



## **Dept. of Electrical and Electronics Engineering**

**19ELC312 – DATABASE SYSTEMS AND PROGRAMMING**

### **CASE STUDY REPORT ON**

**“Data enabled Smart Water Distribution/Management System”**

**Team Number: 13**

#### **Team Members:**

Nivedhitha G	-	CB.EN.U4ELC22036
Pranav Karthikeyan	-	CB.EN.U4ELC22062

# 19ELC312 – Data Base Systems and Programming

## Case Study – Group 13

### **DATA-ENABLED SMART WATER DISTRIBUTION SYSTEM**

**Nivedhitha G** – CB.EN.U4ELC22036

**Pranav Karthikeyan** – CB.EN.U4ELC22062

#### **Abstract:**

Water wastage and inefficiencies in distribution remain critical challenges, leading to resource depletion, high operational costs, and energy inefficiency. This project proposes an IoT-enabled Smart Water Distribution System that integrates real-time monitoring, database management, and adaptive control to optimize water usage, detect leakages, and enhance sustainability.

The system employs IoT sensors to continuously track water flow, pressure, consumption, and leakage. This data is processed using a Python-MySQL backend, ensuring efficient storage, analysis, and anomaly detection. A user-friendly dashboard provides real-time insights, automated alerts, and remote-control functionalities. Predictive analytics enhance demand forecasting, enabling smart water allocation and energy-efficient distribution.

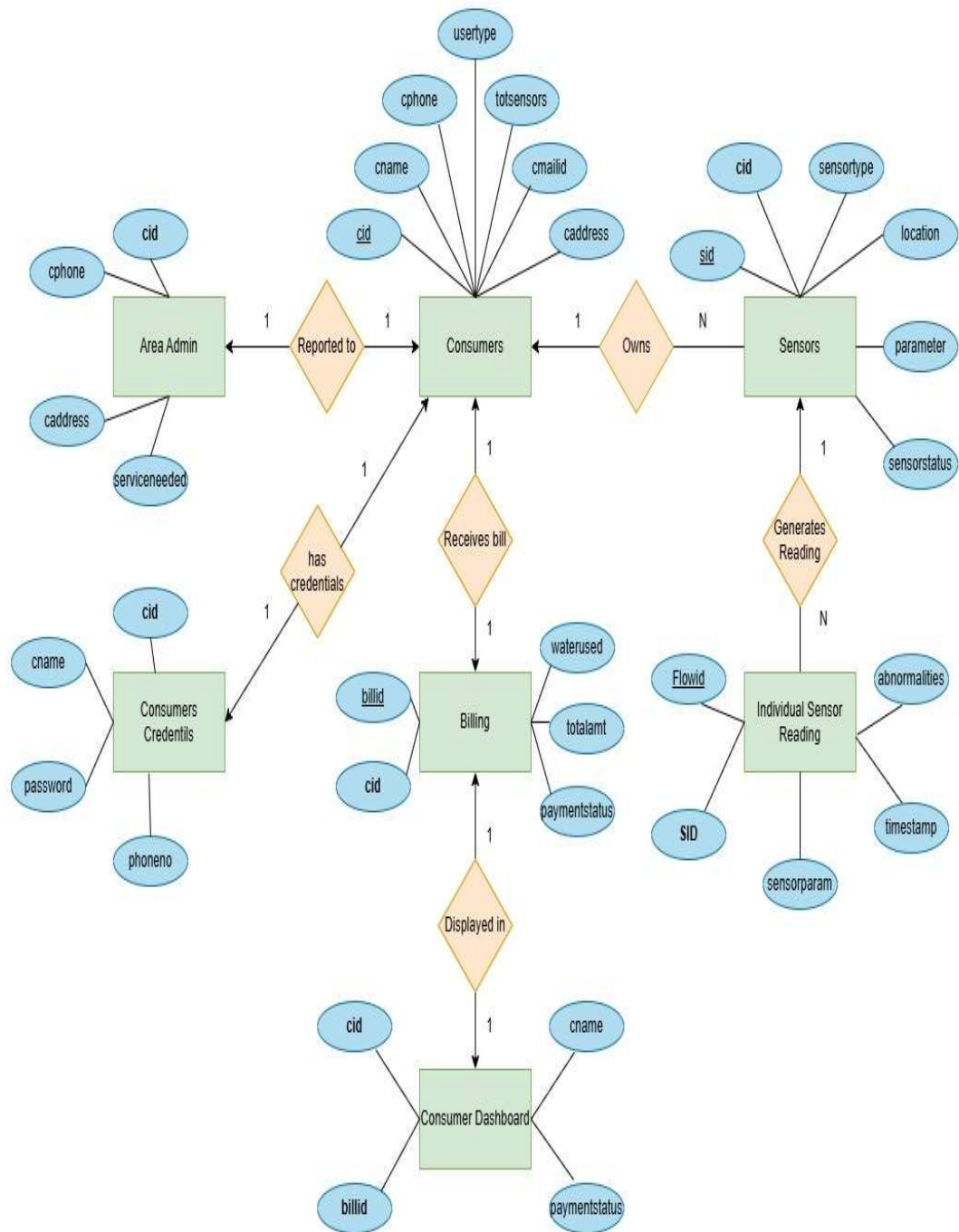
Key features include:

- Real-time Monitoring & Alerts – Detects anomalies like leakages and excessive usage.
- Adaptive Water Distribution – Dynamically adjusts supply based on demand patterns.
- Energy Efficiency & Cost Reduction – Optimizes resource allocation to minimize losses.
- Data Analytics & Predictive Maintenance – Prevents failures and improves long-term planning.

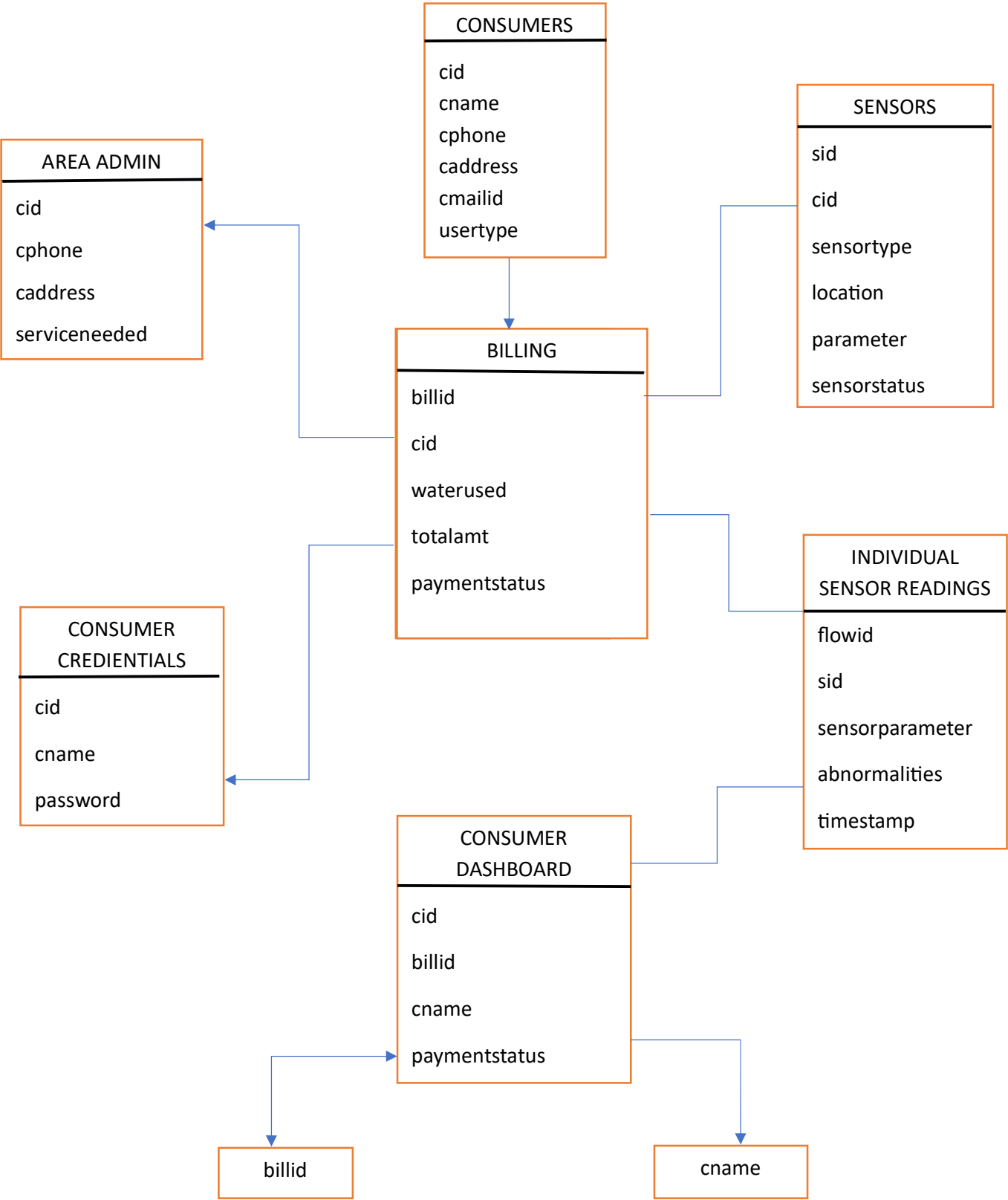
This scalable and intelligent solution empowers stakeholders—municipalities, industries, and residential users—to ensure a sustainable, efficient, and resilient water management

