RESOURCE MANAGEMENT IN A COMPANY

Name of the Team: Data Beta

Sno	**RollNo**	**Name**	**Role** **(Project leader, Designer, Developer, Test & Document)**
1	17364	UDHAY AAKASH R	Project leader
2	17348	SHRAVAN	Designer
3	17348	RAGUL S V	Developer
4	17324	GOKUL	Test & Document

TABLE OF CONTENTS

ABSTRACT AND PREVIEW FOR THE PROJECT

Introduce about the system:

A Company constitutes of resources such as financial resources, equipment and labor resources such as employees. The progress of a company mainly depends on the efficient usage of company's resources. The main aim of this system is to automate the process of managing the resources wherein the process of allocation of tangible resources to the employee is done in the most efficient way possible. The system ensures optimized utilization and uniform distribution of the resources. It saves time as it speeds up every aspect of the employee database management.

Describe the functionalities of the system:

This system stores and manages the details of the company such as the company's resources, location of those resources, projects, requirements of projects, employee details, their skills and the allotment of resources to projects using database and its queries.

There are two main functionalities of this system:

1)Analyzing employee's skills and allotting projects to selected employees based on his/her skill.

2)Allocation of resources to project based on the requirements of the project (Number of days to complete), priority of the project, deadlines of the project and availability of resources.

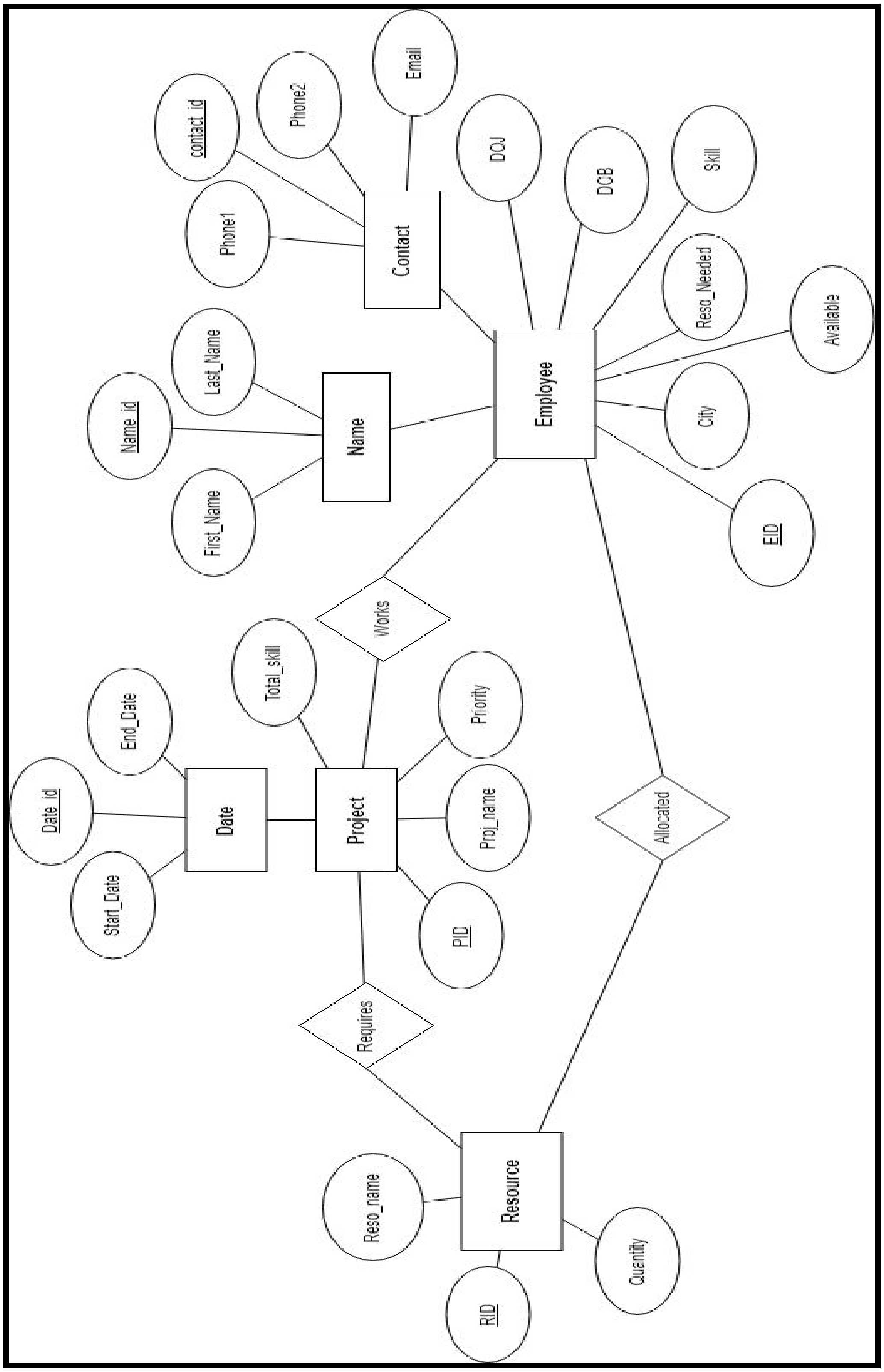
Describe the modules in the system:

- Interface creation for master table management.
- Storing Employee details, Resource information and Project Information in the database.
- Allocating resources to projects based on number of days required to finish the project, start and end date of the project(deadlines) and priority of the project.
- Project allocation based on skill of employee.

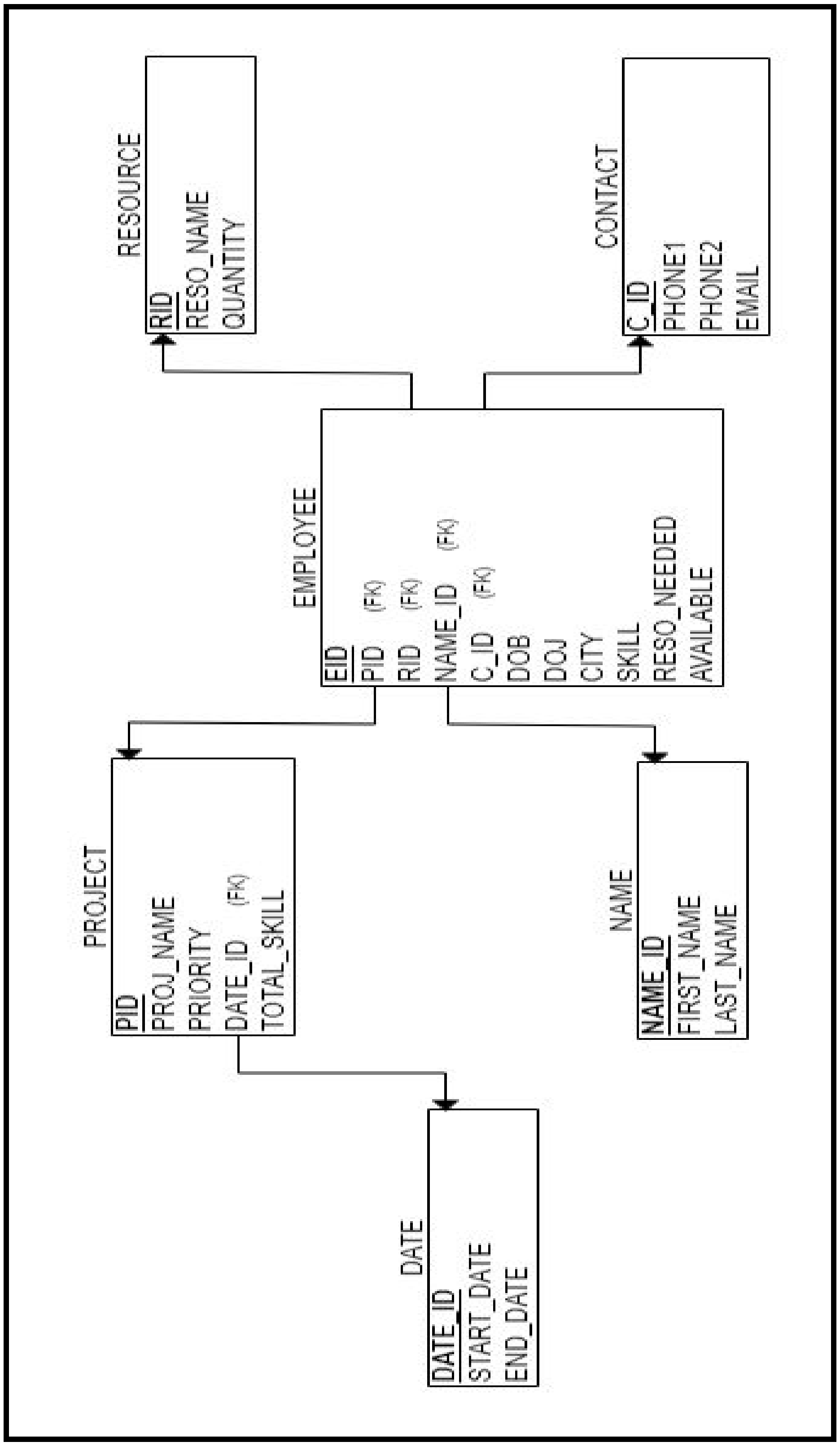
Describe the benefits of the system:

