

**Visvesvaraya Technological University  
Belgaum, Karnataka-590 014**



A Project Report on  
**“AUTOMATION OF WATER MANAGEMENT  
SYSTEM OF A CITY”**

Project Report submitted upon the completion of Mini Project  
for the 5<sup>th</sup> Semester's  
**Database Management System Laboratory (18CSL58)**

**Submitted by**

**Veer S Jadimath                      2JR18CS091**

**Sourabh G Patil                      2JR18CS079**

**Under the Guidance of  
Prof. Basavraj Madagouda  
Prof. Raghavendra Katagal**



**Jain Group of Institutions'**

**Jain College of Engineering and Research, Belagavi**

**Department of Computer Science and Engineering**

**2020-21**

Jain Group of Institutions'  
**Jain College of Engineering and Research, Belagavi**  
Department of Computer Science and Engineering



## CERTIFICATE

This is to certified that the project work entitled “**AUTOMATION OF WATER MANAGEMENT SYSTEM OF A CITY**” carried out by Mr. Veer Jadimath, USN 2JR18CS091, and Mr. Sourabh Patil, USN 2JR18CS079, bonafide students of Jain College of Engineering and Research, Belagavi, in partial fulfillment for the Mini Project in the subject **Database Management System Laboratory** in **Computer Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said subject.

**Prof. Basavraj Madagouda**  
Guide

**Prof. Raghavendra Katagall**  
Guide

**Dr. Pritam M**  
Head of Dept.

**Examiner**

**Signature with date:**

- 1.
- 2.

# List of Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>1</b>
<b>3</b>	<b>Literature Survey</b>	<b>2</b>
3.1	Existing Systems .....	2
<b>4</b>	<b>Data Flow Diagrams</b>	<b>3</b>
4.1	Entity Relationship .....	3
4.2	Relationship schema .....	3
<b>5</b>	<b>Methodology</b>	<b>4</b>
5.1	Tools Used .....	4
5.2	Code snippets .....	4
5.2.1	Backend .....	4
5.2.2	Frontend .....	7
<b>6</b>	<b>Results</b>	<b>10</b>
<b>7</b>	<b>Conclusion and Recommendations</b>	<b>14</b>
7.1	Conclusion .....	14
7.2	Recommendation .....	14
<b>8</b>	<b>References</b>	<b>16</b>

# 1 ABSTRACT

Most of the houses in a city or town get their water supply from their particular City Corporation or Water Board. Majority of the areas in a city have a water supply schedule planned. The people living in such areas manage their water usage according to the schedule and have to manually note down the next date for their water supply. Due to circumstances such as technical difficulties with the reservoir or the water board, a sector may not get the water supply on the expected day which might lead the household with no water and also create much commotion due to confusion. Our project automates this process by informing the users on what day their particular sector is going to get the water supply and if there are any delays to a particular sector, the local officer can inform to the database admin and the water supply date will be updated accordingly. In addition to this the user can also check the water bill information and can also get the contact information of their sector's water supply contractor. Thus, our project will act as a solution to all the water supply problems of a household.

# 2 INTRODUCTION

There are a lot of responsibilities on a municipality and one of it is the Water management system. Water is a precious resource to all living beings and should be used with caution. The water distribution system of a city is dependent upon the local natural resources available and its efficiency is measured by how well is it able to satisfy the city's needs. An efficient system can be built with the help proper set of pumps and pipes situated strategically to ensure proper flow of water throughout the city. From a perspective of power consumption and usage requirement, water is supplied at a regular interval such that the people are able to use it within that time period with the stimulation of the need for proper storage of water and to manage any unnecessary consumption. A proper timetable is required to be maintained so that every region of the city gets adequate water supply. This project aims to build a database for the water management system for a city and also automate the billing sector for the ease of the consumers. In normal circumstances one would have to know about the last date of water supply and then calculate the next date. And if suppose a change in the schedule is implemented then it could be a headache if you are not up to date with people. The necessity of designing this application is that let the people know supply is started and they will be ready to store water thus reducing the water wastage too. When it comes to paying the bills the scenario would be a person coming to your house and handing over a bill which you would have to then pay at a government handled information center or at the water management station of the city. This project pools all of the actions above to one website with net database which will produce accurate results for every individual with no errors.

### 3 LITERATURE SURVEY

There are many important research studies and experimental work carried out using various techniques such as Cloud computing, micro controllers like Arduino and microprocessors like Raspberry Pi. The data offered by these studies and experiments inspired us to take up this project.

A. Merchant, M.S. Mohan Kumar, P.N. Ravindra, P. Vyas, U. Manohar [1] introduced a new water management software that is centered around powerful dash boarding, background analytics, management through exception and codifying standard operating procedures. This new water management software supports customizable key performance indicators (KPIs), business rules for managing water flow, real time reporting on a rich geo-spatial visual. The paper describes the implementation and results of the first phase of a roadmap to provide a state-of-the-art water management system to Bangalore.

