**Rural Water Supply and Sanitation System**

**BTech/III Year CSE/V Semester**

**15CSE302/Database Management Systems**

**Project Review -3**

|  |  |
| --- | --- |
| Rollno | Name |
| CB.EN.U4CSE18266 | **Venkatasubramanian N** |
| CB.EN.U4CSE18212 | **V Ashwin** |
| CB.EN.U4CSE18227 | **Gubbala Sri Ram** |
| CB.EN.U4CSE18234 | **K. Chandra Mohan Reddy** |

**Amrita School of Engineering, Coimbatore**

**Department of Computer Science and Engineering**

**2020 -2021 Odd Semester**

Table of Contents

**Title Page number**

Chapter 1- Introduction

[Objective 6](#_Toc56562181)

[STACK used: 6](#_Toc56562182)

[Abstract: 7](#_Toc56562183)

[Chapter 2 Logical Database Design ER Diagram 8](#_Toc56562184)

[Enhanced ER Diagram: 9](#_Toc56562185)

[Entities: 9](#_Toc56562186)

[Attributes: 10](#_Toc56562187)

[Relationships: 12](#_Toc56562188)

[Chapter 3 ER to Relational Schema Mapping 13](#_Toc56562189)

[Chapter 4 User Interface Screens 14](#_Toc56562190)

[Home Screen 14](#_Toc56562191)

[Abstract Page 15](#_Toc56562192)

[Donate Form – Blank 15](#_Toc56562193)

[Donate Form – Filled 16](#_Toc56562194)

[Donor Table 16](#_Toc56562195)

[Initial Water\_Supply\_and\_Sanitation Table: 17](#_Toc56562196)

[Functional Dependencies: 17](#_Toc56562197)

[19](#_Toc56562198)

[Functional Dependency Closure: 20](#_Toc56562199)

[Attribute Closure: 22](#_Toc56562200)

[Canonical Cover: 22](#_Toc56562201)

[Superkeys: 24](#_Toc56562202)

[Anomalies: 24](#_Toc56562203)

[To check if the relation is in First normal form 24](#_Toc56562204)

[Conditions satisfied and violated by the relation Water\_Supply\_And\_Sanitation\_System: 24](#_Toc56562205)

[Identify the Partial dependencies and Decompose the table and check if it is 2NF 25](#_Toc56562206)

[Conditions for a relation to be in Second Normal Form: 25](#_Toc56562207)

[Dependencies: 25](#_Toc56562208)

[Check Transitive dependencies and Decompose the table and check if it is 3NF 26](#_Toc56562209)

[Conditions for Third normal form: 26](#_Toc56562210)

[Check if it is Lossless decomposition using chase method and Check if it is Dependency preserving 27](#_Toc56562211)

[Dependency preserving: 29](#_Toc56562212)

[Normalize to BCNF: 29](#_Toc56562213)

[Final schema with Primary keys and dependency diagram. 30](#_Toc56562214)

[Dependency Diagram 31](#_Toc56562215)

[Location 32](#_Toc56562216)

[Jobs 32](#_Toc56562217)

[Employee 33](#_Toc56562218)

[WaterSources 33](#_Toc56562219)

[WaterUsage 34](#_Toc56562220)

[SanitationSystems 34](#_Toc56562221)

[Families 35](#_Toc56562222)

[Donations 35](#_Toc56562223)

[Expenditure 36](#_Toc56562224)

[A. Include the create command for ALL THE TABLES, sample insert commands B. Minimum 10 meaningful records to be inserted C. Data from all the tables 37](#_Toc56562225)

[Location Table: 44](#_Toc56562226)

[Jobs Table: 45](#_Toc56562227)

[Employee Table: 46](#_Toc56562228)

[Water Sources Table: 47](#_Toc56562229)

[Water Usage Table: 48](#_Toc56562230)

[Sanitation Systems Table: 49](#_Toc56562231)

[Expenditure Table: 50](#_Toc56562232)

[Donation Table: 51](#_Toc56562233)

[Connectivity 53](#_Toc56562234)

[Connectivity Code: 53](#_Toc56562235)

[Database Initialization: 53](#_Toc56562236)

[UI Screen – Home Page 61](#_Toc56562237)

[UI Screen – Employees 61](#_Toc56562238)

[Code – Back end: 61](#_Toc56562239)

[Code – Front end: 64](#_Toc56562240)

[Analytics 80](#_Toc56562241)

[UI Screen 80](#_Toc56562242)

[Back End 80](#_Toc56562243)

[Front end code 110](#_Toc56562244)

[Conclusion 129](#_Toc56562245)

[References 130](#_Toc56562246)

Chapter 1 Introduction:

The Failure of Rural Communities to understand the need for water resources as a social good and their inability to adhere to hygienic sanitation practices has led to the improper usage of water resources. There is also an acute need for a proper and easy to use health and sanitation system in rural areas. This makes the aim of the project to focus on four different components.

1. Improved Water Supply Facilities.
2. Hygienic Sanitation System.
3. Community Mobilization and Education.
4. Advanced Monitoring, Evaluating, and Supporting System.

# Objective

To improve the quality of lives of people living in rural areas by building a composite and decentralised rural water supply and sanitation system.

# STACK used:

Database – MySQL

Back end – NodeJs

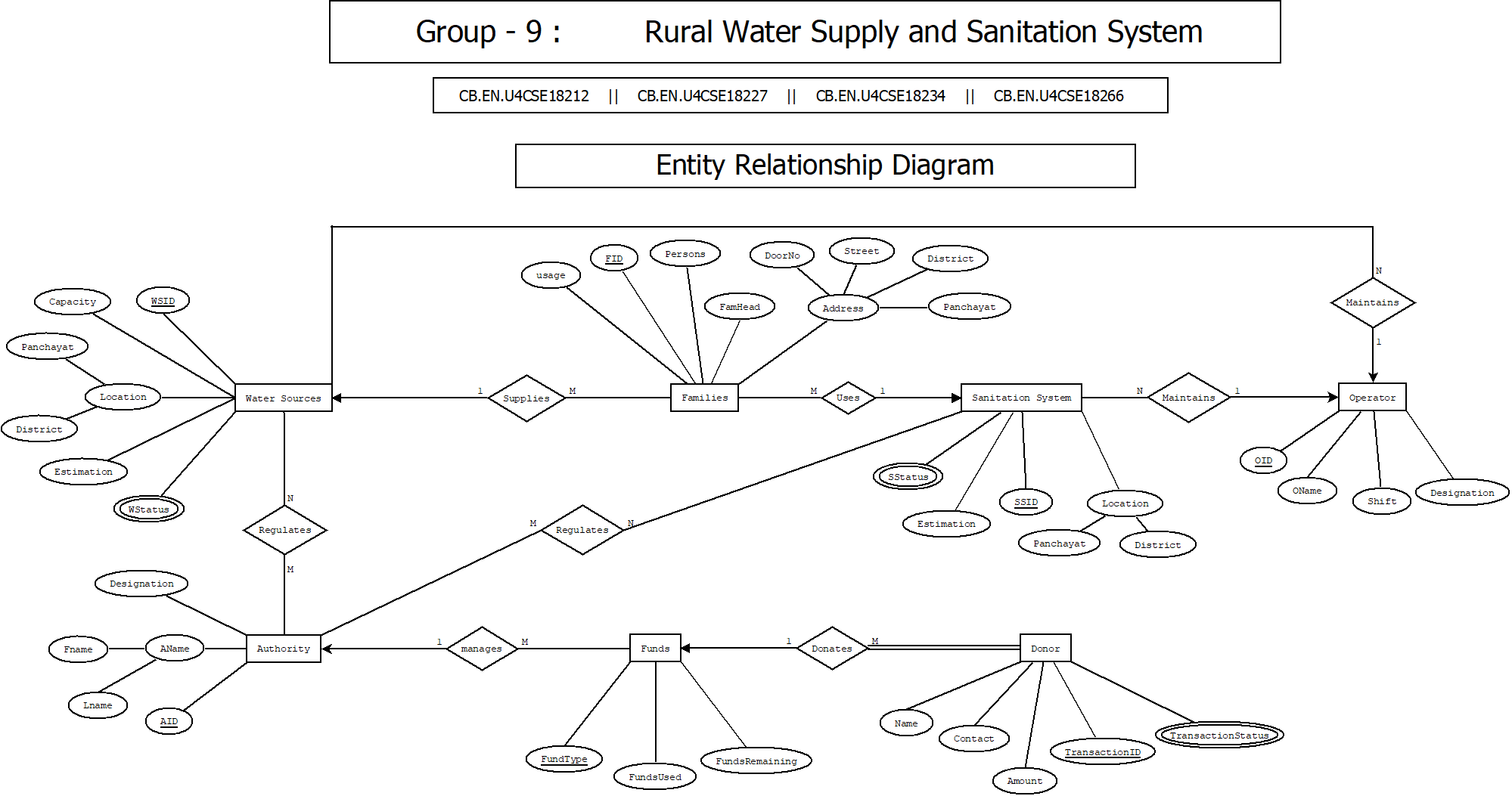
Front end – ReactJs

# Abstract:

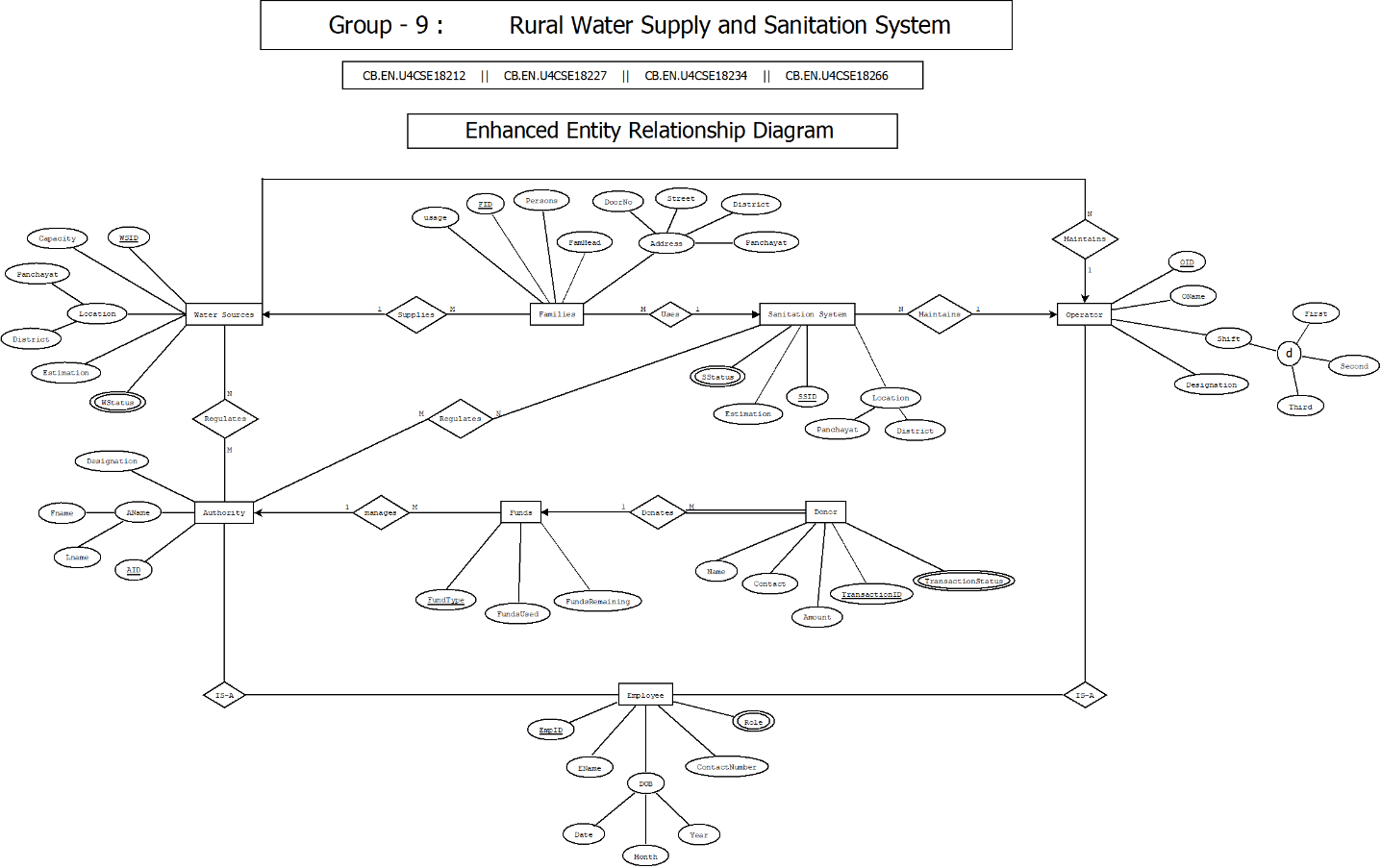
Starting with the first component, water for the village is considered as a single resource and is distributed equally among the villagers in appropriate time intervals. We also have a reserve to be used in the case of an emergency. The second component targets on the sanitary investments like constructing safe wastewater and excreta disposal systems, public installations of latrines on a pilot basis, small sewer collectors, community septic tanks, and lagoon-type wastewater treatment plant. The third component focuses on educating the villagers by connecting with them in a way they understand using community mobilization. The last and important component focuses on using the latest and advanced technology to monitor the whole system in real-time and evaluate the efficiency of both the villagers and the system in frequent time intervals. This whole data is stored in a relational database and can be used anytime to keep track of the progress of the project and gauge the success of this project especially in the area of funding.

Eventually, this project will ensure that the villagers are benefitted by the implementation of a proper water resource management system along with a hygienic sanitation system and are educated regarding the usage of the same. The monitoring system built will be used to keep track of the efficiency of the system and the project as a whole.

# Chapter 2 Logical Database Design ER Diagram



# Enhanced ER Diagram:



# Entities:

1. Water Source
2. Sanitation System
3. Employee
4. Authorities
5. Operator
6. Families
7. Funds
8. Donor

# Attributes:

1. Water Sources

* WSID – Primary Key
* Location
* Capacity
* Estimation
* WStatus

2. Sanitation System

* SSID – Primary Key
* Location
* Estimation
* SStatus

3. Employee

* EmpID – Primary Key
* EName
* DOB
* ContactNumber
* Role

4. Authority

* EmpID – Primary Key
* EName
* DOB
* ContactNumber
* Role

5. Operator

* OID – Primary Key
* OName
* Shift
* Designation

6. Families

* FID – Primary Key
* Persons
* FamHead
* Address
* Usage

7. Funds

* FundType – Primary Key
* FundsUsed
* FundsRemaining

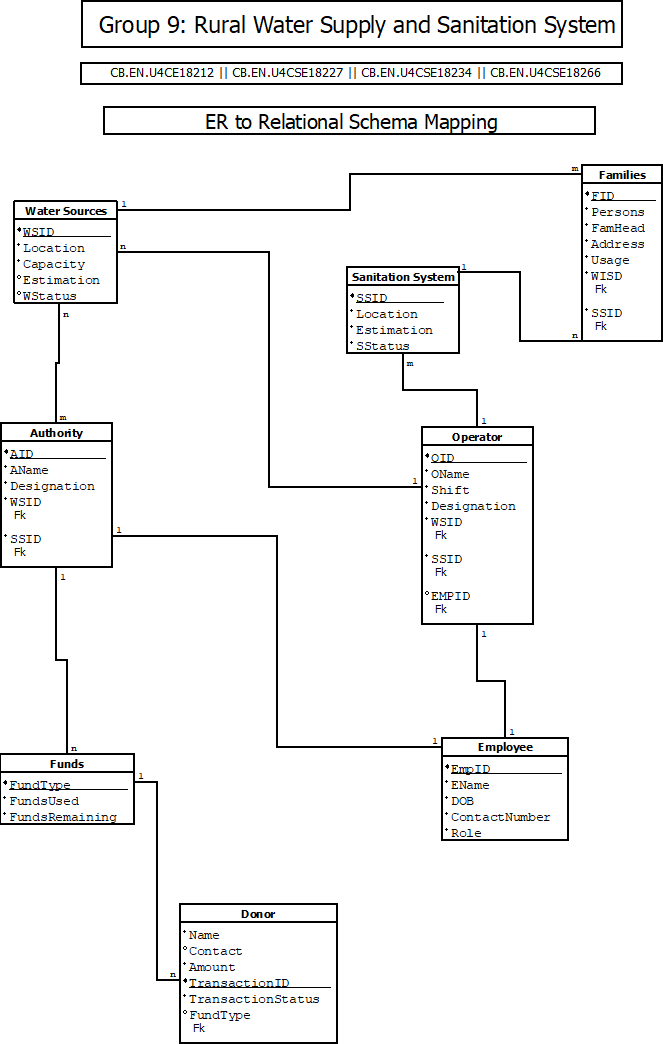
8. Donor

* TransactionID – Primary Key
* Name
* Contact
* TransactionStatus

# Relationships:

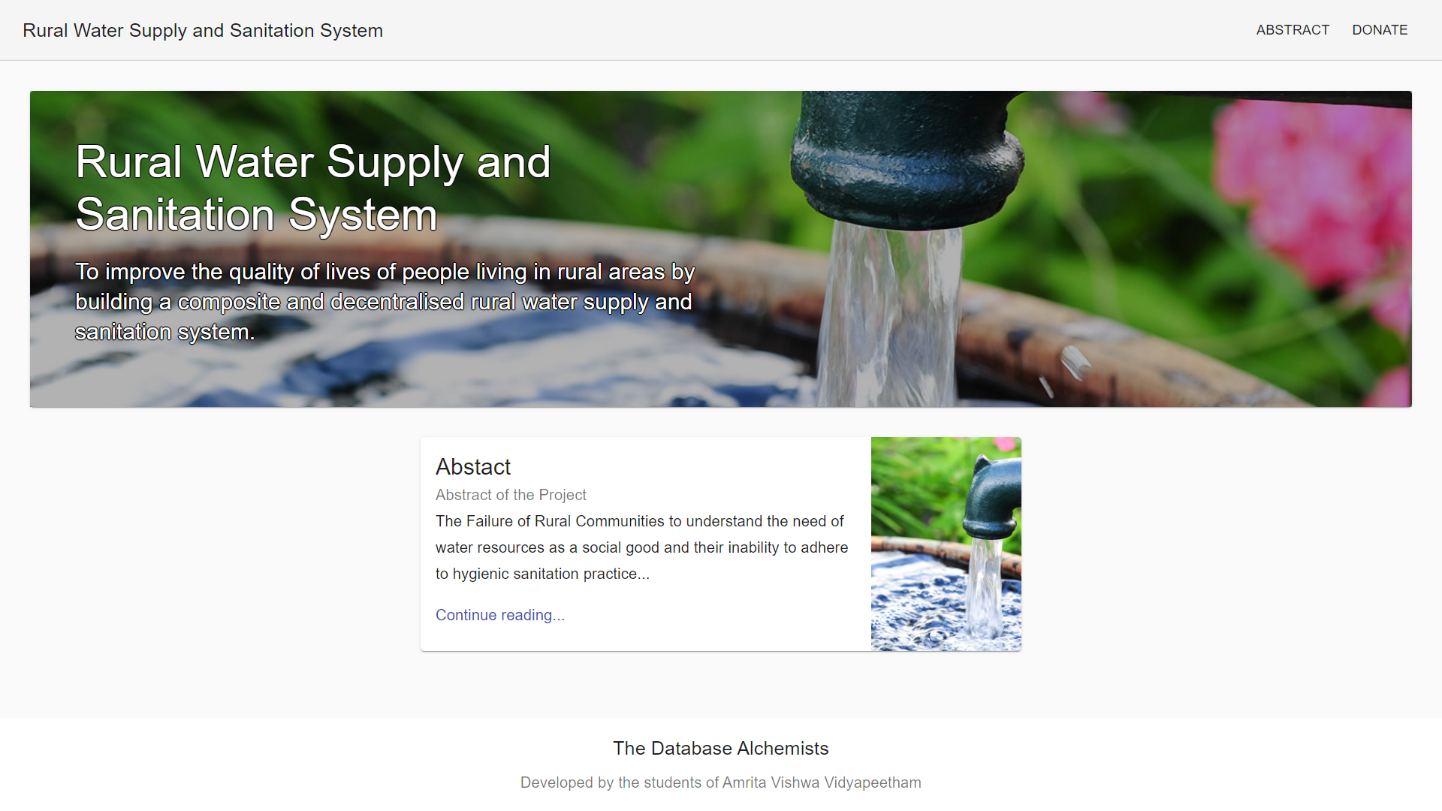
* Water Sources SUPPLIES Families
* Authority REGULATES Water Sources
* Authority REGULATES Sanitation Systems
* Authority MANAGES Funds
* Authority IS-A Employee
* Operator IS-A Employee
* Operator MAINTAINS Water Sources
* Operator MAINTAINS Sanitation System
* Families USES Sanitation System
* Donor DONATES Funds