- Reducing Downtime of the resources
- Improving profit margin for the company
- Effective Management of Time
- Proper Utilization of Resources

ER DIAGRAM


Contact
contact id
Phone1
Phone2
Email
Employee
DOJ
DOB
Skill
Reso_Needed
Available
City
EID
Name
Name id
Last_Name
First_Name
Works
Project
Total_skill
Priority
Proj_name
PID
Date
Date id
End_Date
Start_Date
Requires
Allocated
Resource
Reso_name
RID
Quantity

NORMALIZED SCHEMA



NORMALIZATION

Relation with all the attributes:
R(eid, name_id, first_name, last_name, c_id, phone1, phone2, emailid, dob, doj, city, skill, reso_needed, available, rid, reso_name, quantity, pid, proj_name, priority, date_id, start_date, end_date, total_skill)

Functional Dependencies:

eid → name_id, c_id, dob, doj, city, skill
name_id → first_name, last_name
c_id → phone1, phone2, emailid
eid → pid
pid → proj_name, priority, date_id, total_skill
date_id → start_date, end_date
eid → rid
rid → reso_name, quantity

Candidate Key: eid

1-NF:
The table is already in 1-NF since there are no attributes with more than one value.

2-NF:
The table is already in 2-NF since there are no partial dependencies.

3-NF:
The relation R has transitive dependency. Hence in order for the relations to be in 3-NF, the relation R is further split into six relations.

Initial Relation	Functional Dependencies	Normalized Relation
R(eid, name_id, first_name, last_name, c_id, phone1, phone2, emailid, dob, doj, city, skill, reso_needed, available, rid, reso_name, quantity, pid, proj_name, priority, date_id, start_date, end_date, total_skill)	eid → name_id name_id → first_name, last_name	R1(name_id, first_name. last_name)

R(eid, c_id, phone1, phone2, emailid, dob, doj, city, skill, reso_needed, available, rid, reso_name, quantity, pid, proj_name, priority, date_id, start_date, end_date, total_skill)	eid → c_id c_id → phone1, phone2, emailid	R2(c_id, phone1, phone2, emailid)
R(eid, dob, doj, city, skill, reso_needed, available, rid, reso_name, quantity, pid, proj_name, priority, date_id, start_date, end_date, total_skill)	eid → pid pid → date_id date_id → start_date, end_date	R3(date_id, start_date, end_date)
R(eid, dob, doj, city, skill, reso_needed, available, rid, reso_name, quantity, pid, proj_name, priority, total_skill)	eid → pid pid → proj_name, priority, date_id, total_skill	R4(pid, proj_name, priority, date_id, total_skill)
R(eid, dob, doj, city, skill, reso_needed, available, rid, reso_name, quantity)	eid → rid rid → reso_name, quantity	R5(rid, reso_name, quantity)
R(eid, pid, rid, name_id, c_id, dob, doj, city, skill, reso_needed, available)	-	R6(eid, pid, rid, name_id, c_id, dob, doj, city, skill, reso_needed, available)

BCNF:

There are no non-trivial functional dependencies of attributes on anything other than a superset of a candidate key. Therefore all the decomposed relation which are in 3-NF are also in BC-NF.