K. Mohammed Shahanas, P. Bagavathi Sivakumar [2] reviewed different technologies and platforms that are required for a smart environment. An architecture design for Smart water management is proposed and an implementation detail of Smart water monitoring system is discussed.

Mahesh Junnare, Nikhil Khairnar, Vaibhav Karpe, Maruti ware [3] proposed a system for monitoring the water supply provided in the city. Their application is handled by the municipal corporation and it is usable for people who daily surviving peak flow problem. The proposed system overcomes the problem of household women or people who don't know the information about today's water supply.

#### 3.1 Existing Systems

In circumstances where the water supply schedule is changed, the information to the public is not conveyed properly. Generally, it is conveyed via the newspaper or by local News channels. For this system to work people have to be up to date with the regular changes in the neighborhood and be in sync with newspapers and TV channels. This is a good practice but however there may arise some circumstances where you aren't in contact with these details therefore it becomes rather a bothersome task

The current system however gives the option to pay the bills digitally but is very inefficient to do so. Much problems are faced in this process as the login procedure is way too complicated.

This system logic can be tweaked in a certain way to make it easier for the people to process their transaction.

## 4 DATA FLOW DIAGRAMS

### 4.1 Entity Relationship

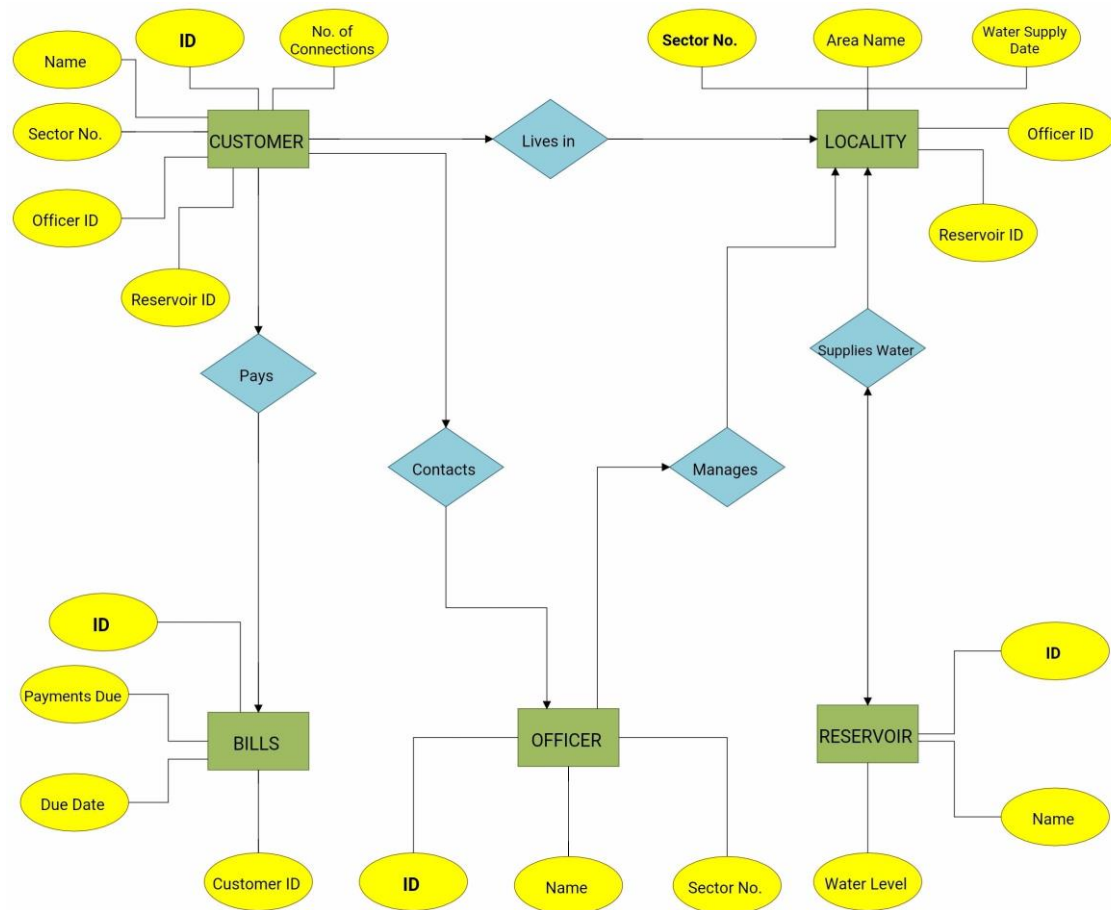


Figure 1: ER Diagram for Water management system.

