**• SLOT : L13+L14**

**• TEAM NUMBER: 5**

**• REGISTRATION NUMBER: 19BCE2119**

**• NAME: GAURAV KUMAR SINGH**

**REVIEW-3**

Cement Industry Project Management Database

**INDEX**

1. **Phase 1 Document Page 2**
2. **Phase 2 Document Page 4**
3. **Hardware and Software Requirements for the Application Page 13**
4. **Configuration and Setup Instructions for the Application Page 14**
5. **Flow of Control of Application Page 16**

* **Login Page Page 16**
* **Edit/View Menu Page 16**
* **Edit Menu Page 17**
* **View Menu Page 19**
* **Logging Out of Application Page 21**

1. **Screenshots of the Working Application Page 22**

* **Login Page Page 22**
* **Edit/View Menu Page 23**
* **Edit Menu Page 23**
* **View Menu Page 26**

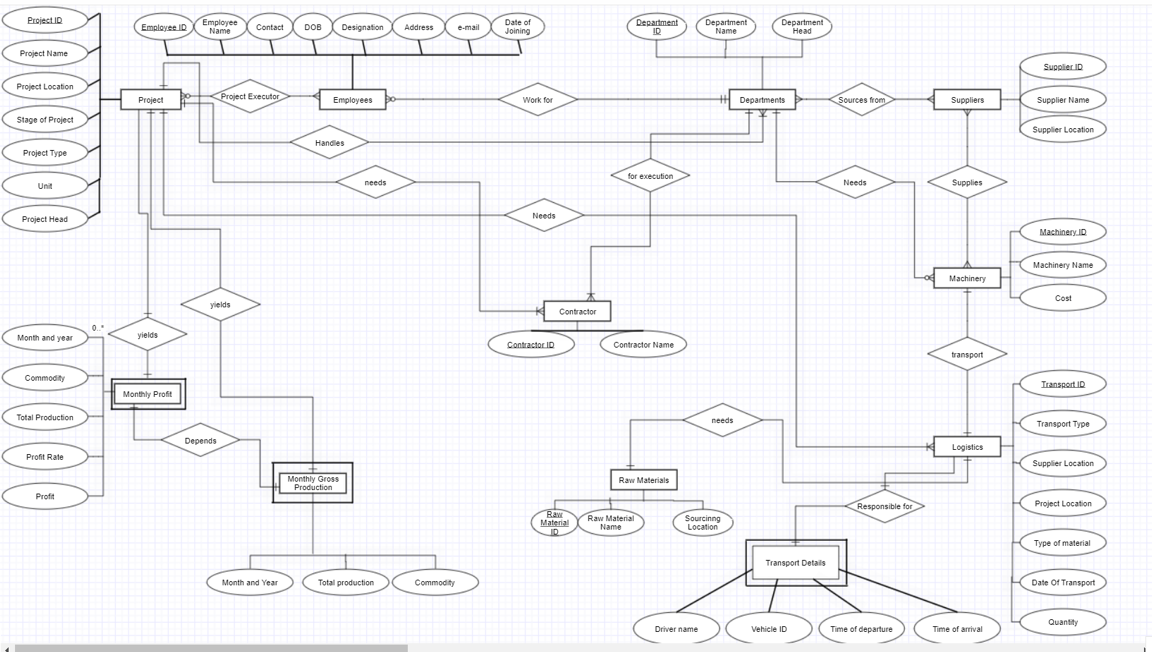
**Previous Phase Submission Documents:**

**Phase 1 Document**

**CEMENT INDUSTRY PROJECT MANAGEMENT DATABASE**

Cement Manufacturing is the infrastructure of establishing many other infrastructures. As important as it is, its also a very complex process to manufacture cement. Sourcing and mining of limestone and other ores, clinker manufacturing, to mixing and packaging, a lot of raw materials, sourcing and machineries are involved. Because of all this the units and projects have to established in several different parts of the country or even world and to keep track of everything can often get confusing but also plays important role since everything is unit plays an important role for the manufacturing process. To reduce this confusion, DBMS can be used to reduce overall complexity and make it easier and faster for us to access the data when needed, increasing our overall efficiency.

**ENTITY-RELATIONSHIP DIAGRAM**



**CONCEPTUAL SCHEMA**

**PROJECT**

Project\_ID, Project\_Name, Project\_Location, Stage\_of\_Project, Project\_Type, Unit, Project\_Head

**EMPLOYEES**

Emp\_ID, Emp\_Name, Contact, DOB, Designation, Address, E-mail, Date\_of\_Joining, Project\_ID, Dept\_ID

**DEPARTMENTS**

Dept\_ID, Dept\_Name, Dept\_Head, Project\_ID

**SUPPLIERS**

Supplier\_ID, Supplier\_Name, Supplier Location

**MACHINERY**

Machinery\_ID, Machinery\_Name, Cost, Supplier\_ID

**RAW MATERIALS**

Raw\_Material\_ID, Raw\_Material\_Name, Unit, Cost per unit

**LOGISTICS**

Tran\_ID, Trans\_Type, From, To, Mat\_ID, Quantity

**CONTRACTOR**

Contractor\_ID, Contractor\_Name, Project\_ID, Dept\_ID

**MONTHLY PROFIT**

Month\_and\_Year, Project\_ID, Commodity, Total\_Production, Profit\_Rate, Profit

**TRANSPORT DETAILS**

Driver\_Name, Vehicle\_ID, Date\_and\_Time\_of\_Departure, Date\_and\_Time\_of\_Arrival, Trans\_ID

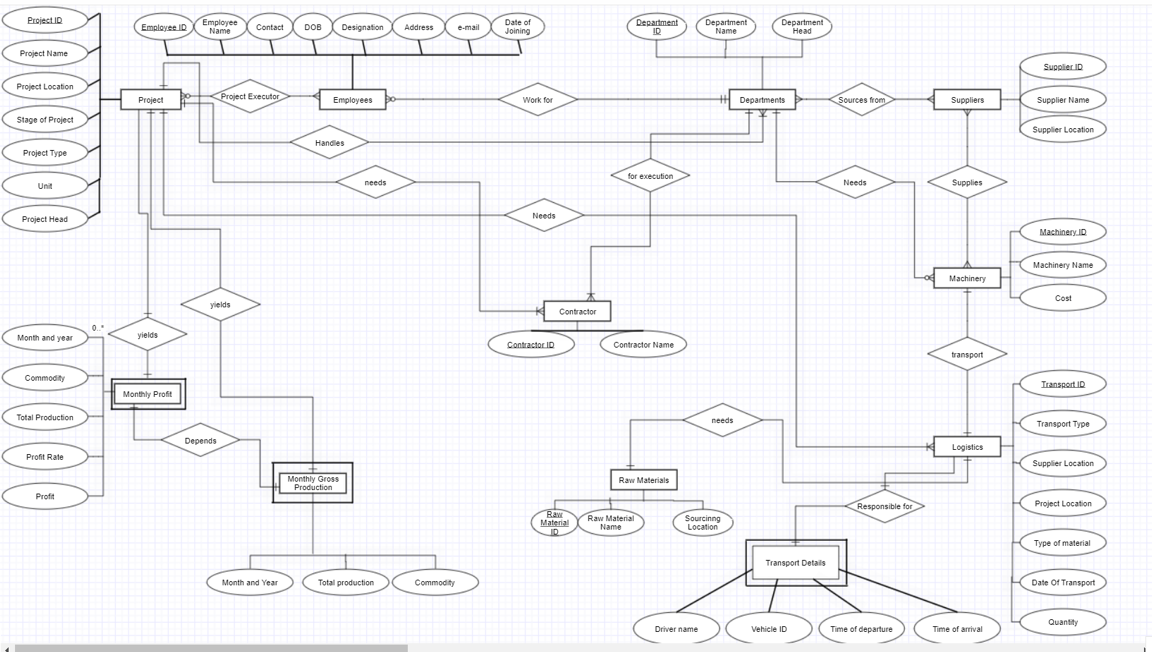
**xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx**

**Phase 2 Document**

**CEMENT INDUSTRY PROJECT MANAGEMENT DATABASE**

Cement Manufacturing is the infrastructure of establishing many other infrastructures. As important as it is, its also a very complex process to manufacture cement. Sourcing and mining of limestone and other ores, clinker manufacturing, to mixing and packaging, a lot of raw materials, sourcing and machineries are involved. Because of all this the units and projects have to established in several different parts of the country or even world and to keep track of everything can often get confusing but also plays important role since everything is unit plays an important role for the manufacturing process. To reduce this confusion, DBMS can be used to reduce overall complexity and make it easier and faster for us to access the data when needed, increasing our overall efficiency.

