VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590014

A DBMS Mini-Project Report
On

"Real Estate Management System "

Submitted in partial fulfillment of the requirements for the 5th semester of
Bachelor of Engineering in Computer Science and Engineering
of Visvesvaraya Technological University, Belagavi

Submitted by:

Name: Ajay Umakanth **USN: 1RN16CS006**
Name: Matharishwa B **USN: 1RN16CS052**

Under the Guidance of:

Mrs. S Mamatha Jajur
Assistant Professor
Dept. of CSE

Department of Computer Science and Engineering
RNS Institute of Technology
Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098

2017-2018

RNS Institute of Technology

Channasandra, Dr.Vishnuvardhan Road,
Bengaluru-560 098

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

Certified that the DBMS mini-project work entitled **"Real Estate Management System"** has been successfully carried out by **Ajay Umakanth** bearing USN **1RN16CS006** and **Matharishwa B** bearing USN **1RN16CS052**, bonafide students of **RNS Institute of Technology** in partial fulfillment of the requirements for the **5th semester Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, Belagavi, during the academic year 2017-2018. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the mini-project requirements of DBMS lab of 5th semester BE in CSE.

Mrs. S Mamatha Jajur	**Mr. Sanjay P K**	**Dr. G T Raju**
Assistant Professor	**Assistant Professor**	**Vice Principal and**
Dept. of CSE	**Dept of CSE**	**Head of Dept. of CSE**

External Viva:

Name of the Examiners **Signature with Date**

1.

2.

ACKNOWLEDGMENTS

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

We would like to thank **Dr. H N Shivashankar**, Director, RNSIT, Bangalore, for his moral support towards completing our project.

We are grateful to **Dr. M K Venkatesha,** Principal, RNSIT, Bangalore, for his support towards completing this mini project.

We would like to thank **Dr. G T Raju**, Vice Principal, Prof. and Head, Department of Computer Science & Engineering, RNSIT, Bangalore, for his valuable suggestions and expert advice.

We deeply express my sincere gratitude to my guide **Mrs. S Mamatha Jajur** and **Mr. Sanjay PK**, Asst Prof, Department of CSE, RNSIT, Bangalore, for their able guidance, regular source of encouragement and assistance throughout this project.

We would like to thank all the teaching and non-teaching staff of department of Computer Science & Engineering, RNSIT, Bengaluru for their constant support and encouragement.

Name: Ajay Umakanth

USN: 1RN16CS006

Date : 27/11/2018

Name:Matharishwa

Place : Bengaluru

USN: 1RN16CS052

ABSTRACT

- Our project "**REAL ESTATE MANAGEMENT SYSTEM**" is related to application of Real Estate.

- Real estate management system is advanced solution for his/her estate problem.

- User can bid on the property, the owner can put his properties on sale &verify them and the online service is provided by us free of charge.

- Here registration is also free of cost. So user can registration by using Real System then Buy the property & verify them. The software is so reliable to user .Our main concept is give best & quick result to user.

CONTENT

Chapter No. Title Page No

5. Results , snapshots and discussions

6. Conclusion and Future Enhancements

Bibliography

CHAPTER 1

INTRODUCTION

1.1 DATABASE TECHNOLOGIES

The essential feature of database technology is that it provides an internal representation (model) of the external world of interest. Examples are the representation of a particular date/time/flight/aircraft in airline reservation or of item code/item description/quantity on hand/reorder level/reorder quantity in a stock control system.
The technology involved is concerned primarily with maintaining the internal representation consistent with external reality; this involves the results of extensive R&D over the past 30 years in areas such as user requirements analysis, data modelling, process modelling, data integrity, concurrency, transactions, file organisation, indexing, rollback and recovery, persistent programming, object-orientation, logic programming, deductive database systems, active database systems... and in all these (and other) areas there remains much to be done.
The essential point is that database technology is a CORE TECHNOLOGY with links to:

- Information management / processing
- Data analysis / statistics
- Data visualization / presentation
- Multimedia and hypermedia
- Office and document systems
- Business processes, workflow, CSCW (computer-supported cooperative work)

Relational DBMS is the modern base technology for many business applications. It offers flexibility and easy-to-use tools at the expense of ultimate performance. More recently relational systems have started to extend their facilities in the directions of information retrieval, object-orientation and deductive/active systems leading to the so-called 'Extended Relational Systems'.

Information Retrieval Systems started with handling library catalogues and extended to full free-text utilizing inverted index technology with a lexicon or thesaurus. Modern systems utilize some KBS (knowledge-based systems) techniques to improve retrieval.

Object-Oriented DBMS started for engineering applications where objects are complex, have versions and need to be treated as a complete entity. OODBMSs share many of the OOPL features such as identity, inheritance, late binding, overloading and overriding.

OODBMSs have found favour in engineering and office systems but have not yet been successful in traditional application areas.

Deductive / Active DBMS have emerged over the last 20 years and combine logic programming technology with database technology. This allows the database itself to react to external events an to maintain dynamically its integrity with respect to the real world.

1.2 CHARACTERISTICS OF DATABASE APPROACH

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics −

- Real-world entity − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.

- Relation-based tables − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

- Isolation of data and application − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.

- Less redundancy − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.

- Consistency − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

- Query Language − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.

- ACID Properties − DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.

- Multiuser and Concurrent Access − DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

- Multiple views − DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.

- Security − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

1.3 APPLICATIONS OF DBMS

Applications where we use Database Management Systems are:

- **Telecom**: There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry**: Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.
- **Banking System**: For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Education sector**: Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc. There is a hell lot amount of inter-related data that needs to be stored and retrieved in an efficient manner.
- **Online shopping**: You must be aware of the online shopping websites such as Amazon, Flip kart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

1.4 PROBLEM DESCRIPTION/STATEMENT

- This is a real estate business application through which a user can access its information and manage all the adding, updating, deleting the assets and some of its tasks.

