**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: Matharishwa B USN: 1RN16CS052**

Under the Guidance of:

**Mrs. S Mamatha Jajur ……………….**

**Assistant Professor Assistant Professor**

**Dept. of CSE Dept. of CSE**



**Department of Computer Science and Engineering**

**RNS Institute of Technology**

**Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560 098**

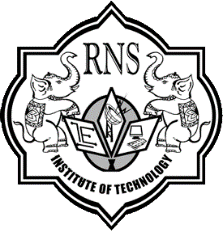
**2017-2018**

**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 **Matharishwa B** bearing USN **1RN16CS052** and **“Ajay Umakanth”** bearing USN **1RN16CS006**, 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.

Date : 27/11/2018 Name: Matharishwa B

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** | | 1.Introduction |  | | * 1. Database technologies |  | | * 1. Characteristics of database approach |  | | * 1. Applications of DBMS   2. Problem description/ statement |  | | 2. Requirements Analysis |  | | 2.1 Hardware Requirements |  | | 2.2 Software Requirements  2.3 Functional Requirements  2.3.1 Major Entities  2.3.2 End User Requirements  2.3.3 c# Applets  2.3.4 c#  2.3.5 SQLite |  | | 3. Database Design |  | | 3.1 Entities, Attributes and Relationships |  | | 3.2 Identify major entities/Attributes and relationships  3.3 ER Schema  3.4 Relational Schema |  | | 4. Implementation  4.1 Database connectivity  4.2 Pseudo code For Major Functionalities |  | | 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. **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**

**(**Description about the major entities**)**

**2.3.2 End User Requirements**

**(**Description about the end user requirements **)**

**2.3.3 C# Applets**

**(**CHANGE THIS PART **)**

A **Java applet** is a small application that is written in the [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29) programming language, or another [programming language](https://en.wikipedia.org/wiki/Programming_language) that compiles to [Java byte code](https://en.wikipedia.org/wiki/Java_bytecode), and delivered to users in the form of Java [byte code](https://en.wikipedia.org/wiki/Bytecode). The user launches the Java applet from a [web page](https://en.wikipedia.org/wiki/Web_page), and the applet is then executed within a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) in a [process](https://en.wikipedia.org/wiki/Process_%28computing%29) separate from the [web browser](https://en.wikipedia.org/wiki/Web_browser) itself. A Java applet can appear in a frame of the web page, a new application window, [Sun](https://en.wikipedia.org/wiki/Sun_Microsystems)'s [AppletViewer](https://en.wikipedia.org/wiki/AppletViewer), or a stand-alone tool for testing applets. Java applets were introduced in the first version of the Java language, which was released in 1995.Java applets are usually written in Java, but other languages such as [Jython](https://en.wikipedia.org/wiki/Jython), [JRuby](https://en.wikipedia.org/wiki/JRuby), [Pascal](https://en.wikipedia.org/wiki/Pascal_%28programming_language%29), [Scala](https://en.wikipedia.org/wiki/Scala_%28programming_language%29), or [Eiffel](https://en.wikipedia.org/wiki/Eiffel_%28programming_language%29) (via [SmartEiffel](https://en.wikipedia.org/wiki/SmartEiffel))may be used as well.

Java applets run at very fast speeds and, until 2011, they were many times faster than [JavaScript](https://en.wikipedia.org/wiki/JavaScript).[[11]](https://en.wikipedia.org/wiki/Java_applet#cite_note-11) Unlike JavaScript, Java applets had access to 3D [hardware acceleration](https://en.wikipedia.org/wiki/Hardware_acceleration), making them well-suited for non-trivial, computation-intensive visualizations. As browsers have gained support for hardware-accelerated graphics thanks to the [canvas](https://en.wikipedia.org/wiki/Canvas_element) technology (or specifically [WebGL](https://en.wikipedia.org/wiki/WebGL) in the case of 3D graphics), as well as [just-in-time compiled](https://en.wikipedia.org/wiki/Just-in-time_compilation) JavaScript, the speed difference has become less noticeable. Since Java bytecode is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) (or platform independent), Java applets can be executed by browsers (or other [clients](https://en.wikipedia.org/wiki/Client_%28computing%29)) for many platforms, including [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows), [FreeBSD](https://en.wikipedia.org/wiki/FreeBSD), [Unix](https://en.wikipedia.org/wiki/Unix), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux). Java applet technology has been marked for [deprecation](https://en.wikipedia.org/wiki/Deprecation).

