***INTRODUCTION TO PROJECT***

Our project is based on an offline version of justdial or we may say an offline replica of justdial using java and SQL. It is a desktop application.

It tells some categories on thing nearby which have been saved in the database it just retrieves the data and show to the user and if the user wants to change the details it can be changed. It helps to locate the nearby shops, garages easily.

To enter the you need to login without it you can’t enter the app. This app also let us add new entries from user so as we can get more of data to show other user. It also increases the efficiency of the app.

It also lets the user delete the data which lets the app to get rid of incorrect data. The modification of data is also possible it lets correction in data which improve the app performance.

The most important option which is provided that is search which let the user to search the nearby location based on the criteria provided in the app to get the nearby location present in the database.

***TECHNOLOGY USED***

The technology used are: -

1. JAVA
2. SQL

***JAVA***

It is a programming language widely used across world among programmer for developing app, websites and many other things.

This language was being in programming world by James Gosling in 1991. It was brought to overcome the features of C/C++.

The language can be used to develop both offline and online applications. Unlike C++ it’s a complete or pure object-oriented language.

This language is different from other as it does not just compile the code and runs it but first it compiles the code then convert it into a byte code and then interprets it and executes. The benefit of creating the byte code makes the language architecturally neutral and portable.

To execute we use Java Virtual Machine (JVM). JVM is basically used to interpret and execute the byte code.

Java provided support to the coder by providing an endless inbuilt library which helps to create application designs.

There were five primary goals in the creation of the Java language:

1. It must be simple, object-oriented, and familiar.
2. It must be robust and secure.
3. It must be architecture-neutral and portable.
4. It must execute with high performance.
5. It must be interpreted, threaded, and dynamic.

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate run time support. This is achieved by compiling the Java language code to an intermediate representation called [Java bytecode](https://en.wikipedia.org/wiki/Java_bytecode), instead of directly to architecture-specific [machine code](https://en.wikipedia.org/wiki/Machine_code). Java bytecode instructions are analogous to machine code, but they are intended to be executed by a [virtual machine](https://en.wikipedia.org/wiki/Virtual_machine) (VM) written specifically for the host hardware. [End users](https://en.wikipedia.org/wiki/End_user) commonly use a [Java Runtime Environment](https://en.wikipedia.org/wiki/Java_virtual_machine) (JRE) installed on their own machine for standalone Java applications, or in a web browser for [Java applets](https://en.wikipedia.org/wiki/Java_applet).

Standard libraries provide a generic way to access host-specific features such as graphics, [threading](https://en.wikipedia.org/wiki/Thread_(computing)), and [networking](https://en.wikipedia.org/wiki/Computer_network).

The use of universal bytecode makes porting simple. However, the overhead of [interpreting](https://en.wikipedia.org/wiki/Interpreter_(computing)) bytecode into machine instructions made interpreted programs almost always run more slowly than native [executables](https://en.wikipedia.org/wiki/Executable). [Just-in-time](https://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) compilers that compile byte-codes to machine code during runtime were introduced from an early stage. Java itself is platform-independent and is adapted to the particular platform it is to run on by a [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) for it, which translates the [Java bytecode](https://en.wikipedia.org/wiki/Java_bytecode) into the platform's machine language.

Java uses an [automatic garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) to manage memory in the [object lifecycle](https://en.wikipedia.org/wiki/Object_lifetime). The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the [unreachable memory](https://en.wikipedia.org/wiki/Unreachable_memory) becomes eligible to be freed automatically by the garbage collector. Something similar to a [memory leak](https://en.wikipedia.org/wiki/Memory_leak) may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a non-existent object are called, a [null pointer](https://en.wikipedia.org/wiki/Null_pointer) exception is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the [stack](https://en.wikipedia.org/wiki/Stack_(abstract_data_type)) or explicitly allocated and deallocated from the [heap](https://en.wikipedia.org/wiki/Memory_management#DYNAMIC). In the latter case, the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a [memory leak](https://en.wikipedia.org/wiki/Memory_leak) occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable or crash. This can be partially remedied by the use of [smart pointers](https://en.wikipedia.org/wiki/Smart_pointer), but these add overhead and complexity. Note that garbage collection does not prevent [logical memory](https://en.wikipedia.org/wiki/Logical_address) leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style [pointer arithmetic](https://en.wikipedia.org/wiki/Pointer_(computer_programming)), where object addresses can be arithmetically manipulated (e.g. by adding or subtracting an offset). This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's [primitive data types](https://en.wikipedia.org/wiki/Primitive_data_type) are either stored directly in fields (for objects) or on the [stack](https://en.wikipedia.org/wiki/Stack-based_memory_allocation) (for methods) rather than on the heap, as is commonly true for non-primitive data types (but see [escape analysis](https://en.wikipedia.org/wiki/Escape_analysis)). This was a conscious decision by Java's designers for performance reasons.