- The Admin user can change the update the information regarding property selling and buying and cancellation.

- The system is very useful for the companies who develope apartments, hotels, villa, residential properties and commercial properties.

- Companies or individual agents can also advertise their property .

CHAPTER 2

REQUIREMENT ANALYSIS

2.1 HARDWARE REQUIREMENTS

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor	:	Intel i5 processor
Processor Speed	:	3.1 GHz
RAM	:	4 GB
Storage Space	:	40 GB
Monitor Resolution	:	1024*768 or 1336*768 or 1280*1024

2.2 SOFTWARE REQUIREMENTS

Operating System	:	Windows 10
IDE	:	Visual Studio IDE

2.3 FUNCTIONAL REQUIREMENTS

2.3.1 Major Entities

User, Property, Locality, City, Bids, Transaction, Buy, Rent

2.3.2 End User Requirements

Should have Windows platform.

2.3.3 Windows Forms

Window Forms (WinForms) is a graphical (GUI) class library included as a part of Microsoft .NET Framework, providing a platform to write rich client applications for desktop, laptop, and tablet PCs. While it is seen as a replacement for the earlier and more complex C++ based Microsoft Foundation Class Library, it does not offer a comparable paradigm and only acts as a platform for the user interface tier in a multi-tier solution.

A Windows Forms application is an event-driven application supported by Microsoft's .NET Framework. Unlike a batch program, it spends most of its time simply waiting for the user to do something, such as fill in a text box or click a button.

Windows Forms provides access to native Windows User Interface Common Controls by wrapping the existent Windows API in managed code. With the help of Windows Forms, the .NET Framework provides a more comprehensive abstraction above the Win32 API than Visual Basic or MFC did.

Windows Forms is similar to Microsoft Foundation Class(MFC)library in developing client applications. it provides a wrapper consisting of a set of C++ classes for development of Windows applications. However, it does not provide a default application frame work like the MFC. Every control in Windows Forms application is a concrete instance of a class.[from programming of c sharp book]

All visual elements in the Windows Forms class library derive from the Control class. This provides a minimal functionality of a user interface element such as location, size, color, font, text, as well as common events like click and drag/drop. The Control class also has docking support to let a control rearrange its position under its parent. The Microsoft Active Accessibility support in the Control class also helps impaired users to use Windows Forms better.

Besides providing access to native Windows controls like button, textbox, checkbox and listview, Windows Forms added its own controls for ActiveX hosting, layout arrangement, validation and rich data binding. Those controls are rendered using GDI+

The Visual Studio integrated development environment is a creative launching pad that you can use to edit, debug, and build code, and then publish an app. An integrated development environment (IDE) is a feature-rich program that can be used for many aspects of software development. Over and above the standard editor and debugger that most IDEs provide, Visual Studio includes compilers, code completion tools, graphical designers, and many more features to ease the software development process.

2.3.4 C#

C# (pronounced C sharp) is a general-purpose, multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines.[16] It was developed around 2000 by Microsoft within its .NET initiative and later approved as a

standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure.

C#'s development team is led by Anders Hejlsberg. The most recent version is C# 7.3, which was released in 2018 alongside Visual Studio 2017 version 15.7.2.

C# is a general object-oriented programming (OOP) language for networking and Web development. C# is specified as a common language infrastructure (CLI) language.

In January 1999, Dutch software engineer Anders Hejlsberg formed a team to develop C# as a complement to Microsoft's NET framework. Initially, C# was developed as C-Like Object Oriented Language (Cool). The actual name was changed to avert potential trademark issues. In January 2000, NET was released as C#. Its NET framework promotes multiple Web technologies. The term is sometimes spelled as C Sharp or C-Sharp.C# improved and updated many C and C++ features, including the following:

- C# has a strict Boolean data variable type, such as bool, whereas C++ bool variable types may be returned as integers or pointers to avoid common programming errors.
- C# automatically manages inaccessible object memory using a garbage collector, which eliminates developer concerns and memory leaks.
- C# type is safer than C++ and has safe default conversions only (for example, integer widening), which are implemented during compile or runtime.
- No implicit conversions between Booleans, enumeration members and integers (other than 0) may be converted to an enumerated type. User-defined conversions must be specified as explicit or implicit, versus the C++ default implicit conversion operators and copy constructors.

The language is intended to be a simple, modern, general-purpose, object-oriented programming language. The language, and implementations thereof, should provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important.

The language is intended for use in developing software components suitable for deployment in distributed environments. Portability is very important for source code and programmers, especially those already familiar with C and C++. Support for internationalization is very important. C# is intended to be suitable for writing applications for both hosted and embedded systems, ranging from the very large that use sophisticated operating systems, down to the very small having dedicated functions. Although C# applications are intended to

be economical with regard to memory and processing power requirements, the language was not intended to compete directly on performance and size with C or assembly language.

During the development of the .NET Framework, the class libraries were originally written using a managed code compiler system called Simple Managed C (SMC). In January 1999, Anders Hejlsberg formed a team to build a new language at the time called Cool, which stood for "C-like Object Oriented Language". Microsoft had considered keeping the name "Cool" as the final name of the language, but chose not to do so for trademark reasons. By the time the .NET project was publicly announced at the July 2000 Professional Developers Conference, the language had been renamed C#, and the class libraries and ASP.NET runtime had been ported to C#.

Hejlsberg is C#'s principal designer and lead architect at Microsoft, and was previously involved with the design of Turbo Pascal, Embarcadero Delphi (formerly CodeGear Delphi, Inprise Delphi and Borland Delphi), and Visual J++. In interviews and technical papers he has stated that flaws in most major programming languages (e.g. C++, Java, Delphi, and Smalltalk) drove the fundamentals of the Common Language Runtime (CLR), which, in turn, drove the design of the C# language itself.

