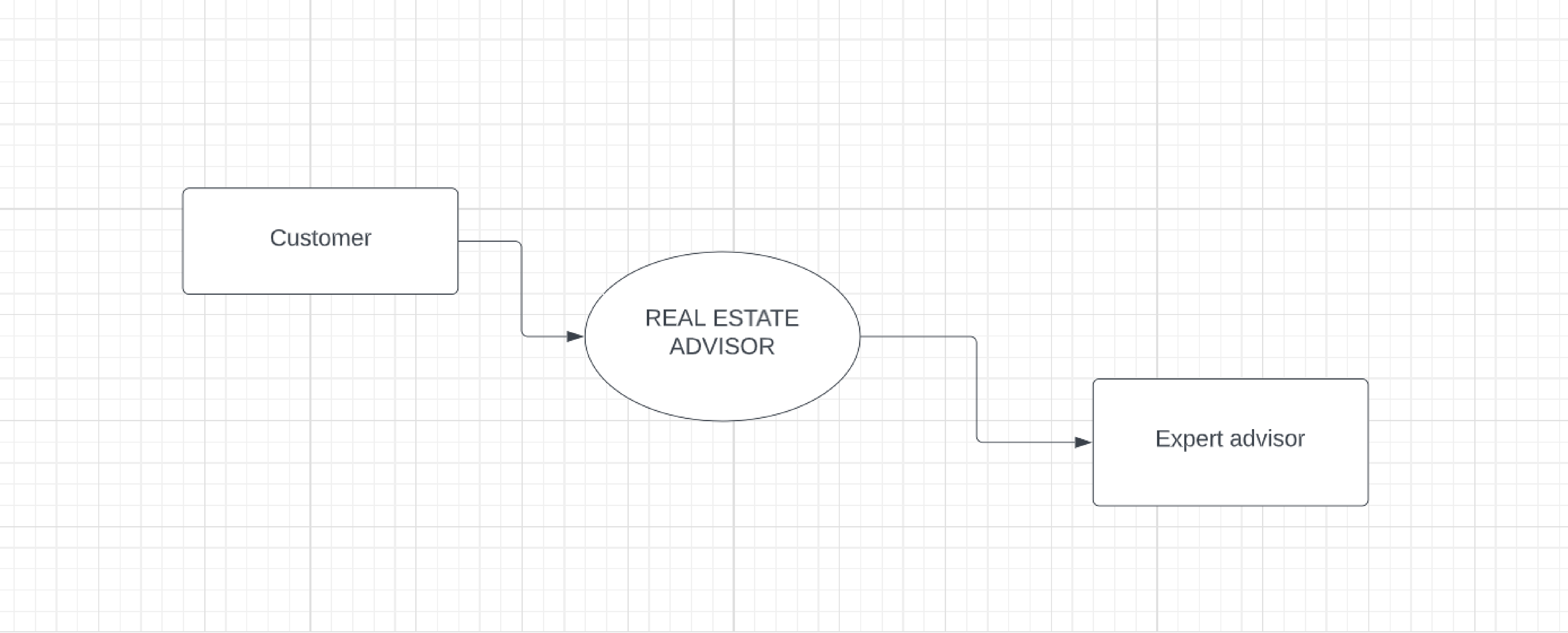
1. **Add New Record**: This module allows the user to add a new record of property for sale or purchase. The user will input all the details of the property, including its location, type (flat, land, commercial property), number of bedrooms/bathrooms, area, and price.
2. **Display the available property**: This module will display all the available properties based on the user's search criteria. The user can filter the results by location, type, number of bedrooms/bathrooms, area, and price.

3. **Search**: The search module allows the user to search for properties based on their specific requirements. The user can input their desired location, type of property, number of bedrooms/bathrooms, area, and price range, and the system will return all the matching properties.

4. **Buy/Sell**: This module allows the user to buy or sell a property. The user can choose to list their property for sale or browse available properties for purchase. The module will facilitate the transaction by providing the necessary information and documentation for both parties.

5. **Edit/Delete Record**: This module allows the user to edit or delete their existing property records. If the user needs to make any changes to their property listing or remove it from the system, they can use this module to do so.