# Chapter 3 ER to Relational Schema Mapping

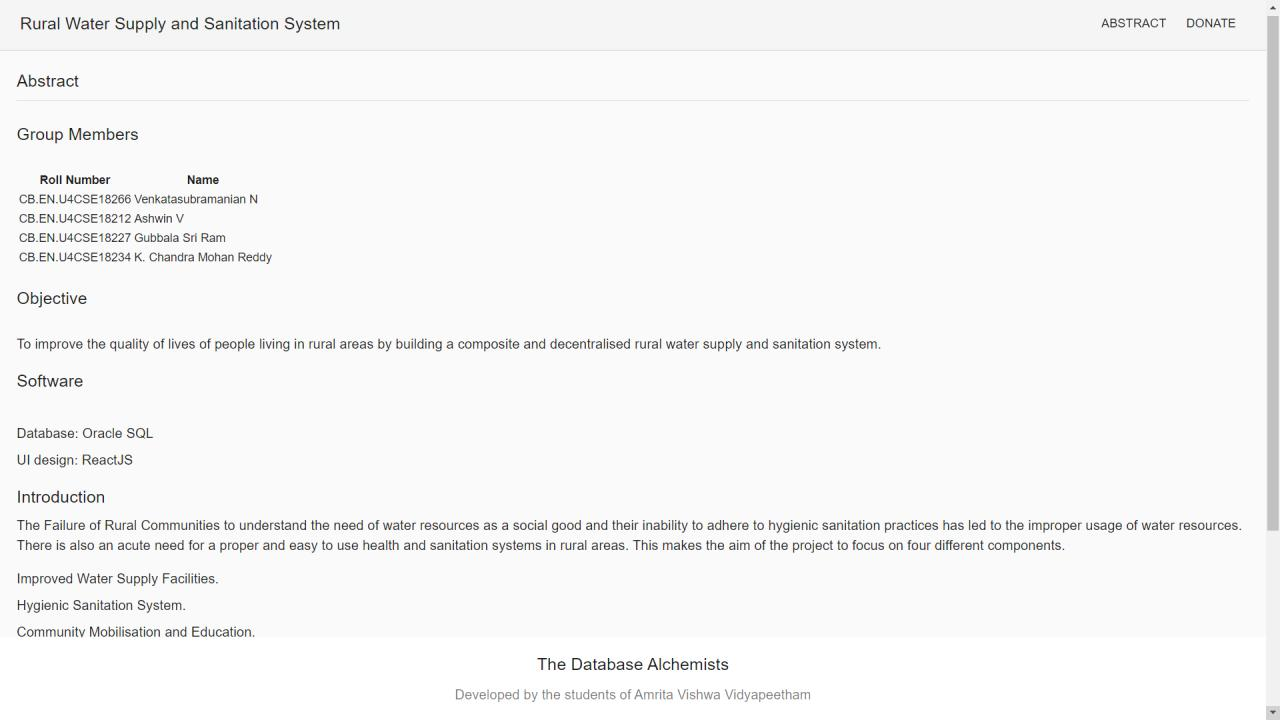


# Chapter 4 User Interface Screens

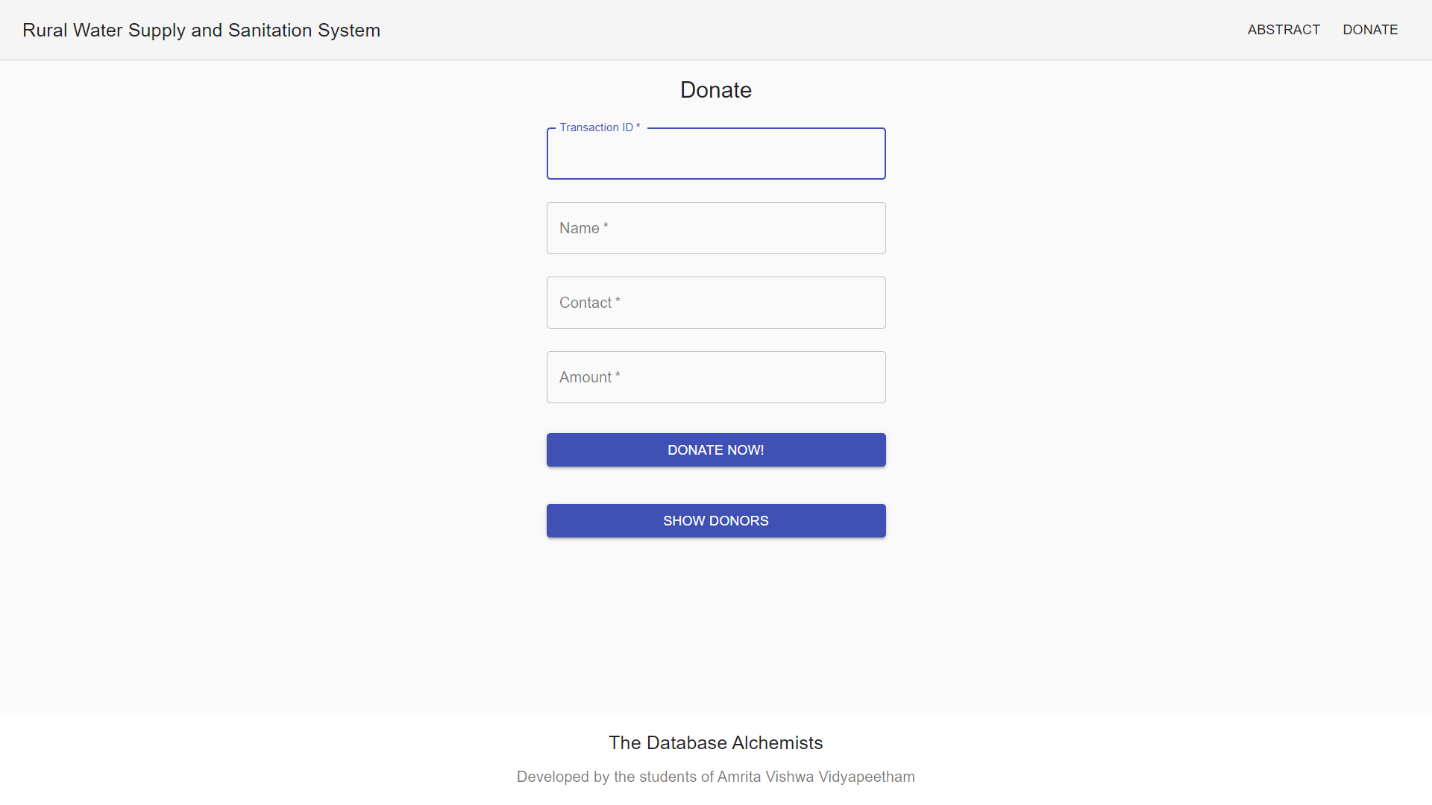
## Home Screen



## Abstract Page



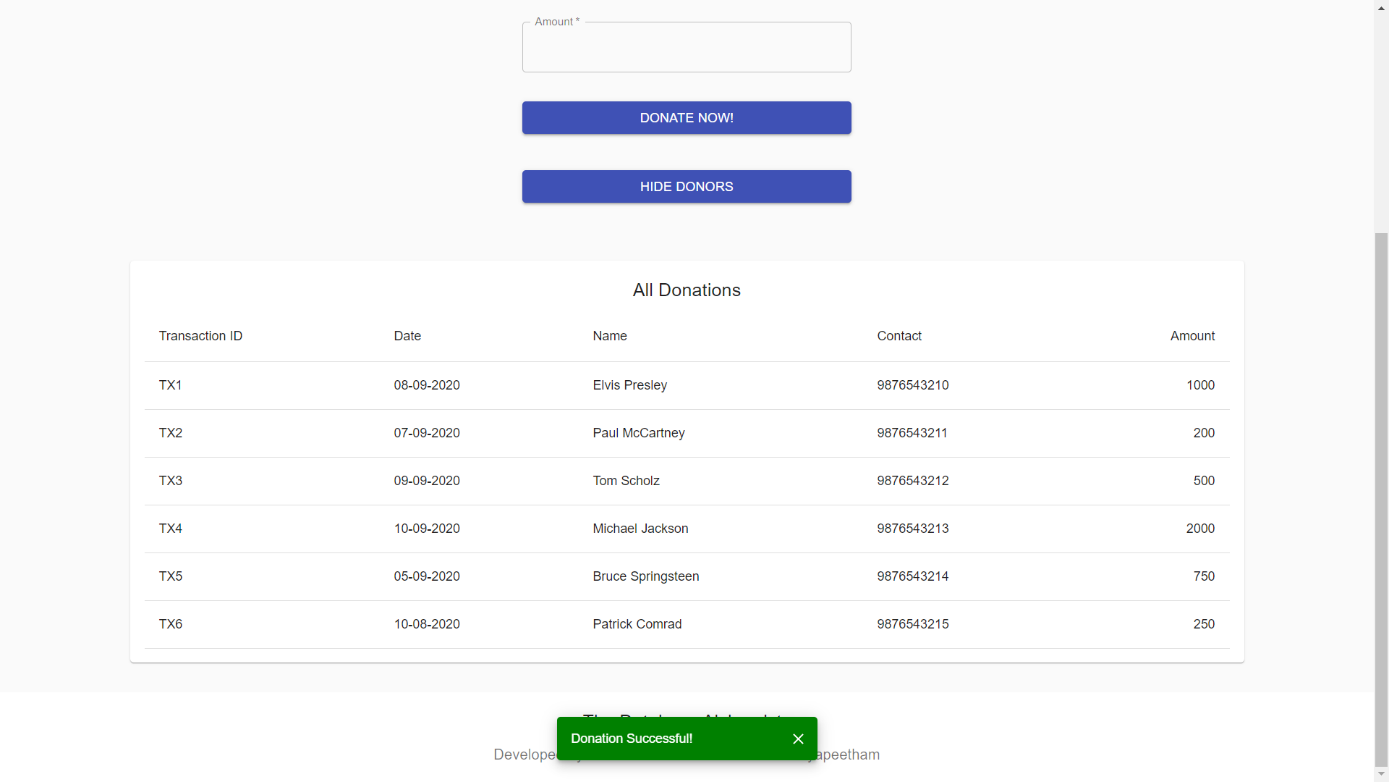
## Donate Form – Blank



## Donate Form – Filled



## Donor Table



Chapter 4 Normalization

**Write out the relation including all attribute names. Provide at least 3-5 records as sample data for the relation. Write down all Functional Dependencies. and draw dependency diagram.**

## Initial Water\_Supply\_and\_Sanitation Table:

**Water\_Supply\_and\_Sanitation**(Pincode, Panchayat, District, EmpID, FName, LName, EContact, JobCode, Designation, Shift, WSID, WStatus, WCapacity, WEstimation, Month, Year, Usage, SSID, SStatus, SEstimation, FID, Persons, FHead, Consumption, FContact, TransactionID, AccountNumber, Amount, DDate, ExpenseID, EDate, EAmount)

## Functional Dependencies:

Pincode  Panchayat, District

EmpID  EContact, DOB, LName, FName, JobCode

JobCode  Designation, Shift

WSID  WEstimation, WCapacity, WStatus, Pincode

WSID, Month, Year  Usage

SSID  SEstimation, SStatus, Pincode

FID  FContact, Consumption, FHead, Persons, Pincode

TransactionID  Amount, AccountNumber, DDate

ExpenseID  EDate, EAmount, EMPID, WSID, SSID

## 

A. Find FD closure and attribute closure B. Determine Canonical Cover C. Determine superkeys D. Identify Anomalies

## Functional Dependency Closure:

{Pincode}+ = {Panchayat, District}

{Panchayat}+ = {Panchayat}

{District}+ = {District}

{EmpID}+ = {EmpID, FName, LName, EContact, JobCode, Pincode, Panchayat, District}

{FName}+ = {FName}

{LName}+ = {LName}

{EContact}+ = {EContact}

{JobCode}+ = {JobCode, Designation, Shift}

{Designation}+ = {Designation}

{Shift}+ = {Shift}

{WSID}+ = {WSID, WEstimation, WStatus, WCapacity, Pincode, Panchayat, District}

{WStatus}+ = {WStatus}

{WCapacity}+ = {WCapacity}

{WEstimation}+ = {WEstimation}

{Month}+ = {Month}

{Year}+ = {Year}

{Usage}+ = {Usage}

{WSID, Month, Year} + = {WSID, WEstimation, WCapacity, WStatus, Pincode, Panchayat,  District ,Month, Year, Usage}

{SSID}+ = {SSID, SEstimation, SStatus, Pincode, Panchayat, District}

{SStatus}+ = {SStatus}

{SEstimation}+ = {SEstimation}

{FID}+ = {FID, FContact, Consumption, FHead, Persons, Pincode, Panchayat, District}

{Persons}+ = {Persons}

{FHead}+ = {FHead}

{Consumption}+ = {Consumption}

{FContact}+ = {FContact}

{TransactionID}+ = {TransactionID, Amount, AccountNumber, DDate, DContact}

{AccountNumber}+ = {AccountNumber}

{Amount}+ = {Amount}

{DDate}+ = {DDate}

{DContact}+ = {DContact}

{ExpenseID}+ = {ExpenseID, EDate, EAmount}

{EDate}+ = {EDate}

{EAmount}+ = {EAmount}

## Attribute Closure:

{Pincode}+ = {Pincode, Panchayat, District}

{JobCode}+ = {Designation, Shift}

{EmpID}+ = {EmpID, EContact, LName, FName, Pincode, Panchayat, District}

{WSID}+ = {WSID, WEstimation, Capacity, WStatus, Pincode, Panchayat, District}

{WSID, Month, Year} + = {WSID, WEstimation, WCapacity, WStatus, Pincode, Panchayat,  District , Month, Year, Usage}

{SSID}+ = {SSID, SEstimation, SStatus, Pincode, Panchayat, District}

{FID}+ = {FID, FContact, Consumption, FHead, Persons, Pincode, Panchayat, District}

{TransactionID}+ =  {TransactionID , Amount, AccountNumber, DDate, DContact}

{ExpenseID}+ = {ExpenseID,  EDate, EAmount, EMPID, WSID, SSID}

## Canonical Cover:

EmpID  FName

EmpID  LName

EmpID  EContact

EmpID  JobCode

EmpID  Pincode

JobCode  Designation

JobCode  Shift

WSID  WStatus

WSID  WCapacity

WSID  WEstimation

WSID  Pincode

SSID  SStatus

SSID  SEstimation

SSID  Pincode

FID  Persons

FID  FHead

FID  Consumption

FID  FContact

FID  Pincode

WSID Month Year  Usage

ExpenseID  EDate

ExpenseID  EAmount

ExpenseID  WSID

ExpenseID  EmpID

ExpenseID  SSID

ExpenseID  DContact

TransactionID  AccountNumber

TransactionID  Amount

TransactionID  DDate

Pincode  Panchayat

Pincode  District

## Superkeys:

EmpID, WSID, SSID, Pincode, Month, Year, ExpenseID, TransactionID

## Anomalies:

In Employee Table, Shift depends on Designation which is not a primary key.

This brought in partial dependency which created creation, updation and deletion anomalies.

We resolved this by decomposing those attributes to a new Jobs Table which acted as a foreign key in the Employee Table.

# To check if the relation is in First normal form

* Each table should have a primary key.
* The values in each column of a table should be atomic.
* There should not be any repeating groups

## Conditions satisfied and violated by the relation Water\_Supply\_And\_Sanitation\_System:

1. Composite Primary Key is present.
2. There are non-atomic values present in the relation.
   1. There can be more than one EmpID, WSID and SSID mapped to one Pincode.
   2. There is a many to many relationship present here.
3. There are no repeating groups.

**Solution:**

To reduce the table into 1NF, we make the values atomic by splitting them into different tuples in the same relations.

# Identify the Partial dependencies and Decompose the table and check if it is 2NF

## Conditions for a relation to be in Second Normal Form:

* The relation must be in 1NF
* All non-key attributes are fully functionally dependent on the primary key.

## Dependencies:

Pincode  Panchayat, District

EmpID  EContact, DOB, LName, FName, JobCode

JobCode Designation, Shift

WSID  WEstimation, WCapacity, WStatus, Pincode

WSID, Month, Year Usage

SSID  SEstimation, SStatus, Pincode

FID  FContact, Consumption, FHead, Persons, Pincode

TransactionID  Amount, AccountNumber, DDate

ExpenseID  EDate, EAmount, EMPID, WSID, SSID

* **The partial dependencies have been normalized by decomposing the relations into the following relations.**

Location - (**Pincode**, WSID, SSID, EmpID, FID, Panchayat, District)

Employee - (**EmpID**, FName, LName, EContact, JobCode, Pincode, Designation, Shift)

Water Sources - (**WSID**, WStatus, WCapacity, WEstimation, Pincode )

WaterUsage - (**WSID, Month, Year**, Usage)

Sanitation Systems - (**SSID**, SStatus, SEstimation, Pincode )

Families - (**FID**, Persons, FHead, Consumption, FContact, Pincode)

Donation - ( **TransactionID**, AccountNumber, Amount, DDate )

Expenditure - ( **ExpenseID**, EDate, EmpID, WSID, SSID, EAmount )

# Check Transitive dependencies and Decompose the table and check if it is 3NF

## Conditions for Third normal form:

* The relation must be in 2NF
* No non-key attribute should be transitively dependent on the primary key.

EmpID  EContact, DOB, LName, FName, JobCode

JobCode  Designation, Shift

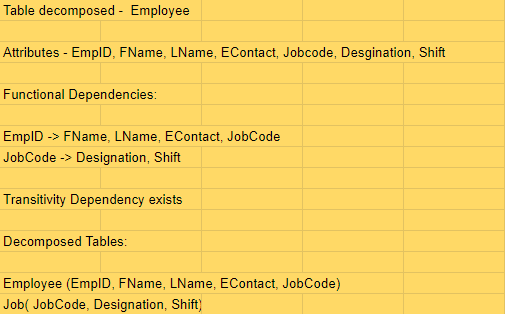
* **The transitive dependencies have been normalized by decomposing the relations into the following relations**

Employee - (**EmpID**, FName, LName, EContact, JobCode, Pincode)

Job - ( **JobCode**, Designation, Shift )

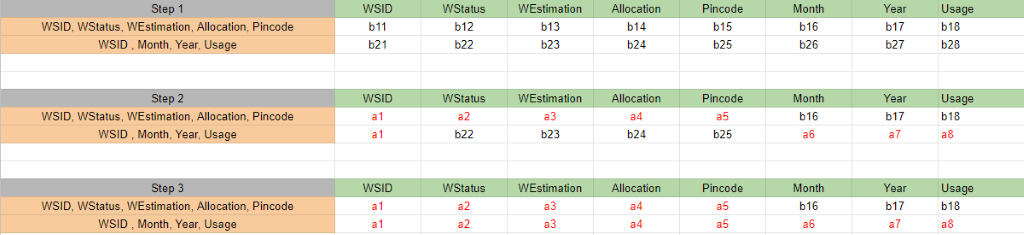
# Check if it is Lossless decomposition using chase method and Check if it is Dependency preserving

By chase method, we have found that the decompositions are lossless.









## Dependency preserving:

All the below listed dependencies have can be found in the decomposed tables as shown in the functional dependency diagram. Hence dependency is preserved.

Dependencies:

Pincode  Panchayat, District

EmpID  EContact, DOB, LName, FName, JobCode

JobCode Designation, Shift

WSID  WEstimation, WCapacity, WStatus, Pincode

WSID, Month, Year Usage

SSID  SEstimation, SStatus, Pincode

FID  FContact, Consumption, FHead, Persons, Pincode

TransactionID  Amount, AccountNumber, DDate

ExpenseID  EDate, EAmount, EMPID, WSID, SSID

## Normalize to BCNF:

All the relations after being normalized to 3NF are present in BCNF as there are no non-key attribute determining is determining a key attribute.