### 4.2 Relationship schema

OFFICER (id, Name, sector\_no) RESERVOIR (id, Name, Water\_level)

BILL (id, customer\_id, Payments\_Due, due\_Date)

LOCALITY (sector\_no, Area\_Name, Water\_Supply\_Date, officer\_id, reservoir\_id)

CUSTOMER (id, Name, Address, sector\_no, officer\_id, reservoir\_id, no\_of\_connection)

## 5 METHODOLOGY

### 5.1 Tools Used

- **Python** was used as a database connecting language.
- **Sqlite3** with the inbuilt Sqlite3 module of Python was used for the back-end part.
- **Tkinter** is the Python interface to the Tk GUI toolkit shipped with Python

Python provides a SQL interface that is compatible with the DB-API 2.0 specification described in PEP 249. You may not need to load this module separately, as it is shipped with Python by default.

To use the Sqlite3 module, you must first create a connection object that represents a database and then, optionally, create a cursor object that will allow you to execute all SQL statements.

Tkinter is the standard Python GUI library. When paired with Tkinter, Python offers a quick and simple way to build GUI applications. Tkinter offers a versatile object-oriented interface for the Tk GUI toolkit.

Creation of GUI application is an easy task which needs the following steps:

- Import the Tkinter module.
- Create the main GUI application window.
- Add the desired widgets to the GUI application.
- Enter the main Event loop to take action against each event triggered by the user.

### 5.2 Code snippets

#### 5.2.1 Backend

To show the logic used in the back-end part of the project, the Officer relation is elaborated. We first

import the Sqlite3 library in the code so as to enable communication with Sqlite3 module using the statement:

```
import sqlite3
def officerData():
    con = sqlite3.connect("backend.db")
    cur = con.cursor()
    cur.execute("PRAGMA foreign_keys = ON")
    cur.execute("CREATE TABLE IF NOT EXISTS officer (id INTEGER
    PRIMARY KEY, Name text, sector_no text, FOREIGN KEY( ' sector_no' )
    REFERENCES ' Locality' ( ' sector_no' ) ON UPDATE CASCADE ON
    DELETE CASCADE)")
    con.commit()
    con.close()
```

**The above code is used to create the Table Officer in the database.**

‘con’ is used to store the value of the database which is being handled.

‘cur’ is used as a cursor which will be used to manipulate the elements of the database.

Officer has multiple attributes:

1.id

This attribute is a unique identification number for every officer.

This attribute is also used as the primary key for the table since it uniquely identifies each record in the table.

The attribute is in Integer format.

2.Name

This attribute provides us with the name of the officer. This attribute is in text format.

3.sector\_no


This attribute gives us the sector number of the locality for which the officer is assigned. The attribute is a foreign key which references the relation

‘Locality’

‘ON UPDATE CASCADE ON DELETE CASCADE’ is used to maintain the integrity constrain of the database.

We then save the data in the database by using ‘con.commit()’. We then close the connection to our database.

We get the following result after the above code snippet is executed.

Table:  officer

id	Name	sector_no
Filter	Filter	Filter

Figure 2: Officer Table

```
def addOfficer(id, Name, sector_no):  
    con = sqlite3.connect("backend.db")  
    cur = con.cursor()  
    cur.execute("PRAGMA foreign_keys = ON")  
    cur.execute("INSERT INTO officer VALUES (?, ?, ?)", (id, Name,  
    sector_no)) con.commit()  
    con.close()
```

**The above code is used to add a record to the Officer relation.**



The values for the attribute id, Name and sector\_no are to be provided while executing. The following is the output table for Officer.

Table: officer			
	id	Name	sector_no
	Filter	Filter	Filter
1	1	Rohan	1

Figure 3: Inserted value

```
def viewOfficer():
    con = sqlite3.connect("backend.db")
    cur = con.cursor()
    cur.execute("PRAGMA foreign_keys = ON")
    cur.execute("SELECT * FROM officer")
    rows = cur.fetchall()
    con.close()
    return rows
```

**The above code is used to view the Officer table from the database.**

The code returns all the tuples present in the Officer table which will be further used for other queries.

```
def delOfficer(id):
    con = sqlite3.connect("backend.db")
    cur = con.cursor()
    cur.execute("PRAGMA foreign_keys = ON")
    cur.execute("DELETE FROM officer WHERE id=?", (id,))
    con.commit()
    con.close()
```

**The above code is used to delete a record from the Officer Table.**

To automate the dates of water supply to each area of the city we use datetime library to manipulate and store date and time information.

```
def updateDateEveryday():
    seclist = []
    nextdate = 4
    d = 0
    today = (datetime.date.today())
    timedelta = (datetime.timedelta(days = nextdate))
    con = sqlite3.connect('backend.db')
    cur = con.cursor()
    cur.execute("SELECT Water_Supply_Date FROM Locality")
    dAteS = cur.fetchall()
    for every in dAteS: EveryDate = every[0]
```

```

string_date = changeToDate(EveryDate)
seclist.append(string_date)
appended_Date = today + timedelta
for all in seclist:
    if all == datetime.date.today():
        seclist[d] = appended_Date
        d=d+1
for all in seclist:
    dst = changeToString(all)
cur.execute("UPDATE Locality SET Water_Supply_Date = ? WHERE
TRUE",(dst,)) con.commit()
cur.close()

```

**The above code offers a variability to change the frequency between the water supply dates in the event of water shortage.**

The dates are stored in string format in the database and are manipulated in the form of date object through datetime module.