Relations after normalising upto BCNF:

R1(name_id, first_name. last_name)
R2(c_id, phone1, phone2, emailid)
R3(date_id, start_date, end_date)
R4(pid, proj_name, priority, date_id, total_skill)
R5(rid, reso_name, quantity)
R6(eid, pid, rid, name_id, c_id, dob, doj, city, skill, reso_needed, available)

TABLE CREATION

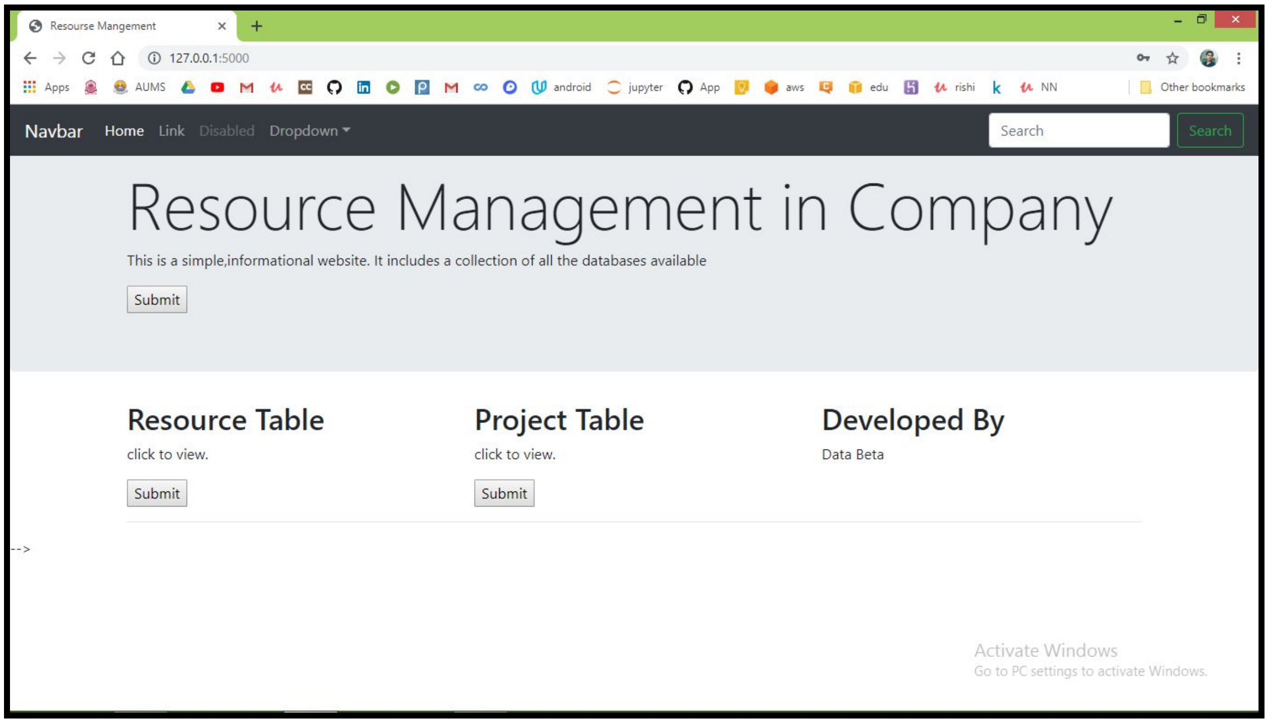
A total of 6 tables are created corresponding to the normalized relations. The name of the tables are:

Employee (eid, pid, rid, name_id, c_id, dob, doj, city, skill, reso_needed, available)
Resource (rid, reso_name, quantity)
Project (pid, proj_name, priority, date_id, total_skill)
Date (date_id, start_date, end_date)
Name (name_id, first_name. last_name)
Contact (c_id, phone1, phone2, emailid)