# Final schema with Primary keys and dependency diagram.

Location - (**Pincode**, WSID, SSID, EmpID, FID, Panchayat, District)

Employee - (**EmpID**, FName, LName, EContact, JobCode, Pincode)

Job - ( **JobCode**, Designation, Shift )

Water Sources - (**WSID**, WStatus, WCapacity, WEstimation, Pincode )

WaterUsage - ( **WSID, Month, Year**, Usage)

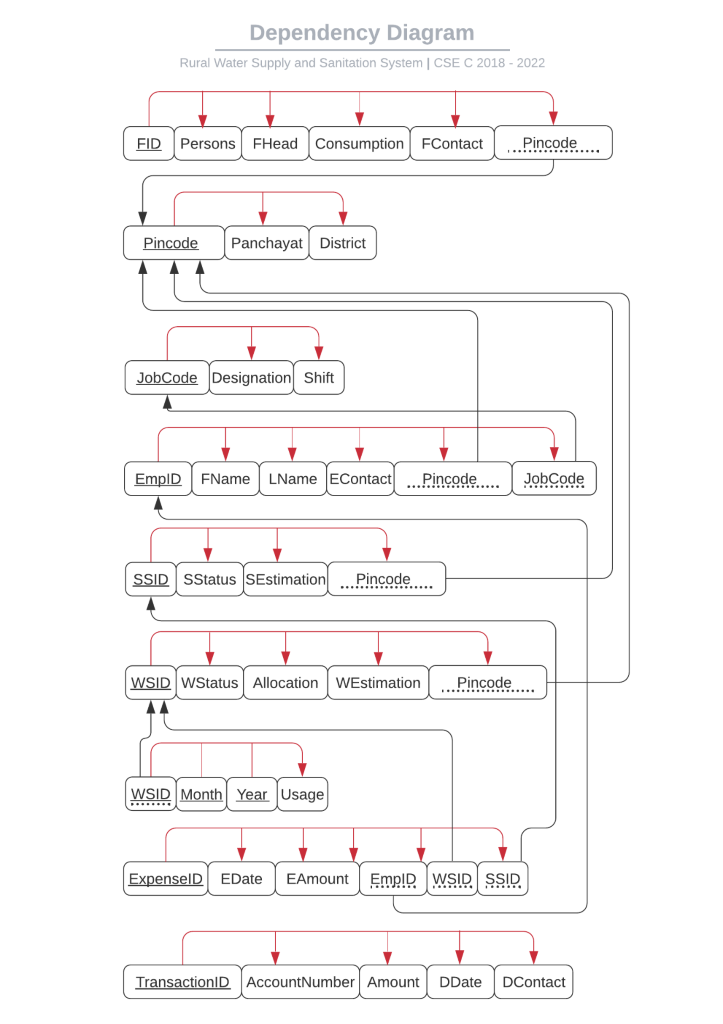
Sanitation Systems - (**SSID**, SStatus, SEstimation, Pincode )

Families - ( **FID**, Persons, FHead, Consumption, FContact, Pincode)

Donation - ( **TransactionID**, AccountNumber, Amount, DDate )

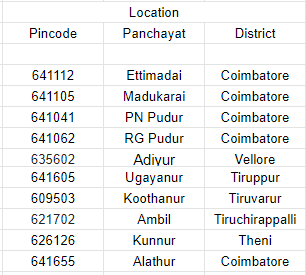
Expenditure - ( **ExpenseID**, EDate, EmpID, WSID, SSID, EAmount )

# Dependency Diagram

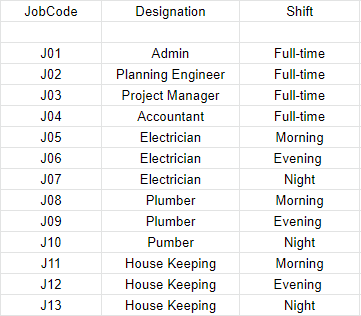


**Relations:**

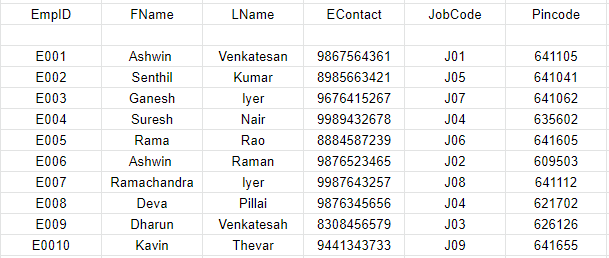
## Location



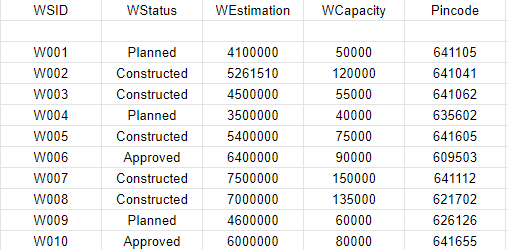
## Jobs



## Employee



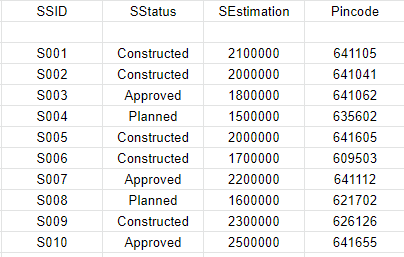
## WaterSources



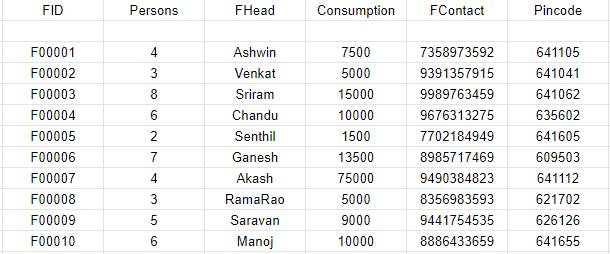
## WaterUsage



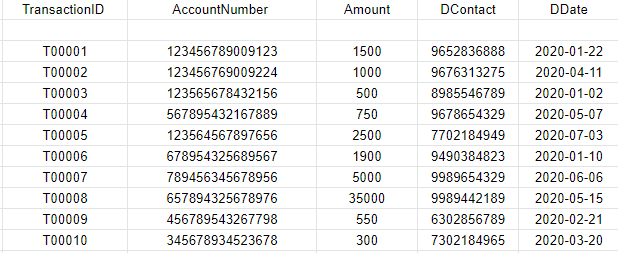
## SanitationSystems



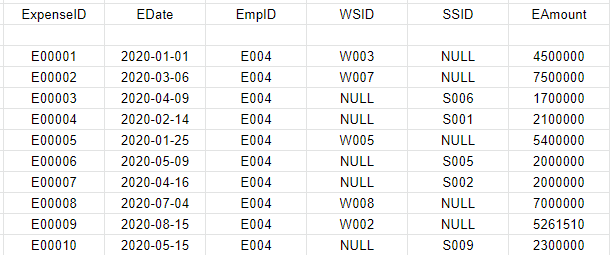
## Families



## Donations



## Expenditure



Chapter 5 Creation of Tables

# A. Include the create command for ALL THE TABLES, sample insert commands B. Minimum 10 meaningful records to be inserted C. Data from all the tables

**Location Table:**

create table Location

(

Pincode number(6) not null,

Panchayat varchar2(15),

District varchar2(20),

constraint pk\_Pincode primary key (Pincode)

)

insert into Location values(641112,'Ettimadai','Coimbatore');

insert into Location values(641105,'Madukarai','Coimbatore');

insert into Location values(641041,'PN Pudur','Coimbatore');

insert into Location values(641062,'RG Pudur','Coimbatore');

insert into Location values(635602,'Adiyur','Vellore');

insert into Location values(641605,'Ugayanur','Tiruppur');

insert into Location values(609503,'Koothanur','Tiruppur');

insert into Location values(621702,'Ambil','Tiruchirappalli');

insert into Location values(626126,'Kunnur','Theni');

insert into Location values(641655,'Alathur','Coimbatore');

**Job** **Table:**

create table Job

(

JobCode varchar2(3) not null,

Designation varchar2(20),

Shift varchar2(15),

constraint pk\_JobCode primary key (JobCode)

)

insert into Job values('J01','Admin','Full-time');

insert into Job values('J02','Planning Engineer','Full-time');

insert into Job values('J03','Project Manager','Full-time');

insert into Job values('J04','Accountant','Full-time');

insert into Job values('J05','Electrician','Morning');

insert into Job values('J06','Electrician','Evening');

insert into Job values('J07','Electrician','Night');

insert into Job values('J08','Plumber','Morning');

insert into Job values('J09','Plumber','Evening');

insert into Job values('J10','Pumber','Night');

insert into Job values('J11','House Keeping','Morning');

insert into Job values('J12','House Keeping','Evening');

insert into Job values('J13','House Keeping','Night');

**Employee Table:**

create table Employee

(

EmpID varchar2(4) not null,

FName varchar2(20),

LName varchar2(20),

EContact number(10),

JobCode varchar2(3),

Pincode number(6),

constraint pk\_EmpID primary key (EmpID),

constraint fk\_JobCode FOREIGN KEY(JobCode) REFERENCES Job(JobCode),

constraint fk\_Pincode FOREIGN KEY(Pincode) REFERENCES Location(Pincode)

)

insert into Employee values('E001','Ashwin','Venkatesan',9867564361,'J01',641105);

insert into Employee values('E002','Senthi','kumar',8985663421,'J05',641041);

insert into Employee values('E003','Ganesh','Iyer',9676415267,'J07',641062);

insert into Employee values('E004','Suresh','Nair',9989432678,'J04',635602);

insert into Employee values('E005','Rama','Rao',8884587239,'J06',641605);

insert into Employee values('E006','Ashwin','Raman',9876523465,'J02',609503);

insert into Employee values('E007','Ramachandra','Iyer',9987643257,'J08',641112);

insert into Employee values('E008','Deva','Pillai',9876345656,'J04',621702);

insert into Employee values('E009','Dharun','Venkatesah',6308456579,'J03',626126);

insert into Employee values('E0010','Kavin','Thevar',9441343733,'J09',641655);

**WaterSource** **Table:**

create table WaterSource

(

WSID varchar2(4) not null,

WStatus varchar2(15),

WEstimation number(7),

WCapacity number(6),

Pincode number(6),

constraint pk\_WSID primary key (WSID),

constraint fk\_WPincode FOREIGN KEY(Pincode) REFERENCES Location(Pincode)

)

insert into WaterSource values('W002','Constructed',5261510,120000,641041);

insert into WaterSource values('W007','Constructed',7500000,150000,641112);

insert into WaterSource values('W008','Constructed',7000000,135000,621702);

**WaterUsage Table:**

create table WaterUsage

(

WSID varchar2(4) not null,

MONTH varchar2(10) not null,

YEAR number(4) not null,

USAGE number(6),

constraint pk\_WSID\_MONTH\_YEAR primary key (WSID, MONTH, YEAR),

constraint fk\_WSID FOREIGN KEY(WSID) REFERENCES WaterSource(WSID)

)

insert into WaterUsage values('W002','January',2020,20000);

insert into WaterUsage values('W007','March',2020,30000);

insert into WaterUsage values('W008','November',2020,25000);

insert into WaterUsage values('W002','February',2020,9000);

insert into WaterUsage values('W008','January',2020,30000);

insert into WaterUsage values('W002','April',2020,25000);

insert into WaterUsage values('W007','February',2020,17000);

insert into WaterUsage values('W007','September',2020,22000);

insert into WaterUsage values('W008','June',2020,15000);

insert into WaterUsage values('W002','July',2020,21000);

**SanitationSystems** **Table:**

create table SanitationSystems

(

SSID varchar2(4) not null,

SStatus varchar2(15),

SEstimation number(10),

Pincode number(6),

constraint pk\_SSID primary key (SSID),

constraint fk\_SPincode FOREIGN KEY(Pincode) REFERENCES Location(Pincode)

)

insert into SanitationSystems values('S001','Constructed',2100000,641105);

insert into SanitationSystems values('S005','Constructed',2000000,641605);

insert into SanitationSystems values('S006','Constructed',1700000,609503);

**Families** **Table:**

create table Families

(

FID varchar2(6) not null,

Persons number(2),

FHead varchar2(10),

Consumption number(5),

FContact number(10),

Pincode number(6),

constraint pk\_FID primary key (FID),

constraint fk\_FPincode FOREIGN KEY(Pincode) REFERENCES Location(Pincode)

)

insert into Families values('F00001',4,'Ashwin',7500,7358973592,641105);

insert into Families values('F00002',3,'Venkat',5000,9391357915,641041);

insert into Families values('F00003',8,'Sriram',15000,9989763459,641062);

insert into Families values('F00004',6,'Chandu',10000,9676313275,635602);

insert into Families values('F00005',2,'Senthil',1500,7702184949,641605);

insert into Families values('F00006',7,'Ganesh',13500,8985717469,609503);

insert into Families values('F00007',4,'Akash',75000,9490384823,641112);

insert into Families values('F00008',3,'RamaRao',5000,8356983593,621702);

insert into Families values('F00009',5,'Saravan',9000,9441754535,626126);

insert into Families values('F00010',6,'Manoj',10000,8886433659,641655);

**Donations** **Table:**

create table Donations

(

TransactionID varchar2(6) not null,

AccountNumber number(15),

Amount number(10,2),

DContact number(10),

DDate varchar2(25),

constraint pk\_TransactionID primary key (TransactionID)

)

insert into Donations values('T00001',123456789009123,1500,9652836888,'2020-01-22');

insert into Donations values('T00002',123456769009224,1000,9676313275,'2020-04-11');

insert into Donations values('T00003',123565678432156,500,8985546789,'2020-01-02');

insert into Donations values('T00004',567895432167889,750,9678654329,'2020-05-07');

insert into Donations values('T00005',123564567897656,2500,7702184949,'2020-07-03');

insert into Donations values('T00006',678954325689567,1900,9490384823,'2020-01-10');

insert into Donations values('T00007',789456345678956,5000,9989654329,'2020-06-06');

insert into Donations values('T00008',657894325678976,35000,9989442189,'2020-05-15');

insert into Donations values('T00009',456789543267798,550,6302856789,'2020-02-21');

insert into Donations values('T00010',345678934523678,300,7302184965,'2020-03-20');

**Expenditure Table:**

create table Expenditure

(

ExpenseID varchar2(6) not null,

EDate varchar2(25),

EmpID varchar2(4),

WSID varchar2(4),

SSID varchar2(4),

EAmount number(10,2),

constraint pk\_ExpenseID primary key (ExpenseID),

constraint fk\_EmpID FOREIGN KEY(EmpID) REFERENCES Employee(EmpID),

constraint fk\_EWSID FOREIGN KEY(WSID) REFERENCES WaterSource(WSID),

constraint fk\_SSID FOREIGN KEY(SSID) REFERENCES SanitationSystems(SSID)

)

insert into Expenditure values('E00001','2020-01-01','E004','W003','NULL',15000);

insert into Expenditure values('E00002','2020-03-06','E009','W007','NULL',20000);

insert into Expenditure values('E00003','2020-04-09','E005','NULL','S006',10000);

insert into Expenditure values('E00004','2020-02-14','E001','NULL','S003',25000);

insert into Expenditure values('E00005','2020-01-25','E002','W004','NULL',5000);

insert into Expenditure values('E00006','2020-05-09','E003','NULL','S005',7500);

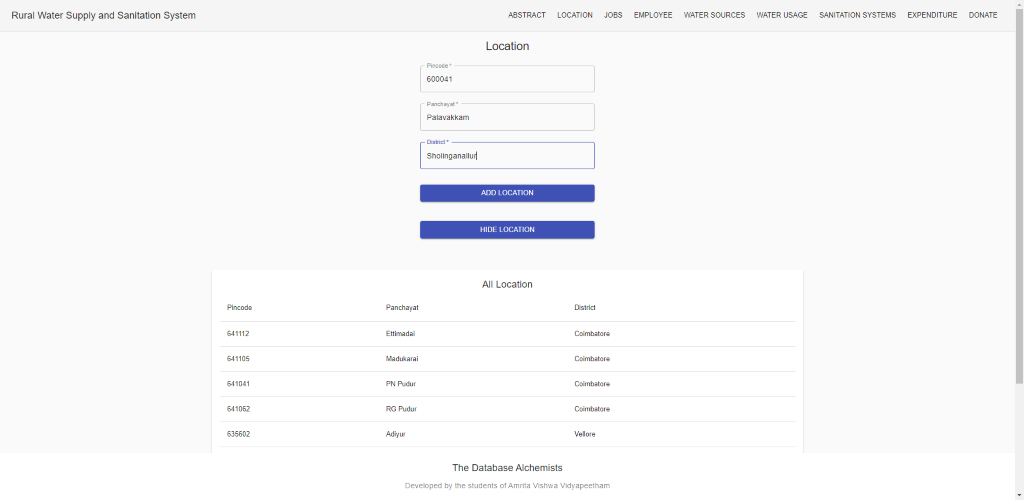
insert into Expenditure values('E00007','2020-04-16','E007','NULL','S002',12500);

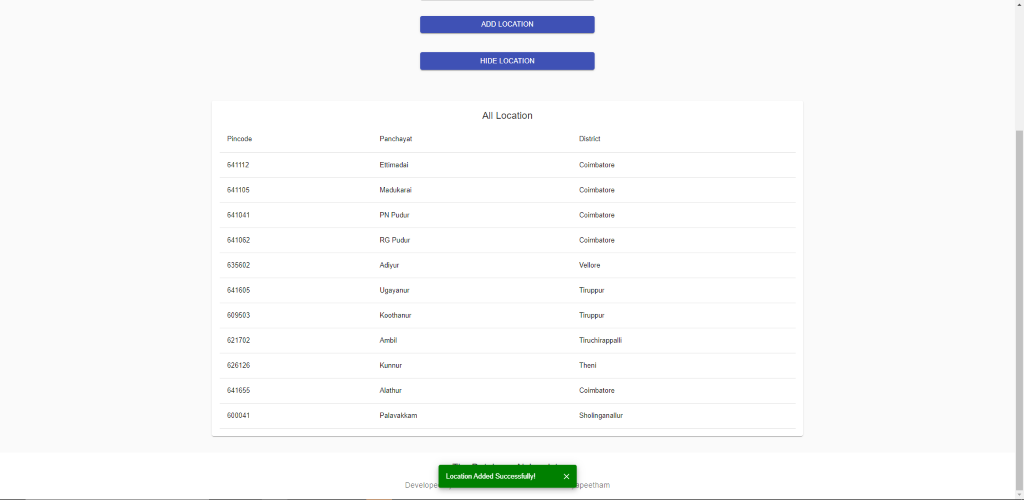
insert into Expenditure values('E00009','2020-08-15','E004','W002','NULL',15000);

insert into Expenditure values('E00010','2020-05-15','E008','NULL','S008',12000);

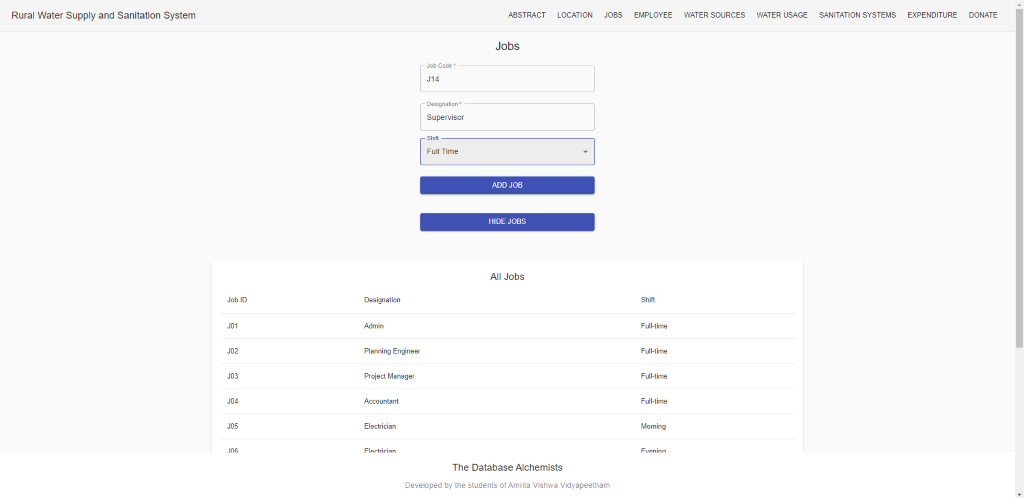
Chapter 6 User Interface Design

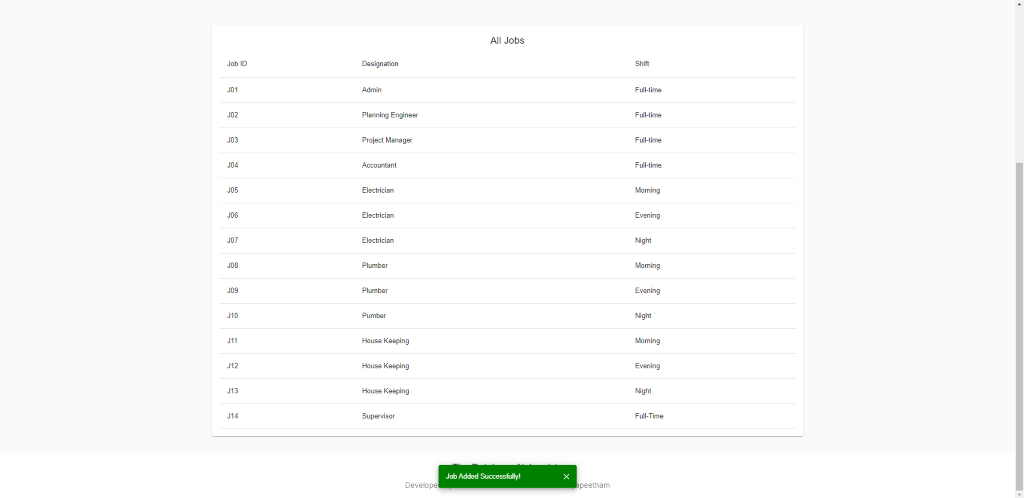
# Location Table:



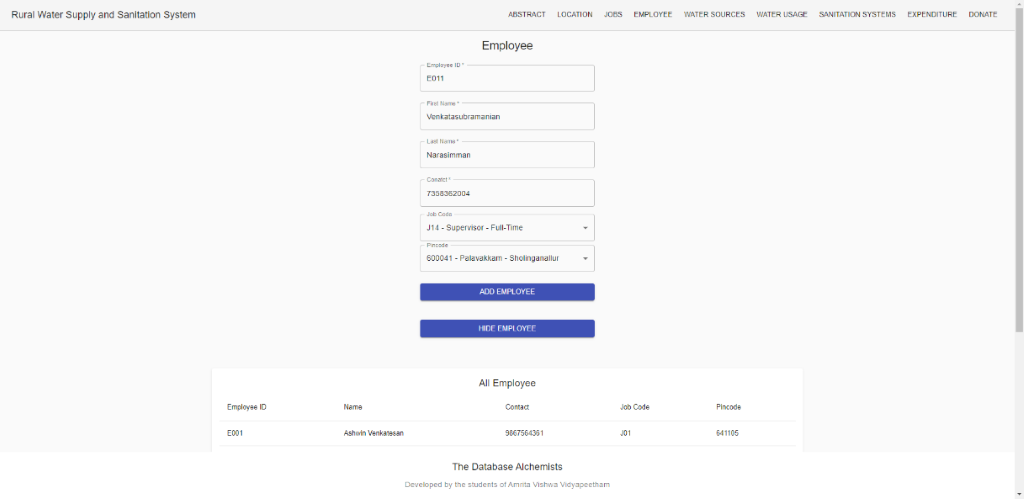


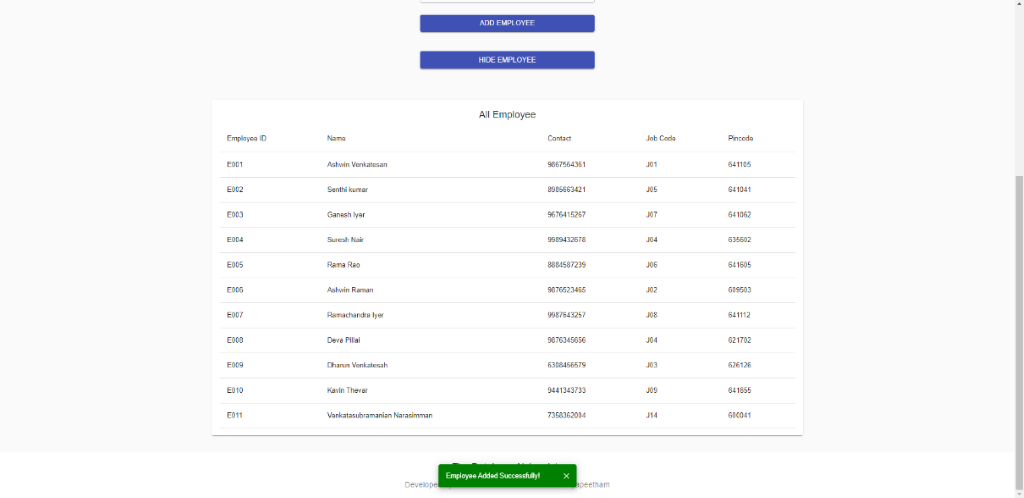
# Jobs Table:

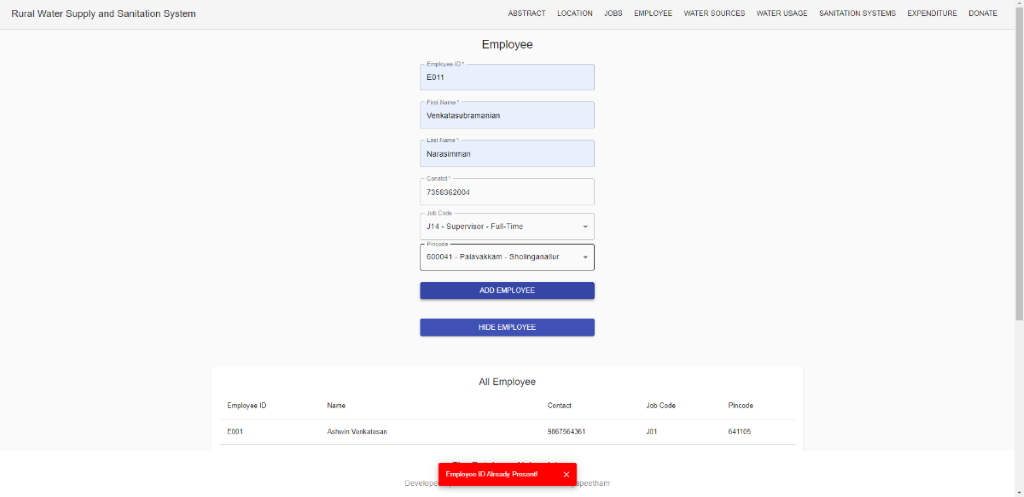




# Employee Table:

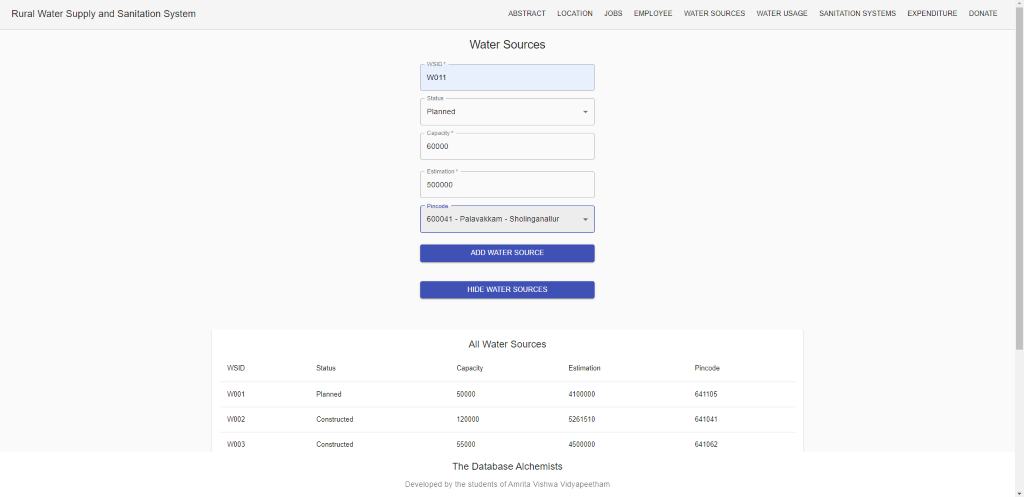


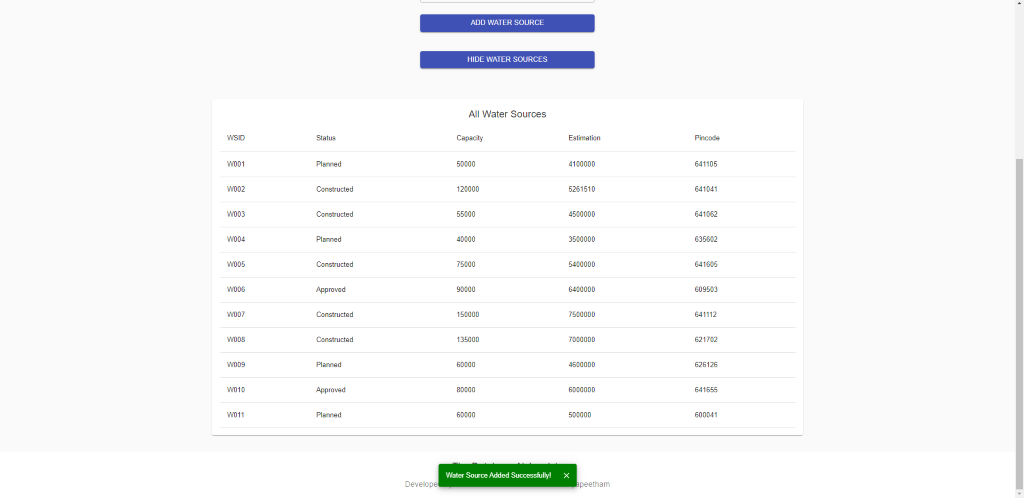




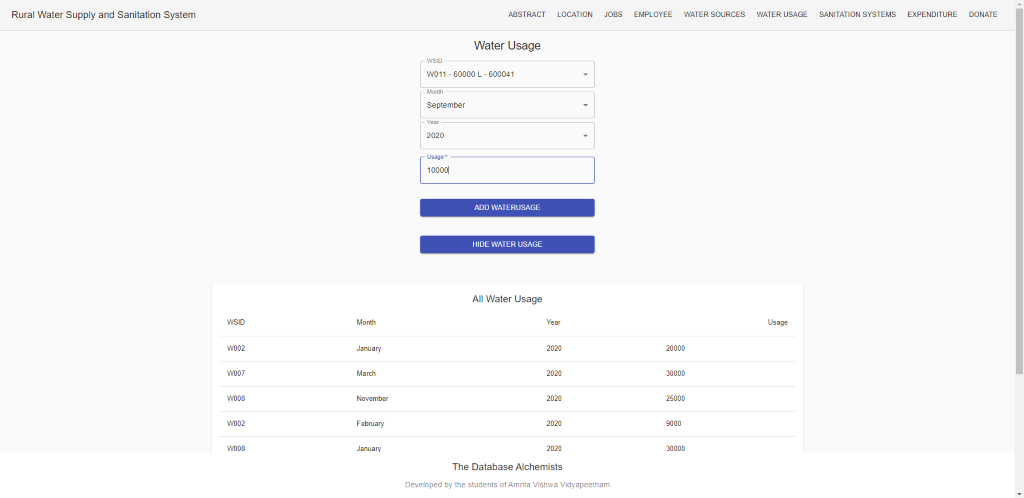
Primary Key EmpID Unique Validation!

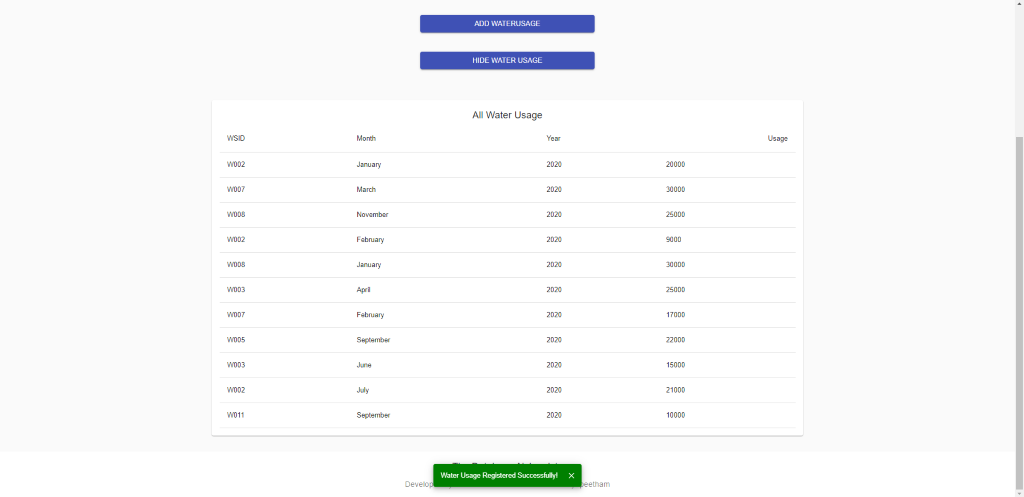
# Water Sources Table:



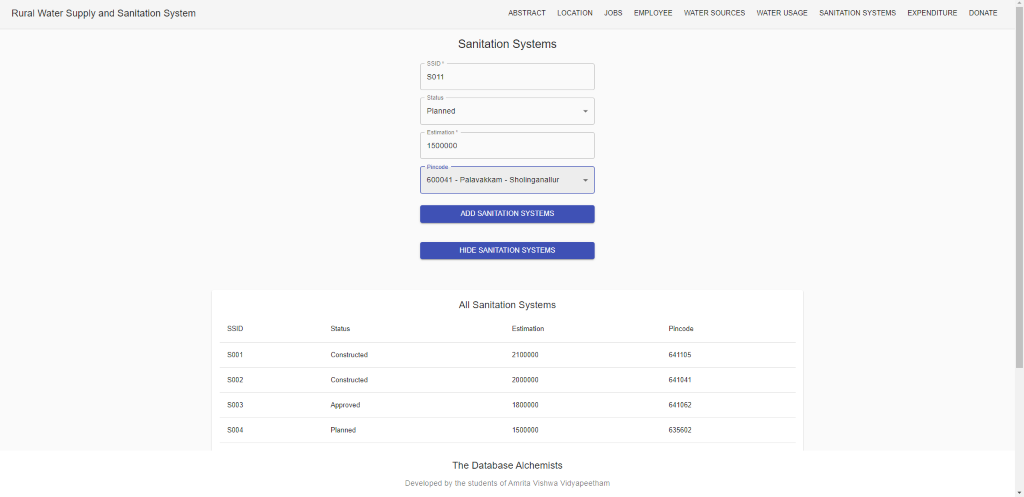


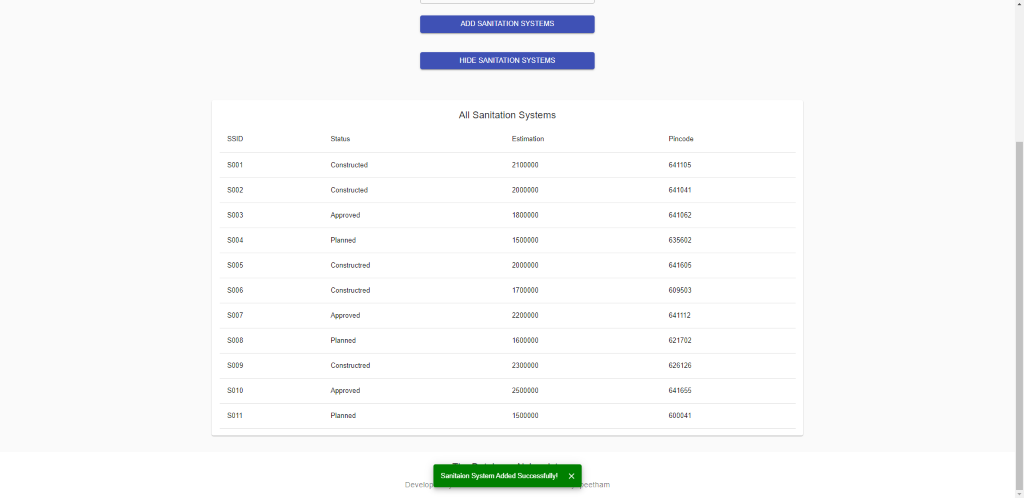
# Water Usage Table:



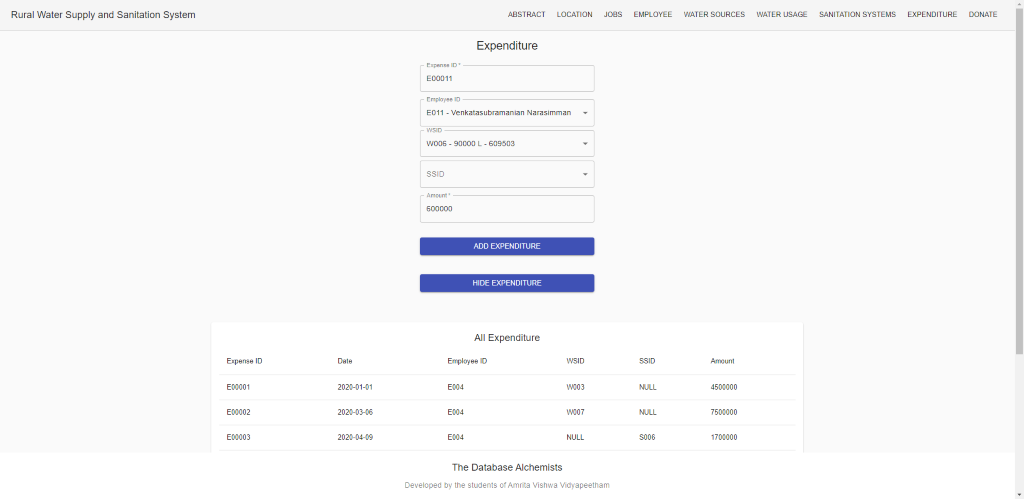


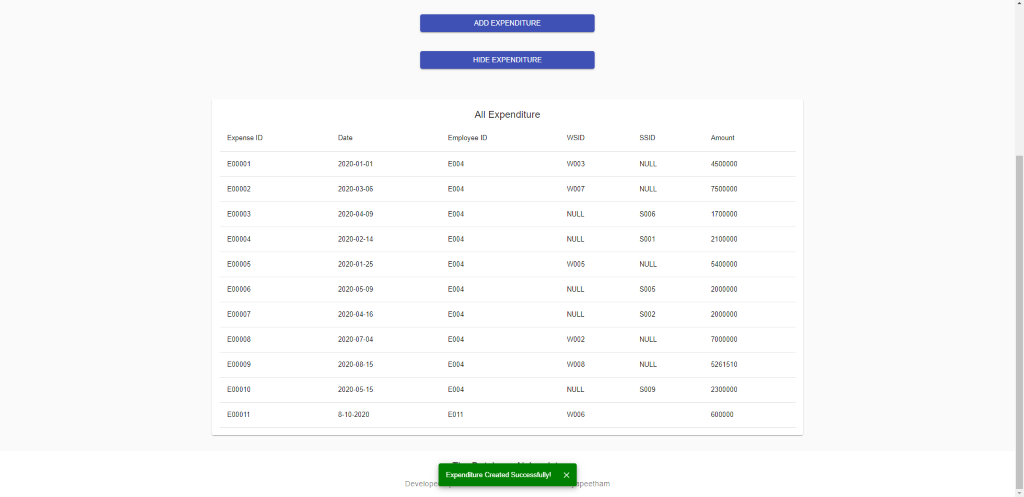
# Sanitation Systems Table:



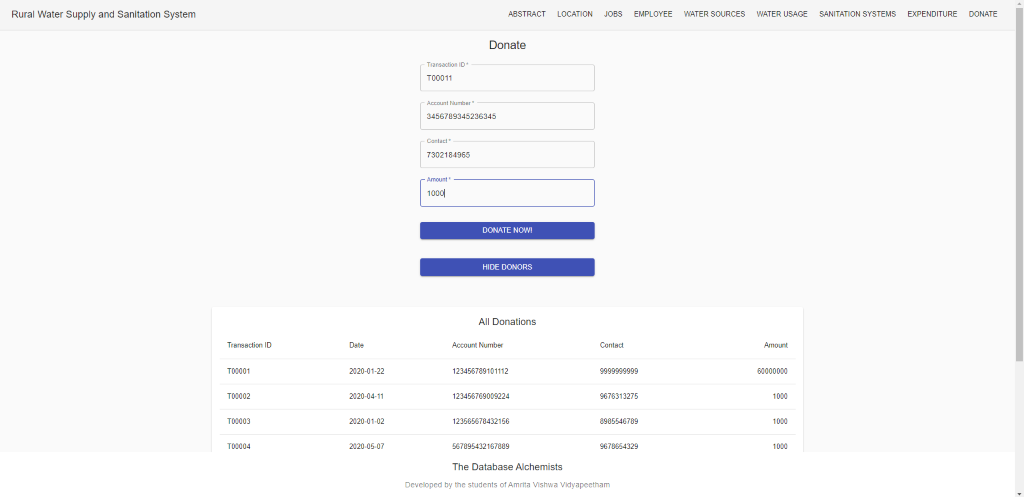


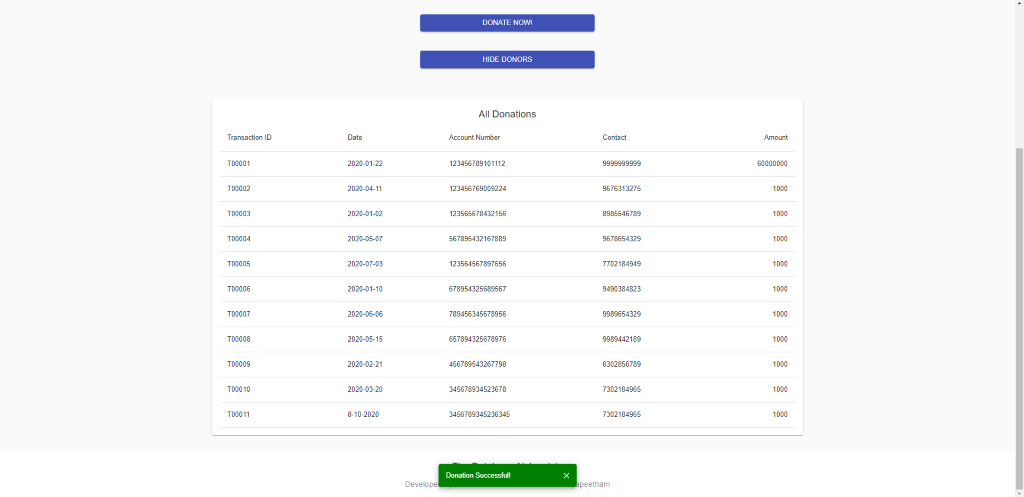
# Expenditure Table:





# Donation Table:





# Connectivity

# Connectivity Code:

const mysql = require('mysql');

const connection = mysql.createConnection({

  host: "localhost",

  user: "root",

  password: "root",

  port: 3306

});

module.exports = connection;

# Database Initialization:

const bcrypt = require("bcryptjs");

const config = require("./config");

const connection = require("./db");

/\* -------------------------------- creating settings and superusers --------------------------------- \*/

function initializer() {

    try {

        connection.connect(function (err) {

            if (err) throw err;

            console.log("Connected to DB Successfully ✅\n");

            console.log("starting to seed admins...");

            const admins = config.admins;

            connection.query(

                `CREATE DATABASE IF NOT EXISTS RuralWaterSupplyAndSanitationSystem`,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `use RuralWaterSupplyAndSanitationSystem`,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `CREATE TABLE IF NOT EXISTS Locations

                    (

                        Pincode int not null,

                        Panchayat varchar(15),

                        District varchar(20),

                        constraint pk\_Pincode primary key (Pincode)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `CREATE TABLE IF NOT EXISTS Jobs

                    (

                        JobCode int not null AUTO\_INCREMENT,

                        Designation varchar(20),

                        Shift varchar(15),

                        constraint pk\_JobCode primary key (JobCode)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `CREATE TABLE IF NOT EXISTS Employees

                    (

                        EmpID int not null AUTO\_INCREMENT,

                        FName varchar(20),

                        LName varchar(20),

                        EContact BIGINT,

                        JobCode int,

                        Pincode int,

                        Username varchar(35),

                        Password varchar(60) not null,

                        constraint pk\_EmpID primary key (EmpID),

                        constraint fk\_JobCode FOREIGN KEY(JobCode) REFERENCES Jobs(JobCode),

                        constraint fk\_Pincode FOREIGN KEY(Pincode) REFERENCES Locations(Pincode)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `CREATE TABLE IF NOT EXISTS WaterSources

                    (

                        WSID int not null AUTO\_INCREMENT,

                        WStatus varchar(20),

                        WEstimation int,

                        WCapacity int,

                        Pincode int,

                        constraint pk\_WSID primary key (WSID),

                        constraint fk\_WPincode FOREIGN KEY(Pincode) REFERENCES Locations(Pincode)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `CREATE TABLE IF NOT EXISTS WaterUsages

                    (

                        WSID int not null,

                        Month varchar(10) not null,

                        Year int not null,

                        Usages int,

                        constraint pk\_WSID\_MONTH\_YEAR primary key (WSID, Month, Year),

                        constraint fk\_WSID FOREIGN KEY(WSID) REFERENCES WaterSources(WSID)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `create table if not exists SanitationSystems

                    (

                        SSID int not null AUTO\_INCREMENT,

                        SStatus varchar(20),

                        SEstimation int,

                        Pincode int,

                        constraint pk\_SSID primary key (SSID),

                        constraint fk\_SPincode FOREIGN KEY(Pincode) REFERENCES Locations(Pincode)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `create table if not exists Families

                    (

                        FID int not null AUTO\_INCREMENT,

                        Persons int,

                        FHead varchar(10),

                        FContact BIGINT,

                        Pincode int,

                        constraint pk\_FID primary key (FID),

                        constraint fk\_FPincode FOREIGN KEY(Pincode) REFERENCES Locations(Pincode)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `create table if not exists Donations

                    (

                        TransactionID varchar(20) not null,

                        AccountNumber varchar(25),

                        Amount decimal(10,2),

                        DContact BIGINT,

                        DDate varchar(25),

                        constraint pk\_TransactionID primary key (TransactionID)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `create table if not exists Expenditures

                    (

                        ExpenseID int not null AUTO\_INCREMENT,

                        EDate varchar(25),

                        EmpID int,

                        WSID int,

                        SSID int,

                        EAmount decimal(10,2),

                        constraint pk\_ExpenseID primary key (ExpenseID),

                        constraint fk\_EmpID FOREIGN KEY(EmpID) REFERENCES Employees(EmpID),

                        constraint fk\_EWSID FOREIGN KEY(WSID) REFERENCES WaterSources(WSID),

                        constraint fk\_SSID FOREIGN KEY(SSID) REFERENCES SanitationSystems(SSID)

                    )

            `,

                function (err, result) {

                    if (err) throw err;

                }

            );

            connection.query(

                `select \* from Jobs where Designation="Admin"`,

                function (err, result) {

                    if (err) throw err;

                    if (result.length === 0) {

                        connection.query(

                            `INSERT into Jobs(Designation,Shift) values('Resigned', 'None')`,

                            function (err, result) {

                                if (err) throw err;

                            }

                        );

                        connection.query(

                            `INSERT into Jobs(Designation,Shift) values('Admin', 'Full-time')`,

                            function (err, result) {

                                if (err) throw err;

                            }

                        );

                    }

                }

            );

            connection.query(

                `select \* from Locations where Pincode=000000`,

                function (err, result) {

                    if (err) throw err;

                    if (result.length === 0) {

                        connection.query(

                            `INSERT into Locations(Pincode, Panchayat, District) values(000000, 'Panchayat', 'District')`,

                            function (err, result) {

                                if (err) throw err;

                            }

                        );

                    }

                }

            );

            for (let admin in admins) {

                connection.query(

                    `select \* from Employees where Username="${admins[admin].Username}"`,

                    function (err, result) {

                        if (err) throw err;

                        if (result.length === 0) {

                            connection.query(

                                `INSERT into Employees(FName,LName,EContact,JobCode,Pincode,Username,Password) values('${

                                    admins[admin].FName

                                }', '${admins[admin].LName}', ${

                                    admins[admin].EContact

                                }, ${admins[admin].JobCode}, ${

                                    admins[admin].Pincode

                                }, '${

                                    admins[admin].Username

                                }', '${bcrypt.hashSync(

                                    admins[admin].Password,

                                    +config.hashsecret

                                )}')`,

                                function (err, result) {

                                    if (err) throw err;

                                    console.log(

                                        `Admin: ${admins[admin].Username} created`

                                    );

                                    if (admin == admins.length - 1) {

                                        console.log(

                                            "finished seeding admins ✅\n"

                                        );

                                        console.log(

                                            "App Initialization Complete!"

                                        );

                                    }

                                }

                            );

                        } else {

                            console.log(

                                `Admin: ${admins[admin].Username} - already exists`

                            );

                            if (admin == admins.length - 1) {

                                console.log("finished seeding admins ✅\n");

                                console.log("App Initialization Complete!");

                            }

                        }

                    }

                );

            }

        });

    } catch (error) {

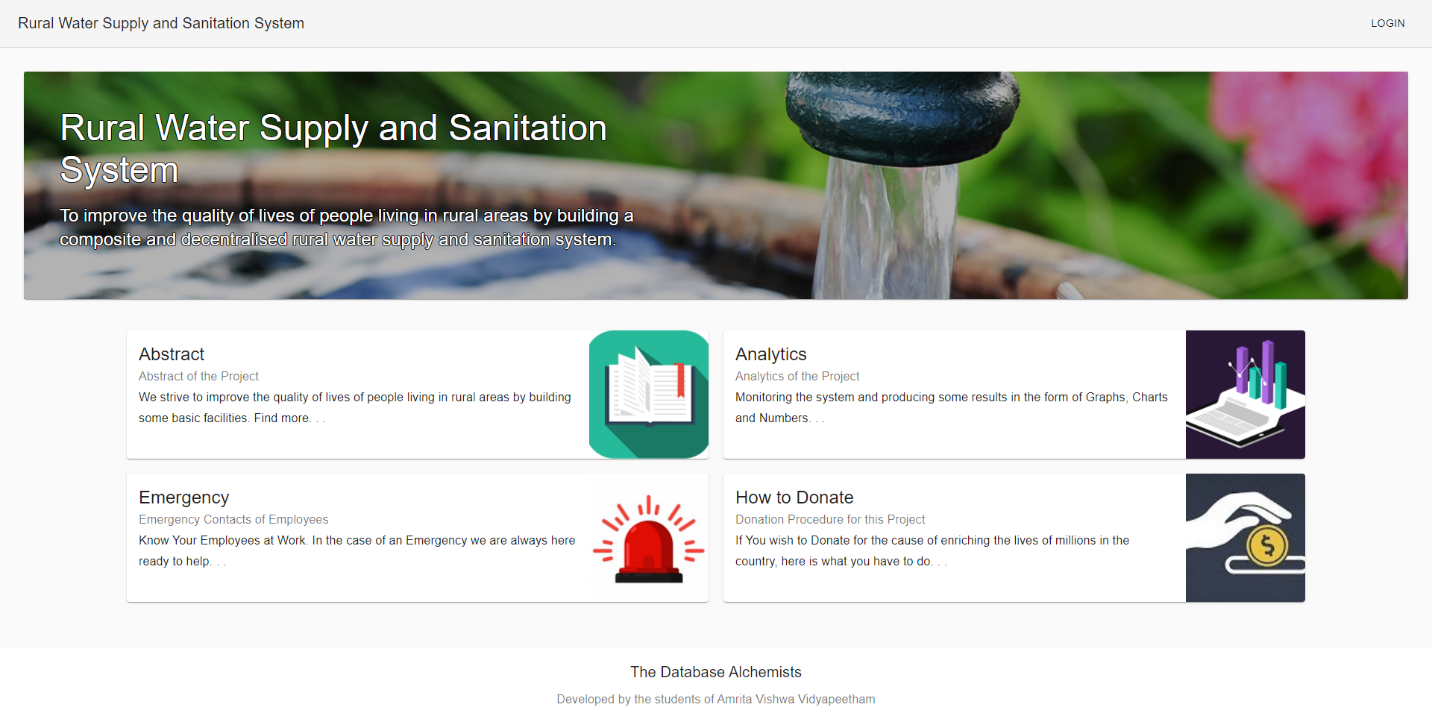
        console.log(error.toString());

    }

}

initializer();

# UI Screen – Home Page



# UI Screen – Employees

## Code – Back end:

const express = require("express");

const router = express.Router();

const connection = require("./../db");

const config = require("./../config");

const bcrypt = require("bcryptjs");

router.get("/getallemployees", (req, res, next) => {

    if(req.userDetails.Designation !== 'Admin'){

        res.status(400).json({ message: "Only Admins are Allowed to get employee details!" });

        return;

    }

    connection.query(`SELECT \* from Employees where JobCode<>1`, function (err, result) {

        if (err) {

            res.status(500).json({ message: err.toString() });

            return;

        }

        res.status(200).json({

            message: "Employees Fetched Successfully!",

            result,

        });

    });

});

router.post("/addemployee", (req, res, next) => {

    if(req.userDetails.Designation !== 'Admin'){

        res.status(400).json({ message: "Only Admins are Allowed to add employees!" });

        return;

    }

    connection.query(

        `INSERT into Employees(FName, LName, EContact, JobCode, Pincode, Username, Password) values('${

            req.body.FName

        }','${req.body.LName}',${req.body.EContact},${req.body.JobCode},${

            req.body.Pincode

        },'${req.body.Username}','${bcrypt.hashSync(

            req.body.Password,

            +config.hashsecret

        )}')`,

        function (err, result) {

            if (err) {

                res.status(500).json({ message: err.toString() });

                return;

            }

            res.status(200).json({ message: "Employee Added Successfully!" });

        }

    );

});

router.post("/markresigned", (req, res, next) => {

    console.log(req.body);

    if(req.userDetails.Designation !== 'Admin'){

        res.status(400).json({ message: "Only Admins are Allowed to mark an employee resignation!" });

        return;

    }

    connection.query(

        `update Employees SET JobCode=1 where EmpID=${req.body.EmpID}`,

        function (err, result) {

            if (err) {

                res.status(500).json({ message: err.toString() });

                return;

            }

            res.status(200).json({ message: "Employee Marked Resigned!" });

        }

    );

});

module.exports = router;

## Code – Front end:

import React, { Component } from "react";

import axios from "axios";

import Table from "@material-ui/core/Table";

import TableBody from "@material-ui/core/TableBody";

import TableCell from "@material-ui/core/TableCell";

import TableHead from "@material-ui/core/TableHead";

import TableRow from "@material-ui/core/TableRow";

import Container from "@material-ui/core/Container";

import Grid from "@material-ui/core/Grid";

import Paper from "@material-ui/core/Paper";

import Typography from "@material-ui/core/Typography";

import Button from "@material-ui/core/Button";

import TextField from "@material-ui/core/TextField";

import InputLabel from "@material-ui/core/InputLabel";

import MenuItem from "@material-ui/core/MenuItem";

import FormControl from "@material-ui/core/FormControl";

import Select from "@material-ui/core/Select";

import Snackbar from "@material-ui/core/Snackbar";

import SnackbarContent from "@material-ui/core/SnackbarContent";

import IconButton from "@material-ui/core/IconButton";

import Dialog from '@material-ui/core/Dialog';

import DialogActions from '@material-ui/core/DialogActions';

import DialogTitle from '@material-ui/core/DialogTitle';

import CloseIcon from "@material-ui/icons/Close";

import { withStyles } from "@material-ui/core/styles";

import Header from "../components/Header";

import Resign from "../static/resign.svg";

async function fetchDB(token) {

  let resdata = [];

  await axios

    .get(`http://localhost:3001/api/employees/getallemployees`, { headers: { Authorization: "Bearer " + token } })

    .then((res) => {

      resdata = res.data.result;

    })

    .catch((err) => {

      console.log(err);

    });

  return resdata;

}

async function fetchJobs(token) {

  let resdata = [];

  await axios

    .get(`http://localhost:3001/api/jobs/getalljobs`, { headers: { Authorization: "Bearer " + token } })

    .then((res) => {

      resdata = res.data.result;

    })

    .catch((err) => {

      console.log(err);

    });

  return resdata;

}

async function fetchLocations() {

  let resdata = [];

  await axios

    .get(`http://localhost:3001/api/location/getalllocations`)

    .then((res) => {

      resdata = res.data.result;

    })

    .catch((err) => {

      console.log(err);

    });

  return resdata;

}

const styles = (theme) => ({

  seeMore: {

    marginTop: theme.spacing(3),

  },

  container: {

    paddingTop: theme.spacing(4),

    paddingBottom: theme.spacing(4),

  },

  paper: {

    padding: theme.spacing(2),

    display: "flex",

    flexDirection: "column",

    alignItems: "center",

  },

  form: {

    width: "100%",

    marginTop: theme.spacing(1),

  },

  submit: {

    margin: theme.spacing(3, 0, 2),

  },

});

class Employee extends React.Component {

  state = {

    rows: [],

    showEmployee: false,

    showEmployeeText: "Show Employee",

    jobSelect: "",

    locationSelect: "",

    deleteSelect: null,

    snackbarColor: "",

    Token: null,

    snackbarMessage: "",

    openDialog: false,

    availableJobs: [],

    availableLocation: [],

    val : {

      FName:"",

        LName: "",

        EContact: "",

        Username: "",

        Password: "",

    }

  };

  async componentDidMount() {

    let Token = sessionStorage.getItem("Token");

    if (!Token || Token.length === 0) {

      this.setState({

        ...this.state,

        snackbarMessage: "Please Login First!!!",

        open: true,

        snackbarColor: "red",

      });

      let self = this;

      setTimeout(function () {

        self.props.history.push("/");

      }, 500);

    }

    await this.setState({ Token });

    let Designation = sessionStorage.getItem("Designation");

    if (Designation !== "Admin") {

      this.setState({

        ...this.state,

        snackbarMessage: "Login as Admin First!!!",

        open: true,

        snackbarColor: "red",

      });

      let self = this;

      setTimeout(function () {

        self.props.history.push("/Dashboard");

      }, 500);

    }

    let newrows = await fetchDB(Token);

    let jobs = await fetchJobs(this.state.Token);

    let locations = await fetchLocations();

    this.setState({

      rows: newrows,

      availableJobs: jobs,

      availableLocation: locations,

    });

  }

  handleSubmit = (e) => {

    e.preventDefault();

    e.persist();

    for(let txt of Object.values(this.state.val)) {

      if(txt.length > 0) {

        this.setState({ open: true, snackbarMessage: "Invalid Values!", snackbarColor: "red" });

        return;

      }

    }

    let ev = e;

    axios

      .post(`http://localhost:3001/api/employees/addemployee`, {

        FName: e.target.FName.value,

        LName: e.target.LName.value,

        EContact: e.target.EContact.value,

        JobCode: e.target.JobCode.value,

        Pincode: e.target.Pincode.value,

        Username: e.target.Username.value,

        Password: e.target.Password.value,

      }, { headers: { Authorization: "Bearer " + this.state.Token } })

      .then(async (res) => {

        let newrows = await fetchDB(this.state.Token);

        this.setState({

          ...this.state,

          rows: newrows,

          snackbarMessage: res.data.message,

          open: true,

          snackbarColor: "green",

          jobSelect: "",

          locationSelect: "",

        });

        ev.target.reset();

      })

      .catch((err) => {

        console.log(err);

        this.setState({

          ...this.state,

          open: true,

          snackbarMessage: err.response.data.message,

          snackbarColor: "red",

        });

      });

  };

  handleClose = (event, reason) => {

    if (reason === "clickaway") {

      return;

    }

    this.setState({ ...this.state, open: false });

  };

  handleDisagree = (event, reason) => {

    if (reason === "clickaway") {

      return;

    }

    this.setState({ ...this.state, openDialog: false, deleteSelect: null  });

  };

  handleAgree = () => {

    console.log("resign",this.state.deleteSelect);

    axios

      .post(`http://localhost:3001/api/employees/markresigned`, {

        EmpID: this.state.deleteSelect.EmpID

      }, { headers: { Authorization: "Bearer " + this.state.Token } })

      .then(async (res) => {

        let newrows = await fetchDB(this.state.Token);

        this.setState({

          ...this.state,

          rows: newrows,

          snackbarMessage: res.data.message,

          open: true,

          snackbarColor: "green",

        });

      })

      .catch((err) => {

        console.log(err);

        this.setState({

          ...this.state,

          open: true,

          snackbarMessage: err.response.data.message,

          snackbarColor: "red",

        });

      });

    this.setState({ ...this.state, openDialog: false, deleteSelect: null });

  }

  deleteComponent(row) {

    this.setState({ deleteSelect: row, openDialog: true });

  }

  renderTable() {

    const { classes } = this.props;

    if (this.state.showEmployee) {

      return (

        <Container maxWidth="lg" className={classes.container}>

          <Grid container spacing={3}>

            <Grid item xs={12}>

              <Paper className={classes.paper}>

                <Typography component="h2" variant="h6" gutterBottom>

                  All Employees

                </Typography>

                <Table size="medium">

                  <TableHead>

                    <TableRow>

                      <TableCell>Employee ID</TableCell>

                      <TableCell>Name</TableCell>

                      <TableCell>Contact</TableCell>

                      <TableCell>Username</TableCell>

                      <TableCell>Job Code</TableCell>

                      <TableCell>Pincode</TableCell>

                      <TableCell>Actions</TableCell>

                    </TableRow>

                  </TableHead>

                  <TableBody>

                    {this.state.rows.map((row) => (

                      <TableRow key={row.EmpID}>

                        <TableCell>{row.EmpID}</TableCell>

                        <TableCell>{`${row.FName} ${row.LName}`}</TableCell>

                        <TableCell>{row.EContact}</TableCell>

                        <TableCell>{row.Username}</TableCell>

                        <TableCell>{row.JobCode}</TableCell>

                        <TableCell>{row.Pincode}</TableCell>

                        <TableCell component="th" scope="row">

                          <IconButton

                            color="secondary"

                            aria-label="mark resigned"

                            component="span"