## 5.2.2 Frontend

The following snippets of code use Tkinter which is used to create the GUI window.

```

from tkinter import* import tkinter.messagebox from tkinter import ttk import random
import time import datetime import pymysql
import tempfile, os import b as backend

```

We start by importing the required modules from the Tkinter Library.

We also import the "b.py" which is the backend file which handles the database operations.

```

def iExit():
    iExit = tkinter.messagebox.askyesno("Exit", "Confirm if you want to
exit")
    if iExit>0:
        root.destroy()
        return

def iReset():
    self.txtid.delete(0, END)
    self.txtName.delete(0, END)
    self.cbsector_no.current(0)

def addData():
    if id.get() == "" or Name.get() == "" or
sector_no.get()=="tkinter.messagebox.askyesno("Error", "Please
enter the correct Data")
    else:
        backend.addOfficer(id.get(), Name.get(), sector_no.get())

    displayData()
    super(self.officerlist, self).delete()
    self.officerlist.insert(END,(id.get(),Name.get(), sector_no.get()))

def displayData():

```

```

        result = backend.viewOfficer()
        if len(result)!=0:
            self.officerlist.delete(*self.officerlist.get_children())
        for row in result:
            self.officerlist.insert(' ', END, values = row)

def deleteData():
    if(len(id.get())!= 0):
        backend.delOfficer(sd[0])
        iReset()
        displayData()
        tkinter.messagebox.showinfo("Delete", "Record successfully deleted")

def update():
    if(len(id.get()) != 0):
        backend.delOfficer(sd[0])

    if(len(id.get())!= 0):
        backend.addOfficer(id.get(), Name.get(), sector_no.get())

    displayData()

```

The above snippets of code describe the various functions used to communicate with their corresponding backend part which manipulates the database accordingly.

```

MainFrame = Frame(self.root, bd = 10, width = 1350, height = 700, relief =
RIDGE, bg = "cadet blue")
MainFrame.grid()

```

This code gives the blueprint of the frame construction which is required to build various GUI elements such as buttons, TreeView of the Database, Textbox, etc.

Numerous other frames can be created by deriving properties of the mainframe.

```

self.lblid = Label(WidgetFrame, font = ('arial',12,'bold'), text = 'Officer ID ',
bd = 7, anchor='w', justify=LEFT)
self.lblid.grid(row=0,column=0,sticky =W,padx=5)
self.txtid = Entry(WidgetFrame, font = ('arial',12,'bold'), bd = 5, width = 4)
self.txtid.grid(row=0, column=1)

```

**This code provides an example of how widgets are created in Tkinter.**

TreeView of the database allows us to build a tree-like structure and insert items accordingly, along with their attributes.

This type of viewing simplifies the maintenance of the database.

```

scroll_x = Scrollbar(TreeViewFrame, orient = HORIZONTAL) scroll_y =

```

```

Scrollbar(TreeViewFrame, orient = VERTICAL)
self.officerlist = ttk.Treeview(TreeViewFrame, height = 12, columns = ("id",
"Name", "sector_no"), xscrollcommand = scroll_x.set, yscrollcommand =
scroll_y.set)

scroll_x.pack(side = BOTTOM, fill = X)
scroll_y.pack(side = BOTTOM, fill = Y)

self.officerlist.heading("id", text = "Officer ID")
self.officerlist.heading("Name", text = "Officer Name")
self.officerlist.heading("sector_no", text = "Sector No")

self.officerlist['show'] = 'headings'
self.officerlist.column("id", width = 70)
self.officerlist.column("Name", width = 150)
self.officerlist.column("sector_no", width = 70)
self.officerlist.pack(fill = BOTH, expand = 1)
self.officerlist.bind("<ButtonRelease-1>", officerREC)
displayData()

```

**The above code snippet allows us to create a TreeView of the database.**

```

self.btnAddNew = Button(ButtonFrame, pady = 1, bd = 4,
font = ('arial', 20, 'bold'), text = "Insert New", padx = 24,
width = 8, height = 1, command = addData).grid(row = 0, column = 0,
padx = 1)