DATA FLOW DIAGRAM



CONCLUSION

The Real Estate Management System is a user-friendly and efficient platform that simplifies the process of buying and selling properties. The system provides an easy to use interface for property owners to list their properties and for potential clients to search for property and connect with property owners. Overall, the real state management system aims to revolutionize the real state industry by providing a simple and effective solution to the complex problems of buying and selling property.

PROJECT CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_RECORDS 50
struct property
{
 char address[50];
 char type[50];
 int price;
};
// Function to add a new property record
void add_new_record(struct property *records, int
*num_records)
{
 // Check if maximum number of records has been reached
 if (*num_records == MAX_RECORDS)
 {
 printf("Maximum number of records reached.\n");
 return;
 }
```

```
// Get new record details from user
 printf("Enter property address: ");
 scanf(" %[^\n]", records[*num_records].address);
 printf("Enter property type: ");
 scanf(" %[^\n]", records[*num_records].type);
 printf("Enter property price: ");
 scanf("%d", &records[*num_records].price);
 // Increase number of records
 (*num_records)++;
 printf("Record added.\n");
}
// Function to display available property
void display_record(struct property *records, int num_records)
{
 // Check if there are any records
 if (num_records == 0)
 {
 printf("No records available.\n");
 return;
 }
```

```
// Display all available properties
 printf("%-20s %-20s %-10s\n", "Address", "Type", "Price");
 for (int i = 0; i < num_records; i++)
 {
 if (records[i].price > 0)
 {
 printf("%-20s %-20s $%-9d\n", records[i].address,
records[i].type, records[i].price);
 }
 }
}
// Function to search for a property record
void search_record(struct property *records, int num_records)
{
 // Check if there are any records
 if (num_records == 0)
 {
 printf("No records available.\n");
 return;
 }
 // Get search query from user
 char search_query[50];
 printf("Enter property address or type to search: ");
 scanf(" %[^\n]", search_query);
 // Search for property record
 int found = 0;
 for (int i = 0; i < num_records; i++)
 {
 if (strcmp(records[i].address, search_query) == 0 ||
strcmp(records[i].type, search_query) == 0)
 {
 printf("%-20s %-20s $%-9d\n", records[i].address,
records[i].type, records[i].price);
 found = 1;
 }
 }
 // Display message if no record was found
 if (!found)
 {
 printf("No matching records found.\n");
 }
}
```