                            onClick={(event) => this.deleteComponent(row)}

                          >

                            <img height="24" width="24" src={Resign}></img>

                          </IconButton>

                        </TableCell>

                      </TableRow>

                    ))}

                  </TableBody>

                </Table>

              </Paper>

            </Grid>

          </Grid>

        </Container>

      );

    }

    return <br />;

  }

  render() {

    const { classes } = this.props;

    return (

      <React.Fragment>

        <Header />

        <Snackbar

          open={this.state.open}

          autoHideDuration={6000}

          onClose={this.handleClose}

        >

          <SnackbarContent

            style={{

              backgroundColor: this.state.snackbarColor,

            }}

            action={

              <React.Fragment>

                <IconButton

                  size="small"

                  aria-label="close"

                  color="inherit"

                  onClick={this.handleClose}

                >

                  <CloseIcon fontSize="small" />

                </IconButton>

              </React.Fragment>

            }

            message={

              <span id="client-snackbar">{this.state.snackbarMessage}</span>

            }

          />

        </Snackbar>

        <Container component="main" maxWidth="xs">

          <div className={classes.paper}>

            <Typography component="h1" variant="h5">

              Employee

            </Typography>

            <form className={classes.form} onSubmit={this.handleSubmit}>

              <TextField

                variant="outlined"

                margin="normal"

                required

                fullWidth

                name="FName"

                label="First Name"

                type="text"

                id="FName"

                autoFocus

                error={(this.state.val.FName.length === 0)? false : true}

                helperText={this.state.val.FName}

                onChange={(e) => {

                  let val = this.state.val;

                  var format = /[!@#$%^&\*()\_+\-=\[\]{};':"\\|,.<>\/?0-9]+/;

                  if (format.test(e.target.value)) val.FName = "Name cannot contain special symbols";

                  else val.FName = "";

                  this.setState({ val });

                }}

                error={(this.state.val.FName.length === 0)? false : true}

              />

              <TextField

                variant="outlined"

                margin="normal"

                required

                fullWidth

                name="LName"

                label="Last Name"

                type="text"

                id="LName"

                error={(this.state.val.LName.length === 0)? false : true}

                helperText={this.state.val.LName}

                onChange={(e) => {

                  let val = this.state.val;

                  var format = /[!@#$%^&\*()\_+\-=\[\]{};':"\\|,.<>\/?0-9]+/;

                  if (format.test(e.target.value)) val.LName = "Name cannot contain special symbols";

                  else val.LName = "";

                  this.setState({ val });

                }}

                error={(this.state.val.LName.length === 0)? false : true}

              />

              <TextField

                variant="outlined"

                margin="normal"

                required

                fullWidth

                name="EContact"

                label="Conatct"

                type="number"

                id="EContact"

                error={(this.state.val.EContact.length === 0)? false : true}

                helperText={this.state.val.EContact}

                onChange={(e) => {

                  var format = /[0-9]+/;

                  var cformat = /[0-5]+/;

                  var val = this.state.val

                  if (!format.test(e.target.value) || e.target.value.length !== 10) val.EContact="Contact Number must have 10 numbers";

                  else if (e.target.value.toString()[0].match(cformat)) val.EContact="Please enter a valid contact number";

                  else val.EContact = "";

                  this.setState({ val });

                }}

                error={(this.state.val.EContact.length === 0)? false : true}

              />

              <FormControl

                variant="outlined"

                fullWidth

                className={classes.form}

              >

                <InputLabel id="Job-Label">Job Code</InputLabel>

                <Select

                  labelId="Job-Label"

                  id="JobCode"

                  label="Job Code"

                  name="JobCode"

                  variant="outlined"

                  value={this.state.jobSelect}

                  onOpen={(e) => {

                    if (this.state.availableJobs.length === 0)

                      this.setState({

                        open: true,

                        snackbarMessage: "Jobs Unavailable!",

                        snackbarColor: "red",

                      });

                  }}

                  onChange={(e) => {

                    this.setState({ jobSelect: e.target.value });

                  }}

                  required

                  fullWidth

                >

                  {this.state.availableJobs.map((job) => (

                    <MenuItem

                      key={job.JobCode}

                      value={job.JobCode}

                    >{`${job.JobCode} - ${job.Designation} - ${job.Shift}`}</MenuItem>

                  ))}

                </Select>

              </FormControl>

              <FormControl

                variant="outlined"

                fullWidth

                className={classes.form}

              >

                <InputLabel id="Pincode-Label">Pincode</InputLabel>

                <Select

                  labelId="Pincode-Label"

                  id="Pincode"

                  label="Pincode"

                  name="Pincode"

                  variant="outlined"

                  value={this.state.locationSelect}

                  onOpen={(e) => {

                    if (this.state.availableLocation.length === 0)

                      this.setState({

                        open: true,

                        snackbarMessage: "Location Unavailable!",

                        snackbarColor: "red",

                      });

                  }}

                  onChange={(e) => {

                    this.setState({ locationSelect: e.target.value });

                  }}

                  required

                  fullWidth

                >

                  {this.state.availableLocation.map((pin) => (

                    <MenuItem

                      key={pin.Pincode}

                      value={pin.Pincode}

                    >{`${pin.Pincode} - ${pin.Panchayat} - ${pin.District}`}</MenuItem>

                  ))}

                </Select>

              </FormControl>

              <TextField

                variant="outlined"

                margin="normal"

                required

                fullWidth

                name="Usename"

                label="Usename"

                type="text"

                id="Username"

                error={(this.state.val.Username.length === 0)? false : true}

                helperText={this.state.val.Username}

                onChange={(e) => {

                  let val = this.state.val;

                  var format = /[!#$%^&\*()+\-=\[\]{};':"\\|,<>\/?]+/;

                  if (format.test(e.target.value)) val.Username = "Username format invalid";

                  else val.Username = "";

                  this.setState({ val });

                }}

                error={(this.state.val.Username.length === 0)? false : true}

              />

              <TextField

                variant="outlined"

                margin="normal"

                required

                fullWidth

                name="Password"

                label="Password"

                type="text"

                id="Password"

                error={(this.state.val.Password.length === 0)? false : true}

                helperText={this.state.val.Password}

                onChange={(e) => {

                  let val = this.state.val;

                  var format = /[!#$%^&\*()\_+\-=\[\]{};':"\\|,<>\/?]+/;

                  if (format.test(e.target.value)) val.Password = "Inalid Password Format";

                  else val.Password = "";

                  this.setState({ val });

                }}

                error={(this.state.val.Password.length === 0)? false : true}

              />

              <Button

                type="submit"

                fullWidth

                variant="contained"

                color="primary"

                className={classes.submit}

              >

                Add Employee

              </Button>

              <Button

                variant="contained"

                color="primary"

                fullWidth

                className={classes.submit}

                onClick={(e) => {

                  console.log("came in");

                  e.preventDefault();

                  if (!this.state.showEmployee)

                    this.setState({

                      ...this.state,

                      showEmployee: true,

                      showEmployeeText: "Hide Employee",

                    });

                  else

                    this.setState({

                      ...this.state,

                      showEmployee: false,

                      showEmployeeText: "Show Employee",

                    });

                  console.log(this.state);

                }}

              >

                {this.state.showEmployeeText}

              </Button>

            </form>

          </div>

        </Container>

        {this.renderTable()}

        <Dialog

          open={this.state.openDialog}

          keepMounted

          onClose={this.handleDisagree}

          aria-labelledby="alert-dialog-slide-title"

          aria-describedby="alert-dialog-slide-description"

        >

          <DialogTitle id="alert-dialog-slide-title">

            {"Are you sure you want to Mark Resignation for this Employee?"}

          </DialogTitle>

          <DialogActions>

            <Button onClick={this.handleDisagree} color="primary">

              Cancel

            </Button>

            <Button onClick={this.handleAgree} color="secondary">

              Mark Resign

            </Button>

          </DialogActions>

        </Dialog>

      </React.Fragment>

    );

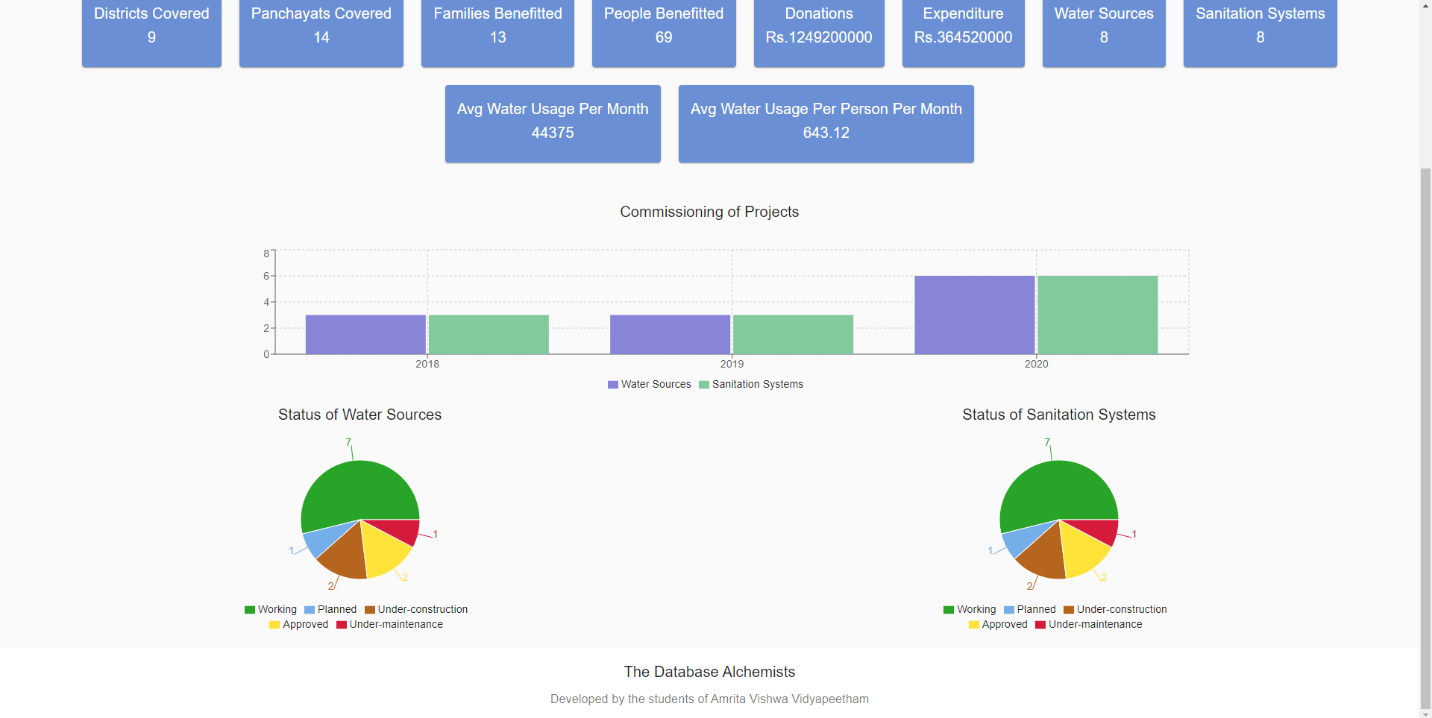
  }

}

export default withStyles(styles, { withTheme: true })(Employee);

# Analytics

## UI Screen



## Back End

const express = require("express");

const router = express.Router();

const connection = require("./../db");

const config = require("./../config");

router.get("/getallstatistics", (req, res, next) => {

    let finalresult = {

        totalDistictsCovered: 0,

        totalPanchayatsCovered: 0,

        totalFamiliesBenefitted: 0,

        totalPeopleBenefitted: 0,

        totalDonations: 0,

        totalExpenditure: 0,

        totalWaterSourcesConstructed: 0,

        totalSanitationSystemsConstructed: 0,

        averageWaterUsagePerPersonPerMonth: 0,

        averageWaterUsagePerMonth: 0,

        totalConstuctedBarGraph: [],

        waterSourcesStatusPieChart: [],

        sanitationSystemsStatusPieChart: [],

    };

    connection.query(

        `Select count(distinct district) as districtcount, count(distinct panchayat) as panchayatcount from Locations`,

        function (err, result) {

            if (err) {

                res.status(500).json({ message: err.toString() });

                return;

            }

            console.log("Locations: ", result);

            finalresult.totalDistictsCovered = result[0].districtcount;

            finalresult.totalPanchayatsCovered = result[0].panchayatcount;

            connection.query(

                `Select count(FID) as familycount, sum(Persons) as peoplecount from Families`,

                function (err, result) {

                    if (err) {

                        res.status(500).json({ message: err.toString() });

                        return;

                    }

                    console.log("Families: ", result);

                    finalresult.totalFamiliesBenefitted = result[0].familycount;

                    finalresult.totalPeopleBenefitted = result[0].peoplecount;

                    connection.query(

                        `Select sum(Amount) as totaldonations, sum(EAmount) as totalexpenditure from Donations, Expenditures`,

                        function (err, result) {

                            if (err) {

                                res.status(500).json({

                                    message: err.toString(),

                                });

                                return;

                            }

                            console.log("Donations: ", result);

                            finalresult.totalDonations =

                                result[0].totaldonations;

                            finalresult.totalExpenditure =

                                result[0].totalexpenditure;

                            connection.query(

                                `SELECT EDate, WSID, SSID from Expenditures`,

                                function (err, result) {

                                    if (err) {

                                        res.status(500).json({

                                            message: err.toString(),

                                        });

                                        return;

                                    }

                                    console.log(

                                        "constructedProjects: ",

                                        result

                                    );

                                    let constructedObjects = {};

                                    for (let eachresult of result) {

                                        let year = eachresult.EDate.split(

                                            "-"

                                        )[2];

                                        if (constructedObjects[year]) {

                                            if (eachresult.WSID)

                                                constructedObjects[

                                                    year

                                                ].ws += 1;

                                            else

                                                constructedObjects[

                                                    year

                                                ].ss += 1;

                                        } else {

                                            if (eachresult.WSID)

                                                constructedObjects[year] = {

                                                    ws: 1,

                                                    ss: 0,

                                                };

                                            else

                                                constructedObjects[year] = {

                                                    ss: 1,

                                                    ws: 0,

                                                };

                                        }

                                    }

                                    for (let key in constructedObjects) {

                                        let eachObject = {

                                            name: key.toString(),

                                            "Water Sources":

                                                constructedObjects[key].ws,

                                            "Sanitation Systems":

                                                constructedObjects[key].ss,

                                        };

                                        finalresult.totalConstuctedBarGraph.push(

                                            eachObject

                                        );

                                    }

                                    console.log(

                                        "Modified Results: ",

                                        finalresult.totalConstuctedBarGraph

                                    );

                                    connection.query(

                                        `Select WStatus from WaterSources`,

                                        function (err, result) {

                                            if (err) {

                                                res.status(500).json({

                                                    message: err.toString(),

                                                });

                                                return;

                                            }

                                            console.log(

                                                "WaterSourceStatuses: ",

                                                result

                                            );

                                            let constructedObjects = {};

                                            for (let eachresult of result) {

                                                if (

                                                    constructedObjects[

                                                        eachresult.WStatus

                                                    ]

                                                )

                                                    constructedObjects[

                                                        eachresult.WStatus

                                                    ] += 1;

                                                else

                                                    constructedObjects[

                                                        eachresult.WStatus

                                                    ] = 1;

                                            }

                                            for (let key in constructedObjects) {

                                                let eachObject = {

                                                    name: key,

                                                    value:

                                                        constructedObjects[key],

                                                };

                                                if (key === "Planned") {

                                                    eachObject.fill = "#75AFE9";

                                                } else if (key === "Approved") {

                                                    eachObject.fill = "#FFE338";

                                                } else if (

                                                    key === "Under-construction"

                                                ) {

                                                    eachObject.fill =

                                                        "#b5651d ";

                                                } else if (key === "Working") {

                                                    eachObject.fill = "#28A428";

                                                } else if (

                                                    key === "Under-maintenance"

                                                ) {

                                                    eachObject.fill = "#D61A3C";

                                                }

                                                finalresult.waterSourcesStatusPieChart.push(

                                                    eachObject

                                                );

                                                if (

                                                    key === "Working" ||

                                                    key === "Under-maintenance"

                                                ) {

                                                    finalresult.totalWaterSourcesConstructed +=

                                                        constructedObjects[key];

                                                }

                                            }

                                            console.log(

                                                "WaterSourceStatus Data: ",

                                                finalresult.waterSourcesStatusPieChart

                                            );

                                            console.log(

                                                "TotalWaterSourcesConstructed: ",

                                                finalresult.totalWaterSourcesConstructed

                                            );

                                            connection.query(

                                                `Select SStatus from SanitationSystems`,

                                                function (err, result) {

                                                    if (err) {

                                                        res.status(500).json({

                                                            message: err.toString(),

                                                        });

                                                        return;

                                                    }

                                                    console.log(

                                                        "SanitationSystemsStatuses: ",

                                                        result

                                                    );

                                                    let constructedObjects = {};

                                                    for (let eachresult of result) {

                                                        if (

                                                            constructedObjects[

                                                                eachresult

                                                                    .SStatus

                                                            ]

                                                        )

                                                            constructedObjects[

                                                                eachresult.SStatus

                                                            ] += 1;

                                                        else

                                                            constructedObjects[

                                                                eachresult.SStatus

                                                            ] = 1;

                                                    }

                                                    for (let key in constructedObjects) {

                                                        let eachObject = {

                                                            name: key,

                                                            value:

                                                                constructedObjects[

                                                                    key

                                                                ],

                                                        };

                                                        if (key === "Planned") {

                                                            eachObject.fill =

                                                                "#75AFE9";

                                                        } else if (

                                                            key === "Approved"

                                                        ) {

                                                            eachObject.fill =

                                                                "#FFE338";

                                                        } else if (

                                                            key ===

                                                            "Under-construction"

                                                        ) {

                                                            eachObject.fill =

                                                                "#b5651d ";

                                                        } else if (

                                                            key === "Working"

                                                        ) {

                                                            eachObject.fill =

                                                                "#28A428";

                                                        } else if (

                                                            key ===

                                                            "Under-maintenance"

                                                        ) {

                                                            eachObject.fill =

                                                                "#D61A3C";

                                                        }

                                                        finalresult.sanitationSystemsStatusPieChart.push(

                                                            eachObject

                                                        );

                                                        if (

                                                            key === "Working" ||

                                                            key ===

                                                                "Under-maintenance"

                                                        ) {

                                                            finalresult.totalSanitationSystemsConstructed +=

                                                                constructedObjects[

                                                                    key

                                                                ];

                                                        }

                                                    }

                                                    console.log(

                                                        "SanitaionSystemsStatus Data: ",

                                                        finalresult.sanitationSystemsStatusPieChart

                                                    );

                                                    console.log(

                                                        "TotalSanitaionSystemsConstructed: ",

                                                        finalresult.totalSanitationSystemsConstructed

                                                    );

                                                    connection.query(

                                                        `Select avg(u.Usages) as averageusagepermonth from WaterUsages u join WaterSources w on w.WSID = u.WSID`,

                                                        function (err, result) {

                                                            if (err) {

                                                                res.status(

                                                                    500

                                                                ).json({

                                                                    message: res.toString(),

                                                                });

                                                            }

                                                            finalresult.averageWaterUsagePerMonth =

                                                                result[0].averageusagepermonth;

                                                            finalresult.averageWaterUsagePerPersonPerMonth =

                                                                finalresult.averageWaterUsagePerMonth /

                                                                finalresult.totalPeopleBenefitted;

                                                            res.status(

                                                                200

                                                            ).json({

                                                                message:

                                                                    "Statistics Fetched Successfully!",

                                                                finalresult,

                                                            });

                                                        }

                                                    );

                                                }

                                            );

                                        }

                                    );

                                }

                            );

                        }

                    );

                }

            );

        }

    );

});

router.post("/getpanchayatstatistics", (req, res, next) => {

    let finalresult = {

        totalFamiliesBenefitted: 0,

        totalPeopleBenefitted: 0,

        totalExpenditure: 0,

        totalWaterSourcesConstructed: 0,

        totalSanitationSystemsConstructed: 0,

        averageWaterUsagePerMonth: 0,

        averageWaterUsagePerPersonPerMonth: 0,

        totalConstuctedBarGraph: [],

        waterSourcesStatusPieChart: [],

        sanitationSystemsStatusPieChart: [],

    };

    connection.query(

        `Select count(FID) as familycount, sum(Persons) as peoplecount from Families where Pincode = ${req.body.Pincode}`,

        function (err, result) {

            if (err) {

                res.status(500).json({ message: err.toString() });

                return;

            }

            console.log("Families: ", result);

            finalresult.totalFamiliesBenefitted = result[0].familycount;

            finalresult.totalPeopleBenefitted = result[0].peoplecount;

            connection.query(

                `Select sum(e.EAmount) as totalwsexpenditure from Expenditures e join WaterSources w on e.WSID = w.WSID where w.Pincode = ${req.body.Pincode}`,

                function (err, result) {

                    if (err) {

                        res.status(500).json({ message: err.toString() });