James Gosling, who created the Java programming language in 1994, and Bill Joy, a co-founder of Sun Microsystems, the originator of Java, called C# an "imitation" of Java; Gosling further said that "[C# is] sort of Java with reliability, productivity and security deleted. Klaus Kreft and Angelika Langer (authors of a C++ streams book) stated in a blog post that "Java and C# are almost identical programming languages. Boring repetition that lacks innovation,Hardly anybody will claim that Java or C# are revolutionary programming languages that changed the way we write programs," and "C# borrowed a lot from Java - and vice versa. Now that C# supports boxing and unboxing, we'll have a very similar feature in Java." In July 2000, Hejlsberg said that C# is "not a Java clone" and is "much closer to C++" in its design.

Since the release of C# 2.0 in November 2005, the C# and Java languages have evolved on increasingly divergent trajectories, becoming two very different languages. One of the first major departures came with the addition of generics to both languages, with vastly different implementations. C# makes use of reification to provide "first-class" generic objects that can be used like any other class, with code generation performed at class-load time. Furthermore, C# has added several major features to accommodate functional-style programming, culminating in the LINQ extensions released with C# 3.0 and its supporting framework of lambda expressions, extension methods, and anonymous types.[27] These features enable

C# programmers to use functional programming techniques, such as closures, when it is advantageous to their application. The LINQ extensions and the functional imports help developers reduce the amount of boilerplate code that is included in common tasks like querying a database, parsing an xml file, or searching through a data structure, shifting the emphasis onto the actual program logic to help improve readability and maintainability.

C# used to have a mascot called Andy (named after Anders Hejlsberg). It was retired on January 29, 2004.

C# was originally submitted to the ISO subcommittee JTC 1/SC 22 for review, under ISO/IEC 23270:2003, was withdrawn and was then approved under ISO/IEC 23270:2006.

2.3.5 MSSQL

MSSQL (Microsoft SQL Server) is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.

Data storage is a database, which is a collection of tables with typed columns. SQL Server supports different data types, including primitive types such as Integer, Float, Decimal, Char (including character strings), Varchar (variable length character strings), binary (for unstructured blobs of data), Text (for textual data) among others. The rounding of floats to integers uses either Symmetric Arithmetic Rounding or Symmetric Round Down (fix) depending on arguments: SELECT Round gives

Microsoft SQL Server also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including views, stored procedures, indexes and constraints, along with a transaction log. A SQL Server database can contain a maximum of 231 objects, and can span multiple OS-level files with a maximum file size of 260 bytes (1 exabyte). The data in the database are stored in primary data files with an extension .mdf. Secondary data files, identified with

a .ndfextension, are used to allow the data of a single database to be spread across more than one file, and optionally across more than one file system. Log files are identified with the .ldf extension.

Storage space allocated to a database is divided into sequentially numbered pages, each 8 KB in size. A page is the basic unit of I/O for SQL Server operations. A page is marked with a 96-byte header which stores metadata about the page including the page number, page type, free space on the page and the ID of the object that owns it. Page type defines the data contained in the page: data stored in the database, index, allocation map which holds information about how pages are allocated to tables and indexes, change map which holds information about the changes made to other pages since last backup or logging, or contain large data types such as image or text. While page is the basic unit of an I/O operation, space is actually managed in terms of an extent which consists of 8 pages. A database object can either span all 8 pages in an extent ("uniform extent") or share an extent with up to 7 more objects ("mixed extent"). A row in a database table cannot span more than one page, so is limited to 8 KB in size. However, if the data exceeds 8 KB and the row contains varchar or varbinary data, the data in those columns are moved to a new page (or possibly a sequence of pages, called an allocation unit) and replaced with a pointer to the data.

For physical storage of a table, its rows are divided into a series of partitions (numbered 1 to n). The partition size is user defined; by default all rows are in a single partition. A table is split into multiple partitions in order to spread a database over a computer cluster. Rows in each partition are stored in either B-tree or heap structure. If the table has an associated, clustered index to allow fast retrieval of rows, the rows are stored in-order according to their index values, with a B-tree providing the index. The data is in the leaf node of the leaves, and other nodes storing the index values for the leaf data reachable from the respective nodes. If the index is non-clustered, the rows are not sorted according to the index keys. An indexed view has the same storage structure as an indexed table. A table without a clustered index is stored in an unordered heap structure. However, the table may have non-clustered indices to allow fast retrieval of rows. In some situations the heap structure has performance advantages over the clustered structure. Both heaps and B-trees can span multiple allocation units.

Buffer management

SQL Server buffers pages in RAM to minimize disk I/O. Any 8 KB page can be buffered in-memory, and the set of all pages currently buffered is called the buffer cache. The amount of

memory available to SQL Server decides how many pages will be cached in memory. The buffer cache is managed by the Buffer Manager. Either reading from or writing to any page copies it to the buffer cache. Subsequent reads or writes are redirected to the in-memory copy, rather than the on-disc version. The page is updated on the disc by the Buffer Manager only if the in-memory cache has not been referenced for some time. While writing pages back to disc, asynchronous I/O is used whereby the I/O operation is done in a background thread so that other operations do not have to wait for the I/O operation to complete. Each page is written along with its checksum when it is written. When reading the page back, its checksum is computed again and matched with the stored version to ensure the page has not been damaged or tampered with in the meantime.

Concurrency and locking

SQL Server allows multiple clients to use the same database concurrently. As such, it needs to control concurrent access to shared data, to ensure data integrity—when multiple clients update the same data, or clients attempt to read data that is in the process of being changed by another client. SQL Server provides two modes of concurrency control: pessimistic concurrency and optimistic concurrency. When pessimistic concurrency control is being used, SQL Server controls concurrent access by using locks. Locks can be either shared or exclusive. Exclusive lock grants the user exclusive access to the data—no other user can access the data as long as the lock is held. Shared locks are used when some data is being read—multiple users can read from data locked with a shared lock, but not acquire an exclusive lock. The latter would have to wait for all shared locks to be released.