he Applets are used to provide interactive features to web applications that cannot be provided by [HTML](https://en.wikipedia.org/wiki/HTML) alone. They can capture [mouse input](https://en.wikipedia.org/wiki/Mouse_%28computing%29) and also have controls like [buttons](https://en.wikipedia.org/wiki/Button_%28computing%29) or [check boxes](https://en.wikipedia.org/wiki/Check_box). In response to user actions, an applet can change the provided graphic content. This makes applets well-suited for demonstration, visualization, and teaching. There are online applet collections for studying various subjects, from physics to heart physiology. An applet can also be a text area only; providing, for instance, a cross-platform [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface) to some remote system. If needed, an applet can leave the dedicated area and run as a separate window. However, applets have very little control over web page content outside the applet's dedicated area, so they are less useful for improving the site appearance in general, unlike other types of [browser extensions](https://en.wikipedia.org/wiki/Browser_extension) Applets can also play media in formats that are not natively supported by the browser.

Pages coded in HTML may embed parameters within them that are passed to the applet. Because of this, the same applet may have a different appearance depending on the parameters that were passed.

As applets were available before [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) and [DHTML](https://en.wikipedia.org/wiki/DHTML) were standard, they were also widely used for trivial effects such as [rollover](https://en.wikipedia.org/wiki/Rollover_%28web_design%29) navigation buttons. Heavily criticized, this usage is now declining.

Java applets are executed in a [*sandbox*](https://en.wikipedia.org/wiki/Sandbox_%28security%29) by most web browsers, preventing them from accessing local data like the [clipboard](https://en.wikipedia.org/wiki/Clipboard_%28software%29) or [file system](https://en.wikipedia.org/wiki/File_system). The code of the applet is downloaded from a [web server](https://en.wikipedia.org/wiki/Web_server), after which the browser either [embeds](https://en.wikipedia.org/wiki/Compound_document) the applet into a web page or opens a new window showing the applet's [user interface](https://en.wikipedia.org/wiki/User_interface).

A Java applet extends the class [java.applet.Applet](https://docs.oracle.com/javase/9/docs/api/java/applet/Applet.html), or in the case of a [Swing](https://en.wikipedia.org/wiki/Swing_%28Java%29) applet, [javax.swing.JApplet](https://docs.oracle.com/javase/9/docs/api/javax/swing/JApplet.html). The class which must override methods from the applet class to set up a user interface inside itself (Applet) is a descendant of [Panel](https://docs.oracle.com/javase/9/docs/api/java/awt/Panel.html) which is a descendant of [Container](https://docs.oracle.com/javase/9/docs/api/java/awt/Container.html). As applet inherits from container, it has largely the same user interface possibilities as an ordinary Java application, including regions with user specific visualization.

The first implementations involved downloading an applet class by class. While classes are small files, there are often many of them, so applets got a reputation as slow-loading components. However, since [.jars](https://en.wikipedia.org/wiki/JAR_%28file_format%29) were introduced, an applet is usually delivered as a single file that has a size similar to an image file (hundreds of kilobytes to several megabytes).The [domain](https://en.wikipedia.org/wiki/Domain_Name_System) from where the applet executable has been downloaded is the only domain to which the usual (unsigned) applet is allowed to communicate. This domain can be different from the domain where the surrounding HTML document is hosted.Java [system libraries](https://en.wikipedia.org/wiki/Static_library) and [runtimes](https://en.wikipedia.org/wiki/Runtime_library) are backwards-compatible, allowing one to write code that runs both on current and on future versions of the Java virtual machine.