                        return;

                    }

                    console.log("WS Expenditures: ", result);

                    finalresult.totalExpenditure +=

                        result[0].totalwsexpenditure;

                    connection.query(

                        `Select sum(e.EAmount) as totalssexpenditure from Expenditures e join SanitationSystems s on e.SSID = s.SSID where s.Pincode = ${req.body.Pincode}`,

                        function (err, result) {

                            if (err) {

                                res.status(500).json({

                                    message: err.toString(),

                                });

                                return;

                            }

                            console.log("SS Expenditures: ", result);

                            finalresult.totalExpenditure +=

                                result[0].totalssexpenditure;

                            connection.query(

                                `SELECT e.EDate, e.WSID from Expenditures e join WaterSources w on e.wsid = w.wsid where w.Pincode = ${req.body.Pincode}`,

                                function (err, result) {

                                    if (err) {

                                        res.status(500).json({

                                            message: err.toString(),

                                        });

                                        return;

                                    }

                                    console.log("EDate,WSID", result);

                                    let constructedObjects = {};

                                    for (let eachresult of result) {

                                        let year = eachresult.EDate.split(

                                            "-"

                                        )[2];

                                        if (constructedObjects[year]) {

                                            constructedObjects[year].ws += 1;

                                        } else {

                                            constructedObjects[year] = {

                                                ws: 1,

                                                ss: 0,

                                            };

                                        }

                                    }

                                    connection.query(

                                        `SELECT e.EDate, e.SSID from Expenditures e join SanitationSystems s on e.ssid = s.ssid where s.Pincode = ${req.body.Pincode}`,

                                        function (err, result) {

                                            if (err) {

                                                res.status(500).json({

                                                    message: err.toString(),

                                                });

                                                return;

                                            }

                                            console.log("EDate,SSID", result);

                                            for (let eachresult of result) {

                                                let year = eachresult.EDate.split(

                                                    "-"

                                                )[2];

                                                if (constructedObjects[year]) {

                                                    constructedObjects[

                                                        year

                                                    ].ss += 1;

                                                } else {

                                                    constructedObjects[year] = {

                                                        ws: 0,

                                                        ss: 1,

                                                    };

                                                }

                                            }

                                            for (let key in constructedObjects) {

                                                let eachObject = {

                                                    name: key.toString(),

                                                    "Water Sources":

                                                        constructedObjects[key]

                                                            .ws,

                                                    "Sanitation Systems":

                                                        constructedObjects[key]

                                                            .ss,

                                                };

                                                finalresult.totalConstuctedBarGraph.push(

                                                    eachObject

                                                );

                                            }

                                            console.log(

                                                "Modified Results: ",

                                                finalresult.totalConstuctedBarGraph

                                            );

                                            connection.query(

                                                `Select WStatus from WaterSources where Pincode = ${req.body.Pincode}`,

                                                function (err, result) {

                                                    if (err) {

                                                        res.status(500).json({

                                                            message: err.toString(),

                                                        });

                                                        return;

                                                    }

                                                    console.log(

                                                        "WaterSourceStatuses: ",

                                                        result

                                                    );

                                                    let constructedObjects = {};

                                                    for (let eachresult of result) {

                                                        if (

                                                            constructedObjects[

                                                                eachresult

                                                                    .WStatus

                                                            ]

                                                        )

                                                            constructedObjects[

                                                                eachresult.WStatus

                                                            ] += 1;

                                                        else

                                                            constructedObjects[

                                                                eachresult.WStatus

                                                            ] = 1;

                                                    }

                                                    for (let key in constructedObjects) {

                                                        let eachObject = {

                                                            name: key,

                                                            value:

                                                                constructedObjects[

                                                                    key

                                                                ],

                                                        };

                                                        if (key === "Planned") {

                                                            eachObject.fill =

                                                                "#75AFE9";

                                                        } else if (

                                                            key === "Approved"

                                                        ) {

                                                            eachObject.fill =

                                                                "#FFE338";

                                                        } else if (

                                                            key ===

                                                            "Under-construction"

                                                        ) {

                                                            eachObject.fill =

                                                                "#b5651d ";

                                                        } else if (

                                                            key === "Working"

                                                        ) {

                                                            eachObject.fill =

                                                                "#28A428";

                                                        } else if (

                                                            key ===

                                                            "Under-maintenance"

                                                        ) {

                                                            eachObject.fill =

                                                                "#D61A3C";

                                                        }

                                                        finalresult.waterSourcesStatusPieChart.push(

                                                            eachObject

                                                        );

                                                        if (

                                                            key === "Working" ||

                                                            key ===

                                                                "Under-maintenance"

                                                        ) {

                                                            finalresult.totalWaterSourcesConstructed +=

                                                                constructedObjects[

                                                                    key

                                                                ];

                                                        }

                                                    }

                                                    console.log(

                                                        "WaterSourceStatus Data: ",

                                                        finalresult.waterSourcesStatusPieChart

                                                    );

                                                    console.log(

                                                        "TotalWaterSourcesConstructed: ",

                                                        finalresult.totalWaterSourcesConstructed

                                                    );

                                                    connection.query(

                                                        `Select SStatus from SanitationSystems where Pincode = ${req.body.Pincode}`,

                                                        function (err, result) {

                                                            if (err) {

                                                                res.status(

                                                                    500

                                                                ).json({

                                                                    message: err.toString(),

                                                                });

                                                                return;

                                                            }

                                                            console.log(

                                                                "SanitationSystemsStatuses: ",

                                                                result

                                                            );

                                                            let constructedObjects = {};

                                                            for (let eachresult of result) {

                                                                if (

                                                                    constructedObjects[

                                                                        eachresult

                                                                            .SStatus

                                                                    ]

                                                                )

                                                                    constructedObjects[

                                                                        eachresult.SStatus

                                                                    ] += 1;

                                                                else

                                                                    constructedObjects[

                                                                        eachresult.SStatus

                                                                    ] = 1;

                                                            }

                                                            for (let key in constructedObjects) {

                                                                let eachObject = {

                                                                    name: key,

                                                                    value:

                                                                        constructedObjects[

                                                                            key

                                                                        ],

                                                                };

                                                                if (

                                                                    key ===

                                                                    "Planned"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#75AFE9";

                                                                } else if (

                                                                    key ===

                                                                    "Approved"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#FFE338";

                                                                } else if (

                                                                    key ===

                                                                    "Under-construction"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#b5651d ";

                                                                } else if (

                                                                    key ===

                                                                    "Working"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#28A428";

                                                                } else if (

                                                                    key ===

                                                                    "Under-maintenance"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#D61A3C";

                                                                }

                                                                finalresult.sanitationSystemsStatusPieChart.push(

                                                                    eachObject

                                                                );

                                                                if (

                                                                    key ===

                                                                        "Working" ||

                                                                    key ===

                                                                        "Under-maintenance"

                                                                ) {

                                                                    finalresult.totalSanitationSystemsConstructed +=

                                                                        constructedObjects[

                                                                            key

                                                                        ];

                                                                }

                                                            }

                                                            console.log(

                                                                "SanitaionSystemsStatus Data: ",

                                                                finalresult.sanitationSystemsStatusPieChart

                                                            );

                                                            console.log(

                                                                "TotalSanitaionSystemsConstructed: ",

                                                                finalresult.totalSanitationSystemsConstructed

                                                            );

                                                            connection.query(

                                                                `Select avg(u.Usages) as averageusagepermonth from WaterUsages u join WaterSources w on w.WSID = u.WSID where w.Pincode = ${req.body.Pincode}`,

                                                                function (

                                                                    err,

                                                                    result

                                                                ) {

                                                                    if (err) {

                                                                        res.status(

                                                                            500

                                                                        ).json({

                                                                            message: res.toString(),

                                                                        });

                                                                    }

                                                                    console.log(

                                                                        "Average Usage Per Month",

                                                                        result[0]

                                                                    );

                                                                    finalresult.averageWaterUsagePerMonth =

                                                                        result[0]

                                                                            .averageusagepermonth ===

                                                                        null

                                                                            ? 0

                                                                            : result[0]

                                                                                    .averageusagepermonth;

                                                                    if (

                                                                        finalresult.averageWaterUsagePerMonth !==

                                                                            0 &&

                                                                        finalresult.totalPeopleBenefitted !==

                                                                            0

                                                                    )

                                                                        finalresult.averageWaterUsagePerPersonPerMonth =

                                                                            finalresult.averageWaterUsagePerMonth /

                                                                            finalresult.totalPeopleBenefitted;

                                                                    res.status(

                                                                        200

                                                                    ).json({

                                                                        message:

                                                                            "Statistics Fetched Successfully!",

                                                                        finalresult,

                                                                    });

                                                                }

                                                            );

                                                        }

                                                    );

                                                }

                                            );

                                        }

                                    );

                                }

                            );

                        }

                    );

                }

            );

        }

    );

});

router.post("/getdistrictstatistics", (req, res, next) => {

    let finalresult = {

        totalPanchayatsCovered: 0,

        totalFamiliesBenefitted: 0,

        totalPeopleBenefitted: 0,

        totalExpenditure: 0,

        totalWaterSourcesConstructed: 0,

        totalSanitationSystemsConstructed: 0,

        averageWaterUsagePerMonth: 0,

        averageWaterUsagePerPersonPerMonth: 0,

        totalConstuctedBarGraph: [],

        waterSourcesStatusPieChart: [],

        sanitationSystemsStatusPieChart: [],

    };

    console.log(req.body.District);

    connection.query(

        `SELECT count(FID) as familycount, sum(Persons) as peoplecount FROM Families

        WHERE Pincode = ANY (SELECT Pincode FROM Locations

                             WHERE Locations.Pincode = Families.Pincode and Locations.District = '${req.body.District}')`,

        function (err, result) {

            if (err) {

                console.log(err);

                res.status(500).json({ message: err.toString() });

                return;

            }

            console.log("Families: ", result);

            finalresult.totalFamiliesBenefitted = result[0].familycount;

            finalresult.totalPeopleBenefitted =

                result[0].peoplecount === null ? 0 : result[0].peoplecount;

            connection.query(

                `Select sum(e.EAmount) as totalwsexpenditure from Expenditures e join WaterSources w on e.WSID = w.WSID where w.Pincode = ANY (SELECT Pincode FROM Locations

                    WHERE Locations.Pincode = w.Pincode and Locations.District = '${req.body.District}')`,

                function (err, result) {

                    if (err) {

                        res.status(500).json({ message: err.toString() });

                        return;

                    }

                    console.log("WS Expenditures: ", result);

                    finalresult.totalExpenditure +=

                        result[0].totalwsexpenditure;

                    connection.query(

                        `Select sum(e.EAmount) as totalssexpenditure from Expenditures e join SanitationSystems s on e.SSID = s.SSID where s.Pincode = ANY (SELECT Pincode FROM Locations

                        WHERE Locations.Pincode = s.Pincode and Locations.District = '${req.body.District}')`,

                        function (err, result) {

                            if (err) {

                                res.status(200).json({

                                    message: err.toString(),

                                });

                            }

                            finalresult.totalExpenditure +=

                                result[0].totalssexpenditure;

                            connection.query(

                                `SELECT e.EDate, e.WSID from Expenditures e join WaterSources w on e.wsid = w.wsid where w.Pincode = ANY (SELECT Pincode FROM Locations

                                WHERE Locations.Pincode = w.Pincode and Locations.District = '${req.body.District}')`,

                                function (err, result) {

                                    if (err) {

                                        res.status(500).json({

                                            message: err.toString(),

                                        });

                                        return;

                                    }

                                    console.log("EDate,WSID", result);

                                    let constructedObjects = {};

                                    for (let eachresult of result) {

                                        let year = eachresult.EDate.split(

                                            "-"

                                        )[2];

                                        if (constructedObjects[year]) {

                                            constructedObjects[year].ws += 1;

                                        } else {

                                            constructedObjects[year] = {

                                                ws: 1,

                                                ss: 0,

                                            };

                                        }

                                    }

                                    connection.query(

                                        `SELECT e.EDate, e.SSID from Expenditures e join SanitationSystems s on e.ssid = s.ssid where s.Pincode = ANY (SELECT Pincode FROM Locations

                                        WHERE Locations.Pincode = s.Pincode and Locations.District = '${req.body.District}')`,

                                        function (err, result) {

                                            if (err) {

                                                res.status(500).json({

                                                    message: err.toString(),

                                                });

                                                return;

                                            }

                                            console.log("EDate,SSID", result);

                                            for (let eachresult of result) {

                                                let year = eachresult.EDate.split(

                                                    "-"

                                                )[2];

                                                if (constructedObjects[year]) {

                                                    constructedObjects[

                                                        year

                                                    ].ss += 1;

                                                } else {

                                                    constructedObjects[year] = {

                                                        ws: 0,

                                                        ss: 1,

                                                    };

                                                }

                                            }

                                            for (let key in constructedObjects) {

                                                let eachObject = {

                                                    name: key.toString(),

                                                    "Water Sources":

                                                        constructedObjects[key]

                                                            .ws,

                                                    "Sanitation Systems":

                                                        constructedObjects[key]

                                                            .ss,

                                                };

                                                finalresult.totalConstuctedBarGraph.push(

                                                    eachObject

                                                );

                                            }

                                            console.log(

                                                "Modified Results: ",

                                                finalresult.totalConstuctedBarGraph

                                            );

                                            connection.query(

                                                `Select w.WStatus from WaterSources w where w.Pincode = ANY (SELECT Pincode FROM Locations

                                                WHERE Locations.Pincode = w.Pincode and Locations.District = '${req.body.District}')`,

                                                function (err, result) {

                                                    if (err) {

                                                        res.status(500).json({

                                                            message: err.toString(),

                                                        });

                                                        return;

                                                    }

                                                    console.log(

                                                        "WaterSourceStatuses: ",

                                                        result

                                                    );

                                                    let constructedObjects = {};

                                                    for (let eachresult of result) {

                                                        if (

                                                            constructedObjects[

                                                                eachresult

                                                                    .WStatus

                                                            ]

                                                        )

                                                            constructedObjects[

                                                                eachresult.WStatus

                                                            ] += 1;

                                                        else

                                                            constructedObjects[

                                                                eachresult.WStatus

                                                            ] = 1;

                                                    }

                                                    for (let key in constructedObjects) {

                                                        let eachObject = {

                                                            name: key,

                                                            value:

                                                                constructedObjects[

                                                                    key

                                                                ],

                                                        };

                                                        if (key === "Planned") {

                                                            eachObject.fill =

                                                                "#75AFE9";

                                                        } else if (

                                                            key === "Approved"

                                                        ) {

                                                            eachObject.fill =

                                                                "#FFE338";

                                                        } else if (

                                                            key ===

                                                            "Under-construction"

                                                        ) {

                                                            eachObject.fill =

                                                                "#b5651d ";

                                                        } else if (

                                                            key === "Working"

                                                        ) {

                                                            eachObject.fill =

                                                                "#28A428";

                                                        } else if (

                                                            key ===

                                                            "Under-maintenance"

                                                        ) {

                                                            eachObject.fill =

                                                                "#D61A3C";

                                                        }

                                                        finalresult.waterSourcesStatusPieChart.push(

                                                            eachObject

                                                        );

                                                        if (

                                                            key === "Working" ||

                                                            key ===

                                                                "Under-maintenance"

                                                        ) {

                                                            finalresult.totalWaterSourcesConstructed +=

                                                                constructedObjects[

                                                                    key

                                                                ];

                                                        }

                                                    }

                                                    console.log(

                                                        "WaterSourceStatus Data: ",

                                                        finalresult.waterSourcesStatusPieChart

                                                    );

                                                    console.log(

                                                        "TotalWaterSourcesConstructed: ",

                                                        finalresult.totalWaterSourcesConstructed

                                                    );

                                                    connection.query(

                                                        `Select s.SStatus from SanitationSystems s where Pincode = ANY (SELECT Pincode FROM Locations

                                                        WHERE Locations.Pincode = s.Pincode and Locations.District = '${req.body.District}')`,

                                                        function (err, result) {

                                                            if (err) {

                                                                res.status(

                                                                    500

                                                                ).json({

                                                                    message: err.toString(),

                                                                });

                                                                return;

                                                            }

                                                            console.log(

                                                                "SanitationSystemsStatuses: ",

                                                                result

                                                            );

                                                            let constructedObjects = {};

                                                            for (let eachresult of result) {

                                                                if (

                                                                    constructedObjects[

                                                                        eachresult

                                                                            .SStatus

                                                                    ]

                                                                )

                                                                    constructedObjects[

                                                                        eachresult.SStatus

                                                                    ] += 1;

                                                                else

                                                                    constructedObjects[

                                                                        eachresult.SStatus

                                                                    ] = 1;

                                                            }

                                                            for (let key in constructedObjects) {

                                                                let eachObject = {

                                                                    name: key,

                                                                    value:

                                                                        constructedObjects[

                                                                            key

                                                                        ],

                                                                };

                                                                if (

                                                                    key ===

                                                                    "Planned"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#75AFE9";

                                                                } else if (

                                                                    key ===

                                                                    "Approved"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#FFE338";

                                                                } else if (

                                                                    key ===

                                                                    "Under-construction"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#b5651d ";

                                                                } else if (

                                                                    key ===

                                                                    "Working"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#28A428";

                                                                } else if (

                                                                    key ===

                                                                    "Under-maintenance"

                                                                ) {

                                                                    eachObject.fill =

                                                                        "#D61A3C";

                                                                }

                                                                finalresult.sanitationSystemsStatusPieChart.push(

                                                                    eachObject

                                                                );

                                                                if (

                                                                    key ===

                                                                        "Working" ||

                                                                    key ===

                                                                        "Under-maintenance"

                                                                ) {

                                                                    finalresult.totalSanitationSystemsConstructed +=

                                                                        constructedObjects[

                                                                            key

                                                                        ];

                                                                }

                                                            }

                                                            console.log(

                                                                "SanitaionSystemsStatus Data: ",

                                                                finalresult.sanitationSystemsStatusPieChart

                                                            );

                                                            console.log(

                                                                "TotalSanitaionSystemsConstructed: ",

                                                                finalresult.totalSanitationSystemsConstructed

                                                            );

                                                            connection.query(

                                                                `select count(Panchayat) as panchayatcount from Locations where District = '${req.body.District}'`,

                                                                function (

                                                                    err,

                                                                    result

                                                                ) {

                                                                    if (err) {

                                                                        res.status(

                                                                            500

                                                                        ).json({

                                                                            message: res.toString(),

                                                                        });

                                                                    }

                                                                    finalresult.totalPanchayatsCovered =

                                                                        result[0].panchayatcount;

                                                                    connection.query(

                                                                        `Select avg(u.Usages) as averageusagepermonth from WaterUsages u join WaterSources w on w.WSID = u.WSID where w.Pincode = ANY (SELECT Pincode FROM Locations

                                                                    WHERE Locations.Pincode = w.Pincode and Locations.District = '${req.body.District}')`,

                                                                        function (

                                                                            err,

                                                                            result

                                                                        ) {

                                                                            if (

                                                                                err

                                                                            ) {

                                                                                res.status(

                                                                                    500

                                                                                ).json(

                                                                                    {

                                                                                        message: res.toString(),

                                                                                    }

                                                                                );

                                                                            }

                                                                            console.log(

                                                                                "Average Usage Per Month",

                                                                                result[0]

                                                                            );

                                                                            finalresult.averageWaterUsagePerMonth =

                                                                                result[0]

                                                                                    .averageusagepermonth ===

                                                                                null

                                                                                    ? 0

                                                                                    : result[0]

                                                                                            .averageusagepermonth;

                                                                            if (

                                                                                finalresult.averageWaterUsagePerMonth !==

                                                                                    0 &&

                                                                                finalresult.totalPeopleBenefitted !==

                                                                                    0

                                                                            )

                                                                                finalresult.averageWaterUsagePerPersonPerMonth =

                                                                                    finalresult.averageWaterUsagePerMonth /

                                                                                    finalresult.totalPeopleBenefitted;

                                                                            res.status(

                                                                                200

                                                                            ).json(

                                                                                {

                                                                                    message:

                                                                                        "Statistics Fetched Successfully!",

                                                                                    finalresult,

                                                                                }

                                                                            );

                                                                        }

                                                                    );

                                                                }

                                                            );

                                                        }

                                                    );

                                                }

                                            );

                                        }

                                    );

                                }

                            );

                        }

                    );

                }

            );

        }

    );

});

module.exports = router;