Locks can be applied on different levels of granularity—on entire tables, pages, or even on a per-row basis on tables. For indexes, it can either be on the entire index or on index leaves. The level of granularity to be used is defined on a per-database basis by the database administrator. While a fine-grained locking system allows more users to use the table or index simultaneously, it requires more resources, so it does not automatically yield higher performance. SQL Server also includes two more lightweight mutual exclusion solutions—latches and spinlocks—which are less robust than locks but are less resource intensive. SQL Server uses them for DMVs and other resources that are usually not busy. SQL Server also monitors all worker threads that acquire locks to ensure that they do not end up in deadlocks—in case they do, SQL Server takes remedial measures, which in many cases are to kill one of the threads entangled in a deadlock and roll back the transaction it started. To implement locking, SQL Server contains the Lock Manager. The Lock Manager maintains an in-memory table that manages the database objects and locks, if any, on them along with

other metadata about the lock. Access to any shared object is mediated by the lock manager, which either grants access to the resource or blocks it.

SQL Server also provides the optimistic concurrency control mechanism, which is similar to the multiversion concurrency control used in other databases. The mechanism allows a new version of a row to be created whenever the row is updated, as opposed to overwriting the row, i.e., a row is additionally identified by the ID of the transaction that created the version of the row. Both the old as well as the new versions of the row are stored and maintained, though the old versions are moved out of the database into a system database identified as Tempdb. When a row is in the process of being updated, any other requests are not blocked (unlike locking) but are executed on the older version of the row. If the other request is an update statement, it will result in two different versions of the rows—both of them will be stored by the database, identified by their respective transaction IDs.

Data retrieval and programmability

The main mode of retrieving data from a SQL Server database is querying for it. The query is expressed using a variant of SQL called T-SQL, a dialect Microsoft SQL Server shares with Sybase SQL Server due to its legacy. The query declaratively specifies what is to be retrieved. It is processed by the query processor, which figures out the sequence of steps that will be necessary to retrieve the requested data. The sequence of actions necessary to execute a query is called a query plan. There might be multiple ways to process the same query. For example, for a query that contains a join statement and a select statement, executing join on both the tables and then executing select on the results would give the same result as selecting from each table and then executing the join, but result in different execution plans. In such case, SQL Server chooses the plan that is expected to yield the results in the shortest possible time. This is called query optimization and is performed by the query processor itself.

SQL Server includes a cost-based query optimizer which tries to optimize on the cost, in terms of the resources it will take to execute the query. Given a query, then the query optimizer looks at the database schema, the database statistics and the system load at that time. It then decides which sequence to access the tables referred in the query, which sequence to execute the operations and what access method to be used to access the tables. For example, if the table has an associated index, whether the index should be used or not: if the index is on a column which is not unique for most of the columns (low "selectivity"), it might not be worthwhile to use the index to access the data. Finally, it decides whether to execute the query concurrently or not. While a concurrent execution is more costly in terms of total processor time, because the execution is actually split to different processors might

mean it will execute faster. Once a query plan is generated for a query, it is temporarily cached. For further invocations of the same query, the cached plan is used. Unused plans are discarded after some time.

SQL Server also allows stored procedures to be defined. Stored procedures are parameterized T-SQL queries, that are stored in the server itself (and not issued by the client application as is the case with general queries). Stored procedures can accept values sent by the client as input parameters, and send back results as output parameters. They can call defined functions, and other stored procedures, including the same stored procedure (up to a set number of times). They can be selectively provided access to. Unlike other queries, stored procedures have an associated name, which is used at runtime to resolve into the actual queries. Also because the code need not be sent from the client every time (as it can be accessed by name), it reduces network traffic and somewhat improves performance.[25] Execution plans for stored procedures are also cached as necessary.

Services

SQL Server also includes an assortment of add-on services. While these are not essential for the operation of the database system, they provide value added services on top of the core database management system. These services either run as a part of some SQL Server component or out-of-process as Windows Service and presents their own API to control and interact with them.

Machine Learning Services

The SQL Server Machine Learning services operates within the SQL server instance, allowing people to do machine learning and data analytics without having to send data across the network or be limited by the memory of their own computers. The services come with Microsoft's R and Python distributions that contain commonly used packages for data science, along with some proprietary packages (e.g. revoscalepy, RevoScaleR, microsoftml) that can be used to create machine models at scale.

Analysts can either configure their client machine to connect to a remote SQL server and push the script executions to it, or they can run a R or Python scripts as an external script inside a T-SQL query. The trained machine learning model can be stored inside a database and used for scoring.

Service Broker

Used inside an instance, programming environment. For cross-instance applications, Service Broker communicates over TCP/IP and allows the different components to be synchronized,

via exchange of messages. The Service Broker, which runs as a part of the database engine, provides a reliable messaging and message queuing platform for SQL Server applications.

Replication Services

SQL Server Replication Services are used by SQL Server to replicate and synchronize database objects, either in entirety or a subset of the objects present, across replication agents, which might be other database servers across the network, or database caches on the client side. Lulla follows a publisher/subscriber model, i.e., the changes are sent out by one database server ("publisher") and are received by others ("subscribers"). SQL Server supports three different types of replication:

Transaction replication

Each transaction made to the publisher database (master database) is synced out to subscribers, who update their databases with the transaction. Transactional replication synchronizes databases in near real time.

Merge replication

Changes made at both the publisher and subscriber databases are tracked, and periodically the changes are synchronized bi-directionally between the publisher and the subscribers. If the same data has been modified differently in both the publisher and the subscriber databases, synchronization will result in a conflict which has to be resolved, either manually or by using pre-defined policies. rowguid needs to be configured on a column if merge replication is configured.