Code for creating tables:

```
create table EMPLOYEE(eid varchar(20), pid varchar(20), rid varchar(20), name_id varchar(20),
c_id varchar(20), dob date, doj date, city varchar(20), skill varchar(20), reso_needed varchar(20),
available numeric(1,0),
primary key(eid), foreign key(pid) references PROJECT,
foreign key(rid) references RESOURCE,
foreign key(name_id) references NAME,
foreign key(c_id) references CONTACT);

create table RESOURCE(r_id varchar(20), reso_name varchar(20), quantity numeric(10,0),
primary key(r_id));

create table PROJECT (p_id varchar(20), proj_name varchar(20), priority numeric(3,0), date_id
varchar(20), total_skill varchar(20),
primary key(p_id),
foreign key(date_id) references DATE);

create table DATE(date_id varchar(20) NOT NULL, start_date date, end_date date,
primary key(date_id));

create table NAME(name_id varchar(20), first_name varchar(20), last_name varchar(20),
primary key(name_id));

create table CONTACTS (c_id varchar(20), phone1 numeric(10,0), phone2 numeric(10,0),
emailid varchar(20),
primary key(c_id));
```

UI DESIGN SCREENSHOTS



Resource Management in Company

This is a simple,informational website. It includes a collection of all the databases available

Resource Table

click to view.

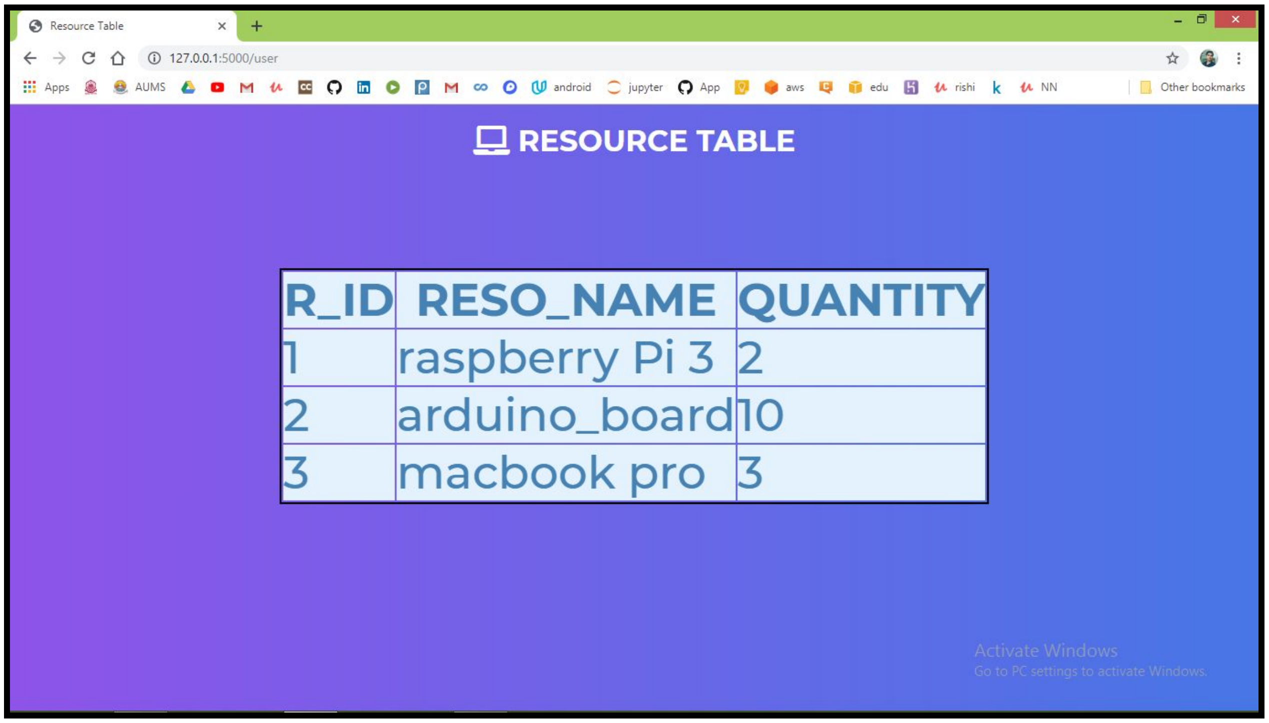
Project Table

click to view.