**ENTITY-RELATIONSHIP DIAGRAM**



**CONCEPTUAL SCHEMA**

**PROJECT**

Project\_ID, Project\_Name, Project\_Location, Stage\_of\_Project, Project\_Type, Unit, Project\_Head

**EMPLOYEES**

Emp\_ID, Emp\_Name, Contact, DOB, Designation, Address, E-mail, Date\_of\_Joining, Project\_ID, Dept\_ID

**DEPARTMENTS**

Dept\_ID, Dept\_Name, Dept\_Head, Project\_ID

**SUPPLIERS**

Supplier\_ID, Supplier\_Name, Supplier Location

**MACHINERY**

Machinery\_ID, Machinery\_Name, Cost, Supplier\_ID

**RAW MATERIALS**

Raw\_Material\_ID, Raw\_Material\_Name, Unit, Cost per unit

**LOGISTICS**

Tran\_ID, Trans\_Type, From, To, Mat\_ID, Quantity

**CONTRACTOR**

Contractor\_ID, Contractor\_Name, Project\_ID, Dept\_ID

**MONTHLY PROFIT**

Month\_and\_Year, Project\_ID, Commodity, Total\_Production, Profit\_Rate, Profit

**TRANSPORT DETAILS**

Driver\_Name, Vehicle\_ID, Date\_and\_Time\_of\_Departure, Date\_and\_Time\_of\_Arrival, Trans\_ID

**NORMALIZATION**

**Original Employees Table**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Employees | | | | | | | | | |
| Emp  ID | Emp  Name | Contact | Date of birth | Designation | Address | E-mail | Date of joining | Project  ID | Dept  ID |
| 1 | Ajay | 99999 | 01-Jan-1994 | DGM | A-192 | ajay@iti.com | 21-Feb-2020 | P0001  P0002 | DP1 |
| 2 | Bhola | 88888 | 02-Mar-1992 | GM | B-143 | bhola@iti.com | 22-March-2018 | P0001 | DP2 |
| 3 | Chris | 77777 | 03-May-1900 | Add.GM | C-150 | chris@iti.com | 23-Dec-2015 | P0002  P0003 | DP1 |

**1NF**

**Functional Dependency:**

Emp ID 🡪 Emp Name, Contact, Date of Birth, Address, E-mail, Date of Joining, Dept ID

Project ID🡪 Project ID

Contact 🡪 Emp ID

E-mail 🡪 Emp ID

Emp ID, Project ID 🡪 R

**Candidate Key:** Emp ID, Project ID

In order for a table to be in 1NF:

1. It should have atomic attributes only.
2. Each column must have unique names.
3. Order of the entries does not matter.
4. Each column must contain values of same variable type.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Employees | | | | | | | | | |
| Emp  ID | Emp  Name | Contact | Date of birth | Designation | Address | E-mail | Date of joining | Project  ID | Dept  ID |
| 1 | Ajay | 99999 | 01-Jan-1994 | DGM | A-192 | ajay@iti.com | 21-Feb-2020 | P0001 | DP1 |
| 1 | Ajay | 99999 | 01-Jan-1994 | DGM | A-192 | ajay@iti.com | 21-Feb-2020 | P0002 | DP1 |
| 2 | Bhola | 88888 | 02-Mar-1992 | GM | B-143 | bhola@iti.com | 22-March-2018 | P0001 | DP2 |
| 3 | Chris | 77777 | 03-May-1900 | Add.GM | C-150 | chris@iti.com | 23-Dec-2015 | P0002 | DP1 |
| 3 | Chris | 77777 | 03-May-1900 | Add.GM | C-150 | chris@iti.com | 23-Dec-2015 | P0003 | DP1 |

**2NF**

In order for a table to be in 2NF form:

1. It should be in 1NF form.
2. It should not have any partial dependencies.

Candidate key: Emp ID

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Employees | | | | | | | | |
| Emp  ID | Emp  Name | Contact | Date of birth | Designation | Address | E-mail | Date of joining | Dept  ID |
| 1 | Ajay | 99999 | 01-Jan-1994 | DGM | A-192 | ajay@iti.com | 21-Feb-2020 | DP1 |
| 2 | Bhola | 88888 | 02-Mar-1992 | GM | B-143 | bhola@iti.com | 22-March-2018 | DP2 |
| 3 | Chris | 77777 | 03-May-1900 | Add.GM | C-150 | chris@iti.com | 23-Dec-2015 | DP1 |

Candidate key: Emp ID, Project ID

|  |  |
| --- | --- |
| Employee’s Ongoing Project | |
| Emp ID | Project ID |
| 1 | P0001 |
| 1 | P0002 |
| 2 | P0001 |
| 3 | P0002 |
| 3 | P0003 |

The decomposition is lossless because:

1. Attributes(r1) U Attributes(r2) = Attributes(R) {R}
2. Attributes(r1) ∩ Attributes(r2) != null {Emp ID}
3. Attributes(r1) ∩ Attributes(r2) 🡪 r1 {Emp ID🡪Employees}

Where,

r1: Employees table after decomposition

r2: Employee’s Ongoing Project table

R: Employees table before decomposition

**3NF**

In order for a table to be in 3NF form:

1. It should be in 2NF form.
2. It should not have any transitive dependency.

Candidate key: Emp ID

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Employees | | | | | | | | |
| Emp  ID | Emp  Name | Contact | Date of birth | Designation | Address | E-mail | Date of joining | Dept  ID |
| 1 | Ajay | 99999 | 01-Jan-1994 | DGM | A-192 | ajay@iti.com | 21-Feb-2020 | DP1 |
| 2 | Bhola | 88888 | 02-Mar-1992 | GM | B-143 | bhola@iti.com | 22-March-2018 | DP2 |
| 3 | Chris | 77777 | 03-May-1900 | Add.GM | C-150 | chris@iti.com | 23-Dec-2015 | DP1 |

Candidate Key: Emp ID, Project ID

|  |  |
| --- | --- |
| Employee’s Ongoing Project | |
| Emp ID | Project ID |
| 1 | P0001 |
| 1 | P0002 |
| 2 | P0001 |
| 3 | P0002 |
| 3 | P0003 |

**BCNF**

In order for a table to be in BCNF (also known as 3.5NF) form:

1. It should be in 3NF form.
2. No non-prime attribute must be able to derive a prime attribute i.e.

For a relation A🡪B, B cannot be prime attribute if A is a non-prime attribute

Candidate Key: Emp ID

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Employees | | | | | | |
| Emp  ID | Emp  Name | Date of birth | Designation | Address | Date of joining | Dept  ID |
| 1 | Ajay | 01-Jan-1994 | DGM | A-192 | 21-Feb-2020 | DP1 |
| 2 | Bhola | 02-Mar-1992 | GM | B-143 | 22-March-2018 | DP2 |
| 3 | Chris | 03-May-1900 | Add.GM | C-150 | 23-Dec-2015 | DP1 |

Candidate Key: Emp ID, Project ID

|  |  |
| --- | --- |
| Employee’s Ongoing Project | |
| Emp ID | Project ID |
| 1 | P0001 |
| 1 | P0002 |
| 2 | P0001 |
| 3 | P0002 |
| 3 | P0003 |

Candidate Key: Emp ID

|  |  |  |
| --- | --- | --- |
| Employee Contact Details | | |
| Emp Id | Contact No. | Email ID |
| 1 | 99999 | ajay@iti.com |
| 2 | 88888 | bhola@iti.com |
| 3 | 77777 | chris@iti.com |

The decomposition is lossless because:

1. Attributes(r1) U Attributes(r2) = Attributes(R) {R}
2. Attributes(r1) ∩ Attributes(r2) != null {Emp ID}
3. Attributes(r1) ∩ Attributes(r2) 🡪 r1 {Emp ID🡪Employees,