**Swing** is a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit) for [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29). It is part of [Oracle](https://en.wikipedia.org/wiki/Oracle_Corporation)'s [Java Foundation Classes](https://en.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) – an [API](https://en.wikipedia.org/wiki/Application_programming_interface) for providing a [graphical user interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI [components](https://en.wikipedia.org/wiki/Software_component) than the earlier [Abstract Window Toolkit (AWT)](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit). Swing provides a native [look and feel](https://en.wikipedia.org/wiki/Look_and_feel) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](https://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

Swing is a highly modular-based architecture, which allows for the "plugging" of various custom implementations of specified framework interfaces: Users can provide their own custom implementation(s) of these components to override the default implementations using Java's inheritance mechanism.

Swing is a **component-based framework**, whose components are all ultimately derived from the javax.swing.JComponent class. Swing objects asynchronously fire events, have bound properties, and respond to a documented set of methods specific to the component. Swing components are [Java Beans](https://en.wikipedia.org/wiki/Java_Beans) components, compliant with the Java Beans Component Architecture specifications.

Since early versions of Java, a portion of the [Abstract Window Toolkit](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT) has provided platform-independent APIs for user interface components. In AWT, each component is rendered and controlled by a native peer component specific to the underlying windowing system.

By contrast, Swing components are often described as *lightweight* because they do not require allocation of native resources in the operating system's windowing toolkit. The AWT components are referred to as heavyweight components.

Much of the Swing API is generally a complementary extension of the AWT rather than a direct replacement. In fact, every Swing lightweight interface ultimately exists within an AWT heavyweight component because all of the top-level components in Swing ([JApplet](https://docs.oracle.com/javase/9/docs/api/javax/swing/JApplet.html), [JDialog](https://docs.oracle.com/javase/9/docs/api/javax/swing/JDialog.html), [JFrame](https://docs.oracle.com/javase/9/docs/api/javax/swing/JFrame.html), and [JWindow](https://docs.oracle.com/javase/9/docs/api/javax/swing/JWindow.html)) extend an AWT top-level container.

Swing's high level of flexibility is reflected in its inherent ability to override the native host [operating system](https://en.wikipedia.org/wiki/Operating_system) (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. Thus, a Swing component does not have a corresponding native OS GUI component, and is free to render itself in any way that is possible with the underlying graphics GUIs.

However, at its core, every Swing component relies on an [AWT](https://en.wikipedia.org/wiki/Abstract_Window_Toolkit) container, since (Swing's) [JComponent](https://docs.oracle.com/javase/9/docs/api/javax/swing/JComponent.html) extends (AWT's) Container. This allows Swing to plug into the host OS's GUI management framework, including the crucial device/screen mappings and user interactions, such as key presses or mouse movements. Swing simply "transposes" its own (OS-agnostic) semantics over the underlying (OS-specific) components. So, for example, every Swing component paints its rendition on the graphic device in response to a call to component.paint(), which is defined in (AWT) Container. But unlike AWT components, which delegated the painting to their OS-native "heavyweight" widget, Swing components are responsible for their own rendering.

This transposition and decoupling is not merely visual, and extends to Swing's management and application of its own OS-independent semantics for events fired within its component containment hierarchies. Generally speaking, the Swing architecture delegates the task of mapping the various flavors of OS GUI semantics onto a simple, but generalized, pattern to the AWT container. Building on that generalized platform, it establishes its own rich and complex GUI semantics in the form of the [JComponent](https://docs.oracle.com/javase/9/docs/api/javax/swing/JComponent.html) model.