system. By integrating automation, AI-driven analytics, and IoT-based monitoring, the system revolutionizes modern water distribution networks.

ER Diagram:



Schema Diagram:



Normalizations:

### 1. First Normal Form (1NF)

- **Atomicity of Attributes:** All attributes must contain atomic (indivisible) values. This means no multi-valued attributes should exist in a single row.
- **Uniqueness of Rows:** Every table should have a **primary key** that uniquely identifies each record.
- **No Repeating Groups:** Each record should be represented in one row, and each field in the row should store a single value.

### Schema Example in 1NF:

#### 1. Consumers Table

- cid (Primary Key)
- cname
- cphone
- cemailid
- caddress
- usertype
- totsensors

#### 2. Consumers Credentials Table

- cid (Primary Key, Foreign Key referencing Consumers.cid)
- password
- phoneno

#### 3. Sensors Table

- sid (Primary Key)
- cid (Foreign Key referencing Consumers.cid)
- sensortype
- location
- parameter
- sensorstatus

#### 4. Individual Sensor Reading Table

- flowid (Primary Key)
- sid (Foreign Key referencing Sensors.sid)
- sensorparam
- abnormalities
- timestamp

## 5. Billing Table

- billid (Primary Key)
- cid (Foreign Key referencing Consumers.cid)
- waterused
- totalamt
- paymentstatus

## 2. Second Normal Form (2NF)

- **Eliminate Partial Dependency:** Non-prime attributes (attributes not part of the primary key) must depend on the **entire** primary key and not just part of it.
- **Remove Redundancy:** If attributes are partially dependent on a composite primary key, split them into separate tables.

### Schema Example in 2NF:

#### 1. Startup Table

- Startup\_ID (Primary Key)
- Name
- Founding\_Year

#### 2. Startup Details Table

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Industry
- Location

#### 3. Startup Funding Table

- Startup\_Record\_ID (Primary Key)

- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Growth\_Stage
- Total\_Funding

#### 4. User Adoption Table

- User\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Number\_of\_Users

#### 5. Startup Adoption Metrics Table

- Startup\_ID (Primary Key)
- Growth\_Rate
- Feedback\_Score

### 3. Third Normal Form (3NF)

- **Eliminate Transitive Dependency:** Non-prime attributes must not depend on other non-prime attributes.
- **Ensure that every non-key attribute is only dependent on the key,** not on other attributes.

#### Schema Example in 3NF:

##### 1. Startup Table

- Startup\_ID (Primary Key)
- Name
- Founding\_Year

##### 2. Startup Details Table

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Industry

##### 3. Industry Details Table

- Industry\_ID (Primary Key)
- Industry

- Location

#### 4. Startup Funding Table

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Growth\_Stage
- Total\_Funding

#### 5. Technology Table

- Tech\_ID (Primary Key)
- Name

#### 6. Tech Year Table

- Tech\_ID (Foreign Key referencing Technology.Tech\_ID)
- First\_Use\_Year

#### 7. Tech Details Table

- Tech\_Record\_ID (Primary Key)
- Tech\_ID (Foreign Key referencing Technology.Tech\_ID)
- Type

### 1. First Normal Form (1NF)

- **Atomicity of Attributes:** All attributes must contain atomic (indivisible) values. This means no multi-valued attributes should exist in a single row.
- **Uniqueness of Rows:** Every table should have a **primary key** that uniquely identifies each record.
- **No Repeating Groups:** Each record should be represented in one row, and each field in the row should store a single value.

