|  |
| --- |
|  |
| User Requirement Analysis |
| By Shakir Alexander |

|  |
| --- |
| Shakir Alexander  6-27-2024 |



# Task 1A: SCENARIO AND SCOPE

The topic chosen is an Agency Management program. The property market is a massive contributor to the growth and stability of our economy; therefore, many people would like join this market in order to make a living. Real Estate agencies are utilised to streamline this process of buying and selling properties.

Real Estate agencies provide an efficient way of buying and selling properties, from a client perspective. However, real estate agencies themselves need a modern and effective way of efficiently organizing and streamlining their management of properties, agents and clients.

A solution to this problem involves a program that centralises all aspects involving real estate. For example, managing all the properties listed under the agency, all agents working for the agency and clients conducting business with the agency. This creates a centralised hub for the agency and its agents to efficiently do business.

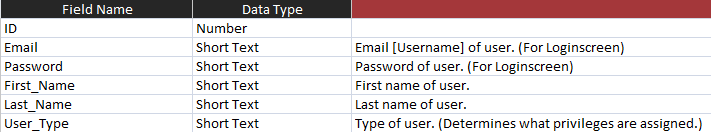
This program will be able to Log in and out as either an Admin or an Agent. Admin users are able to access all information regarding the properties under the agencies management, the information on all agents working for the agency. Admin users will also have access to statistics relevant to the agency. Agent users are able to access all information regarding properties listed under the agencies management as well as access to information on the clients they are working with. Agent users will have access to statistics relevant to them. The program will include features that makes the agency more efficient.

# Task 1B: USER REQUIREMENTS

|  |  |  |
| --- | --- | --- |
|  | User 1: Agent | User 2: Administrator |
| Role | The role of the agent is to manage the buying and selling of houses for their clients under the agency name. | The role of the administrator is to manage all the agents, properties and clients under the agency. |
| Activity | * Ability to log into the program * See all clients being worked with by the agent * See all properties that are being bought and sold by the agency by clients. * Access Information on Properties and Clients. | * Ability to log into program * Access to all information on Agents and Properties. * Access to Agency statistics. |
| Limitation | * No access to admin features such as company statistics etc. * No access to other agent information | * Client information is classified. |

# Task 2: DATABASE

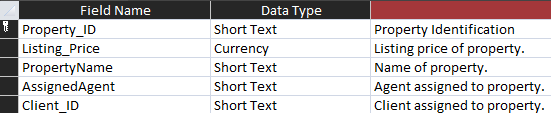
## tblLogin



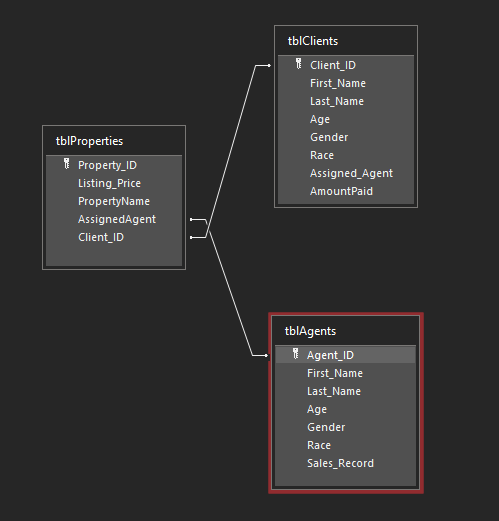
## tblAgent

## tblClients

## tblProperties



## Relationships



# DATA DICTIONARY

## Task 3A: Classes and objects

|  |  |
| --- | --- |
| Key | |
| + | Public |
| - | Private |

|  |  |
| --- | --- |
| TInfo | * Connected to the Agent form. * In form activate it instantiates the object automatically. * Retrieves information on the Agents name and the current level of the Agent. |

|  |
| --- |
| TInfo |
| Attributes: |
| * fFirstName : String * fLastName : String * fCurrent\_Level: integer |
| Methods: |
| * Constructor create (sFirstName, sLastName: string) * Function getFullName:string * Function DetermineTier(iCurrentLevel: integer): string; |

## Task 3B: Text files

|  |  |
| --- | --- |
| PropertyListings.txt | * Contains information: Property ID and their Listing Prices. |
| Log[Date].txt | * Exported information on the logged activity of the Agent and the time it took place. |

## Data structures

The program will make use of a Parallel Array. arrPropertyCode consists of data from PropertyListings.txt. arrPropertyListing consists of data from PropertyListings.txt.