**2.3.4 C#**

**C#** (pronounced C [sharp](https://en.wikipedia.org/wiki/Sharp_(music))) is a general-purpose, [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) encompassing [strong typing](https://en.wikipedia.org/wiki/Strong_typing), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [declarative](https://en.wikipedia.org/wiki/Declarative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming), [generic](https://en.wikipedia.org/wiki/Generic_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) ([class](https://en.wikipedia.org/wiki/Class_(computer_science))-based), and [component-oriented](https://en.wikipedia.org/wiki/Component-based_software_engineering) programming disciplines.[[16]](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-ECMA-334-18) It was developed around 2000 by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) within its .NET initiative and later approved as a standard by [Ecma](https://en.wikipedia.org/wiki/Ecma_International" \o "Ecma International) (ECMA-334) and [ISO](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO/IEC 23270:2006). C# is one of the programming languages designed for the [CommonLanguageInfrastructure](https://en.wikipedia.org/wiki/Common_Language_Infrastructure).

C#'s development team is led by [AndersHejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg). The most recent version is C# 7.3, which was released in 2018 alongside [Visual Studio](https://en.wikipedia.org/wiki/Microsoft_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](https://en.wikipedia.org/wiki/Object-oriented_programming) language. The language, and implementations thereof, should provide support for software engineering principles such as [strong type](https://en.wikipedia.org/wiki/Strong_type) checking, array [bounds checking](https://en.wikipedia.org/wiki/Bounds_checking), detection of attempts to use [uninitialized variables](https://en.wikipedia.org/wiki/Uninitialized_variable), and automatic [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). Software robustness, durability, and programmer productivity are important.

The language is intended for use in developing [software components](https://en.wikipedia.org/wiki/Software_components) suitable for deployment in distributed environments. Portability is very important for source code and programmers, especially those already familiar with [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B). Support for [internationalization](https://en.wikipedia.org/wiki/Internationalization_and_localization) is very important. C# is intended to be suitable for writing applications for both hosted and [embedded systems](https://en.wikipedia.org/wiki/Embedded_system), ranging from the very large that use sophisticated [operating systems](https://en.wikipedia.org/wiki/Operating_system), down to the very small having dedicated functions. Although C# applications are intended to be economical with regard to memory and [processing power](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Base_Class_Library) were originally written using a [managed code](https://en.wikipedia.org/wiki/Managed_code) compiler system called Simple Managed C (SMC). In January 1999, [Anders Hejlsberg](https://en.wikipedia.org/wiki/Anders_Hejlsberg) formed a team to build a new language at the time called Cool, which stood for "[C-like](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Professional_Developers_Conference), the language had been renamed C#, and the class libraries and [ASP.NET](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Turbo_Pascal), [Embarcadero Delphi](https://en.wikipedia.org/wiki/Embarcadero_Delphi) (formerly CodeGear Delphi, Inprise Delphi and Borland Delphi), and [Visual J++](https://en.wikipedia.org/wiki/Visual_J%2B%2B). In interviews and technical papers he has stated that flaws in most major programming languages (e.g. [C++](https://en.wikipedia.org/wiki/C%2B%2B), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Delphi](https://en.wikipedia.org/wiki/Embarcadero_Delphi), and [Smalltalk](https://en.wikipedia.org/wiki/Smalltalk)) drove the fundamentals of the [Common Language Runtime](https://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), which, in turn, drove the design of the C# language itself.

[James Gosling](https://en.wikipedia.org/wiki/James_Gosling), who created the [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) programming language in 1994, and [Bill Joy](https://en.wikipedia.org/wiki/Bill_Joy), a co-founder of [Sun Microsystems](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Boxing_(Computer_Science)) 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](https://en.wikipedia.org/wiki/Generic_programming) to both languages, with vastly different implementations. C# makes use of [reification](https://en.wikipedia.org/wiki/Reification_(computer_science)) 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](https://en.wikipedia.org/wiki/Language_Integrated_Query) extensions released with C# 3.0 and its supporting framework of [lambda expressions](https://en.wikipedia.org/wiki/Lambda_expressions), [extension methods](https://en.wikipedia.org/wiki/Extension_method), and [anonymous types](https://en.wikipedia.org/wiki/Anonymous_type).[[27]](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)#cite_note-29) These features enable C# programmers to use functional programming techniques, such as [closures](https://en.wikipedia.org/wiki/Closure_(computer_science)), when it is advantageous to their application. The LINQ extensions and the functional imports help developers reduce the amount of [boilerplate code](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/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 SQLite**

**SQLite** is a relational database management system contained in a C programming library. In contrast to many other database management systems, SQLite is not a client–server database engine. Rather, it is embedded into the end program.