#### Schema Example in 1NF:

##### 1. Consumers Table

- cid (Primary Key)
- cname
- cphone
- cemailid



- caddress
- usertype
- totsensors

## 2. Consumers Credentials Table

- cid (Primary Key, Foreign Key referencing Consumers.cid)
- password
- phoneno

## 3. Sensors Table

- sid (Primary Key)
- cid (Foreign Key referencing Consumers.cid)
- sensortype
- location
- parameter
- sensorstatus

## 4. Individual Sensor Reading Table

- flowid (Primary Key)
- sid (Foreign Key referencing Sensors.sid)
- sensorparam
- abnormalities
- timestamp

## 5. Billing Table

- billid (Primary Key)
- cid (Foreign Key referencing Consumers.cid)
- waterused
- totalamt
- paymentstatus

## 2. Second Normal Form (2NF)

- **Eliminate Partial Dependency:** Non-prime attributes (attributes not part of the primary key) must depend on the **entire** primary key and not just part of it.
- **Remove Redundancy:** If attributes are partially dependent on a composite primary key, split them into separate tables.

#### **Schema Example in 2NF:**

##### **1. Startup Table**

- Startup\_ID (Primary Key)
- Name
- Founding\_Year

##### **2. Startup Details Table**

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Industry
- Location

##### **3. Startup Funding Table**

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Growth\_Stage
- Total\_Funding

##### **4. User Adoption Table**

- User\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Number\_of\_Users

##### **5. Startup Adoption Metrics Table**

- Startup\_ID (Primary Key)
- Growth\_Rate
- Feedback\_Score

### **3. Third Normal Form (3NF)**

- **Eliminate Transitive Dependency:** Non-prime attributes must not depend on other non-prime attributes.
- **Ensure that every non-key attribute is only dependent on the key,** not on other attributes.

**Schema Example in 3NF:**

**1. Startup Table**

- Startup\_ID (Primary Key)
- Name
- Founding\_Year

**2. Startup Details Table**

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Industry

**3. Industry Details Table**

- Industry\_ID (Primary Key)
- Industry
- Location

**4. Startup Funding Table**

- Startup\_Record\_ID (Primary Key)
- Startup\_ID (Foreign Key referencing Startup.Startup\_ID)
- Growth\_Stage
- Total\_Funding

**5. Technology Table**

- Tech\_ID (Primary Key)
- Name

**6. Tech Year Table**

- Tech\_ID (Foreign Key referencing Technology.Tech\_ID)
- First\_Use\_Year

**7. Tech Details Table**

- Tech\_Record\_ID (Primary Key)
- Tech\_ID (Foreign Key referencing Technology.Tech\_ID)
- Type

### First Normal Form (1NF):

cid	cname	usertype	cphone	cemailid	caddress	sid	sensortype	sensorstatus	readingvalue	billdet
C001	Alice	Domestic	9876543210	alice@mail.com	City A	S001	Water Flow	Active	3.5 L	\$30
C001	Alice	Domestic	9876543210	alice@mail.com	City A	S002	Water Level	Inactive	60%	\$30
C002	Bob	Industrial	8765432109	bob@mail.com	City B	S003	Water Flow	Active	5.0 L	\$50

### Second Normal Form (2NF):

#### Consumers Table

cid	cname	usertype	cphone	cemailid	caddress
C001	Alice	Domestic	9876543210	alice@mail.com	City A
C002	Bob	Industrial	8765432109	bob@mail.com	City B

#### Sensors Table

sid	sensortype	sensorstatus
S001	Water Flow	Active
S002	Water Level	Inactive
S003	Water Flow	Active