```

**The above code shows an example of how a button is design.**



6 RESULTS

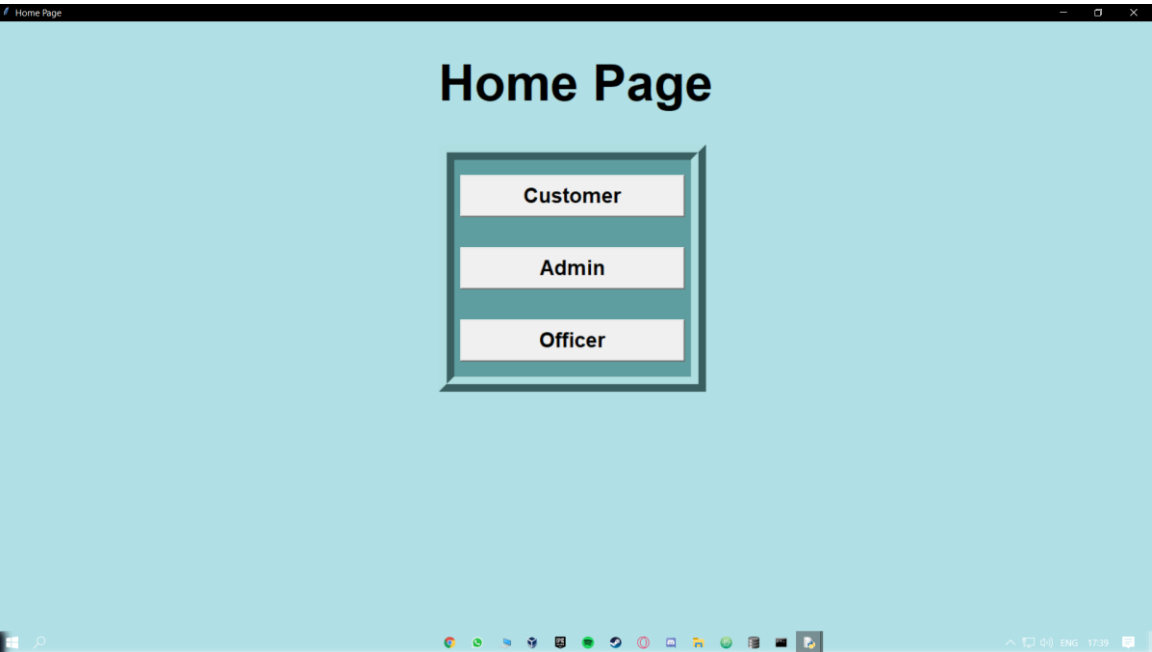


Figure 4: Homepage

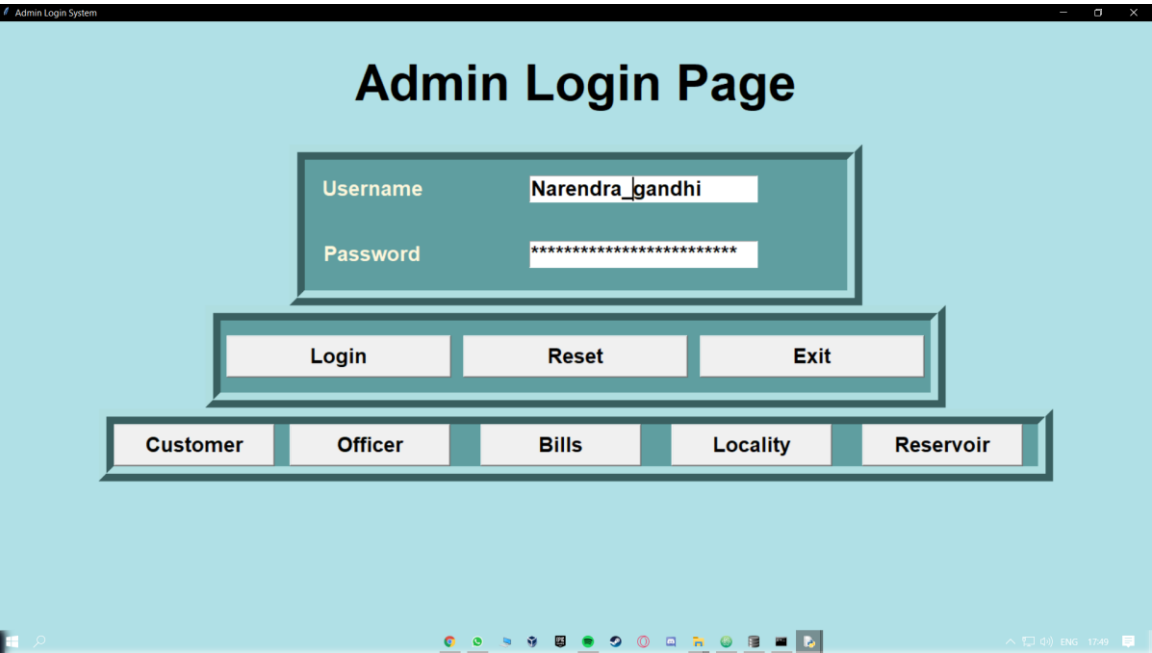


Figure 5: The Admin Login window

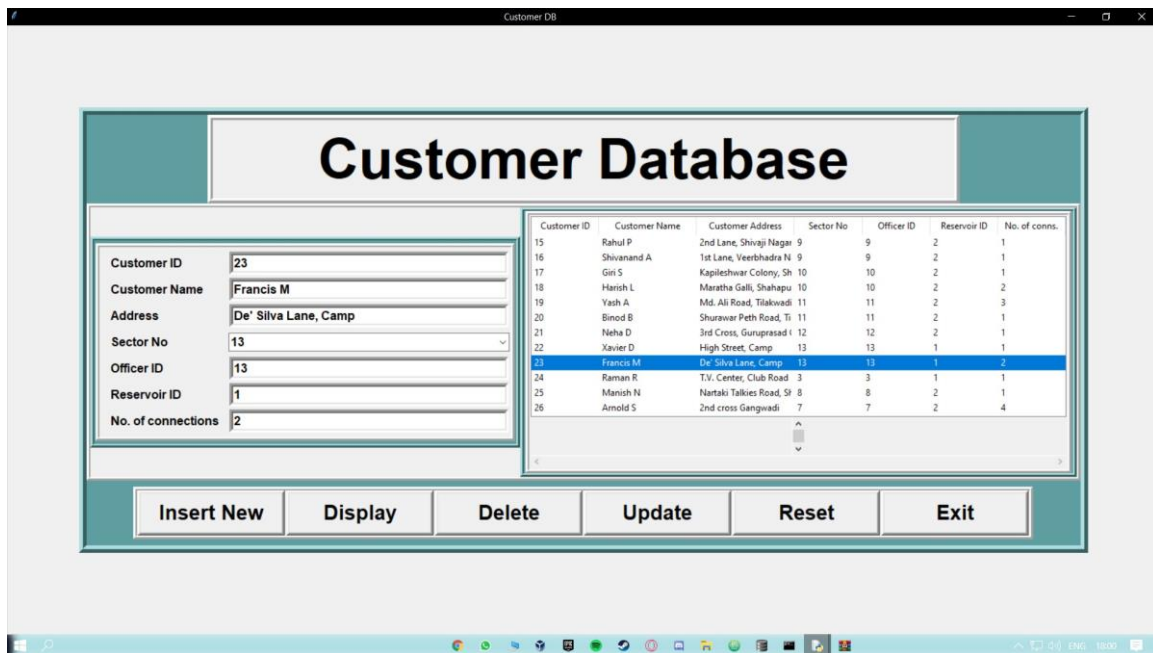


Figure 6: Customer Database

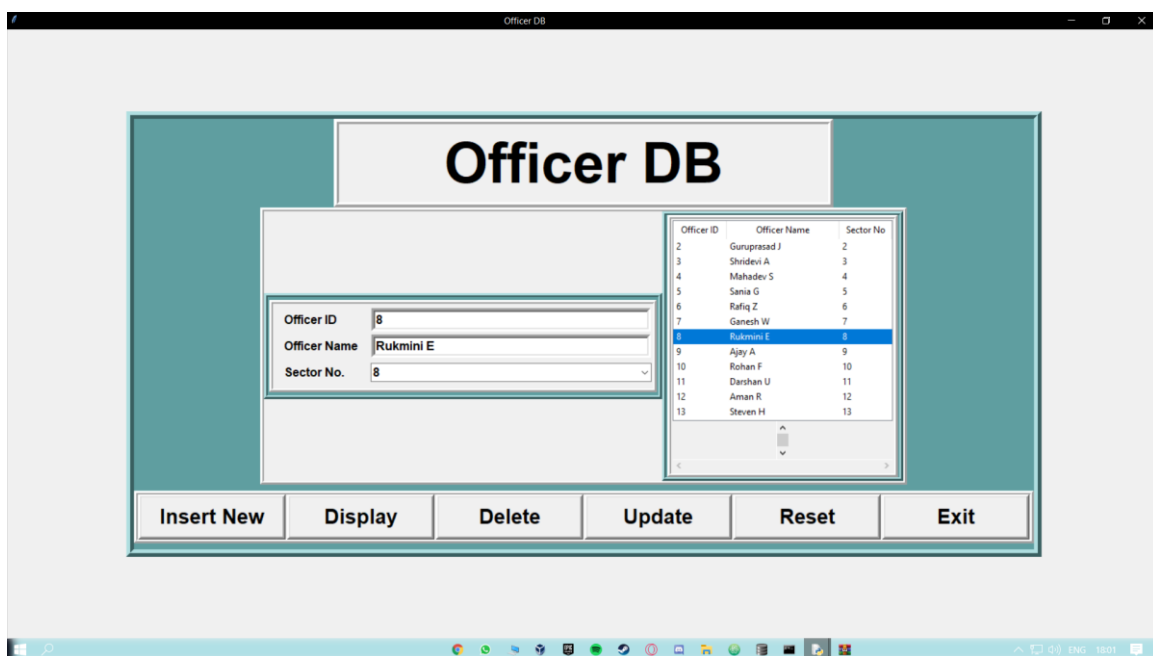


Figure 7: Officer Database

Locality DB

Sector No.	Area Name	Water Supply Date	Officer ID	Reservoir ID
1	College Road	01 February 2021	1	2
2	Hanuman Nagar	02 February 2021	2	1
3	Club Road	03 February 2021	3	1
4	Sadashiv Nagar	04 February 2021	4	1
5	Shahu Nagar	05 February 2021	5	1
6	Azam Nagar	07 February 2021	6	1
7	Mahantesh Nagar	08 February 2021	7	2
8	Belgaum Fort	09 February 2021	8	2
9	R.T.O Office	10 February 2021	9	2
10	Shivaji Garden, Shahapur	12 February 2021	10	2
11	1st Railway Gate, Tilakwadi	13 February 2021	11	2
12	Guruprasad Colony	15 February 2021	12	2