SQLite is ACID-compliant and implements most of the SQL standard, using a dynamically and weakly typed SQL syntax that does not guarantee the domain integrity.

SQLite is a popular choice as embedded database software for local/client storage in application software such as web browsers. It is arguably the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems (such as mobile phones), among others. SQLite has bindings to many programming languages.

Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. Linking may be static or dynamic. The application program uses SQLite's functionality through simple function calls, which reduce latency in database access: function calls within a single process are more efficient than inter-process communication. SQLite stores the entire database (definitions, tables, indices, and the data itself) as a single cross-platform file on a host machine. It implements this simple design by locking the entire database file during writing. SQLite read operations can be multitasked, though writes can only be performed sequentially.

Due to the server-less design, SQLite applications require less configuration than client-server databases. SQLite is called zero-confbecause it does not require service management (such as startup scripts) or access control based on GRANT and passwords. Access control is handled by means of file system permissions given to the database file itself. Databases in client-server systems use file system permissions which give access to the database files only to the daemon process.

Another implication of the serverless design is that several processes may not be able to write to the database file. In server-based databases, several writers will all connect to the same daemon, which is able to handle its locks internally. SQLite on the other hand has to rely on file-system locks. It has less knowledge of the other processes that are accessing the database at the same time. Therefore, SQLite is not the preferred choice for write-intensive deployments.However, for simple queries with little concurrency, SQLite performance profits from avoiding the overhead of passing its data to another process.

SQLite uses PostgreSQL as a reference platform. “What would PostgreSQL do is used to make sense of the SQL standard. One major deviation is that, with the exception of primary keys, SQLite does not enforce type checking; the type of a value is dynamic and not strictly constrained by the schema (although the schema will trigger a conversion when storing, if such a conversion is potentially reversible). SQLite strives to follow Postel's Rule.

SQLite implements most of the SQL-92 standard for SQL but it lacks some features. For example, it partially provides triggers, and it cannot write to views (however it provides INSTEAD OF triggers that provide this functionality). While it provides complex queries, it still has limited ALTER TABLE function, as it cannot modify or delete columns.

SQLite uses an unusual type system for an SQL-compatible DBMS; instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values; in language terms it is *dynamically typed*. Moreover, it is *weakly typed* in some of the same ways that Perl is: one can insert a string into an integer column (although SQLite will try to convert the string to an integer first, if the column's preferred type is integer). This adds flexibility to columns, especially when bound to a dynamically typed scripting language. However, the technique is not portable to other SQL products. A common criticism is that SQLite's type system lacks the data integrity mechanism provided by statically typed columns in other products. The SQLite web site describes a "strict affinity" mode, but this feature has not yet been added.However, it can be implemented with constraints like CHECK(typeof(x)='integer'). Tables normally include a hidden *rowid* index column which gives faster access.[17] If a database includes an Integer Primary Key column SQLite will typically optimize it by treating it as an alias for rowid, causing the contents to be stored as a strictly typed 64-bit signed integer and changing its behavior to be somewhat like an auto-incrementing column. Future versions of SQLite may include a command to introspect whether a column has behavior like that of rowid to differentiate these columns from weakly-typed, non-autoincrementing Integer Primary Keys.

SQLite with full Unicode function is optional.