Consumer-Sensor Mapping Table

cid	sid
C001	S001
C001	S002
C002	S003

Sensor Readings Table

sid	readingvalue
S001	3.5 L
S002	60%
S003	5.0 L

Billing Table

billid	cid	billdetails	paymentstatus
B001	C001	\$30	Paid
B002	C002	\$50	Unpaid

**Third Normal Form (3NF):**

Payment Status Table

billid	paymentstatus
B001	Paid
B002	Unpaid

Sensor Parameters Table

sensorparamid	sid	parameter
SP001	S001	Flow Rate
SP002	S002	Water Level

## Final Decomposition for BCNF

### Consumer-Sensor Mapping (BCNF)

mapid	cid	sid
M001	C001	S001
M002	C001	S002
M003	C002	S003

### Sensor Readings (BCNF)

readingid	sid	sensorparamid	value	timestamp
R001	S001	SP001	3.5 L	2025-02-01 10:00:00
R002	S002	SP002	60%	2025-02-01 11:00:00

### CODE SAMPLES:

#### Models:

```
import sqlite3
```

```
from werkzeug.security import check_password_hash, generate_password_hash
```

```
# Create a database connection
```

```
def create_connection():
```

```
    conn = sqlite3.connect('database.db', timeout=10) # Timeout of 10 seconds
```

```
    conn.row_factory = sqlite3.Row # Rows behave like dictionaries
```

```
    conn.execute('PRAGMA journal_mode=WAL;') # Enable WAL mode
```

```
    return conn
```

```
# Function to initialize and create tables
```

```
def init_db():
```

```
    conn = create_connection()
```

```
cursor = conn.cursor()
```

```
# Drop the old sensors table if it exists
```

```
#cursor.execute("DROP TABLE IF EXISTS sensors")
```

```
# Create Users table (with added columns: profile_pic, address, user_type)
```

```
cursor.execute("""
```

```
CREATE TABLE IF NOT EXISTS users (
```

```
    id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    name TEXT NOT NULL,
```

```
    email TEXT NOT NULL UNIQUE,
```

```
    password_hash TEXT NOT NULL,
```

```
    profile_pic TEXT DEFAULT 'default.png',
```

```
    address TEXT DEFAULT "",
```

```
    user_type TEXT DEFAULT 'consumer'
```

```
)
```

```
""")
```

```
# Create Sensors table (fixed schema)
```

```
cursor.execute("""
```

```
CREATE TABLE IF NOT EXISTS sensors (
```

```
    sensor_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    user_id INTEGER,
```

```
    sensor_type TEXT,
```

```
    location TEXT,
```

```
    status TEXT,
```

```
    FOREIGN KEY(user_id) REFERENCES users(id)
```

```
)
```

```
''')
```

```
# Create Billing table (with payment_status column)
```

```
cursor.execute('''
```

```
CREATE TABLE IF NOT EXISTS billing (
```

```
    bill_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
    user_id INTEGER,
```

```
    amount_due REAL,
```

```
    due_date TEXT,
```

```
    payment_status TEXT DEFAULT 'Pending',
```

```
    FOREIGN KEY(user_id) REFERENCES users(id)
```

```
)
```

```
''')
```

```
cursor.execute('''
```

```
CREATE TABLE if not exists area_admin (
```

```
    user_id INTEGER PRIMARY KEY,
```

```
    service_needed TEXT,
```

```
    email TEXT NULL,
```

```
    service_status TEXT
```

```
);
```

```
''')
```

```
# Commit and close connection
```

```
conn.commit()
```

```
conn.close()
```



*# Function to get user details (including profile\_pic, address, user\_type)*

```
def get_user_details(user_id):  
    conn = create_connection()  
    cursor = conn.cursor()  
    cursor.execute("SELECT * FROM users WHERE id=?", (user_id,))  
    user = cursor.fetchone()  
    conn.close()  
    return user
```

*# Function to update user details (name, address, profile\_pic, user\_type)*

```
def update_user_details(user_id, name, address, profile_pic, user_type='consumer'):  
    conn = create_connection()  
    cursor = conn.cursor()  
    cursor.execute("""  
        UPDATE users SET name=?, address=?, profile_pic=?, user_type=? WHERE id=?  
    """, (name, address, profile_pic, user_type, user_id))  
    conn.commit()  
    conn.close()
```

*# Function to update user password*

```
def update_user_password(user_id, password):  
    password_hash = generate_password_hash(password)  
    conn = create_connection()  
    cursor = conn.cursor()  
    cursor.execute("""  
        UPDATE users SET password_hash=? WHERE id=?  
    """, (password_hash, user_id))  
    conn.commit()
```

```
conn.close()
```

