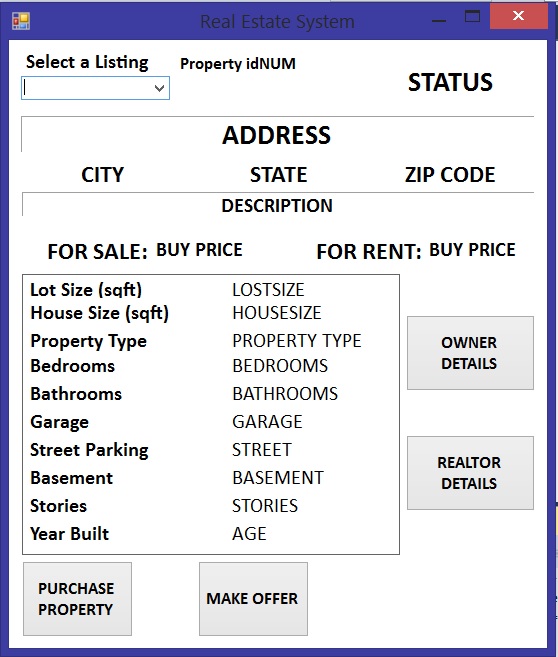
**PROJECT 2**

DESIGN SPECIFICATION

Billy Kong

The design of the Real Estate System will function around a GUI built in C#. The GUI will gather information from a database. The database used is an SQL database.

Sample output as follows:



SQL databases implement rows (X Axis) and columns (Y Axis) which allows for easy organization of data. Rows are organized with positions, [0 - ~] Information in rows will fall under an address (which is also a row).

* Address – row[0], datatype - VARCHAR(50)
* idNum – row[1], datatype - INT
* city – row[2], datatype – VARCHAR(20)
* state – row[3], datatype – VARCHAR(20)
* zip – row[4], datatype – INT
* description – row[5], datatype – VARCHAR(100)
* status – row[6], datatype – VARCHAR(10)
* buyprice – row[7], datatype – INT
* rentprice – row[8], datatype – INT
* lotsize – row[9], datatype – INT
* housesize – row[10], datatype – INT
* propertytype – row[11], datatype – VARCHAR(20)
* bedrooms – row[12], datatype – INT
* bathrooms – row[13], datatype – INT
* garage – row[14], datatype – VARCHAR(3)
* streetparking – row[15], datatype – VARCHAR(3)
* stories – row[16], datatype – INT
* basement – row[17], datatype – VARCHAR(10)
* yearbuilt – row[18], datatype – INT

There will be an option to filter state listings, 3 states – New York, New Jersey, and Pennsylvania

Buttons included in the user interface:

* [Purchase Property] – Will change the status of the property from its default, “active”, to “sold”
* [Make Offer] – Will open a separate form that allows users to input their price offer. It will also have a list of past offers
  + name[1-5] will take in information is positions [19-23] from the database
  + offer[1-5] will take in information is positions [24-28] from the database
  + there will be a name[6] and offer[6] where the user will find his name and offer, this will just replicate information onto labels, will not store into the database.
* [Owner Details] – Separate form that will display property owners information
  + Owner name [29]
  + Contact number and email address [30] [31]
  + Realtor [32]
* [Realtor Details] – separate form will open displaying details of the realtor overseeing the property
  + Agent name [32]
  + Contact number and email address [33] [34]
  + Agent comments of the property [36]
  + Website [35]
* [ADD A LISTING] – will allow users to add their own listing to the collection
  + Form that takes your input from textboxes and relays it to the database

A comboBox will be used to allow users to switch through property addresses. Labels will make the rest of the user interface by displaying appropriate information.

Two utility buttons

* Reset selected – resets selected property status from ‘Sold’ to ‘Active’
* Reset all – resets all property status from ‘Sold’ to ‘Active’

NEW:

Use of Inheritance and Polymorphism in this program:

Added a fifth form that activates when “PURCHASE PROPERTY” has been clicked.

Within the namespace of this form, a base class **CreditCard** is defined. Derived classes:

* MasterCard
* Visa
* American Express
* DiscoverCard

Virtual function PrintCard – overridden in all derived classes

* Returns Card Provider (a string) to a function in Form5, which then sets the label equal to the name of the card provider. Card Providers – Master Card, Visa, American Express, Discover Card

Code in the form then allows for user inputted text to be set to data members of the class:

* name(class data member) = name.Text (form textbox)
* ccNum = cc.Text
* sNum = csc.Text
* expDate = exp.Text