Several computer processes or threads may access the same database concurrently. Several read accesses can be satisfied in parallel. A write access can only be satisfied if no other accesses are currently being serviced. Otherwise, the write access fails with an error code (or can automatically be retried until a configurable timeout expires). This concurrent access situation would change when dealing with temporary tables. This restriction is relaxed in version 3.7 when write-ahead logging (WAL) is turned on enabling concurrent reads and writes.

SQLite version 3.7.4 first saw the addition of the FTS4 (full text search) module, which features enhancements over the older FTS3 module.FTS4 allows users to perform full text searches on documents similar to how search engines search webpages. Version 3.8.2 added support for creating tables without rowid,[23] which may provide space and performance improvements. Common table expressions support was added to SQLite in version 3.8.3.

In 2015, with the json1 extensionand new subtype interfaces, SQLite version 3.9 introduced JSON content managing.

**CHAPTER 3**

**DATABASE DESIGN**

**3.1 Entities, Attributes and Relationships**

(Description to be filled according to your project)

**3.2 Identify Major entities, attributes and relationships**

(Description to be filled according to your project)

**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**

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is Java based data access technology and used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

**FUNCTIONALITY**

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection. JDBC represents statements using one of the following classes:

* Statement – the statement is sent to the database server each and every time.
* PreparedStatement – the statement is cached and then the execution path is pre-determined on the database server allowing it to be executed multiple times in an efficient manner.
* CallableStatement – used for executing stored procedures on the database.

Update statements such as INSERT, UPDATE and DELETE return an update count that indicates how many rows were affected in the database. These statements do not return any other information.

Query statements return a JDBC row result set. The row result set is used to walk over the result set. Individual columns in a row are retrieved either by name or by column number. There may be any number of rows in the result set. The row result set has metadata that describes the names of the columns and their types.

There is an extension to the basic JDBC API in the javax.sql.

JDBC connections are often managed via a connection pool rather than obtained directly from the driver.

**JDBC DRIVERS**

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. Types Commercial and free drivers provide connectivity to most relational-database servers. These drivers fall into one of the following types:

a. Type 1 that calls native code of the locally available ODBC driver.

b. Type 2 that calls database vendor native library on a client side. This code then talks to database over the network.

c. Type 3, the pure-java driver that talks with the server-side middleware that then talks to the database.

d. Type 4, the pure-java driver that uses database native protocol.

Note also a type called an internal JDBC driver - a driver embedded with JRE in Java-enabled SQL databases. It is used for Java stored procedures.

**Steps to connect to the database in java**

There are 5 steps to connect any java application with the database in java using JDBC.

They are as follows:

1. Register the driver class
2. Creating connection
3. Creating statement
4. Executing queries
5. Closing connection

**Register the driver class**

The forName() method of Class class is used to register the driver class. This method is used to dynamically load the driver class.

Syntax of forName() method :

public static void forName(String className )throws ClassNotFoundException .

**Create the connection object**

The getConnection() method of DriverManager class is used to establish connection with the database.

Syntax of getConnection() method:

* public static Connection getConnection(String url)throws SQLException
* public static Connection getConnection(String url,String name,String password) throws SQLException

**Create the Statement object**

The createStatement() method of Connection interface is used to create statement. The object of statement is responsible to execute queries with the database.

Syntax of createStatement() method:

public Statement createStatement()throws SQLException

**Execute the query**

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

Syntax of executeQuery() method:

public ResultSet executeQuery(String sql)throws SQLException

**Close the connection object**

By closing connection object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

Syntax of close() method:

public void close()throws SQLException

**4.2 Pseudo Code for Major Functionalities**

**(Respective pseudo code for different functionalities to be inserted)**

**CHAPTER 5**

**RESULTS, SNAPSHOTS AND DISCUSSIONS**

(Snapshots to be inserted and description of each snapshot to be written. Each snapshot to be given as Fig 5.1 and so on and what the figure is)

**CHAPTER 6**

**CONCLUSION AND FUTURE ENHANCEMENTS**

(According to your project)

**BIBLIOGRAPHY**

(According to your project)