*# Function to get the total number of sensors for a user*

```
def get_total_sensors(user_id):
```

```
    conn = create_connection()
```

```
    cursor = conn.cursor()
```

```
    cursor.execute("SELECT COUNT(*) FROM sensors WHERE user_id=?", (user_id,))
```

```
    total_sensors = cursor.fetchone()[0] # Extract the count value
```

```
    conn.close()
```

```
    return total_sensors
```

```
if __name__ == '__main__':
```




```
    init_db() # This will initialize the database and create the tables
```

Output:

Login:

The image shows a login/signup interface. The left panel is white and contains the 'Sign In' section. It features three social media icons (Facebook, Google, Twitter) with the text 'or use your account' below them. There are input fields for 'Email' and 'Password'. A link 'Forgot your password?' is located below the password field. A red 'SIGN IN' button is at the bottom of the white panel. The right panel is red and contains the 'Hello, Friend!' section. It includes the text 'Enter your personal details and start your journey with us' and a white 'SIGN UP' button.

## Sign In

or use your account

[Forgot your password?](#)

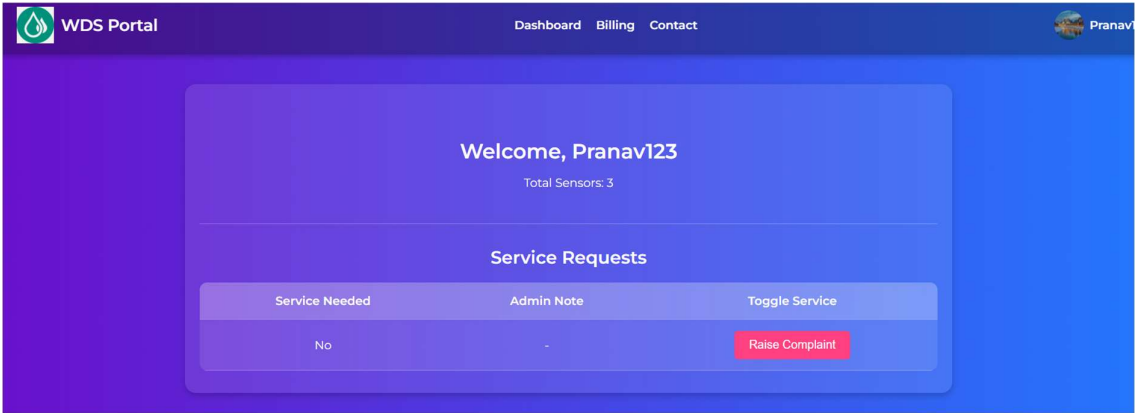
**SIGN IN**

## Hello, Friend!

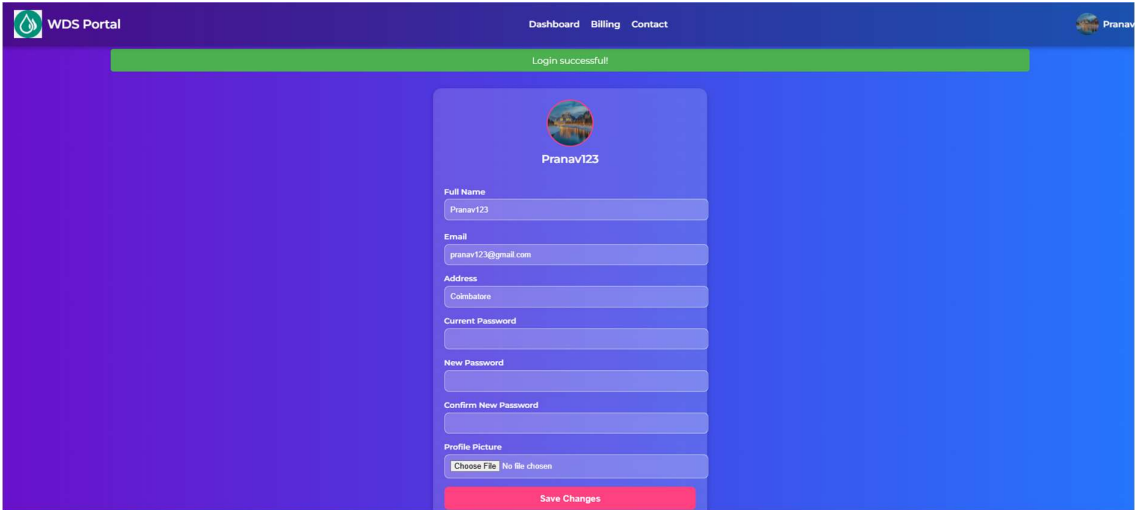
Enter your personal details and start your journey with us