Snapshot replication

Snapshot replication publishes a copy of the entire database (the then-snapshot of the data) and replicates out to the subscribers. Further changes to the snapshot are not tracked.[36]

Analysis Services

Main article: SQL Server Analysis Services

SQL Server Analysis Services adds OLAP and data mining capabilities for SQL Server databases. The OLAP engine supports MOLAP, ROLAP and HOLAP storage modes for data. Analysis Services supports the XML for Analysis standard as the underlying communication protocol. The cube data can be accessed using MDX and LINQ queries. Data mining specific functionality is exposed via the DMX query language. Analysis Services includes various algorithms—Decision trees, clustering algorithm, Naive Bayes algorithm, time series analysis, sequence clustering algorithm, linear and logistic regression analysis, and neural networks—for use in data mining.

Reporting Services

Main article: SQL Server Reporting Services

SQL Server Reporting Services is a report generation environment for data gathered from SQL Server databases. It is administered via a web interface. Reporting services features a web services interface to support the development of custom reporting applications. Reports are created as RDL files.

Reports can be designed using recent versions of Microsoft Visual Studio (Visual Studio.NET 2003, 2005, and 2008) with Business Intelligence Development Studio, installed or with the included Report Builder. Once created, RDL files can be rendered in a variety of formats including Excel, PDF, CSV, XML, BMP, EMF, GIF, JPEG, PNG, and TIFF, and HTML Web Archive.

Notification Services

Main article: SQL Server Notification Services

Originally introduced as a post-release add-on for SQL Server 2000, Notification Services was bundled as part of the Microsoft SQL Server platform for the first and only time with SQL Server 2005SQL Server Notification Services is a mechanism for generating data-driven notifications, which are sent to Notification Services subscribers. A subscriber registers for a specific event or transaction (which is registered on the database server as a trigger); when the event occurs, Notification Services can use one of three methods to send a message to the subscriber informing about the occurrence of the event. These methods include SMTP, SOAP, or by writing to a file in the filesystem. Notification Services was discontinued by Microsoft with the release of SQL Server 2008 in August 2008, and is no longer an officially supported component of the SQL Server database platform.

Integration Services

Main article: SQL Server Integration Services

SQL Server Integration Services (SSIS) provides ETL capabilities for SQL Server for data import, data integration and data warehousing needs. Integration Services includes GUI tools to build workflows such as extracting data from various sources, querying data, transforming data—including aggregation, de-duplication, de-/normalization and merging of data—and then exporting the transformed data into destination databases or files.

CHAPTER 3

DATABASE DESIGN

3.1 Entities, Attributes and Relationships

- **User** : contains details of the user
- **City** : contains available cuties
- **Locality**: locality in the city
- **Transaction**: transaction of selling property between buyer n seller
- **Property**: contains the details about the property owner and property name
- **Buy** & **Rent**: weak entity containg details of properties

3.2 Identify Major entities, Attributes and Relationships

The following are the Major entities and their attributes:

User : UID, name, email, password, phone, income, address, cash, netAssets

City : Name

Locality: Name, cityname

Transaction: TID, customer_UID, Owner_UID, PID, Price,date

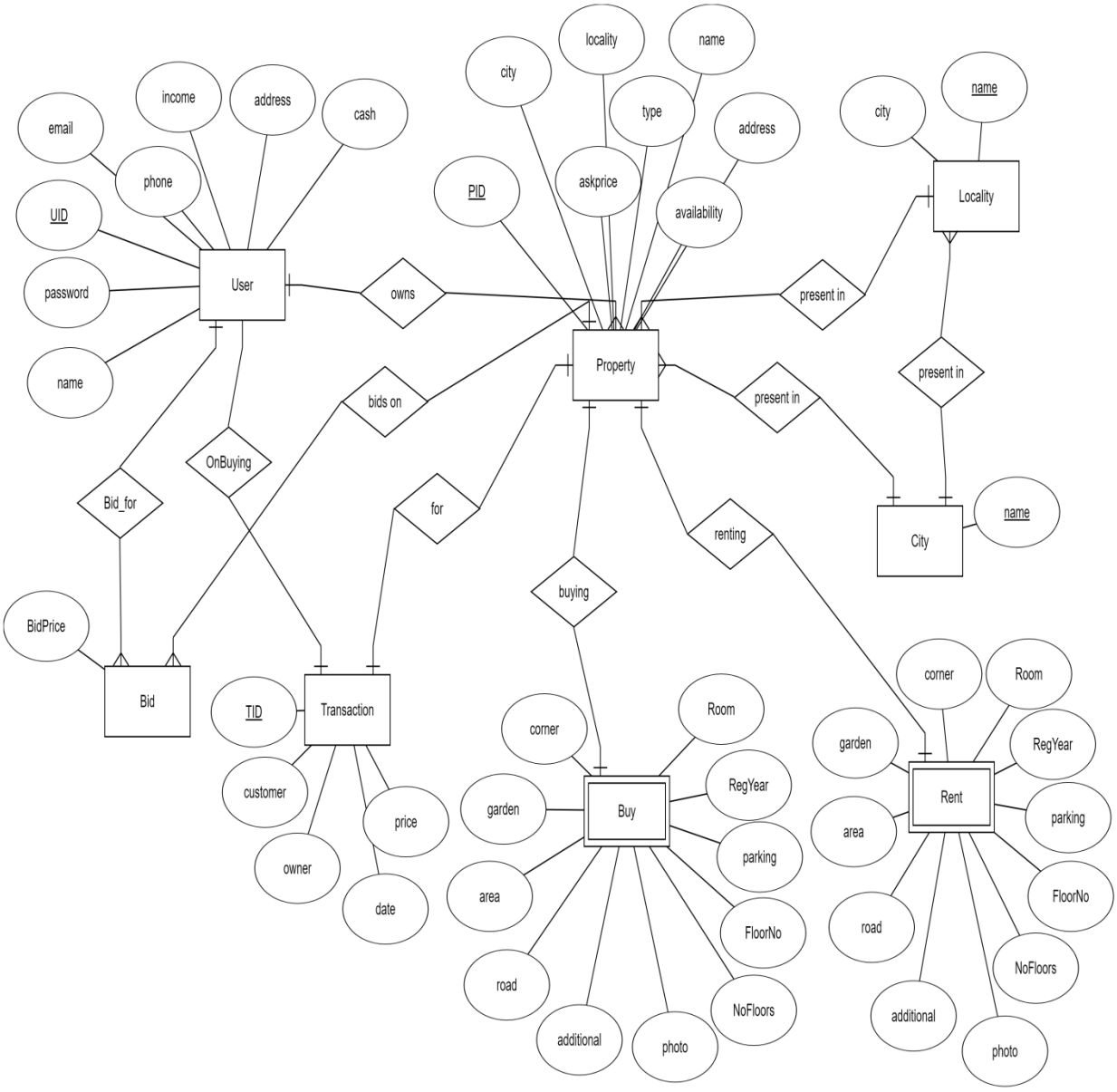
Property: PID,UID,property_name, address, locality_name, city_name, ask_price, availability, type