Java contains multiple types of garbage collectors. By default, HotSpot uses the [parallel scavenge garbage collector](https://en.wikipedia.org/w/index.php?title=Parallel_scavenge_garbage_collector&action=edit&redlink=1). However, there are also several other garbage collectors that can be used to manage the heap. For 90% of applications in Java, the [Concurrent Mark-Sweep](https://en.wikipedia.org/wiki/Concurrent_mark_sweep_collector) (CMS) garbage collector is sufficient. Oracle aims to replace CMS with the [Garbage-First Collector](https://en.wikipedia.org/wiki/Garbage-first_collector) (G1).

Having solved the memory management problem does not relieve the programmer of the burden of handling properly other kind of resources, like network or database connections, file handles, etc., especially in the presence of exceptions. Paradoxically, the presence of a garbage collector has faded the necessity of having an explicit destructor method in the classes, thus rendering the management of these other resources more difficult.

The syntax of Java is largely influenced by [C++](https://en.wikipedia.org/wiki/C%2B%2B). Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language.[[18]](https://en.wikipedia.org/wiki/Java_(programming_language)#cite_note-design_goals-18) All code is written inside classes, and every data item is an object, with the exception of the primitive data types, (i.e. integers, floating-point numbers, [boolean values](https://en.wikipedia.org/wiki/Boolean_data_type" \o "Boolean data type), and characters), which are not objects for performance reasons. Java reuses some popular aspects of C++ (such as the printf method).

Unlike C++, Java does not support [operator overloading](https://en.wikipedia.org/wiki/Operator_overloading) or [multiple inheritance](https://en.wikipedia.org/wiki/Multiple_inheritance) for classes, though multiple inheritance is supported for [interfaces](https://en.wikipedia.org/wiki/Interface_(Java)).

Java uses comments similar to those of C++. There are three different styles of comments: a single line style marked with two slashes (//), a multiple line style opened with /\* and closed with \*/, and the [Javadoc](https://en.wikipedia.org/wiki/Javadoc) commenting style opened with /\*\* and closed with \*/. The Javadoc style of commenting allows the user to run the Javadoc executable to create documentation for the program and can be read by some [integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDEs) such as [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software)) to allow developers to access documentation within the IDE.

***SQL***

It is language which allows user to store data in organized format. It follows all rules RDBMS i.e. Relational Database Management System.

SQL is a [domain-specific language](https://en.wikipedia.org/wiki/Domain-specific_language) used in programming and designed for managing data held in a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS), or for stream processing in a [relational data stream management system](https://en.wikipedia.org/wiki/Relational_data_stream_management_system) (RDSMS). It is particularly useful in handling [structured data](https://en.wikipedia.org/wiki/Data_model) where there are relations between different entities/variables of the data. SQL offers two main advantages over older read/write [APIs](https://en.wikipedia.org/wiki/API) like [ISAM](https://en.wikipedia.org/wiki/ISAM) or [VSAM](https://en.wikipedia.org/wiki/VSAM). First, it introduced the concept of accessing many records with one single command; and second, it eliminates the need to specify how to reach a record, e.g. with or without an [index](https://en.wikipedia.org/wiki/Database_index).