The array will be used to calculate the status of a property: “Available”, “Processing” or “Sold”. Other calculations will also be made using the array.

### arrPropertyListing(Sample)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Listing Price | R1 338 182,00 | R1 338 182,00 | R406 879,00 | R946 263,00 | R1 043 620,00 | R620 817,00 | R984 926,00 | R729 605,00 | R1 161 742,00 |

### Admin Management

### Property Screen

### Property Screen

### Client Screen

### Admin Screen

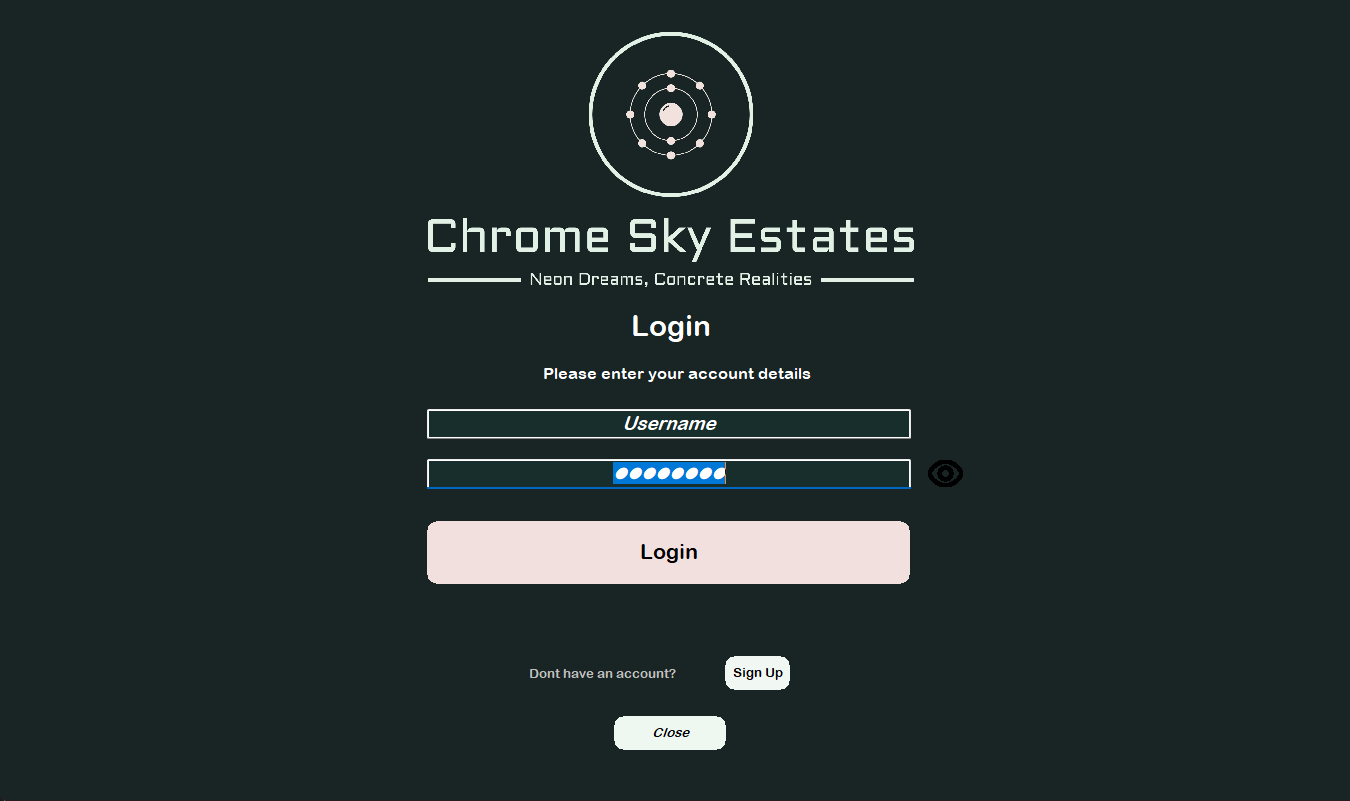
### Agent Screen

### New User Screen

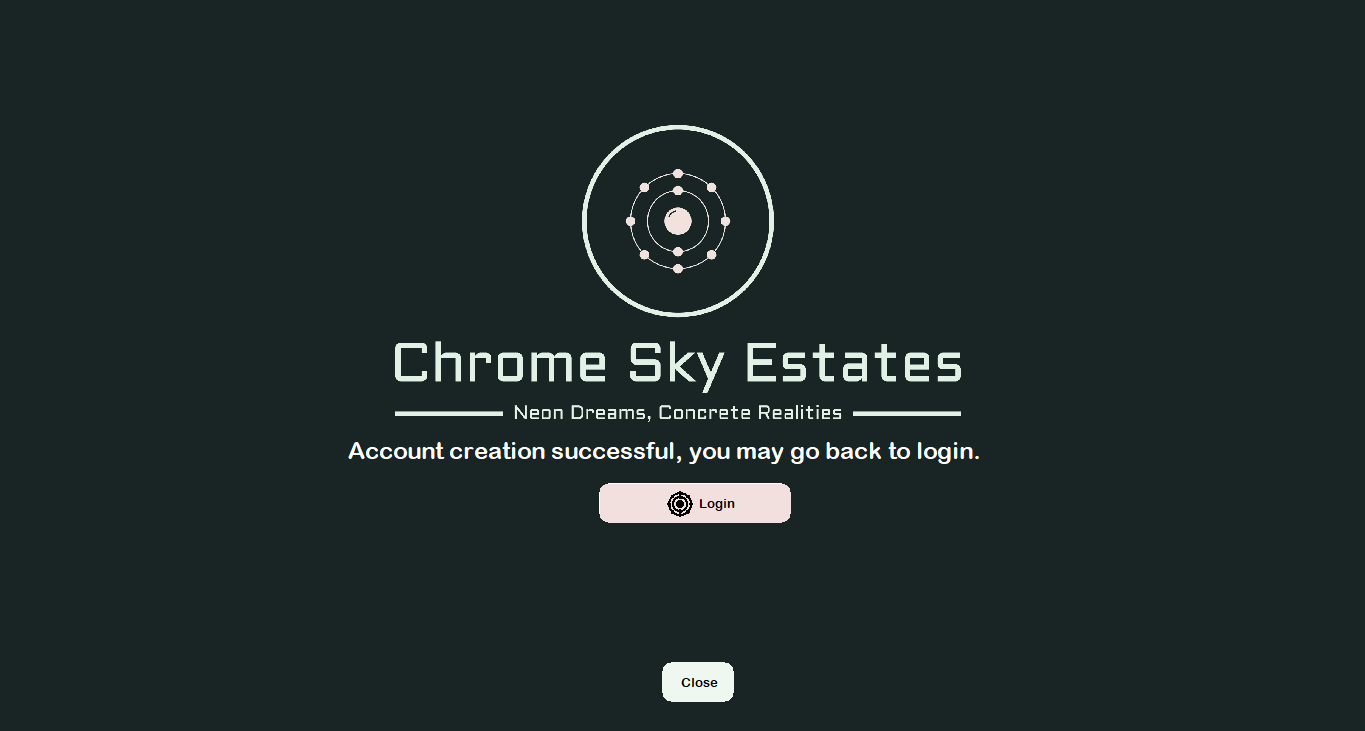
### Login Screen

# Task 4B: GUI

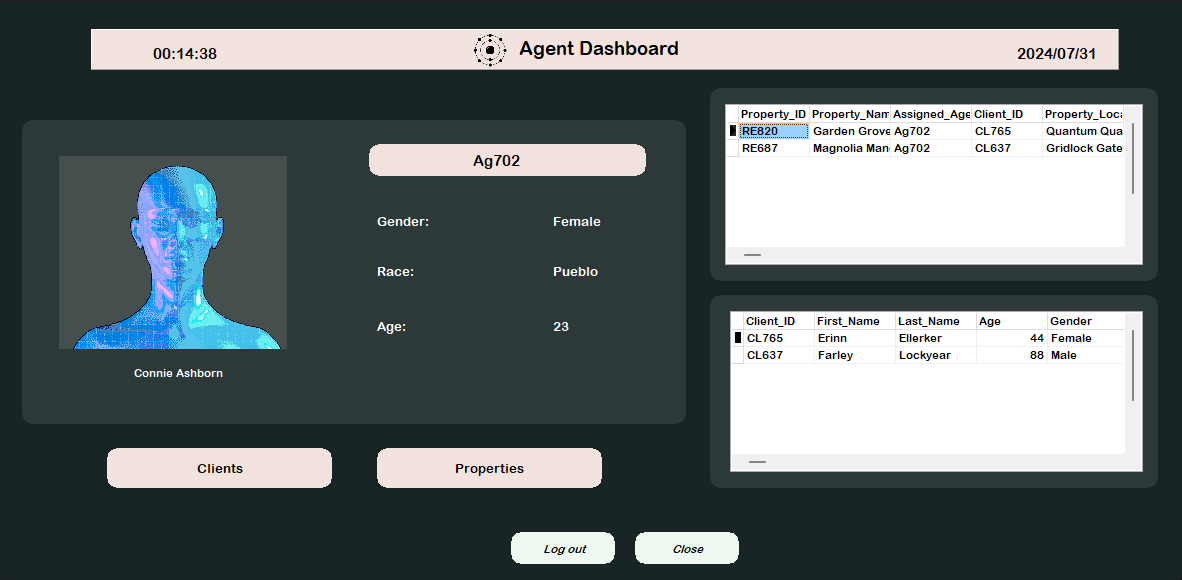
### Login



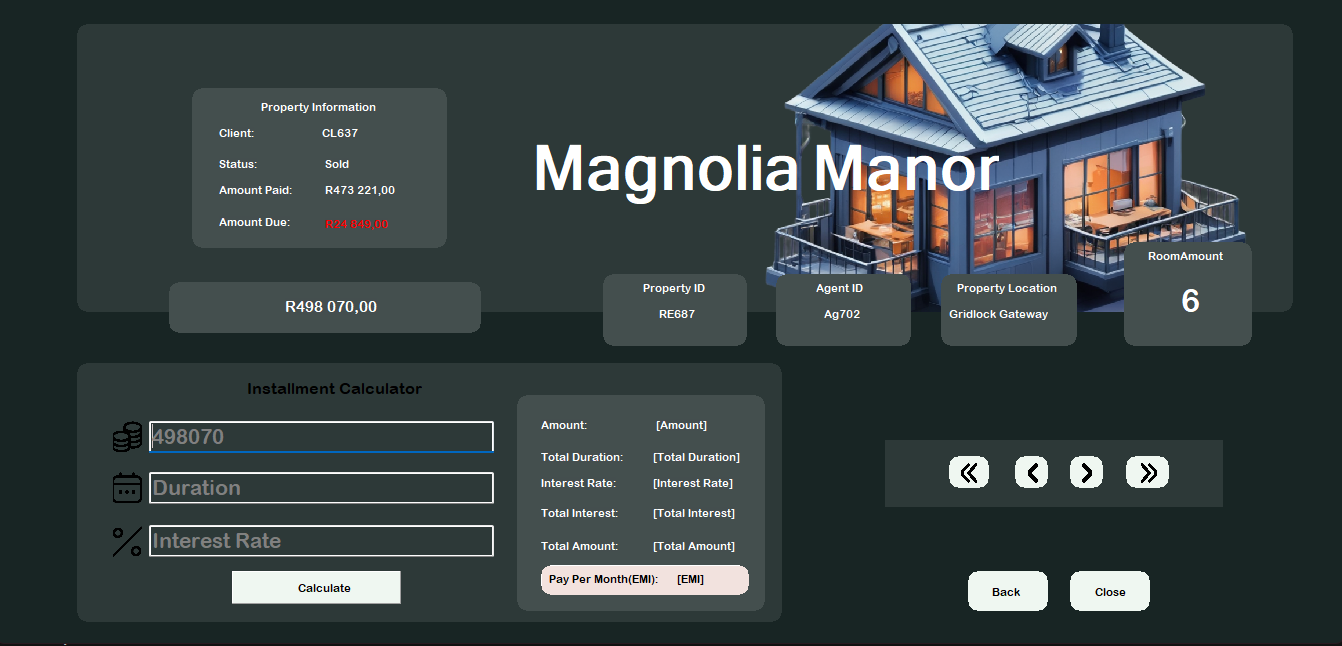
### Register



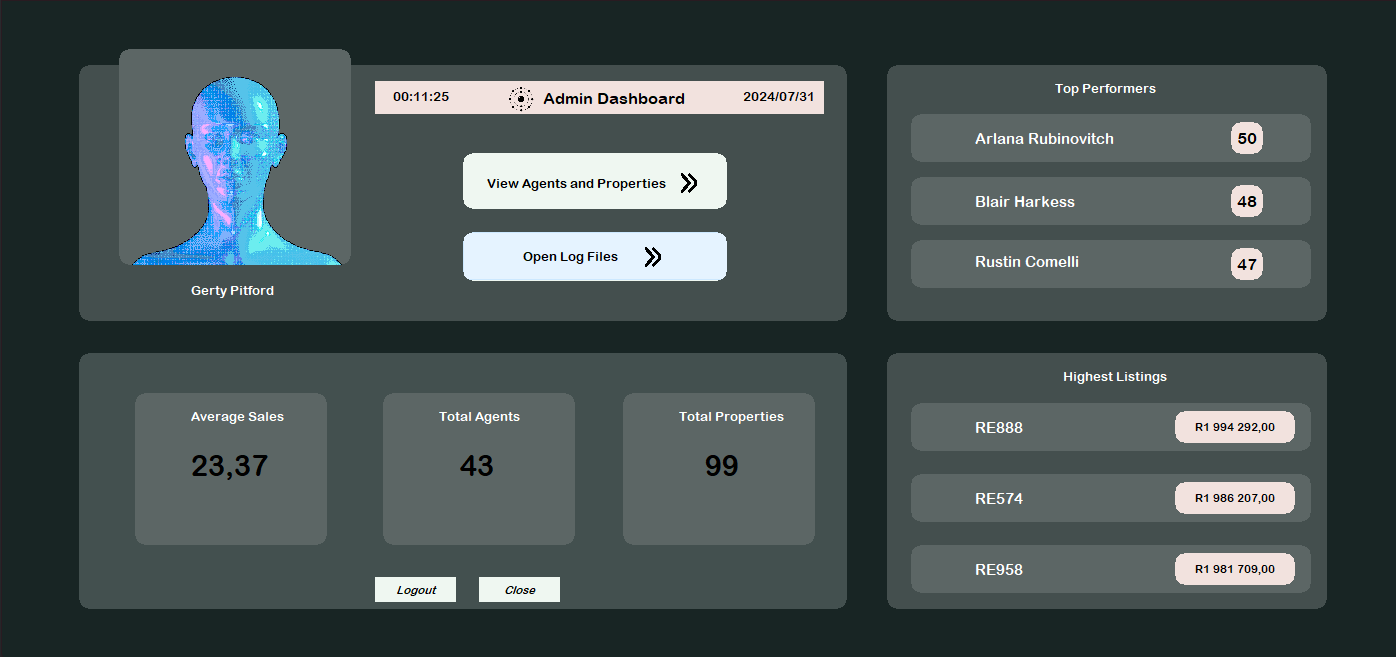
### Agent Dashboard



### Client & Property Management



### Administrator Screen



### Agent and Property Management

### 

# Task 5: IPO

## Task 5A: DATA INPUT

### Login Screen:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Source | Data Type | Format | GUI Component | Validation |
| Username | Keyboard | String | General Text | edtLogInUsername | If the username is blank  Error Message:  “Please enter username” |
| Password | Keyboard | String | General Text | edtLogInPassword | If the password is blank  Error Message: “Please enter Password”  If password is incorrect  Error Message:  “Password or username is incorrect” |
| Button Click: Admin | Mouse Click | String | General Text | btnLoginAdmin | If button click does not correspond with User type  Error Message: “User type incorrect” |
| Button Click: Agent | Mouse Click | String | General Text | btnLoginAdmin | If button click does not correspond with User type  Error Message: “User type incorrect” |