Buy : PID,Area, Rooms, NoFloors, FloorNo, Parking, Road, RegYear, Garden, Corner,Additional,Photo

Rent: PID,Area, Rooms, NoFloors, FloorNo, Parking, Road, RegYear, Garden, Corner,Additional,Photo

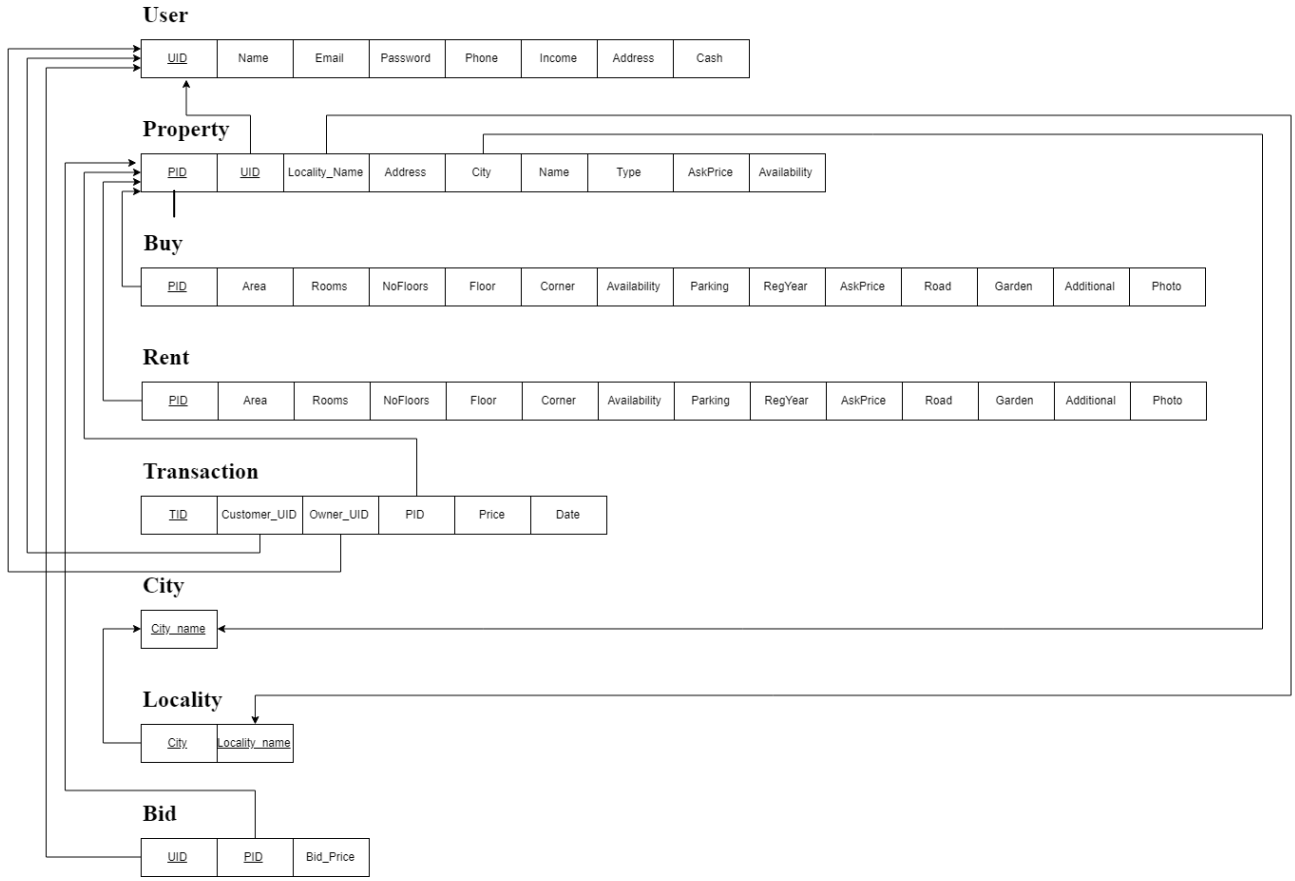
3.3 ER Schema

Fig 3.1 ER Diagram



3.4 Relational Schema

Fig 3.2 Relational Schema



CHAPTER 4

IMPLEMENTATION

4.1 DATABASE CONNECTIVITY

C# and .Net can work with a majority of databases, the most common being Oracle and Microsoft SQL Server. But with every database, the logic behind working with all of them is mostly the same.

In our project, we use the Microsoft SQL Server as our database. For learning purposes, one can download and use the Microsoft SQL Server Express Edition, which is a free database software provided by Microsoft.