Originally based upon [relational algebra](https://en.wikipedia.org/wiki/Relational_algebra) and [tuple relational calculus](https://en.wikipedia.org/wiki/Tuple_relational_calculus), SQL consists of many types of statements, which may be informally classed as [sublanguages](https://en.wikipedia.org/wiki/Sublanguage), commonly:

1. a [data query language](https://en.wikipedia.org/wiki/Data_query_language) (DQL)
2. a [data definition language](https://en.wikipedia.org/wiki/Data_definition_language) (DDL)
3. a [data control language](https://en.wikipedia.org/wiki/Data_control_language) (DCL)
4. a [data manipulation language](https://en.wikipedia.org/wiki/Data_manipulation_language) (DML).

The scope of SQL includes data query, data manipulation (insert, update and delete), data definition ([schema](https://en.wikipedia.org/wiki/Database_schema) creation and modification), and data access control. Although SQL is often described as, and to a great extent is, a [declarative language](https://en.wikipedia.org/wiki/Declarative_programming) ([4GL](https://en.wikipedia.org/wiki/4GL)), it also includes [procedural](https://en.wikipedia.org/wiki/Procedural_programming) elements.

SQL was one of the first commercial languages for [Edgar F. Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd)'s [relational model](https://en.wikipedia.org/wiki/Relational_model). The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to [the relational model as described by Codd](https://en.wikipedia.org/wiki/Codd%27s_12_rules), it became the most widely used database language.

SQL became a [standard](https://en.wikipedia.org/wiki/Technical_standard) of the [American National Standards Institute](https://en.wikipedia.org/wiki/American_National_Standards_Institute) (ANSI) in 1986, and of the [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

SQL was initially developed at [IBM](https://en.wikipedia.org/wiki/IBM) by [Donald D. Chamberlin](https://en.wikipedia.org/wiki/Donald_D._Chamberlin) and [Raymond F. Boyce](https://en.wikipedia.org/wiki/Raymond_F._Boyce) after learning about the relational model from [Ted Codd](https://en.wikipedia.org/wiki/Ted_Codd) in the early 1970s. This version, initially called SEQUEL (Structured English Query Language), was designed to manipulate and retrieve data stored in IBM's original quasi-relational database management system, [System R](https://en.wikipedia.org/wiki/IBM_System_R), which a group at [IBM San Jose Research Laboratory](https://en.wikipedia.org/wiki/IBM_Almaden_Research_Center) had developed during the 1970s.

Chamberlin and Boyce's first attempt of a relational database language was Square, but it was difficult to use due to subscript notation. After moving to the San Jose Research Laboratory in 1973, they began work on SEQUEL. The acronym SEQUEL was later changed to SQL because "SEQUEL" was a [trademark](https://en.wikipedia.org/wiki/Trademark) of the [UK-based](https://en.wikipedia.org/wiki/United_Kingdom) [Hawker Siddeley](https://en.wikipedia.org/wiki/Hawker_Siddeley) Dynamics Engineering Limited company.

After testing SQL at customer test sites to determine the usefulness and practicality of the system, IBM began developing commercial products based on their System R prototype including [System/38](https://en.wikipedia.org/wiki/IBM_System/38), [SQL/DS](https://en.wikipedia.org/wiki/IBM_SQL/DS), and [DB2](https://en.wikipedia.org/wiki/IBM_DB2), which were commercially available in 1979, 1981, and 1983, respectively.

In the late 1970s, Relational Software, Inc. (now [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation)) saw the potential of the concepts described by Codd, Chamberlin, and Boyce, and developed their own SQL-based [RDBMS](https://en.wikipedia.org/wiki/RDBMS) with aspirations of selling it to the [U.S. Navy](https://en.wikipedia.org/wiki/United_States_Navy), [Central Intelligence Agency](https://en.wikipedia.org/wiki/Central_Intelligence_Agency), and other [U.S. government](https://en.wikipedia.org/wiki/Federal_government_of_the_United_States) agencies. In June 1979, Relational Software, Inc. introduced the first commercially available implementation of SQL, [Oracle](https://en.wikipedia.org/wiki/Oracle_Database) V2 (Version2) for [VAX](https://en.wikipedia.org/wiki/VAX) computers.