Emp ID 🡪Employee Contact Details}

Where,

r1: Employees table after decomposition

r2: Employee Contact Details table

R: Employees table before decomposition

**Database Schema in BCNF**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Project | | | | | | |
| Project ID | Project Name | Project Location | Stage of Project | Project Type | Unit | Project Head |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Employees | | | | | | |
| Emp  ID | Emp  Name | Date of birth | Designation | Address | Date of joining | Dept  ID |

|  |  |
| --- | --- |
| Employee’s Ongoing Project | |
| Emp ID | Project ID |

|  |  |  |
| --- | --- | --- |
| Employee Contact Details | | |
| Emp Id | Contact No. | Email ID |

|  |  |
| --- | --- |
| Departments | |
| Department ID | Department Name |

|  |  |  |
| --- | --- | --- |
| Head and Department Relation | | |
| DID | Department ID | Department Head |

|  |  |
| --- | --- |
| Project Departments’ Head | |
| Project ID | DID |

|  |  |
| --- | --- |
| Supplier Names | |
| Supplier ID | Supplier Name |

|  |  |
| --- | --- |
| Supplier’s Location | |
| Supplier ID | Location |

|  |  |
| --- | --- |
| Machinery | |
| Machinery ID | Machinery Name |

|  |  |  |
| --- | --- | --- |
| Machinery Cost | | |
| Machinery ID | Supplier ID | Cost |

|  |  |
| --- | --- |
| Raw Material | |
| Raw Material ID | Raw Material Name |

|  |  |  |
| --- | --- | --- |
| Raw Material Costing | | |
| Raw Material ID | Unit of Measurement | Cost per unit |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Logistics | | | | | |
| Transport ID | Transport Type | From | To | Material  ID | Quantity |

|  |  |
| --- | --- |
| Contractor | |
| Contractor ID | Contractor Name |

|  |  |  |
| --- | --- | --- |
| Contracted Field | | |
| Contractor ID | Project ID | Department ID |

|  |  |
| --- | --- |
| Commodity Produced | |
| Project ID | Commodity |

|  |  |  |  |
| --- | --- | --- | --- |
| Monthly Gross Produce | | | |
| Month and Year | Project ID | Total Production | Profit rate |

|  |  |  |
| --- | --- | --- |
| Profit | | |
| Total Production | Profit Rate | Profit |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Transport Details | | | | |
| Transport ID | Vehicle ID | Date of Departure | Date of Arrival | Driver/Operator |

**xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx**

**Hardware and Software Requirements:**

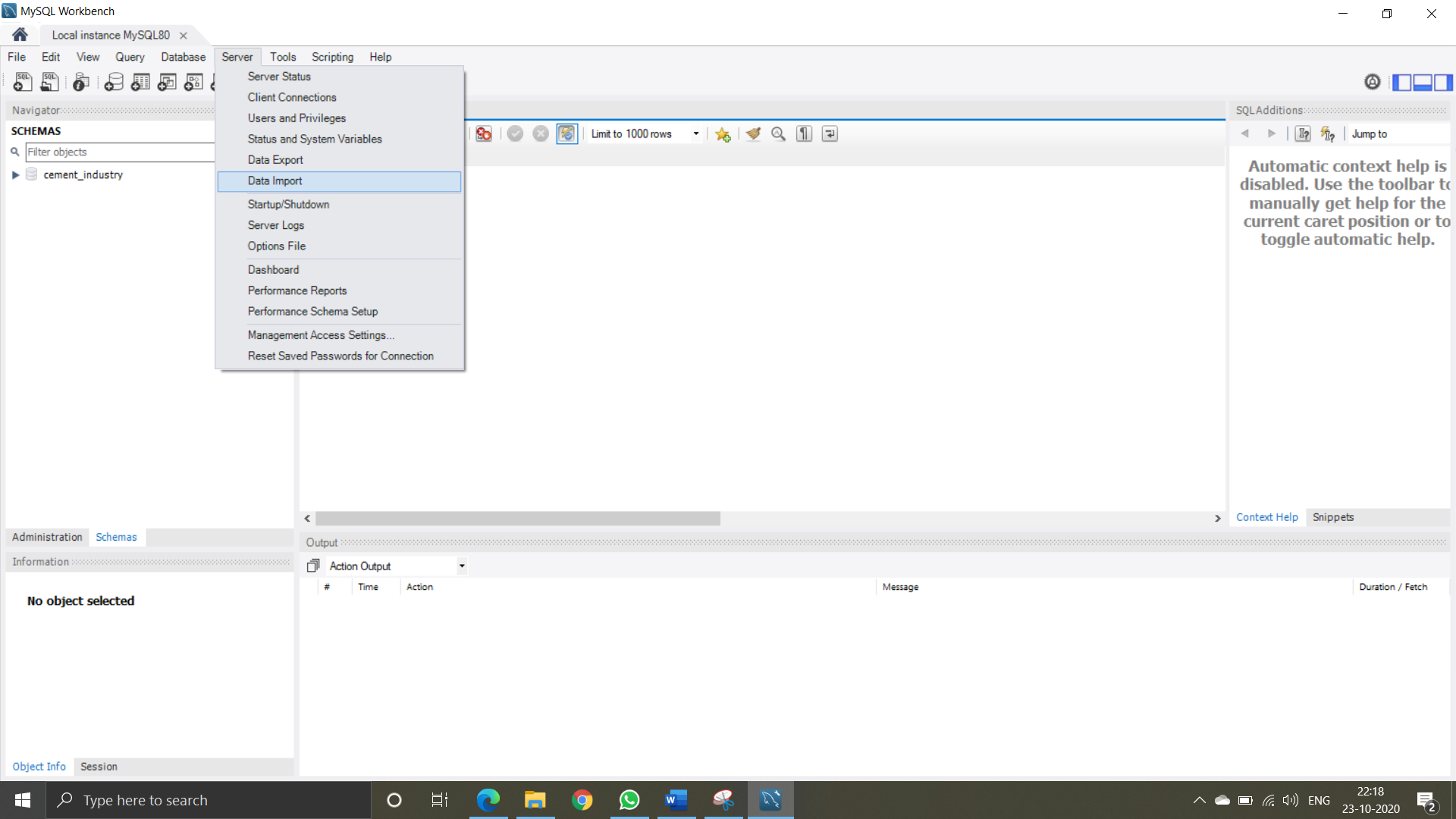
* **Windows OS:** Windows 7 (64-bit, Professional level or higher) {Application Built on Windows 10 Home using Visual Studio 2019 Community Version as frontend for windows form application and MySQL for the backend of the application.}
* **At least 100MB Free Hard Disc Space**
* **MySQL Workbench:** ‘root’ user **MUST** be configured to have the password as ‘1020’ if any for the application to function properly.
* **MySQL for visual studio attachment installed through MySQL setup**
* **.NET Framework 3.5 and above**
* **CPU:** Intel Core or Xeon 3GHz (or Dual Core 2GHz) or equal AMD CPU
* **RAM:** 4 GB (6 GB recommended)

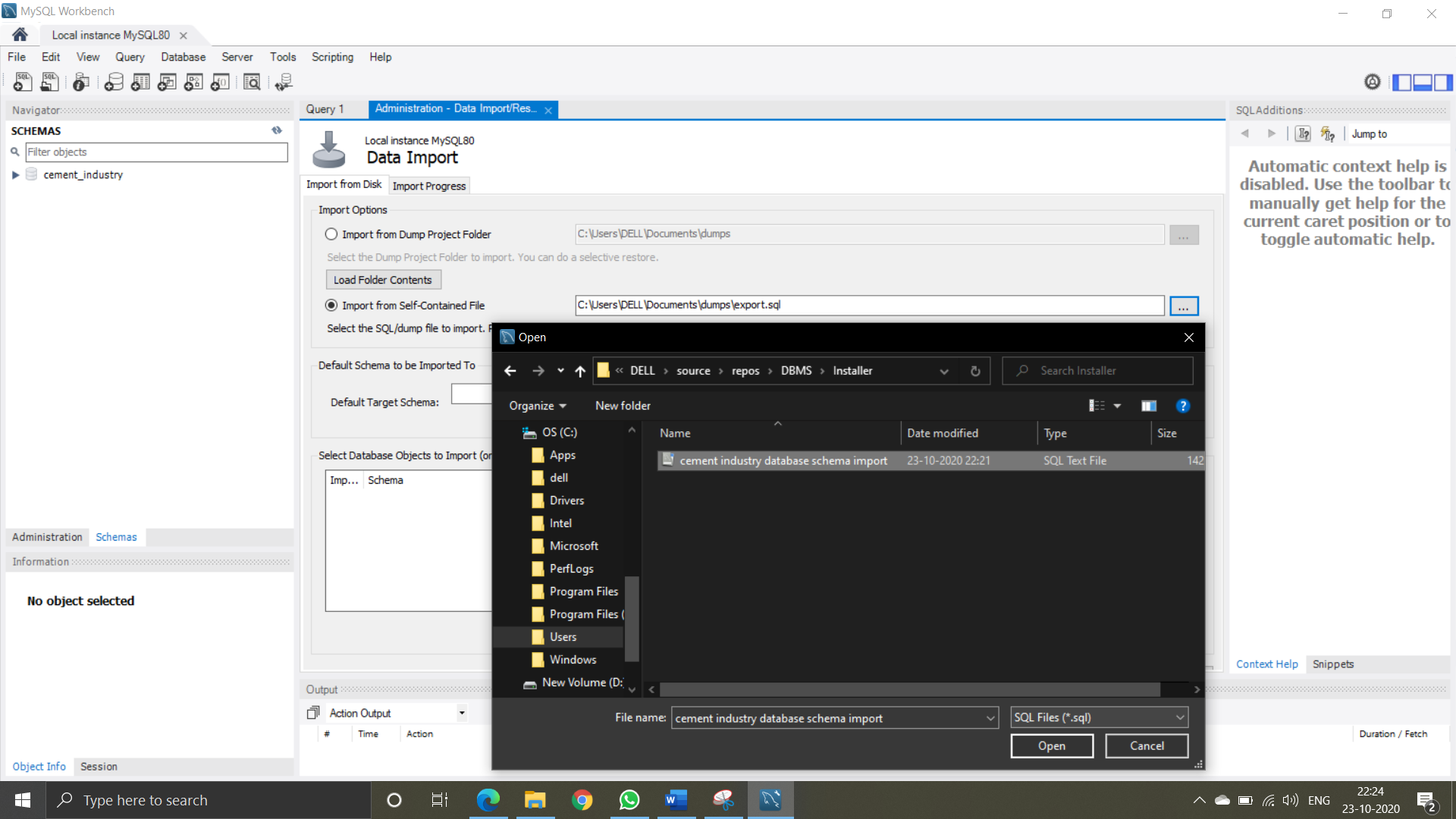
**xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx**