FUNCTIONALITY

- **Connection** – To work with the data in a database, the first obvious step is the connection. The connection to a database normally consists of the below-mentioned parameters.
- **Database name or Data Source** – The first important parameter is the database name to which the connection needs to be established. Each connection can only work with one database at a time.
- **Credentials** – The next important aspect is the username and password which needs to be used to establish a connection to the database. It ensures that the username and password have the necessary privileges to connect to the database.
- **Optional parameters** - For each database type, you can specify optional parameters to provide more information on how .net should handle the connection to the database. For example, one can specify a parameter for how long the connection should stay active. If no operation is performed for a specific period of time, then the parameter would determine if the connection has to be closed.
- **Selecting data from the database** – Once the connection has been established, the next important aspect is to fetch the data from the database. C# can execute 'SQL' select command against the database. The 'SQL' statement can be used to fetch data from a specific table in the database.

- **Inserting data into the database** – C# can also be used to insert records into the database. Values can be specified in C# for each row that needs to be inserted into the database.
- **Updating data into the database** – C# can also be used to update existing records into the database. New values can be specified in C# for each row that needs to be updated into the database.
- **Deleting data from a database** – C# can also be used to delete records into the database. Select commands to specify which rows need to be deleted can be specified in C#.
- Ok, now that we have seen the theory of each operation, let's jump into the further sections to look at how we can perform database operations in C#.

Steps to connect c# to Database

Let's now look at the code, which needs to be kept in place to create a connection to a database. we will connect to a database which has the name of xyz. The credentials used to connect to the database are given below

Username – xyz

Password – xyz123

We will see a simple Windows forms application to work with databases. We will have a simple button called "Connect" which will be used to connect to the database.

So let's follow the below steps to achieve this

Step 1) The first step involves the creation of a new project in Visual Studio. After launching Visual Studio, you need to choose the menu option New->Project.

Step 2) The next step is to choose the project type as a Windows Forms application. Here, we also need to mention the name and location of our project.

In the project dialog box, we can see various options for creating different types of projects in Visual Studio. Click the Windows option on the left-hand side.

When we click the Windows options in the previous step, we will be able to see an option for Windows Forms Application. Click this option.

We then give a name for the application which in our case is "DemoApplication". We also need to provide a location to store our application.

Finally, we click the 'OK' button to let Visual Studio to create our project.

Step 3) Now add a button from the toolbox to the Windows form. Put the text property of the Button as Connect. This is how it will look like

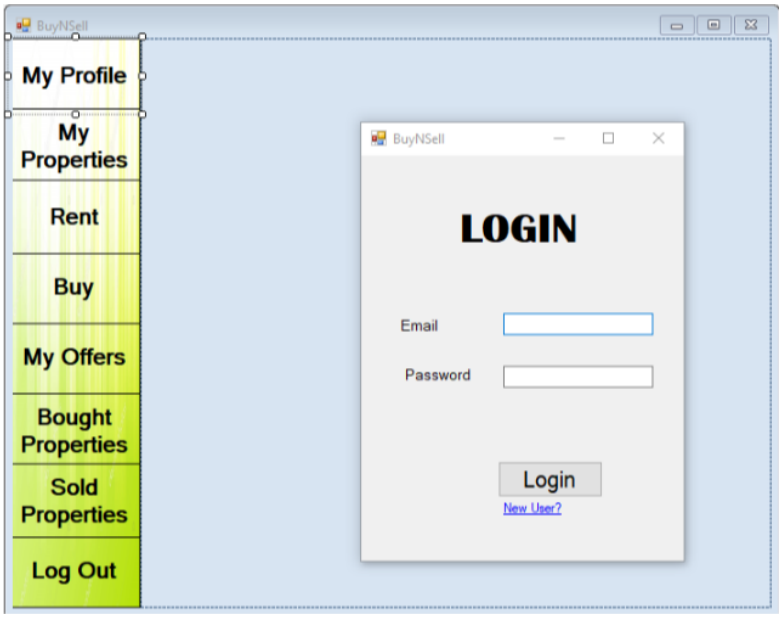
Step 4) Now double click the form so that an event handler is added to the code for the button click event. In the event handler, add the below code.

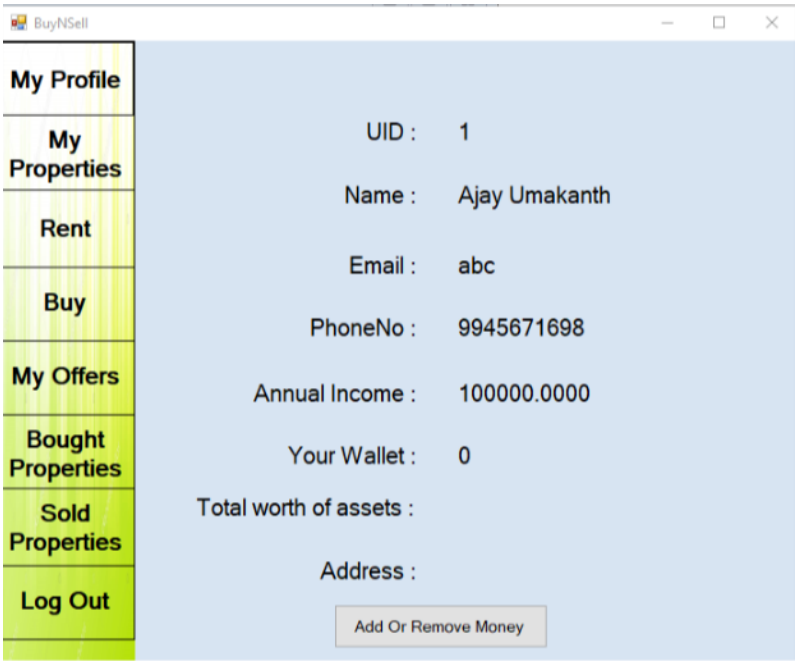
4.2 Pseudo code for a transaction that transfers property between users