```
// Function to buy and sell a property record
void buy_sell_record(struct property *records, int
num_records)
{
 // Check if there are any records
 if (num_records == 0)
 {
 printf("No records available.\n");
 return;
 }
 // Get address of property to buy/sell from user
 char address[50];
 printf("Enter property address to buy/sell: ");
 scanf(" %[^\n]", address);
 // Search for property record
 int index = -1;
 for (int i = 0; i < num_records; i++)
 {
 if (strcmp(records[i].address, address) == 0)
 {
 index = i;
 break;
 }
 }
```

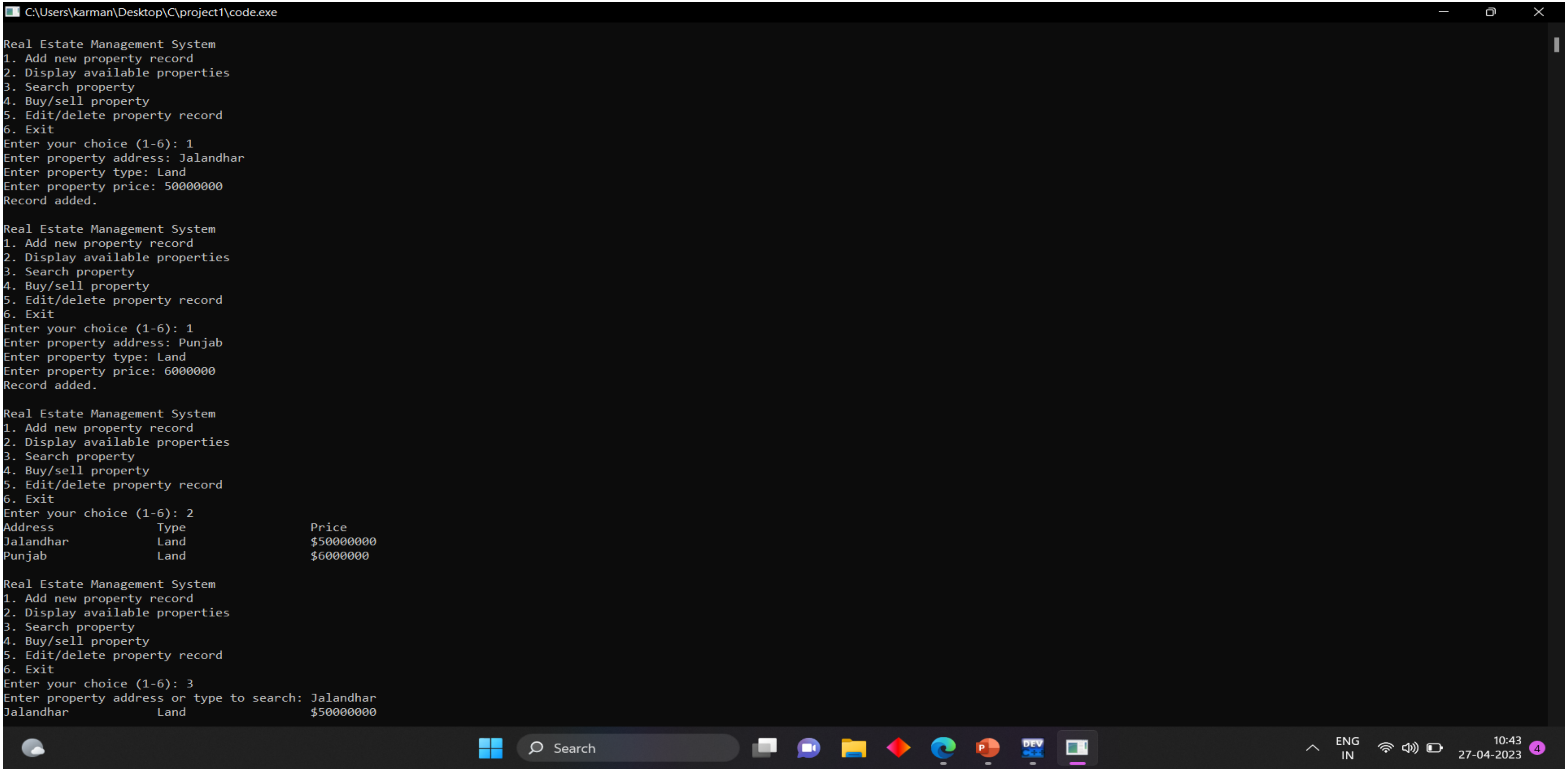
```
// Display message if property was not found
 if (index == -1)
 {
 printf("Property");
 } // Display property details
 printf("%-20s %-20s $%-9d\n", records[index].address,
records[index].type, records[index].price);
 // Get transaction type from user
 printf("Enter 'buy' to buy property or 'sell' to sell property: ");
 char transaction_type[10];
 scanf(" %[^\n]", transaction_type);
 // Perform transaction
 if (strcmp(transaction_type, "buy") == 0)
 {
 if (records[index].price == 0)
 {
 printf("This property is not for sale.\n");
 }
 else
 {
 printf("Property bought for $%d.\n",
records[index].price);
 records[index].price = 0;
 }
}
 else if (strcmp(transaction_type, "sell") == 0)
 {
 if (records[index].price > 0)
 {
 printf("This property is already for sale.\n");
 }
 else
 {
 printf("Enter new property price: ");
 scanf("%d", &records[index].price);
 printf("Property put up for sale for $%d.\n",
records[index].price);
 }
 }
 else
 {
 printf("Invalid transaction type.\n");
 }
}
```