Sector No.   
 Area Name   
 Water Supply Date   
 Officer ID   
 Reservoir ID

Insert New   Display   Delete   Update   Reset   Exit

Figure 8: Locality Database

Billing DB

Bill ID	Customer ID	Payment Due	Due Date
25	18	₹619	05 February 2021
26	18	₹205	05 February 2021
27	19	₹467	30 March 2021
28	19	₹702	30 March 2021
29	19	₹145	30 March 2021
30	20	₹230	15 March 2021
31	21	₹358	16 February 2021
32	22	₹381	19 April 2021
33	23	₹453	19 March 2021
34	23	₹236	01 April 2021
35	24	₹267	05 May 2021
36	25	₹347	07 April 2021

Bill ID   
 Customer ID   
 Payment Due   
 Due Date

Insert New   Display   Delete   Update   Reset   Exit

Figure 9: Billing Database



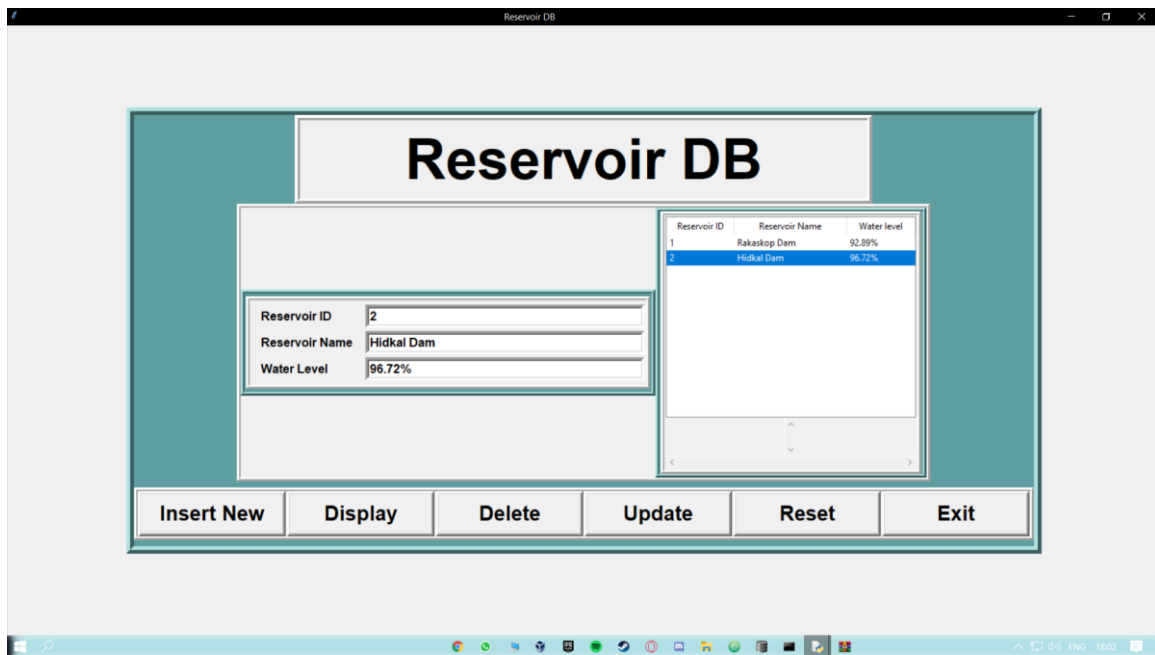


Figure 10: Reservoir Database

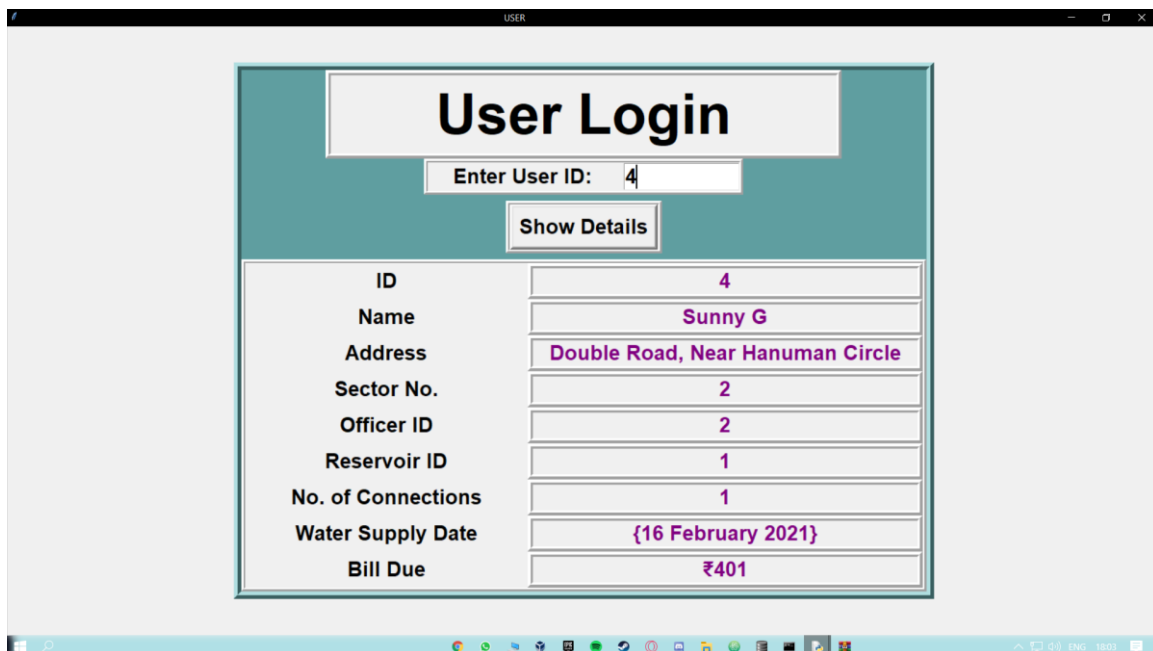


Figure 11: User Login Page

Customer ID	Customer Name	Customer Address	Sector No	Officer ID	Reservoir ID	No. of conns.
12	Twinkl K	Mahantesh Nagar Road	7	7	2	2
26	Arnold S	2nd cross Gangwadi	7	7	2	4

Figure 12: Customers details that fall under a particular officer's management

## 7 CONCLUSION AND RECOMMENDATIONS

### 7.1 CONCLUSION

It clearly evident that in the forthcoming years will approach humanity with times where we will not be able to supply or satisfy the demands of the natural resources due to the over usage of them.

One of these resources would be clean water more importantly clean drinking water. The solution for this problem would be to manage our current resources to limit to our basic needs and not waste them.

In approach to saving resources, the one that do get consumed should be properly managed with utmost precision. The practice of water supply management system has been a huge folly to this matter. The current system can be met with a different approach which ensures that every person gets his water supply on correct time.

In the city of Belgaum, Karnataka the people feel that the current methods do not meet the fulfillment a proper water management system. The current method although maintains a good record of timely supply of water but do not explain the emergency non-notified change of the schedule.