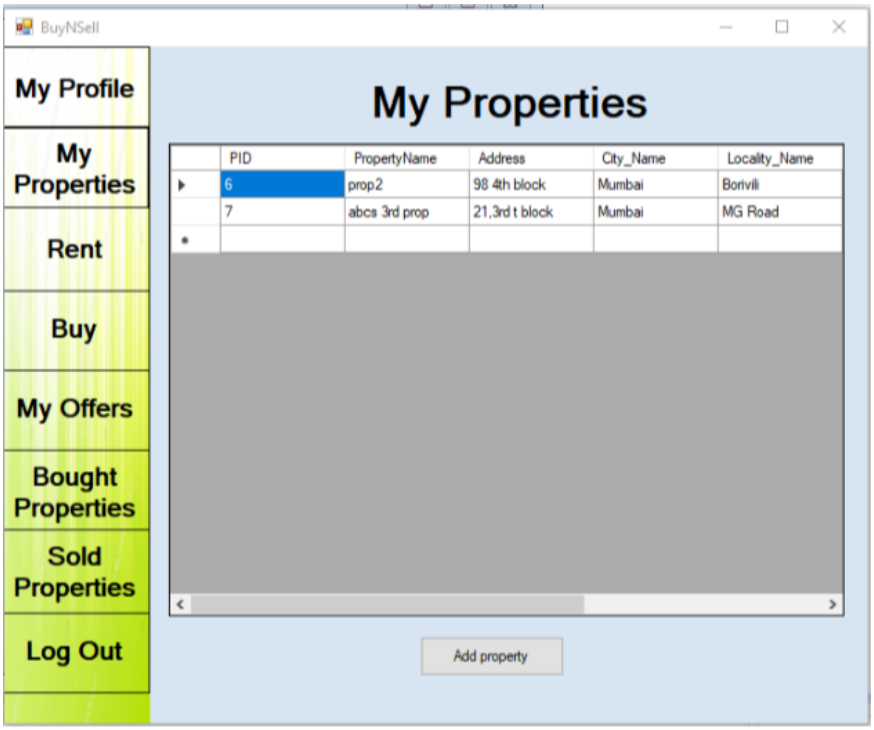
- Deduct offer price from Buyer's wallet
- Credit offer price to seller's wallet
- Update property's UID to buyer's UID
- Set availability as false
- Delete all offers on this property
- Update the transaction table to store the transfer

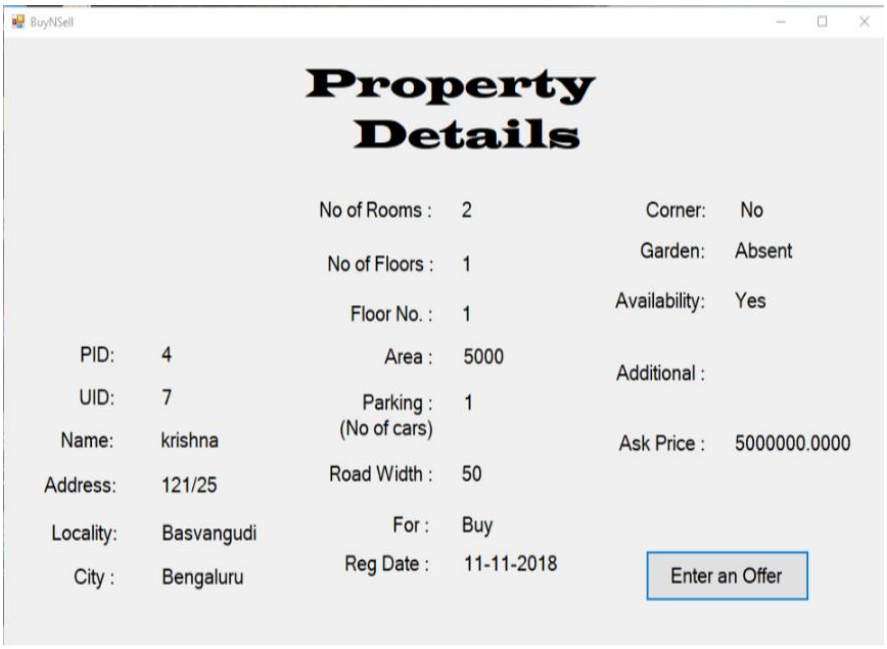
CHAPTER 5

RESULTS, SNAPSHOTS AND DISCUSSIONS






BuyNSell
My Profile
My Properties
Rent
Buy
My Offers
Bought Properties
Sold Properties
Log Out
My Properties
PID
PropertyName
Address
City_Name
Locality_Name
6
prop2
98 4th block
Mumbai
Borivili
7
abcs 3rd prop
21,3rd t block
Mumbai
MG Road
Add property


BuyNSell
Property Details
No of Rooms : 2
No of Floors : 1
Floor No. : 1
Area : 5000
Parking : (No of cars) 1
Road Width : 50
For : Buy
Reg Date : 11-11-2018
Corner: No
Garden: Absent
Availability: Yes
Additional :
Ask Price : 5000000.0000
PID: 4
UID: 7
Name: krishna
Address: 121/25
Locality: Basvangudi
City : Bengaluru
Enter an Offer

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

After completing the initial phases of the project, we are planning to implement machine learning to predict the prices of the property considering many aspects which affect the price of a property so that the user will get to know about the approximate value for which he can buy the property or sell it, so as to reduce the loss which he incurs if he doesn't know the price that his property is worth. Fraud that are happening in Real Estate nowadays will also be reduced. Other enhancements include improving the GUI of our project and building mobile application for the same.

BIBLIOGRAPHY

SQL: https://www.w3schools.com/sql/.

Wikipedia: http://www.wikipedia.com.

StackOverflow: http://stackoverflow.com.

GitHub: http://github.com.

Other Reference: https://www.guru99.com/c-sharp-access-database.html