By 1986, [ANSI](https://en.wikipedia.org/wiki/ANSI) and [ISO](https://en.wikipedia.org/wiki/ISO) standard groups officially adopted the standard "Database Language SQL" language definition. New versions of the standard were published in 1989, 1992, 1996, 1999, 2003, 2006, 2008, 2011, and most recently, 2016.

In this language we use queries to execute. There are many software providing this language but we have used MySQL.

***HARDWARE/SOFTWARE REQUIREMENTS***

* Intel CORE i3
* 2 GB RAM
* Windows 7/8/10, Ubuntu 14.06/16.06/19.06
* HDD 5 GB FREE
* JDK 1.6 and above
* Eclipse or NetBeans
* MySQL or Oracle

***DESCRIPTION OF SOFTWARE LIBRARY USED***

Main Library used are: -

1. Swing
2. SQL jar External
3. Awt

Sub Library used are: -

1. Font
2. JLabel
3. JFrame
4. Color
5. Font
6. Image
7. ImageIcon
8. JPanel
9. JButton
10. ActionListener
11. BorderFactory
12. JTextField
13. FlowLayout
14. JOptionPane
15. Connection
16. DriverManager
17. DriverManager
18. ResultSet
19. SQLException
20. Statement

***java.awt.Font: -***

This class is used for adding font, font size, font type (like bold, italics etc.). it can set it anywhere in code one initialized.

For ex: - f1 = **new** Font("Serif",Font.***BOLD***,25);

***java.awt.Color: -***

This class help to color anything in the code like giving color to a background or giving color to text or a button or to border of panel or background color of a panel.

For ex: - label.setForeground(Color.***white***);

***java.awt.Image: -***

This class help to give changes to a given image like changing size, resizing it etc.

For ex: - Image temp\_img = i.getScaledInstance(1920, 1200, Image.***SCALE\_SMOOTH***);

***java.awt.event.ActionListener: -***

This class help to activate button i.e. to provide operation to the that’s to be performed after clicking it.

For ex: - but3.addActionListener(**this**);

***java.swing.FlowLayout: -***

This class is for setting the frame items in arrangement without providing items there location in a linear format.

For ex: - FlowLayout layout =new FlowLayout();

login.setLayout(layout);

***java.swing.ImageIcon: -***

This class helps to add image the frame on which everything is added. Basically it helps to get the image on the screen i.e. to get the image from the system to the displaying area.

For ex: - img = new ImageIcon("src\\images.jpg ");

***java.swing.JButton: -***

This class helps to create button in the application and can change the properties of button like size, placement, color etc.

For ex: - okay = new JButton("Login ");

***java.swing.JLabel: -***

This class helps to create a label and also can change the properties of label like size, intext, placement etc.

For ex: - lblu = new JLabel("Enter user name here");

***java.swing.JPanel: -***

This class helps to create a panel in the application and let’s change the properties like size, border, placement etc.

For ex: - p1 = new JPanel();

p1.setBounds(500, 100, 400, 180);

***java.swing.JFrame: -***

This class helps to create and manipulate a frame.

For ex: - login = new JFrame("Login window ");

***java.swing.JOptionPane: -***

This class helps to print a message required to known that either a task is performed mostly after pressing the button.

For ex: - JOptionPane.showMessageDialog(login,"Sorry not registered");

***java.swing.JTextField: -***

This class helps to create and manipulate a text box.

For ex: - t3 = new JTextField(30);

***java.sql.Connection: -***

This class let’s us establish a connection between java program and database.

For ex: - con = DriverManager.getConnection("jdbc:mysql://localhost/aroundme", "root", "root");

***java.sql.DriverManager: -***

This class a driver which lets connection class to establish connection between java program and database.

For ex: - con = DriverManager.getConnection("jdbc:mysql://localhost/aroundme", "root", "root");

***java.sql.PreparedStatement: -***

This class helps to execute SQL statement in which there are missing entries and need to get it from user.

For ex: - pstmt = con.prepareStatement(sql);

***java.sql.ResultSet: -***

This class creates a virtual pointer in the SQL table so it can be accessed by the user for insertion, updating, deleting and searching.