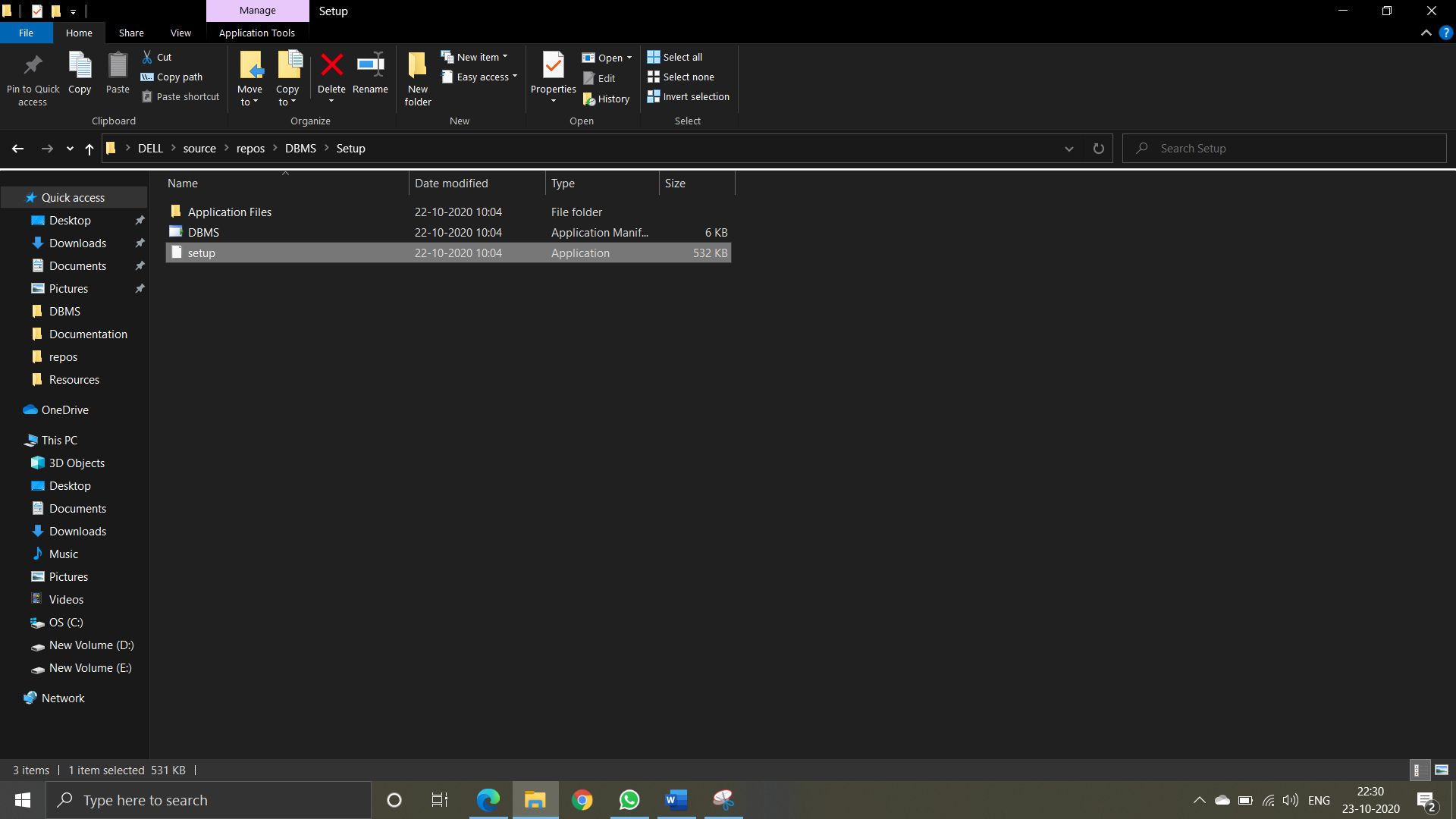
**Configuration and Setup of application**

**IMPORTANT!!!** The password for root user of MySQL MUST be set to ‘1020’ if any for the application to work.

**Installation Steps**

1. Open MySQL Workbench and open a root connection.
2. Open ‘Server’ menu and tap on ‘Data Import’ option. 
3. Check the ‘Import from Self-Contained File’ option and navigate select the ‘cement industry database schema import.sql’ file in the installer folder and make sure one of dump structue or dump data and structure option is selected.



1. Click on ‘Start Import’ to import the database onto the system.
2. Now, to install the frontend application go to ‘Setup’ folder and open the Setup file. 
3. Continue installation as the wizard leads.
4. Your application is now installed and working given the prerequisites are met.

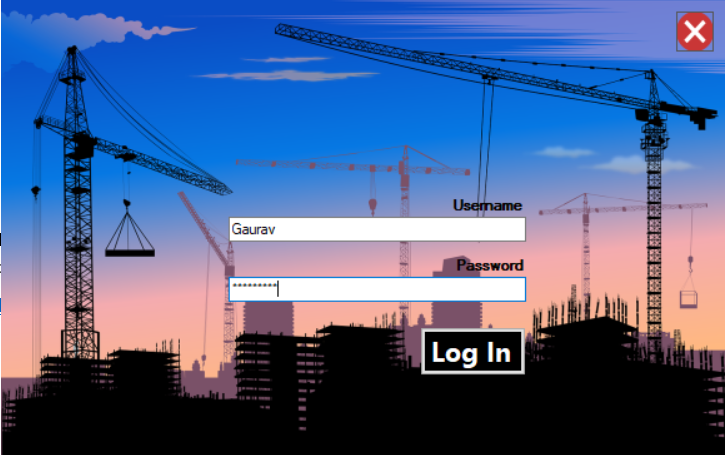
**xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx**

**Flow of Control of Application**

1. As soon as the application is launched, you are asked for your login info.

The default **Username is Gaurav** and **Password is 19BCE2119.** Tap Log IN.

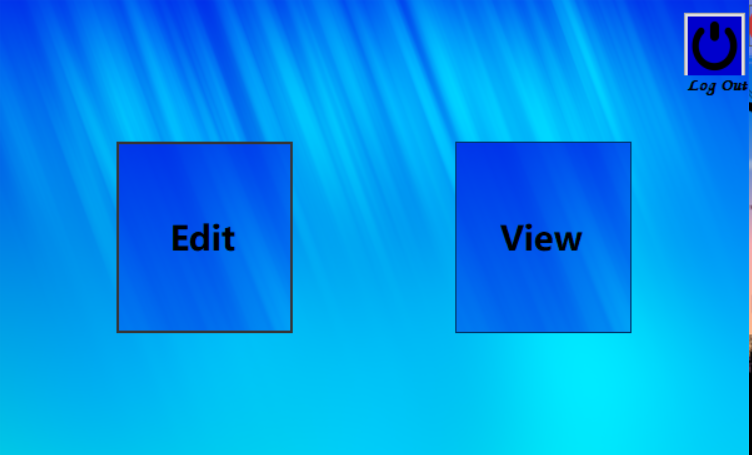
(If a message box telling connection to MySQL database is not properly established or access to database is denied then please cross verify that the MySQL database is properly setup)



1. After Successful Login a message box confirming log in success will pop up and redirect you to the database modification options.