Developed By

Data Beta

-->



R_ID	RESO_NAME	QUANTITY
1	raspberry Pi 3	2
2	arduino_board	10
3	macbook pro	3

CONNECTIVITY CODE IN FLASK

```
from flask import Flask ,render_template,request
from flaskext.mysql import MySQL

app = Flask(__name__)
mysql = MySQL()
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'newrootpassword'
app.config['MYSQL_DATABASE_DB'] = 'dbms_pro'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)

@app.route('/',methods=['GET','POST'])
def index():
        if request.method=='POST':
                userDetails=request.form
                name=userDetails['name']
                email=userDetails['email']
                if name=='ragul_sv@hotmail.com' and email=='******':
                        return render_template('user.html')
                else:
                        return 'failed'

        elif request.method=='GET':
                return render_template('index.html')

@app.route('/user',methods=['GET','POST'])
def users():
        if request.method=='POST':
                conn = mysql.connect()
                cursor =conn.cursor()
                resultvalue=cursor.execute("SELECT * FROM resource7")
                print('hi dude')
                if resultvalue > 0:
                        userDetails=cursor.fetchall();
                        print (userDetails)
                return render_template('resource table.html',userDetails=userDetails)
        elif request.method=='GET':
                return render_template('user.html')

@app.route('/details',methods=['GET','POST'])
```

```
def details():
    print('hi')
    if request.method=='POST':
        conn = mysql.connect()
        cursor =conn.cursor()
        resultvalue=cursor.execute("SELECT * FROM resource1")
        print('hi dude')
        if resultvalue > 0:
            userDetails=cursor.fetchall();
            print (userDetails)

    elif request.method=='GET':
        conn = mysql.connect()
        cursor =conn.cursor()
        resultvalue=cursor.execute("SELECT * FROM employee")
        print('hi dude')
        if resultvalue > 0:
            userDetails=cursor.fetchall();
            print (userDetails)
        return render_template('employee table.html',userDetails=userDetails)

@app.route('/proj',methods=['GET','POST'])
def uusers():
    if request.method=='POST':
        conn = mysql.connect()
        cursor =conn.cursor()
        resultvalue=cursor.execute("SELECT * FROM project1")
        print('hi dude')
        if resultvalue > 0:
            userDetails=cursor.fetchall();
            print (userDetails)
        return render_template('proj.html',userDetails=userDetails)
    elif request.method=='GET':
        conn = mysql.connect()
        cursor =conn.cursor()
        resultvalue=cursor.execute("SELECT * FROM project7")
        print('hi dude')
        if resultvalue > 0:
            userDetails=cursor.fetchall();
            print (userDetails)
        return render_template('project.html',userDetails=userDetails)
if __name__ == '__main__':
    app.run(debug=True)
```

SAMPLE CODE FOR UI PAGE

USER.HTML

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Resourse Mangement</title>
    <!-- Bootstrap core CSS -->
    <link href="../static/bootstrap.min.css" rel="stylesheet">
    <!-- Custom styles for this template -->
<!--     <link href="jumbotron.css" rel="stylesheet">
 -->  </head>
  <body>
    <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
      <a class="navbar-brand" href="#">Navbar</a>
      <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarsExampleDefault">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item active">
            <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Link</a>
          </li>
          <li class="nav-item">
            <a class="nav-link disabled" href="#">Disabled</a>
          </li>
          <li class="nav-item dropdown">
            <a class="nav-link dropdown-toggle" href="http://example.com" id="dropdown01"
data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Dropdown</a>
            <div class="dropdown-menu" aria-labelledby="dropdown01">
              <a class="dropdown-item" href="#">Action</a>
              <a class="dropdown-item" href="#">Another action</a>
              <a class="dropdown-item" href="#">Something else here</a>
            </div>
          </li>
        </ul>
        <form class="form-inline my-2 my-lg-0">
          <input class="form-control mr-sm-2" type="text" placeholder="Search" aria-label="Search">
          <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
        </form>
      </div>
    </nav>
    <main role="main">
      <!-- Main jumbotron for a primary marketing message or call to action -->
      <div class="jumbotron">
        <div class="container">
          <h1 class="display-3">Resource Management in Company</h1>
          <p>This is a simple,informational website. It includes a collection of all the databases
available</p>
            <form action="/details" method="GET">
            <input type="submit"/>
            </form>        </div>
      </div>
      <div class="container">
        <!-- Example row of columns -->
        <div class="row">
          <div class="col-md-4">
            <h2>Resource Table</h2>
            <p>click to view. </p>
            <form action="/user" method="POST">
            <input type="submit"/>
            </form>
          </div>
          <div class="col-md-4">
            <h2>Project Table</h2>
            <p>click to view. </p>
            <form action="/proj" method="GET">
            <input type="submit"/>
            </form>
         </div>
          <div class="col-md-4">
            <h2>Developed By</h2>
            <p>Data Beta</p>
                 </div>
        </div>
        <hr>
      </div>  </main>  </body> </html>
```

RESOURCE TABLE.HTML