**SIGN UP**

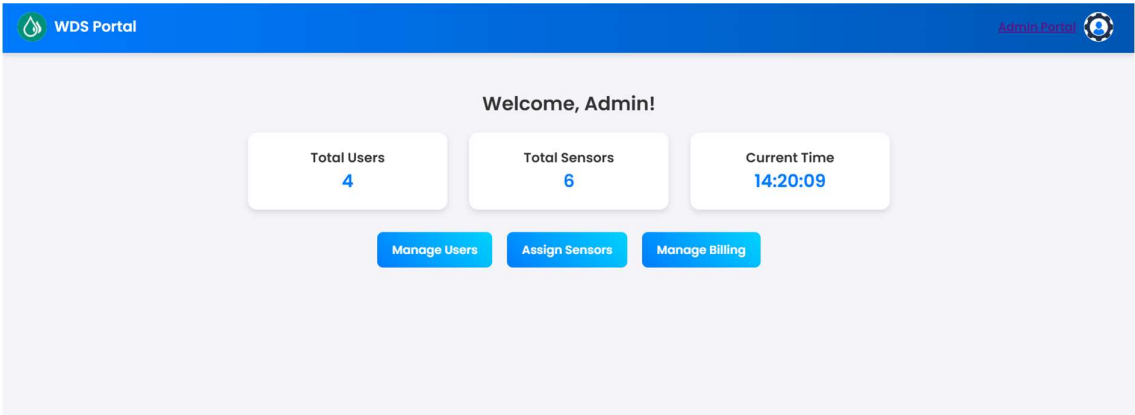
Dashboard:



User settings:



Admin portal:



Manage users:

Admin Dashboard

[Home](#)

Manage Users

User ID	Username	Email	Address	Actions
1	Pranav123	pranav123@gmail.com	Coimbatore	<a href="#">View Details</a>
2	Hello1	hello@gmail.com	Trichy	<a href="#">View Details</a>
4	helloworld	helloworld@gmail.com	Madurai	<a href="#">View Details</a>
5	NivedhithaG	nivedhitha@gmail.com	Vellore	<a href="#">View Details</a>

Assign/Manage sensors:

Manage Sensors

Sensor ID	User	Sensor Type	Location	Status	Actions
1	1 - Pranav123	Temperature	Ceiling	Active	<a href="#">Make Changes</a> <a href="#">Delete</a>
3	1 - Pranav123	Temperature	Ceiling	Active	<a href="#">Make Changes</a> <a href="#">Delete</a>
5	1 - Pranav123	Temperature	Outside	Inactive	<a href="#">Make Changes</a> <a href="#">Delete</a>
6	2 - Hello1	Flow	Wall	Inactive	<a href="#">Make Changes</a> <a href="#">Delete</a>
7	4 - helloworld	Flow	Underground	Active	<a href="#">Make Changes</a> <a href="#">Delete</a>
8	5 - NivedhithaG	Temperature	Underground	Active	<a href="#">Make Changes</a> <a href="#">Delete</a>

[Add New Sensor](#)[Back to Main Page](#)

Billing management:

Water Management

[Dashboard](#)[Billing](#)[Logout](#)

Billing Information

Bill ID	User ID	Amount Due (₹)	Due Date	Payment Status	Actions
1	1	1500	02/04/2025	Paid	<a href="#">Mark as Paid</a> <a href="#">Mark as Unpaid</a>
2	2	2500	01/04/2025	Pending	<a href="#">Mark as Paid</a> <a href="#">Mark as Unpaid</a>
3	5	20000	2025-04-23	Paid	<a href="#">Mark as Paid</a> <a href="#">Mark as Unpaid</a>
4	5	4000	2025-04-03	Pending	<a href="#">Mark as Paid</a> <a href="#">Mark as Unpaid</a>

[Add New Bill](#)