Here you have 3 buttons:

1. **Edit**
2. **View**
3. **Logout**



**EDIT**

Edit button Leads to the following Window

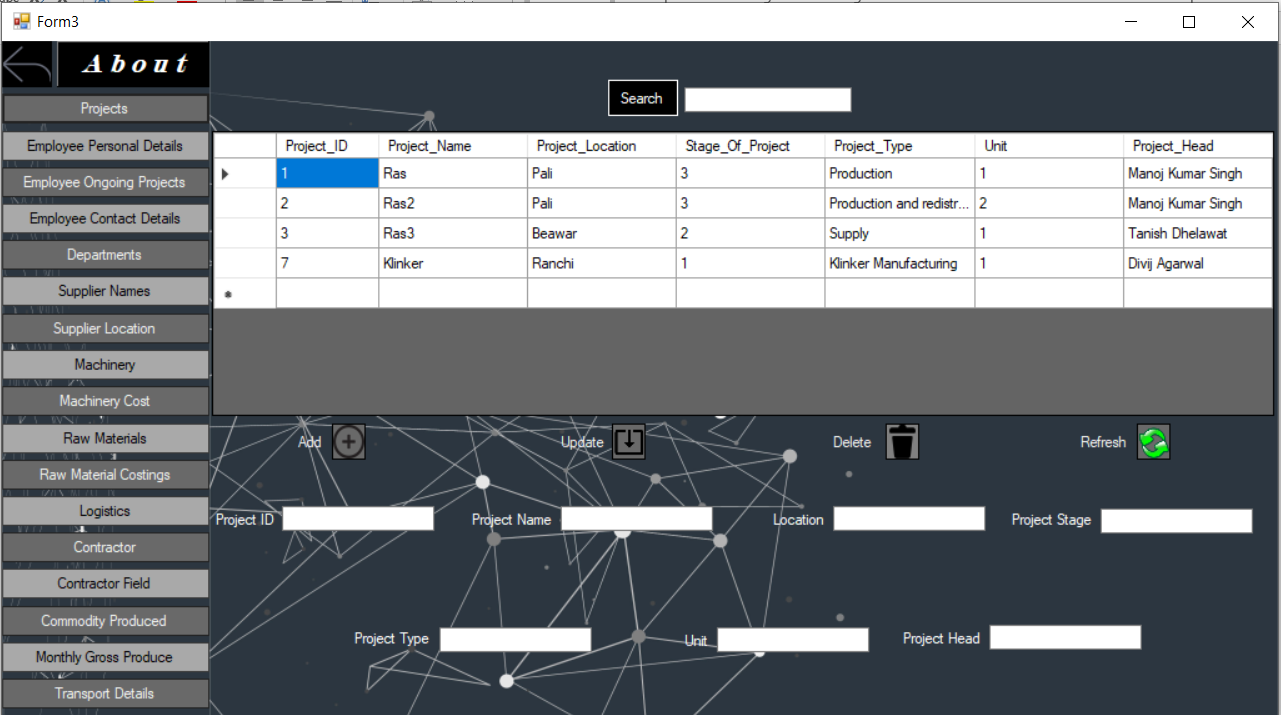


Table access tabs Work area

In Edit Window you get access to all the Normalized tables of the database with the flexibility of adding new tuples, updating existing tuples, deleting tuples and searching for a specific tuple in the particular table.

There are a total of 17 normalized tables present.

Click on the table name u want to access/edit and a ‘User control’ as shown in the following figure will popup on the work area



Back Entry Fields Data Grid DML operations

Every table in Edit Table Windows forms contains 5 buttons in total:

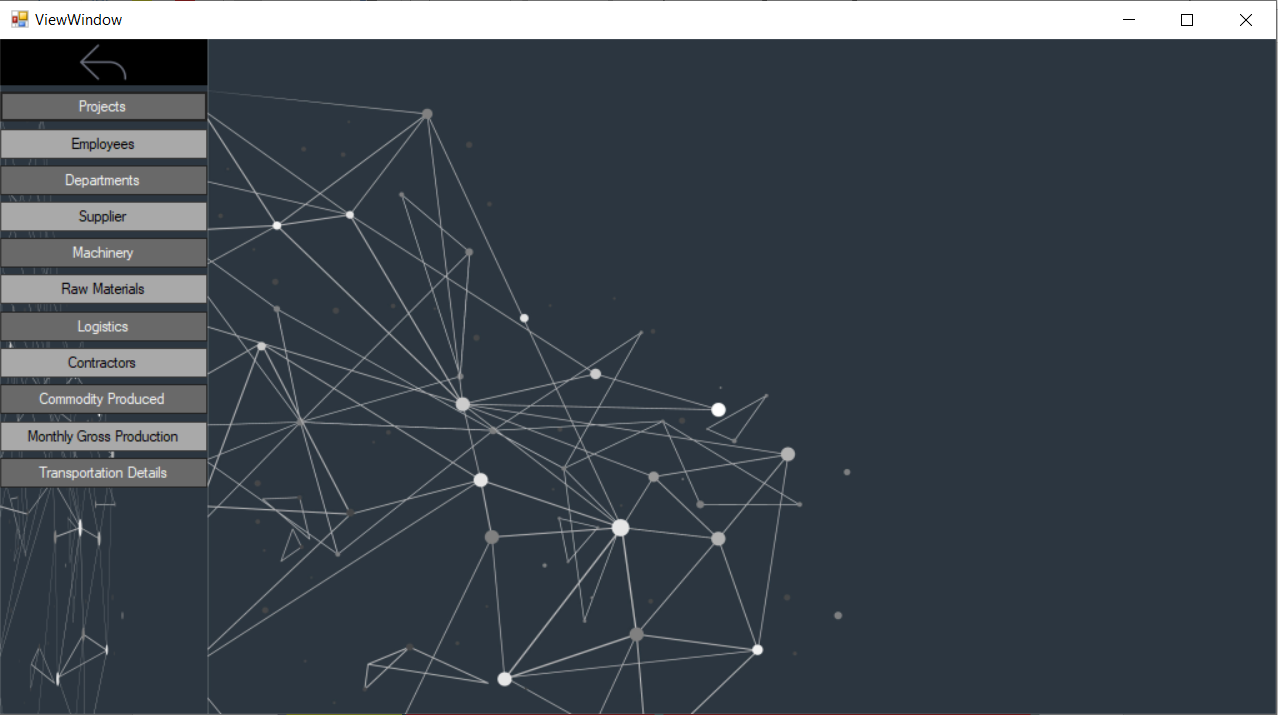
* **Add-** It takes input from the Entry fields and add the tuple into the database if the entries are Valid.
* **Update-**It updates the values into an existing table by taking input from Entry field. Any changed data will be automatically updated, but any null textbox will also nullify that value in table so every text field needs to be filled with appropriate data during updating for the right changes to take place.
* **Delete-**It deletes a tuple from the database by taking input from the primary key entry fields. It only needs that the primary key entry fields are filled to delete the correct tuple and does not care about what the non-prime attribute values are in the text boxes.
* **Refresh-**It can be used to manually refresh the database even though database refreshes itself after every update, add, or delete operation. Its main use is to refresh the data grid after an ‘Search’ is done.
* **Search-**‘Search’ does two functions. ***Firstly****,* it searches for the required tuple from the table and return it into the data grid. ***Secondly***, it auto fills all the entry fields with their corresponding data if they are atomic. It is highly recommended and convenient to perform a search operation on the table you want to update before assigning changed data fields to ensure any unnecessary change are not made during the update.

Every ‘User control’ also contains:

* **Entry Fields-**Every table has its own entry field depending on the columns it contain. This is the area where the input to be read by the buttons is given
* **Data Grid-**The data contained by the table is shown here.