```
// Function to edit/delete a property record
void edit_delete_record(struct property *records, int
*num_records)
{
 // Check if there are any records
 if (*num_records == 0)
 {
 printf("No records available.\n");
 return;
 } // Get address of property to edit/delete from user
 char address[50];
 printf("Enter property address to edit/delete: ");
 scanf(" %[^\n]", address);
 // Search for property record
 int index = -1;
 for (int i = 0; i < *num_records; i++)
 {
 if (strcmp(records[i].address, address) == 0)
 {
 index = i;
 break;
 }
 }
```

```
// Display message if property was not found
 if (index == -1)
 {
 printf("Property not found.\n");
 return;
 }
 // Display property details
 printf("%-20s %-20s $%-9d\n", records[index].address,
records[index].type, records[index].price);
 // Get action from user
 printf("Enter 'edit' to edit property or 'delete' to delete
property: ");
 char action[10];
 scanf(" %[^\n]", action);
 // Perform action
 if (strcmp(action, "edit") == 0)
 {
 // Get new property details from user
 printf("Enter new property address (leave blank to keep
current value \"%s\"): ", records[index].address);
 char new_address[50];
 scanf(" %[^\n]", new_address);
if (strcmp(new_address, "") != 0)
 {
 strcpy(records[index].address, new_address);
 }
 printf("Enter new property type (leave blank to keep current
value \"%s\"): ", records[index].type);
 char new_type[50];
 scanf(" %[^\n]", new_type);
 if (strcmp(new_type, "") != 0)
 {
 strcpy(records[index].type, new_type);
 }
 printf("Enter new property price (leave blank to keep current
value \"$%d\"): ", records[index].price);
 char new_price_str[50];
 scanf(" %[^\n]", new_price_str);
  if (strcmp(new_price_str, "") != 0)
  {
  int new_price = atoi(new_price_str);
  records[index].price = new_price;
  }
  printf("Property details updated.\n");
  }
```