For ex: - rs = stmt.executeQuery(msg);

***java.sql.Statement: -***

This class helps to execute SQL statement in which there is execution or updating.

For ex: - stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,ResultSet.CONCUR\_UPDATABLE);

***java.sql.SQLException: -***

This class is for exception got during programs.

For ex: - catch (SQLException e)***SCREENSHOTS***

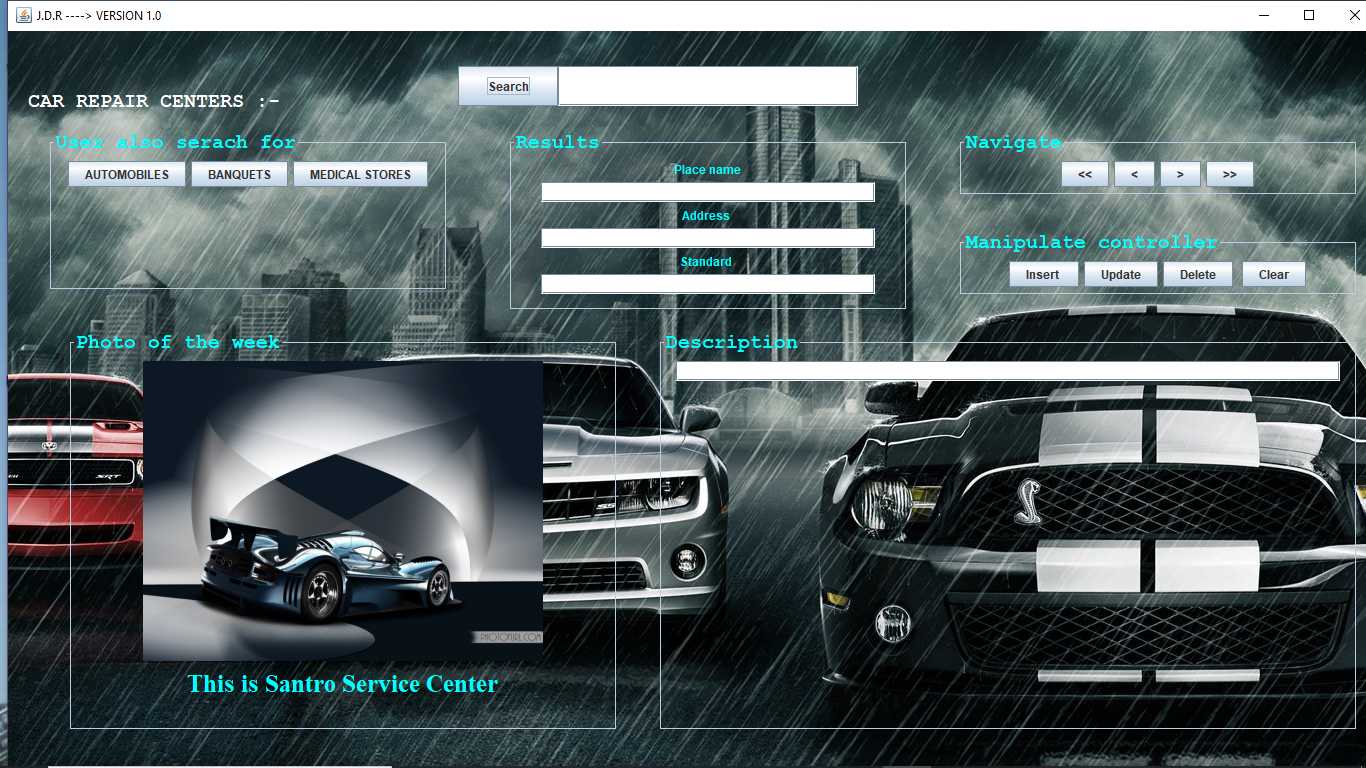
***LOGIN SCREEN: -***



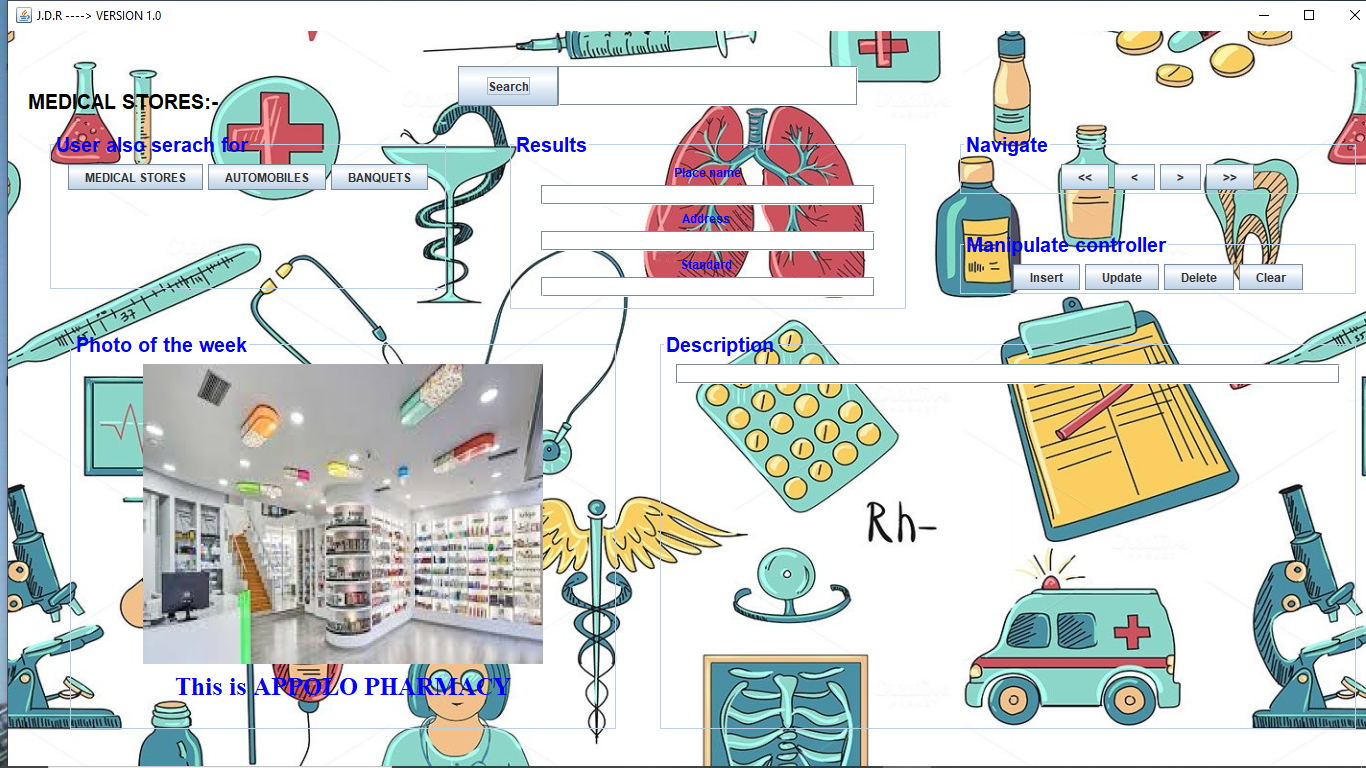
***BANQUET SCREEN: -***



***AUTOMOBILE SCREEN: -***



***MEDICALSTORE SCREEN: -***



***DATABASE TABLES***

**create** **table** automobiles

(

Name **varchar**(20),

Address **varchar**(30),

standard **varchar**(20),

description **varchar**(100),

state **varchar**(20)

);

**create** **table** banquets

(

Name **varchar**(20),

Address **varchar**(30),

standard **varchar**(20),

description **varchar**(100),

state **varchar**(20)

);