The problem's basic roots are enveloped with the lack of proper communication with the concerned authorities and the general public. This project will solve this problem by moving

the information base of this system to a web-based platform which anybody can access at any time.

## 7.2 Recommendation

This project maybe further more improved by adding features such as using IOT based appliances with a combination of server data to automate the process of supplying the water to different areas of the city.

After having a proper schedule which is now accustomed to the public, the consumption of every area can be separately measured and the unnecessary wastage can be cut out.

A smart city should not only have smart supply system but also should contain a good home management system where the user can view their daily consumption.

A mobile application can be created from which the customers can get their notification alerts from the water board itself be it their payments or the next water supply date.

## 8 REFERENCES

- 1.A. Merchant, M.S. Mohan Kumar, P.N. Ravindra, P. Vyas, U. Manohar; Analytics Driven Water Management System for Bangalore City, Procedia Engineering, Volume 70, 2014, Pages 1137- 1146, ISSN 1877-7058, <https://doi.org/10.1016/j.proeng.2014.02.126>.  
(<http://www.sciencedirect.com/science/article/pii/S1877705814001283>)
- 2.K. Mohammed Shahanas, P. Bagavathi Sivakumar, Framework for a Smart Water Management System in the Context of Smart City Initiatives in India, Procedia Computer Science, Volume 92, 2016, Pages 142-147, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2016.07.337>.  
(<http://www.sciencedirect.com/science/article/pii/S1877050916315848>)
3. Mahesh Junnare, Nikhil Khairnar, Vaibhav Karpe, Maruti ware; Water Supply Management System for Municipal Corporation – A SURVEY PAPER, International Journal of Engineering Science and Computing, Volume 6 Issue No. 11, 2016, (<https://ijesc.org/upload/920f37ed2f0af529dc37aaff88cbeabc.Water%20Supply%20Management%20System%20for%20Municipal%20Corporation%20A%20SURVEY%20PAPER.pdf>)
4. K Mutchek, M.; Williams, E. Moving Towards Sustainable and Resilient Smart Water Grids. Challenges 2014, 5, 123-137. <https://doi.org/10.3390/challe5010123>