```
else if (strcmp(action, "delete") == 0)
 {
 // Delete property record
 for (int i = index; i < *num_records - 1; i++)
 {
 records[i] = records[i + 1];
 }
 (*num_records)--;
 printf("Property record deleted.\n");
 }
 else
 {
 printf("Invalid action.\n");
 }
  }
  int main()
  {
   struct property records[MAX_RECORDS];
   int num_records = 0; // Display menu
   while (1)
   {
   printf("\nReal Estate Management System\n");
   printf("1. Add new property record\n");
   printf("2. Display available properties\n”);
printf("3. Search property\n");
 printf("4. Buy/sell property\n");
 printf("5. Edit/delete property record\n");
 printf("6. Exit\n");
 // Get user input
 int choice;
 printf("Enter your choice (1-6): ");
 scanf("%d", &choice);
 // Perform action based on user choice
 switch (choice)
 {  case 1:
     add_new_record(records, &num_records);
     break;
     case 2:
     display_record(records, num_records);
     break;
     case 3:
     search_record(records, num_records);
     break;
     case 4:
     buy_sell_record(records, num_records);
     break;
     case 5:
     edit_delete_record(records, &num_records);
     break;
 case 6:
 exit(0);
 default:
 printf("Invalid choice.\n");
 break;
 }
 }
 return 0;
}
```

OUTPUT

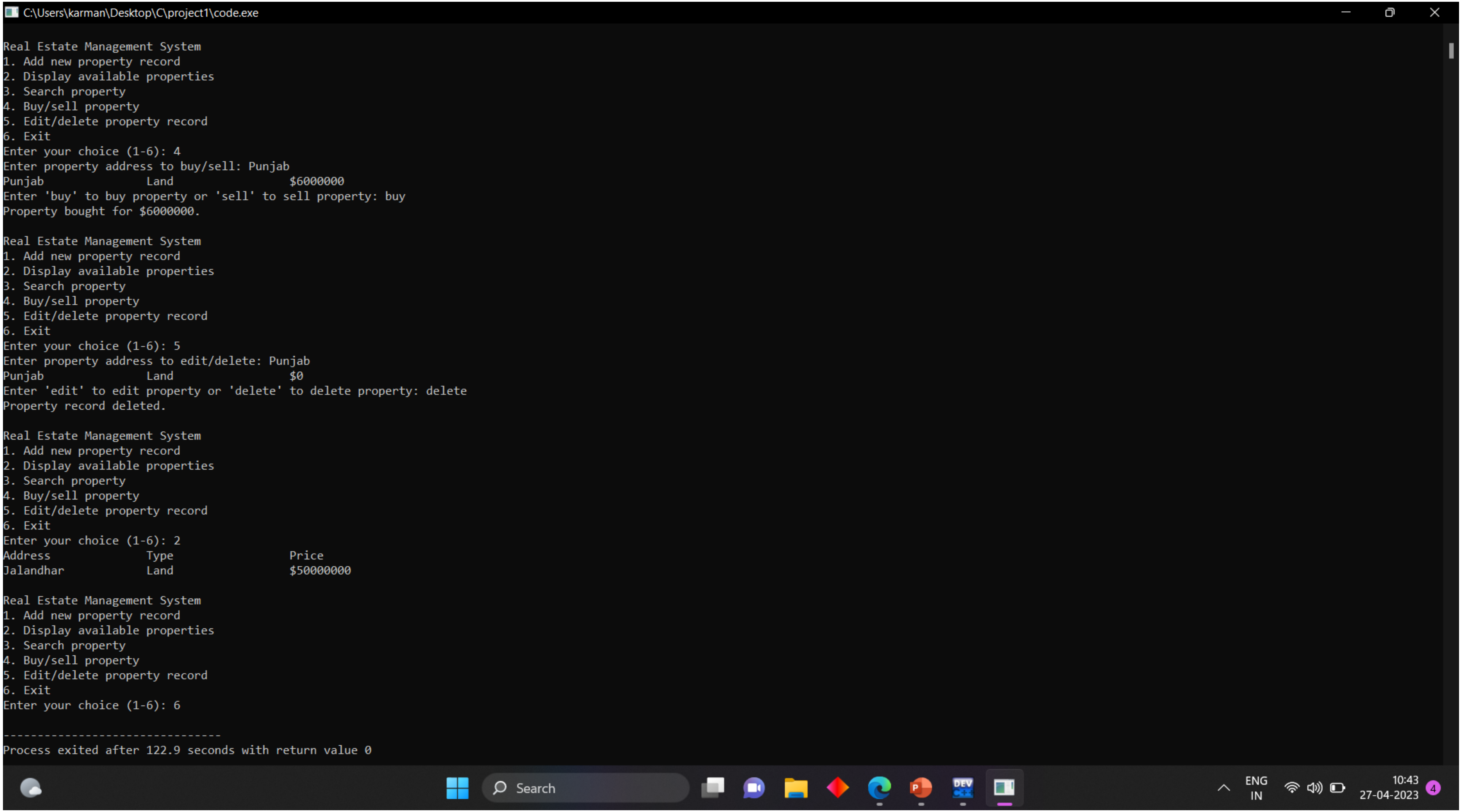


```
Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 1
Enter property address: Jalandhar
Enter property type: Land
Enter property price: 50000000
Record added.

Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 1
Enter property address: Punjab
Enter property type: Land
Enter property price: 6000000
Record added.

Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 2
Address             Type                Price
Jalandhar           Land                $50000000
Punjab              Land                $6000000

Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 3
Enter property address or type to search: Jalandhar
Jalandhar           Land                $50000000
```



C:\Users\karman\Desktop\C\project1\code.exe

```
Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 4
Enter property address to buy/sell: Punjab
Punjab               Land                 $6000000
Enter 'buy' to buy property or 'sell' to sell property: buy
Property bought for $6000000.

Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 5
Enter property address to edit/delete: Punjab
Punjab               Land                 $0
Enter 'edit' to edit property or 'delete' to delete property: delete
Property record deleted.

Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 2
Address              Type                 Price
Jalandhar            Land                 $50000000

Real Estate Management System
1. Add new property record
2. Display available properties
3. Search property
4. Buy/sell property
5. Edit/delete property record
6. Exit
Enter your choice (1-6): 6

--------------------------------
Process exited after 122.9 seconds with return value 0
```