**create** **table** medicalstores

(

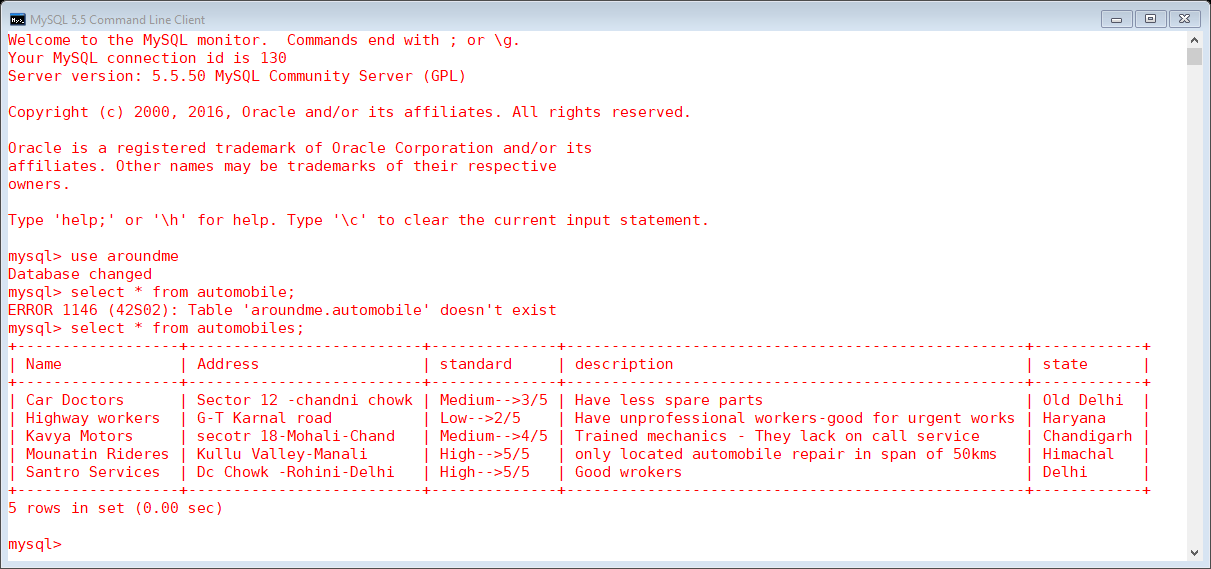
Name **varchar**(20),

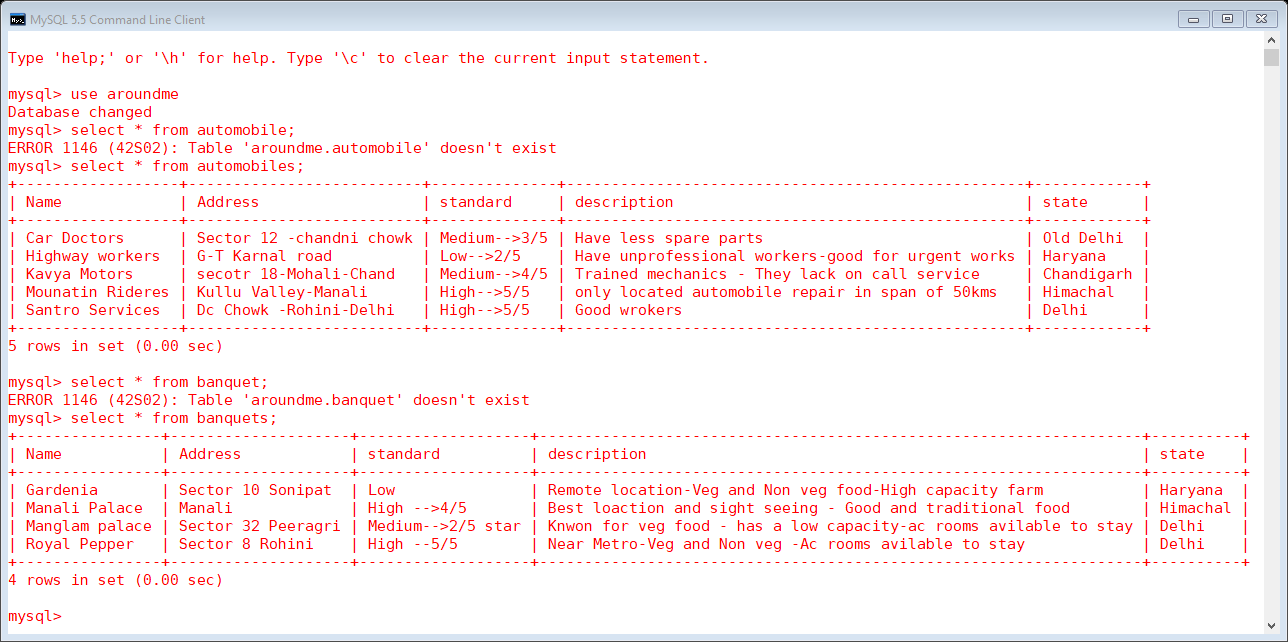
Address **varchar**(30),

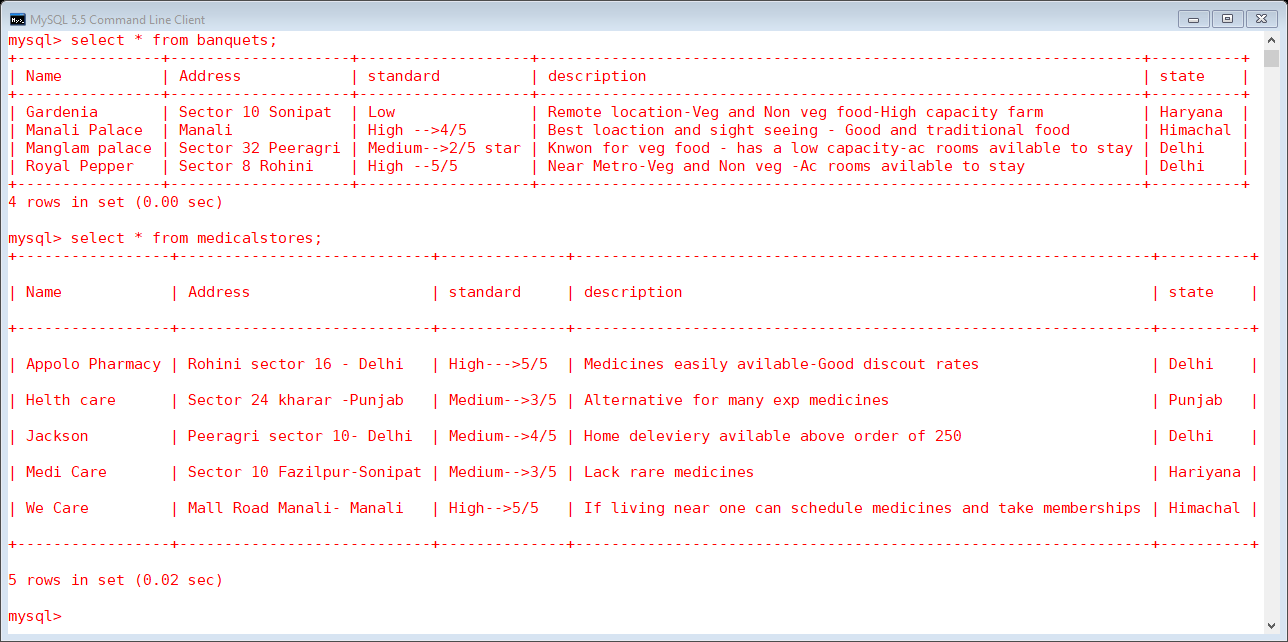
standard **varchar**(20),

description **varchar**(100),

state **varchar**(20));







***ADVANTAGES***

1. It is an offline application/project so it works at all times given all the circumstances.
2. Easy to use
3. User friendly interface
4. Collective information about multiple ventures like hospitals, malls, etc.
5. Easy to modify
6. Time efficient

***DISADVANTAGES***

1. Prime disadvantage is that its not secure by any means whatsoever
2. Requires too much effort in modifying the database.
3. Any malicious information can be inserted in the database
4. Its an offline based application which can’t spread information on a major scale
5. If tried to spread there is problem that database requirement come into play and their entries also

***REFERENCES***

Greekforgeek.com

Google.com

Sandeep Tiwari (Issac It labs)

Youtube.com