**VIEW**

View button Leads to the following Window

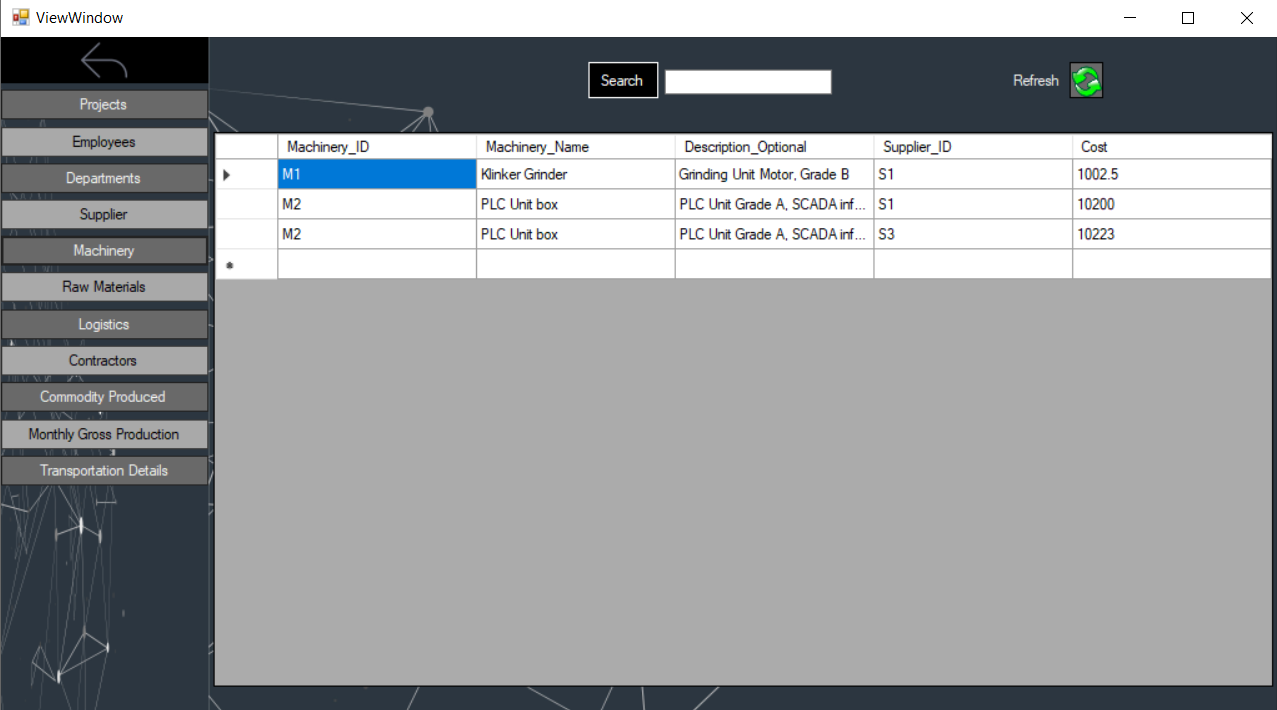


Back Button Table access tabs Work area

In View Window you can get a free view of all the unnormalized forms of the tables. It is easier to search for the complete set of data for a particular entry in views table since it shows all the related fields on one page but you cannot edit any field here since this is not the actual structure of the database. However, you can search for the entries with their prime attributes here.

There are a total of 11 un-normalized tables present.

Click on the table name u want to view and a ‘User control’ as shown in the following figure will pop up on the work area.



Data Grid View Search button Search Textbox Refresh Button

There are just 5 components in the ‘User control’:

* **Search button-**Searches for the tuple with the prime attribute entered into the text box
* **Refresh table button-**Refreshes the data grid view.
* **Search textbox-**The prime attribute with which the search is going to be performed is entered here.
* **Data grid view-**The table is shown here.
* **Back Button-**It sends you to the Edit/View selection form.

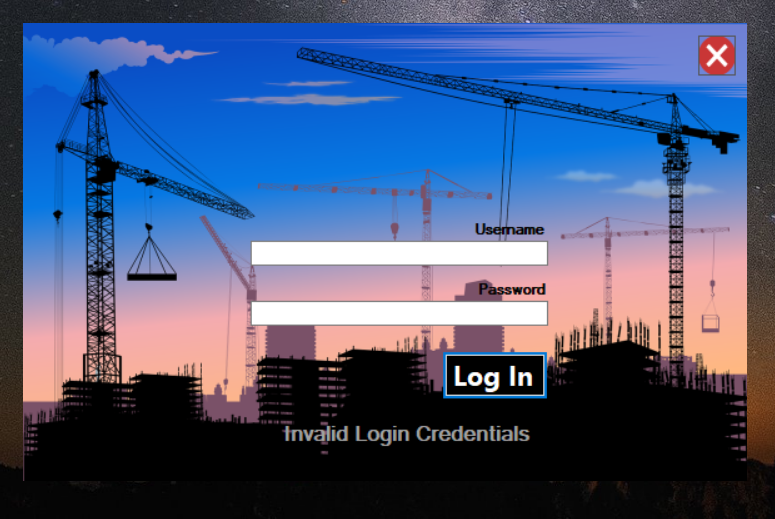
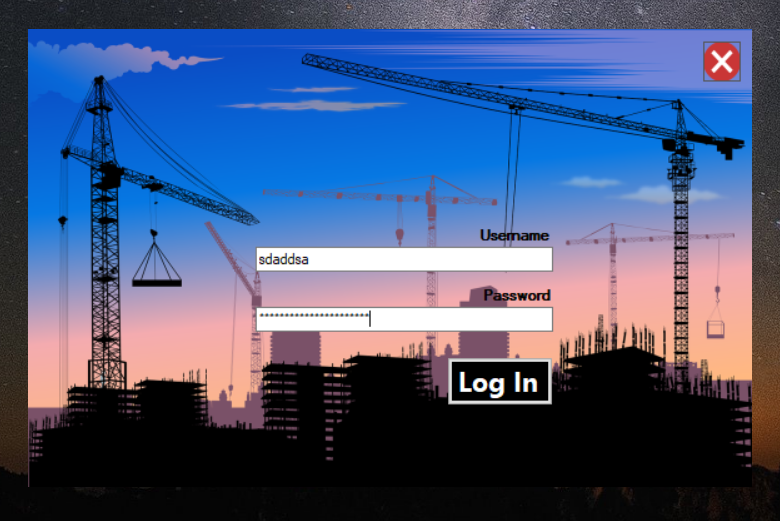
**LOGOUT**



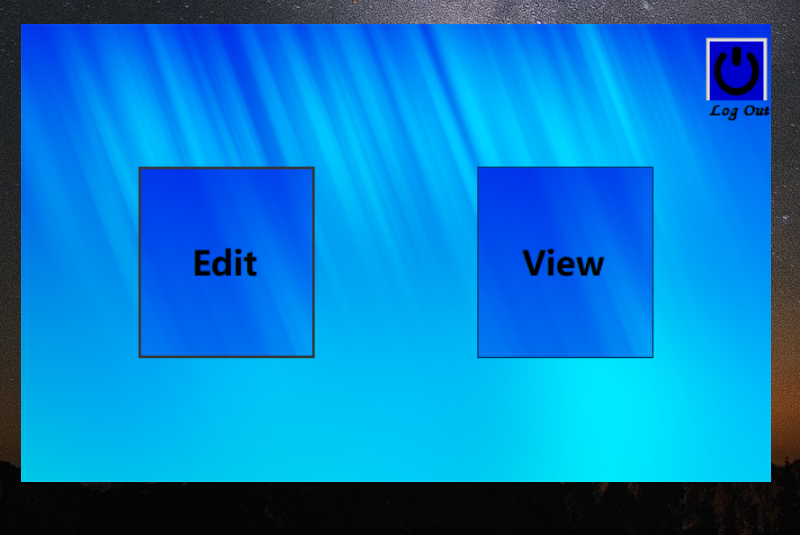
The Logout button takes you back to the Login page.