### frmRegister:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | Source | Data Type | Format | GUI Component | Validation |
| Email | Keyboard | String | General Text | edtRegisterEmail | If the email field is blank  Error Message:  “Please enter email” |
| Password | Keyboard | String | General Text | edtRegisterPassword | If the password field is blank  Error Message:  “Please enter password” |
| First Name | Keyboard | String | General Text | edtRegisterFirstName | If the first name field is blank  Error Message:  “Please enter first name” |
| Last Name | Keyboard | String | General Text | edtRegisterLastName | If the last name field is blank  Error Message:  “Please enter last name” |
| User Type | Combo Box | String | General Text | cmbRegisterUserType | If the user type isn’t selected  Error Message:  “Please choose user type” |
| Age | Keyboard | Integer | General Text | edtAge | If the user is below 18  Error Message: User must be over the age of 18 |
| Gender | Keyboard | String | General Text | edtGender | If the first Gender field is blank  Error Message:  “Please enter Gender” |
| Agent ID | Keyboard | String | General Text | edtAgentID | If agent ID already exists.  Error Message: Agent ID already exists |

### frmClients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Add Amount** | Keyboard | General Text | edtAddAmount | If Amount Added = 0  Error Message: Please enter a value. |
| **Client ID** | Keyboard | General Text | edtClientID |  |
| **Age** | Keyboard | General Text | edtAge |  |
| **First Name** | Keyboard | General Text | edtFirstName |  |
| **Last Name** | Keyboard | General Text | edtLastName |  |
| **Gender** | Keyboard | General Text | edtGender |  |
| **Race** | Keyboard | General Text | edtRace |  |
| **Search** | Keyboard | General Text | edtSearch |  |

|  |  |
| --- | --- |
| What | How |
| Check if username and password are in database. | *tblUsers.first*  *while not tblUsers.eof do*  *begin*  *if tblUser.Username = sUsername AND tblUsers.Password = sPassword then* |
| Check if user is an admin or an agent. | *tblUsers.first*  *while not tblUsers.eof do*  *begin*  *if tblUser.UserType = sUserType then* |
| Checks if Password is valid | tblLogin.first;  if bFound = True and (tblLogin['User\_Type'] = 'Agent') then  begin  frmLoginScreen.Hide;  frmAgentDashboard.Show;  end; |
| Takes property info/client info/agent info into a text file. | *if FileExists('PropertyListings.txt') = false then //Loads text file contents into arrays: arrPropertyCode and arrPropertyListing.*  *begin*  *showmessage('Error: File does not exists');*  *end;*  *while not Eof(tPropertyValues) do*  *begin*  *Readln(tPropertyValues, sline);*  *Inc(N, 1);*  *arrPropertyCode[N] := copy(sline, 1, pos('#', sline) - 1);*  *arrPropertyListings[N] := StrToFloat(copy(sline, pos('#', sline) + 1,*  *length(sline)));*  *end;*  *CloseFile(tPropertyValues);* |
| Posts new registered users into database | *tblUsers.first*  *while not tblUsers.eof do*  *begin*  *tblUser.FirstName := sFirstName*  *tblUser.Email := sEmail*  *…*  *end* |
| Sort Properties and clients alphabetically | *with dmCSE do*  *begin*  *qryAgents.Sort := ('Agent\_ID ASC');*  *qryProperties.Sort := ('Assigned\_Agent ASC')*  *end;* |
| Calculate Amount due from clients | *PropertyPrice – AmountPaid= AmountDue.* |

## Task 5B: DATA PROCESSING

## Task 5C: DATA OUTPUT

|  |  |  |
| --- | --- | --- |
| Data | Format | GUI Component |
| AgentID, Name, Surname, Sales Record | AgentID: <AgentID>  First Name: <First Name>  Last Name: <Last Name>  Sales Record: <Sales Record> | redAgentDetails : TRichEdit |
| ClientID, First Name, Last Name, Age, Gender, Race, Assigned Agent | ClientID: <ClientID>  First Name: <First Name>  Last Name: <Last Name>  Age: <Age>  Gender: <Gender>  Race: <Race>  Assigned Agent: <Assigned Agent> | redClientDetails : TRichEdit |