## Front end code

import React, { Component } from "react";

import axios from "axios";

import Snackbar from "@material-ui/core/Snackbar";

import Typography from "@material-ui/core/Typography";

import SnackbarContent from "@material-ui/core/SnackbarContent";

import IconButton from "@material-ui/core/IconButton";

import CloseIcon from "@material-ui/icons/Close";

import Card from "@material-ui/core/Card";

import InputLabel from "@material-ui/core/InputLabel";

import MenuItem from "@material-ui/core/MenuItem";

import FormControl from "@material-ui/core/FormControl";

import Select from "@material-ui/core/Select";

import CardContent from "@material-ui/core/CardContent";

import Button from "@material-ui/core/Button";

import Grid from "@material-ui/core/Grid";

import {

  ResponsiveContainer,

  BarChart,

  CartesianGrid,

  XAxis,

  YAxis,

  Tooltip,

  Legend,

  Bar,

  PieChart,

  Pie,

} from "recharts";

import { withStyles } from "@material-ui/core/styles";

import Header from "../components/Header";

async function fetchAnalytics() {

  let resdata = [];

  await axios

    .get(`http://localhost:3001/api/analytics/getallstatistics`)

    .then((res) => {

      resdata = res.data.finalresult;

    })

    .catch((err) => {

      console.log(err);

    });

  return resdata;

}

async function fetchDistricts() {

  let resdata = [];

  await axios

    .get(`http://localhost:3001/api/location/getalldistricts`)

    .then((res) => {

      resdata = res.data.result;

    })

    .catch((err) => {

      console.log(err);

    });

  return resdata;

}

async function fetchPanchayats() {

  let resdata = [];

  await axios

    .get(`http://localhost:3001/api/location/getallpanchayats`)

    .then((res) => {

      resdata = res.data.result;

    })

    .catch((err) => {

      console.log(err);

    });

  return resdata;

}

const styles = (theme) => ({

  seeMore: {

    marginTop: theme.spacing(3),

  },

  container: {

    paddingTop: theme.spacing(4),

    paddingBottom: theme.spacing(4),

  },

  paper: {

    padding: theme.spacing(2),

    display: "flex",

    flexDirection: "column",

    alignItems: "center",

    margin: 10,

  },

  form: {

    width: "100%",

    marginTop: theme.spacing(1),

  },

  gridpaper: {

    textAlign: "center",

    backgroundColor: "#6A8FD4",

    color: "white",

  },

  griditem: {

    padding: theme.spacing(2),

    textAlign: "center",

    display: "flex",

    flexDirection: "column",

    alignItems: "center",

  },

  form: {

    width: "100%",

    marginTop: theme.spacing(1),

  },

  submit: {

    margin: theme.spacing(3, 0, 2),

  },

});

class Dashboard extends React.Component {

  state = {

    availableDistricts: [],

    availablePanchayats: [],

    districtSelect: "",

    panchayatSelect: "",

    previousSelect: "",

    snackbarMessage: "",

    snackbarColor: "",

    selectedType: "",

    open: false,

    averageWaterUsagePerMonth: 0,

    averageWaterUsagePerPersonPerMonth: 0,

    totalDistictsCovered: 0,

    totalDistictsCovered: 0,

    totalPanchayatsCovered: 0,

    totalFamiliesBenefitted: 0,

    totalPeopleBenefitted: 0,

    totalDonations: 0,

    totalExpenditure: 0,

    totalWaterSourcesConstructed: 0,

    totalSanitationSystemsConstructed: 0,

    totalConstuctedBarGraph: [],

    waterSourcesStatusPieChart: [],

    sanitationSystemsStatusPieChart: [],

  };

  async componentDidMount() {

    let analytics = await fetchAnalytics();

    this.setState({

      averageWaterUsagePerMonth: analytics.averageWaterUsagePerMonth,

      averageWaterUsagePerPersonPerMonth: analytics.averageWaterUsagePerPersonPerMonth.toFixed(

        2

      ),

      totalDistictsCovered: analytics.totalDistictsCovered,

      totalPanchayatsCovered: analytics.totalPanchayatsCovered,

      totalFamiliesBenefitted: analytics.totalFamiliesBenefitted,

      totalPeopleBenefitted: analytics.totalPeopleBenefitted,

      totalDonations: analytics.totalDonations,

      totalExpenditure: analytics.totalExpenditure,

      totalWaterSourcesConstructed: analytics.totalWaterSourcesConstructed,

      totalSanitationSystemsConstructed:

        analytics.totalSanitationSystemsConstructed,

      totalConstuctedBarGraph: analytics.totalConstuctedBarGraph,

      waterSourcesStatusPieChart: analytics.waterSourcesStatusPieChart,

      sanitationSystemsStatusPieChart:

        analytics.sanitationSystemsStatusPieChart,

    });

    let districts = await fetchDistricts();

    let panchayats = await fetchPanchayats();

    this.setState({

      availableDistricts: districts,

      availablePanchayats: panchayats,

    });

  }

  handleSubmit = async (e) => {

    e.preventDefault();

    e.persist();

    let ev = e;

    if (this.state.panchayatSelect.length !== 0) {

      if (this.state.previousSelect === this.state.panchayatSelect) {

        return;

      }

      await axios

        .post(`http://localhost:3001/api/analytics/getpanchayatstatistics`, {

          Pincode: this.state.panchayatSelect,

        })

        .then(async (res) => {

          let analytics = res.data.finalresult;

          console.log(analytics);

          this.setState({

            averageWaterUsagePerMonth: analytics.averageWaterUsagePerMonth,

            averageWaterUsagePerPersonPerMonth: analytics.averageWaterUsagePerPersonPerMonth.toFixed(

              2

            ),

            totalFamiliesBenefitted: analytics.totalFamiliesBenefitted,

            totalPeopleBenefitted: analytics.totalPeopleBenefitted,

            totalExpenditure: analytics.totalExpenditure,

            totalWaterSourcesConstructed:

              analytics.totalWaterSourcesConstructed,

            totalSanitationSystemsConstructed:

              analytics.totalSanitationSystemsConstructed,

            totalConstuctedBarGraph: analytics.totalConstuctedBarGraph,

            waterSourcesStatusPieChart: analytics.waterSourcesStatusPieChart,

            sanitationSystemsStatusPieChart:

              analytics.sanitationSystemsStatusPieChart,

            snackbarMessage: res.data.message,

            open: true,

            previousSelect: this.state.panchayatSelect,

            selectedType: "panchayat",

            snackbarColor: "green",

          });

          ev.target.reset();

        })

        .catch((err) => {

          this.setState({

            ...this.state,

            open: true,

            snackbarMessage: err.response.data.message,

            snackbarColor: "red",

          });

        });

    } else if (this.state.districtSelect.length !== 0) {

      if (this.state.previousSelect === this.state.districtSelect) {

        return;

      }

      await axios

        .post(`http://localhost:3001/api/analytics/getdistrictstatistics`, {

          District: this.state.districtSelect,

        })

        .then(async (res) => {

          let analytics = res.data.finalresult;

          console.log(analytics);

          this.setState({

            averageWaterUsagePerMonth: analytics.averageWaterUsagePerMonth,

            averageWaterUsagePerPersonPerMonth: analytics.averageWaterUsagePerPersonPerMonth.toFixed(

              2

            ),

            totalFamiliesBenefitted: analytics.totalFamiliesBenefitted,

            totalPeopleBenefitted: analytics.totalPeopleBenefitted,

            totalPanchayatsCovered: analytics.totalPanchayatsCovered,

            totalExpenditure: analytics.totalExpenditure,

            totalWaterSourcesConstructed:

              analytics.totalWaterSourcesConstructed,

            totalSanitationSystemsConstructed:

              analytics.totalSanitationSystemsConstructed,

            totalConstuctedBarGraph: analytics.totalConstuctedBarGraph,

            waterSourcesStatusPieChart: analytics.waterSourcesStatusPieChart,

            sanitationSystemsStatusPieChart:

              analytics.sanitationSystemsStatusPieChart,

            snackbarMessage: res.data.message,

            open: true,

            previousSelect: this.state.districtSelect,

            selectedType: "district",

            snackbarColor: "green",

          });

          ev.target.reset();

        })

        .catch((err) => {

          console.log(err);

          this.setState({

            ...this.state,

            open: true,

            snackbarMessage: err.response.data.message,

            snackbarColor: "red",

          });

        });

    } else {

      if (this.state.previousSelect === "") {

        return;

      }

      let analytics = await fetchAnalytics();

      this.setState({

        averageWaterUsagePerMonth: analytics.averageWaterUsagePerMonth,

        averageWaterUsagePerPersonPerMonth: analytics.averageWaterUsagePerPersonPerMonth.toFixed(

          2

        ),

        totalDistictsCovered: analytics.totalDistictsCovered,

        totalPanchayatsCovered: analytics.totalPanchayatsCovered,

        totalFamiliesBenefitted: analytics.totalFamiliesBenefitted,

        totalPeopleBenefitted: analytics.totalPeopleBenefitted,

        totalDonations: analytics.totalDonations,

        totalExpenditure: analytics.totalExpenditure,

        totalWaterSourcesConstructed: analytics.totalWaterSourcesConstructed,

        totalSanitationSystemsConstructed:

          analytics.totalSanitationSystemsConstructed,

        totalConstuctedBarGraph: analytics.totalConstuctedBarGraph,

        waterSourcesStatusPieChart: analytics.waterSourcesStatusPieChart,

        sanitationSystemsStatusPieChart:

          analytics.sanitationSystemsStatusPieChart,

        snackbarMessage: "Fetched Analytics Successfully!",

        open: true,

        selectedType: "",

        previousSelect: "",

        snackbarColor: "green",

      });

    }

  };

  handleClose = (event, reason) => {

    if (reason === "clickaway") {

      return;

    }

    this.setState({ ...this.state, open: false });

  };

  renderBarGraph() {

    if (this.state.totalConstuctedBarGraph.length !== 0) {

      return (

        <ResponsiveContainer width="70%" height={200}>

          <BarChart data={this.state.totalConstuctedBarGraph}>

            <CartesianGrid strokeDasharray="3 3" />

            <XAxis dataKey="name" />

            <YAxis />

            <Tooltip />

            <Legend />

            <Bar dataKey="Water Sources" fill="#8884d8" />

            <Bar dataKey="Sanitation Systems" fill="#82ca9d" />

          </BarChart>

        </ResponsiveContainer>

      );

    } else {

      return (

        <Typography style={{ margin: 20 }} component="h3" variant="subtitle1">

          No Data in Bar Graph

        </Typography>

      );

    }

  }

  renderPieChart(data) {

    if (data.length !== 0) {

      return (

        <PieChart width={400} height={270}>

          <Legend />

          <Tooltip />

          <Pie

            data={data}

            dataKey="value"

            nameKey="name"

            cx="50%"

            cy="50%"

            outerRadius={80}

            fill="#82ca9d"

            label

          />

        </PieChart>

      );

    } else {

      return (

        <Typography style={{ margin: 20 }} component="h3" variant="subtitle1">

          No Data in Pie Chart

        </Typography>

      );

    }

  }

  renderDistrict() {

    const { classes } = this.props;

    if (this.state.selectedType === "") {

      return (

        <Grid item>

          <Card className={classes.gridpaper}>

            <CardContent>

              <Typography variant="h6" component="h2">

                Districts Covered

              </Typography>

              <Typography variant="h6" component="h2">

                {this.state.totalDistictsCovered}

              </Typography>

            </CardContent>

          </Card>

        </Grid>

      );

    }

  }

  renderPanchayats() {

    const { classes } = this.props;

    if (this.state.selectedType !== "panchayat") {

      return (

        <Grid item>

          <Card className={classes.gridpaper}>

            <CardContent>

              <Typography variant="h6" component="h2">

                Panchayats Covered

              </Typography>

              <Typography variant="h6" component="h2">

                {this.state.totalPanchayatsCovered}

              </Typography>

            </CardContent>

          </Card>

        </Grid>

      );

    }

  }

  renderDonations() {

    const { classes } = this.props;

    if (this.state.selectedType === "") {

      return (

        <Grid item>

          <Card className={classes.gridpaper}>

            <CardContent>

              <Typography variant="h6" component="h2">

                Donations

              </Typography>

              <Typography variant="h6" component="h2">

                Rs.{this.state.totalDonations}

              </Typography>

            </CardContent>

          </Card>

        </Grid>

      );

    }

  }

  render() {

    const { classes } = this.props;

    return (

      <React.Fragment>

        <Header />

        <Snackbar

          open={this.state.open}

          autoHideDuration={6000}

          onClose={this.handleClose}

        >

          <SnackbarContent

            style={{

              backgroundColor: this.state.snackbarColor,

            }}

            action={

              <React.Fragment>

                <IconButton

                  size="small"

                  aria-label="close"

                  color="inherit"

                  onClick={this.handleClose}

                >

                  <CloseIcon fontSize="small" />

                </IconButton>

              </React.Fragment>

            }

            message={

              <span id="client-snackbar">{this.state.snackbarMessage}</span>

            }

          />

        </Snackbar>

        <div className={classes.paper}>

          <Typography style={{ margin: 10 }} component="h1" variant="h5">

            Analytics

          </Typography>

          <form

            className={classes.form}

            style={{ marginBottom: 40, width: "60%" }}

            onSubmit={this.handleSubmit}

          >

            <Grid container justify="center" spacing={3}>

              <Grid item xs={12} sm={4}>

                <FormControl

                  variant="outlined"

                  fullWidth

                  className={classes.form}

                >

                  <InputLabel id="District-Label">District</InputLabel>

                  <Select

                    labelId="District-Label"

                    id="District"

                    label="District"

                    name="District"

                    variant="outlined"

                    value={this.state.districtSelect}

                    onOpen={(e) => {

                      if (this.state.availableDistricts.length === 0)

                        this.setState({

                          open: true,

                          snackbarMessage: "Districts Unavailable!",

                          snackbarColor: "red",

                        });

                    }}

                    onChange={(e) => {

                      this.setState({ districtSelect: e.target.value });

                      axios

                        .post(

                          `http://localhost:3001/api/location/getcorrespondingpanchayats`,

                          { District: e.target.value }

                        )

                        .then((res) => {

                          let panchayats = res.data.result;

                          this.setState({ availablePanchayats: panchayats });

                        })

                        .catch((err) => {

                          console.log(err);

                        });

                    }}

                    fullWidth

                  >

                    {this.state.availableDistricts.map((pin) => (

                      <MenuItem key={pin.District} value={pin.District}>

                        {pin.District}

                      </MenuItem>

                    ))}

                  </Select>

                </FormControl>

              </Grid>

              <Grid item xs={12} sm={4}>

                <FormControl

                  variant="outlined"

                  fullWidth

                  className={classes.form}

                >

                  <InputLabel id="Panchayat-Label">Panchayat</InputLabel>

                  <Select

                    labelId="Panchayat-Label"

                    id="Panchayat"

                    label="Panchayat"

                    name="Panchayat"

                    variant="outlined"

                    value={this.state.panchayatSelect}

                    onOpen={(e) => {

                      if (this.state.availablePanchayats.length === 0)

                        this.setState({

                          open: true,

                          snackbarMessage: "Panchayats Unavailable!",

                          snackbarColor: "red",

                        });

                    }}

                    onChange={(e) => {

                      this.setState({ panchayatSelect: e.target.value });

                    }}

                    fullWidth

                  >

                    {this.state.availablePanchayats.map((pin) => (

                      <MenuItem key={pin.Pincode} value={pin.Pincode}>

                        {pin.Panchayat}

                      </MenuItem>

                    ))}

                  </Select>

                </FormControl>

              </Grid>

              <Grid item xs={6} sm={2}>

                <Button

                  type="button"

                  fullWidth

                  variant="contained"

                  color="primary"

                  className={classes.submit}

                  onClick={async (e) => {

                    e.preventDefault();

                    if (this.state.previousSelect === "") {

                      let panchayats = await fetchPanchayats();

                      this.setState({

                        availablePanchayats: panchayats,

                        districtSelect: "",

                        panchayatSelect: "",

                      });

                      return;

                    }

                    let analytics = await fetchAnalytics();

                    let districts = await fetchDistricts();

                    let panchayats = await fetchPanchayats();

                    this.setState({

                      averageWaterUsagePerMonth:

                        analytics.averageWaterUsagePerMonth,

                      averageWaterUsagePerPersonPerMonth: analytics.averageWaterUsagePerPersonPerMonth.toFixed(

                        2

                      ),

                      totalDistictsCovered: analytics.totalDistictsCovered,

                      totalPanchayatsCovered: analytics.totalPanchayatsCovered,

                      totalFamiliesBenefitted:

                        analytics.totalFamiliesBenefitted,

                      totalPeopleBenefitted: analytics.totalPeopleBenefitted,

                      totalDonations: analytics.totalDonations,

                      totalExpenditure: analytics.totalExpenditure,

                      totalWaterSourcesConstructed:

                        analytics.totalWaterSourcesConstructed,

                      totalSanitationSystemsConstructed:

                        analytics.totalSanitationSystemsConstructed,

                      totalConstuctedBarGraph:

                        analytics.totalConstuctedBarGraph,

                      waterSourcesStatusPieChart:

                        analytics.waterSourcesStatusPieChart,

                      sanitationSystemsStatusPieChart:

                        analytics.sanitationSystemsStatusPieChart,

                      availableDistricts: districts,

                      availablePanchayats: panchayats,

                      snackbarMessage: "Fetched Analytics Successfully!",

                      open: true,

                      selectedType: "",

                      previousSelect: "",

                      snackbarColor: "green",

                      districtSelect: "",

                      panchayatSelect: "",

                    });

                  }}

                >

                  Reset

                </Button>

              </Grid>

              <Grid item xs={6} sm={2}>

                <Button

                  type="submit"

                  fullWidth

                  variant="contained"

                  color="primary"

                  className={classes.submit}

                >

                  Query!

                </Button>

              </Grid>

            </Grid>

          </form>

          <Grid container justify="center" spacing={3}>

            {this.renderDistrict()}

            {this.renderPanchayats()}

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    Families Benefitted

                  </Typography>

                  <Typography variant="h6" component="h2">

                    {this.state.totalFamiliesBenefitted}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    People Benefitted

                  </Typography>

                  <Typography variant="h6" component="h2">

                    {this.state.totalPeopleBenefitted}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

            {this.renderDonations()}

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    Expenditure

                  </Typography>

                  <Typography variant="h6" component="h2">

                    Rs.{this.state.totalExpenditure}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    Water Sources

                  </Typography>

                  <Typography variant="h6" component="h2">

                    {this.state.totalWaterSourcesConstructed}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    Sanitation Systems

                  </Typography>

                  <Typography variant="h6" component="h2">

                    {this.state.totalSanitationSystemsConstructed}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    Avg Water Usage Per Month

                  </Typography>

                  <Typography variant="h6" component="h2">

                    {this.state.averageWaterUsagePerMonth}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

            <Grid item>

              <Card className={classes.gridpaper}>

                <CardContent>

                  <Typography variant="h6" component="h2">

                    Avg Water Usage Per Person Per Month

                  </Typography>

                  <Typography variant="h6" component="h2">

                    {this.state.averageWaterUsagePerPersonPerMonth}

                  </Typography>

                </CardContent>

              </Card>

            </Grid>

          </Grid>

          <Typography

            style={{ margin: 30, marginTop: 50 }}

            component="h1"

            variant="h6"

          >

            Commissioning of Projects

          </Typography>

          {this.renderBarGraph()}

          <Grid container spacing={3}>

            <Grid item xs={12} sm={6} className={classes.griditem}>

              <Typography style={{ margin: 10 }} component="h1" variant="h6">

                Status of Water Sources

              </Typography>

              {this.renderPieChart(this.state.waterSourcesStatusPieChart)}

            </Grid>

            <Grid item xs={12} sm={6} className={classes.griditem}>

              <Typography style={{ margin: 10 }} component="h1" variant="h6">

                Status of Sanitation Systems

              </Typography>

              {this.renderPieChart(this.state.sanitationSystemsStatusPieChart)}

            </Grid>

          </Grid>

        </div>

      </React.Fragment>

    );

  }

}

export default withStyles(styles, { withTheme: true })(Dashboard);

# Conclusion

The goal of the project which was to develop a monitoring system for a Rural Water Supply and Sanitation System was met and the data collected from the model has been used to analyze various parameters and provide insights for the organization undertaking the project work in real life.

# References

**React Tutorial for Beginners - Programming with Mosh**

**Website: YouTube**

**Date of Visit: 01/10/2020 – 09/10/2020**

**URL:** [**https://www.youtube.com/watch?v=Ke90Tje7VS0**](https://www.youtube.com/watch?v=Ke90Tje7VS0)

**Example Live Project - Kerala Rural Water Supply and Sanitation Agency**

**Website: jalanidhi.kerala.gov.in**

**Date of Visit: 07/09/2020 – 09/10/2020**

**URL:** [**https://jalanidhi.kerala.gov.in/page/render/reference/Jalanidhi\_Analytics**](https://jalanidhi.kerala.gov.in/page/render/reference/Jalanidhi_Analytics)