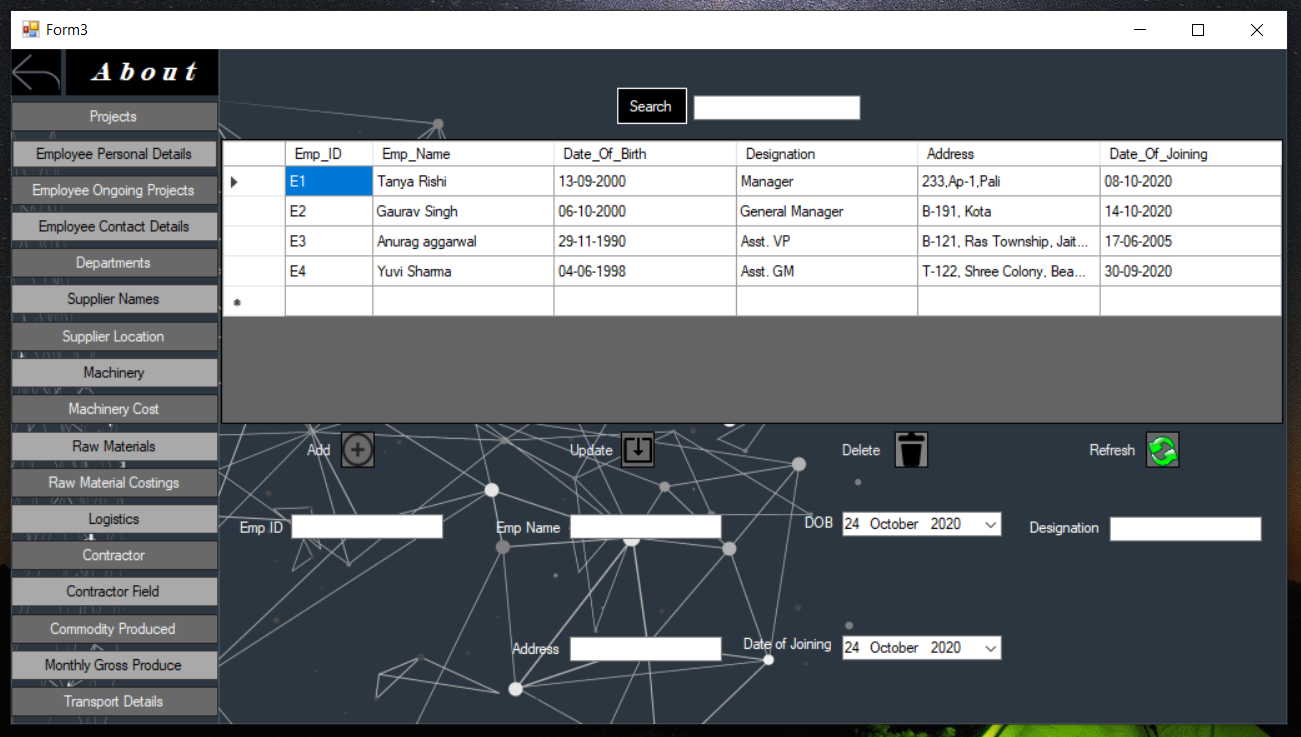
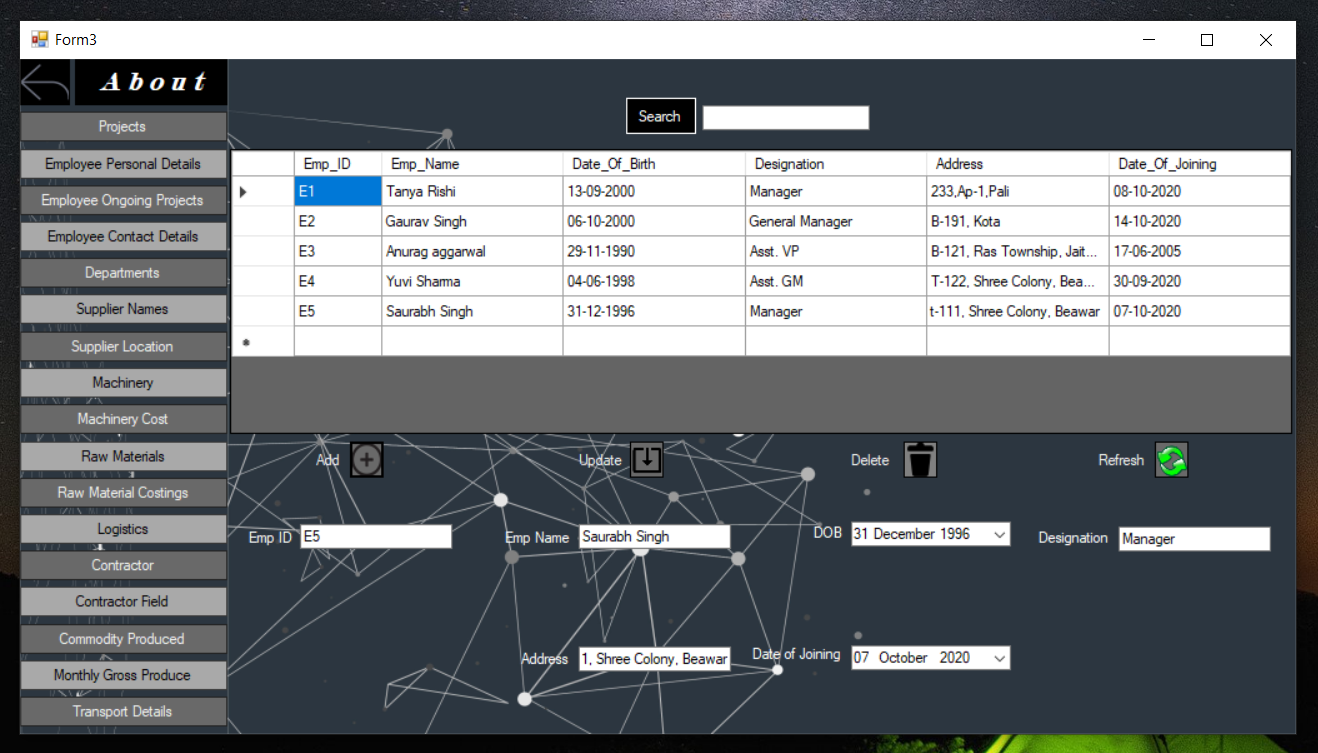
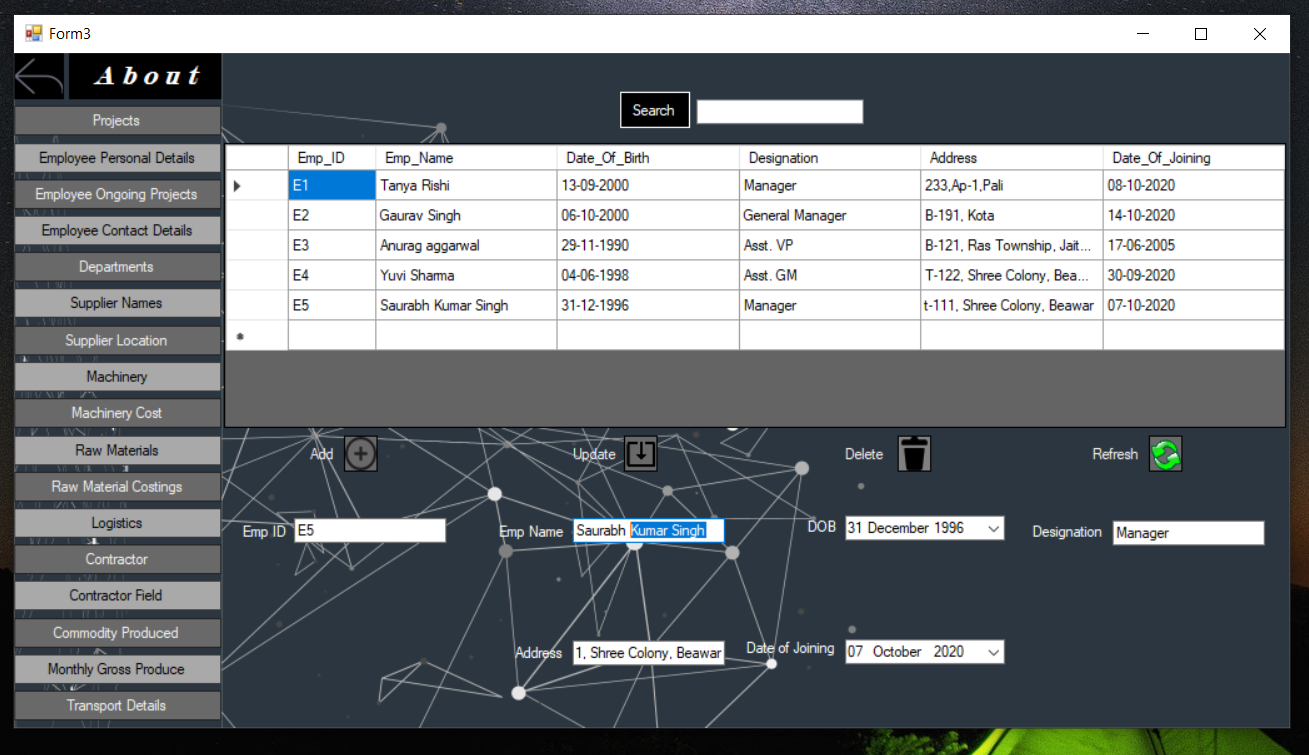
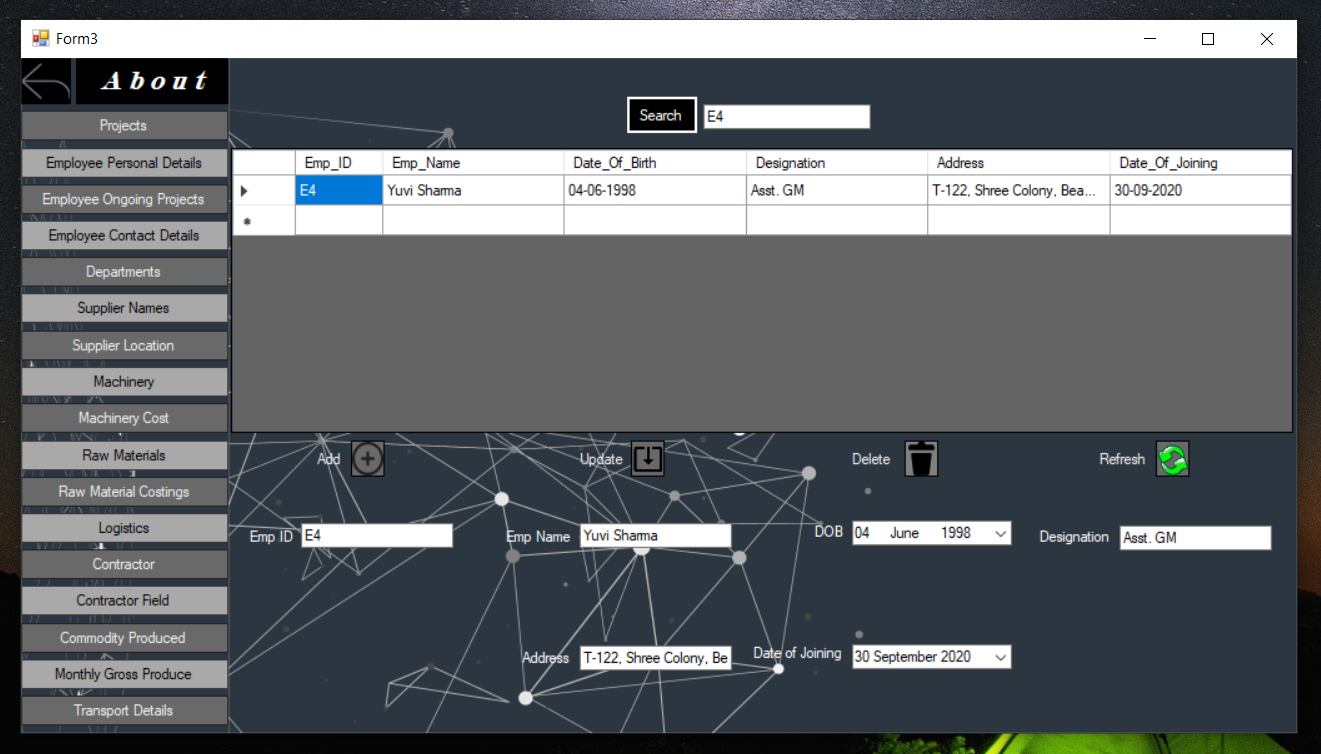
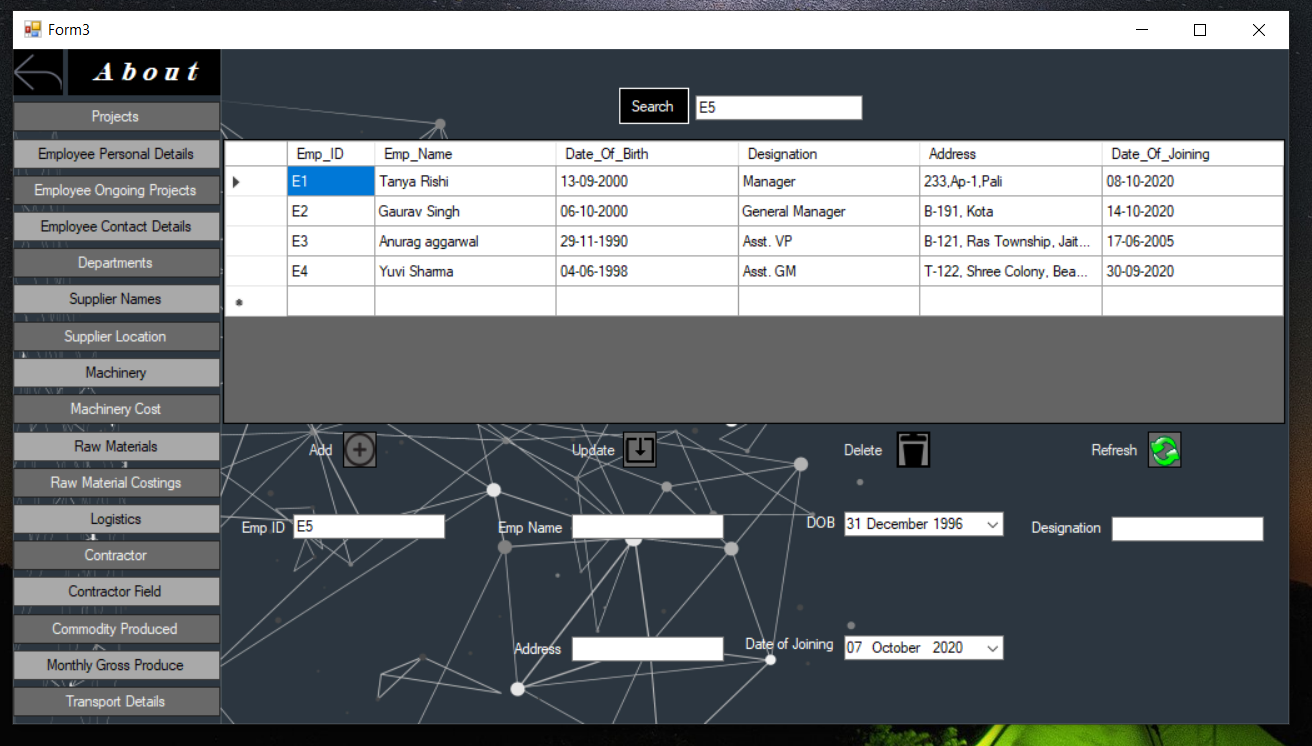
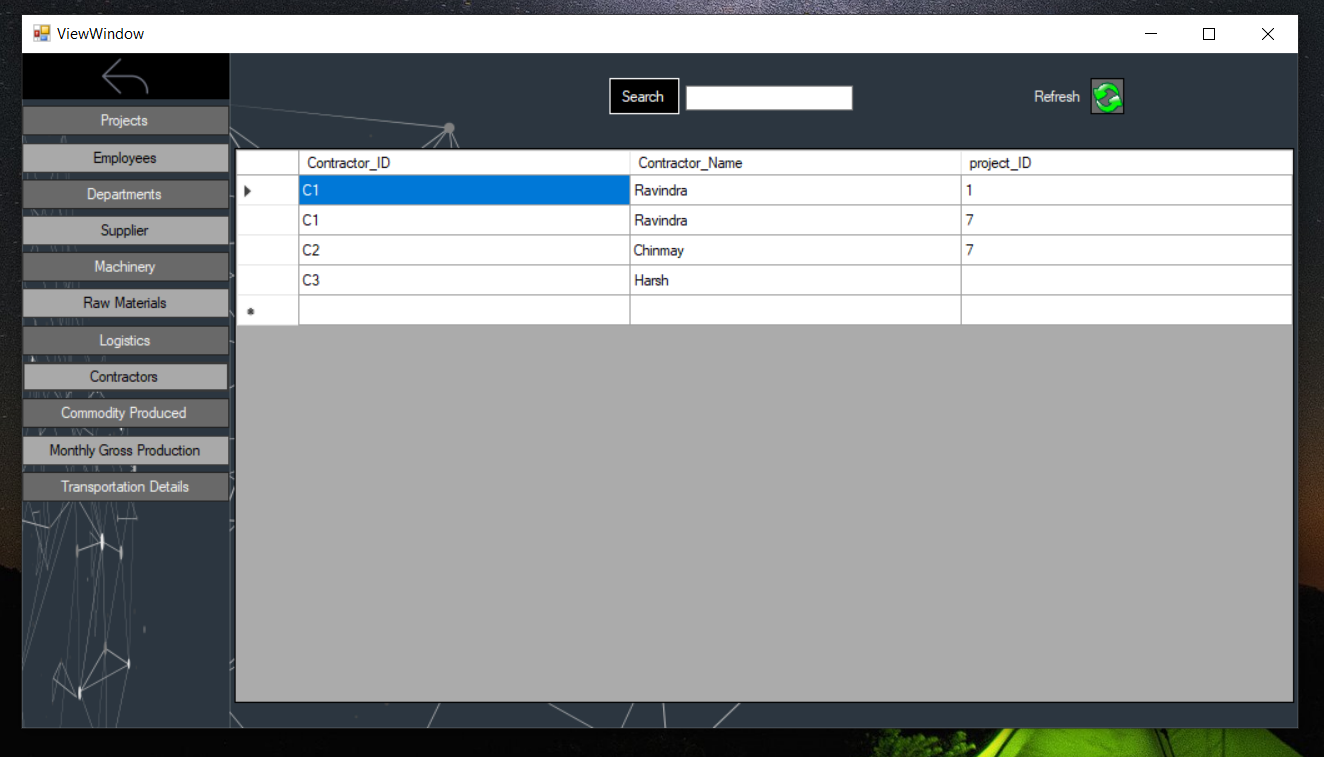
**Screenshots of the working Project**

* **Login Page**



* **Edit/View Menu**



* **Edit Menu**
* **About Menu**
* **Tables Menu Usercontrol**
* **Adding Tuple**
* **Updating Tuple**
* **Searching Tuple**
* **Deleting Tuple**
* **View Menu**
* **User control**
* **Search**