```
<!DOCTYPE html>
<html>
<head>
		<title>Resource Table</title>
		<link rel="stylesheet" type="text/css" href="../static/resource.css">
    <link href="https://fonts.googleapis.com/css?family=Montserrat:300,400,500,600,700&display=swap"
rel="stylesheet">
    <link rel="stylesheet" type="text/css"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.11.2/css/all.css">
</head>
<body>
  <div id="header">
    <h1><i class="fas fa-laptop"></i> RESOURCE TABLE</h1>
  </div>

		<table border=1>
    <tr id="container">
     <th>R_ID</th>
    <th>RESO_NAME</th>
    <th>QUANTITY</th>
    </tr>
  {% for user in userDetails %}
  <tr id="container">
    <td>{{user[0]}}</td>
    <td>{{user[1]}}</td>
    <td>{{user[2]}}</td>
  </tr>
  {% endfor %}
</table>
</body>
</html>
```

AUTOMATION CODE USING PL/SQL

```
set serveroutput on;

DECLARE
B number(12,2);
C number(12,2);
SKIL number(10,0);
SUM1 number(12,0);
qty number(10,0);
counter1 number(10,0);
counter2 number(10,0);
store1 number(10,0);
index1 number(10,0);
available1 number(10,0);
a1 varchar(20);
a2 varchar(20);
flag number(1,0);
BEGIN
select count(*) into B from PROJECT7;
select count(*) into C from employee;
flag:=0;
for counter in 1..B
loop
   select TOTAL_SKILL into SKIL from PROJECT7 where p_id=counter;
   SUM1:=0;
   for counter1 in 1..C
   loop
        select skill into store1 from employee where eid=counter1;
        select avaliable into available1 from employee where eid=counter1;
        if(available1=1) then
          SUM1:=SUM1+store1;
        end if;
        if ((SUM1 >= SKIL)AND(available1=1)) then
             index1:=counter1;
             flag:=1;
             EXIT;
        end if;
   end loop;

   if(flag=0) then
      dbms_output.put_line( 'id of project that cannot be allocated is = ' || to_char( counter )  );
```

```
        continue;
    end if;
      for counter2 in 1..index1
    loop
        select avaliable into available1 from employee where eid=counter2;
        if(available1=1) then
        update employee set pid=counter where eid=counter2;
        update employee set avaliable=0 where eid=counter2;
        select reso_needed into a1 from employee where eid=counter2;
        select r_id into a2 from resource7 where reso_name=a1;
        update employee set rid=a2 where eid=counter2;
        select quantity into qty from resource7 where r_id=a2;
        qty:=qty-1;
        update resource7 set quantity=qty where r_id=a2;
        end if
    end loop;
end loop;
end;
```

CONCLUSION

The above PL/SQL code is used to allocate project and resource to the employees. The available attribute is used to denote whether the employee is available or not(0 or 1). In the outer loop we are looping through the total skill attribute of each project.In the inner loop we are looping through skill attribute of employees and we are adding this skill into a sum variable until it is greater than or equal to the total skill of the project.

If the total skill of the project cannot be accomplished using all the skills of the employee, then that project cannot be scheduled.

So the allocation of resources and projects to the employees is done by updating their respective values of rid and pid using the counter values we used for iterating the loops. After the allocation is done, available attribute of each employee is set back to 0.

The quantity of each resource is also updated and are decremented by 1 after allocation.

Finally, the resource and projects are allocated to the employees of the company successfully.

REFERENCES:

https://stackoverflow.com/questions/9845102/using-mysql-in-flask
https://www.youtube.com/watch?v=6L3HNyXEais
https://www.youtube.com/watch?v=CedOasDoe-w