Szabist

Database Lab Project Report

Welfare Organization Management System

Hira Niaz Malik (2312113)
Alisha Anwar (2312105)

Project Overview

A comprehensive **database and API system** designed to manage operations for **Welfare** organization. The system tracks donations, volunteers, ambulances, orphanages, shelters, and other critical services.

System Features

Module	Key Functionality
Donation Management	Track cash/kind donations, donor history
Volunteer Coordination	Register, assign, and manage volunteers
Ambulance Services	Fleet status, emergency call tracking
Orphanage System	Child records, adoption processes
Shelter Management	Bed allocation, resident tracking
Graveyard Services	Burial plot management
Inventory Control	Medical supplies, food stock tracking

Tables / Attributes

Donor - donor_id , name , contact
Donations - donor_id , donation_id donation_date , donation_ammount
Orphanages - orphange_id , name , location
Orphans - orphan_id , name , age , orphange_id
Volunteers - volunteer_id , name , skill , service_id
Services - service_id , name
Ambulances - ambulance_id , license plate , status
Graveyard - graveyard_id , location , total_plots , status

Employees - employee_id , name , salary
Shelters - shelter_id , location , capacity , type
Inventory - item_id , name , quantity , category

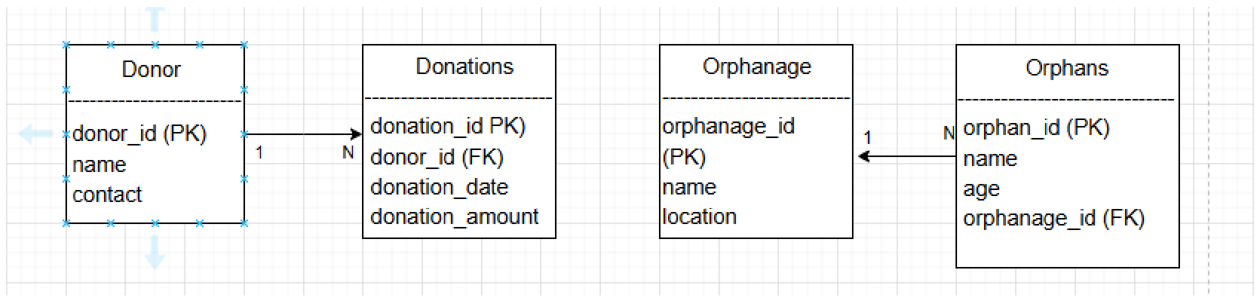
Relationships

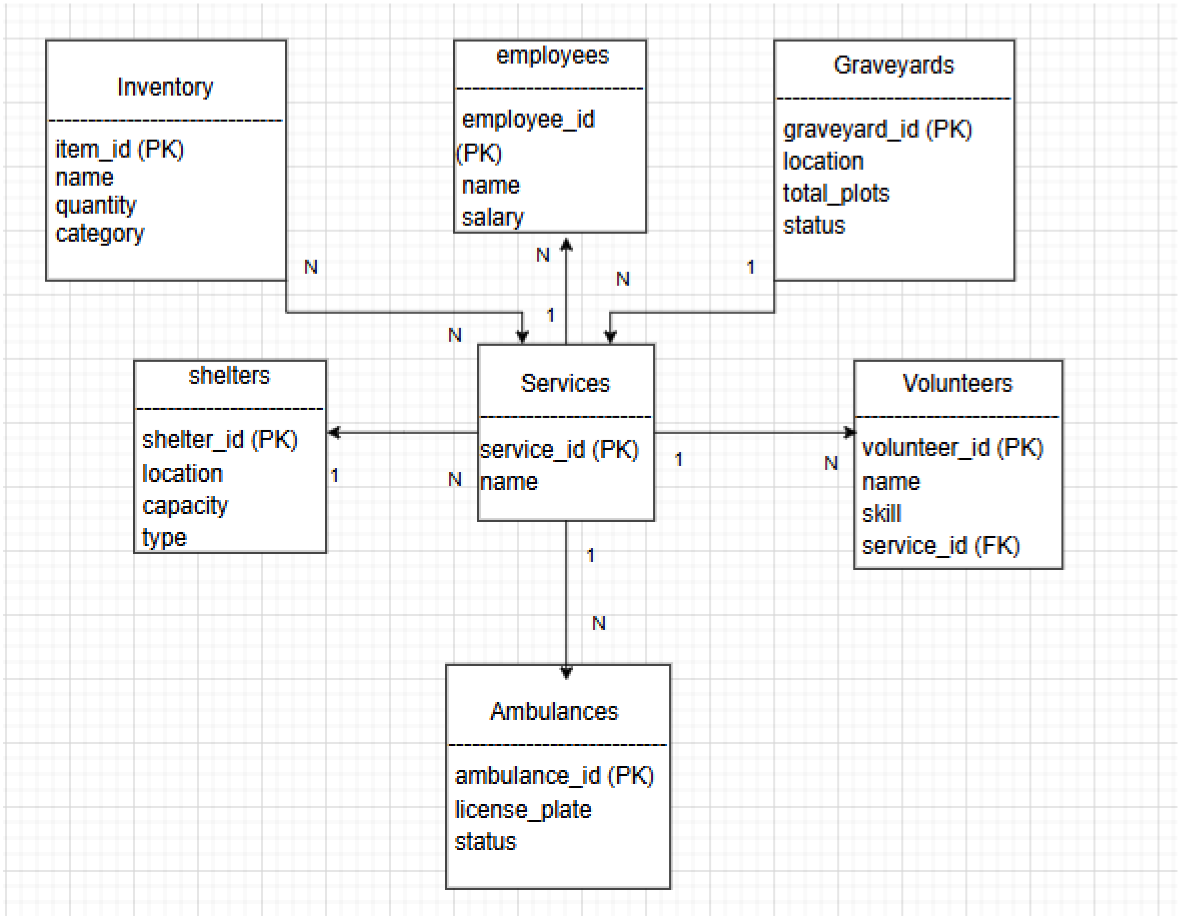
One-to-Many Relationships:
1. Donors → Donations (1 donor makes many donations)
2. Orphanages → Orphans (1 orphanage houses many orphans)
3. Services → Volunteers (1 service has many volunteers)
4. Services → Ambulances (1 service uses many ambulances)
5. Services → Graveyards (1 service manages many graveyards)
6. Services → Employees (1 service coordinates many employees)
7. Services → Shelters (1 service operates many shelters)

Many-to-Many Relationship:
8. Services ↔ Inventory (services need many items, items serve many services)
   - Requires junction table: Service_Inventory(service_id, item_id, quantity)

Foreign Key Locations:
- Donations stores donor_id
- Orphans stores orphanage_id
- Volunteers stores service_id
- Ambulances stores service_id
- Graveyards stores service_id
- Employees stores service_id
- Shelters stores service_id
- Service_Inventory stores both service_id and item_id

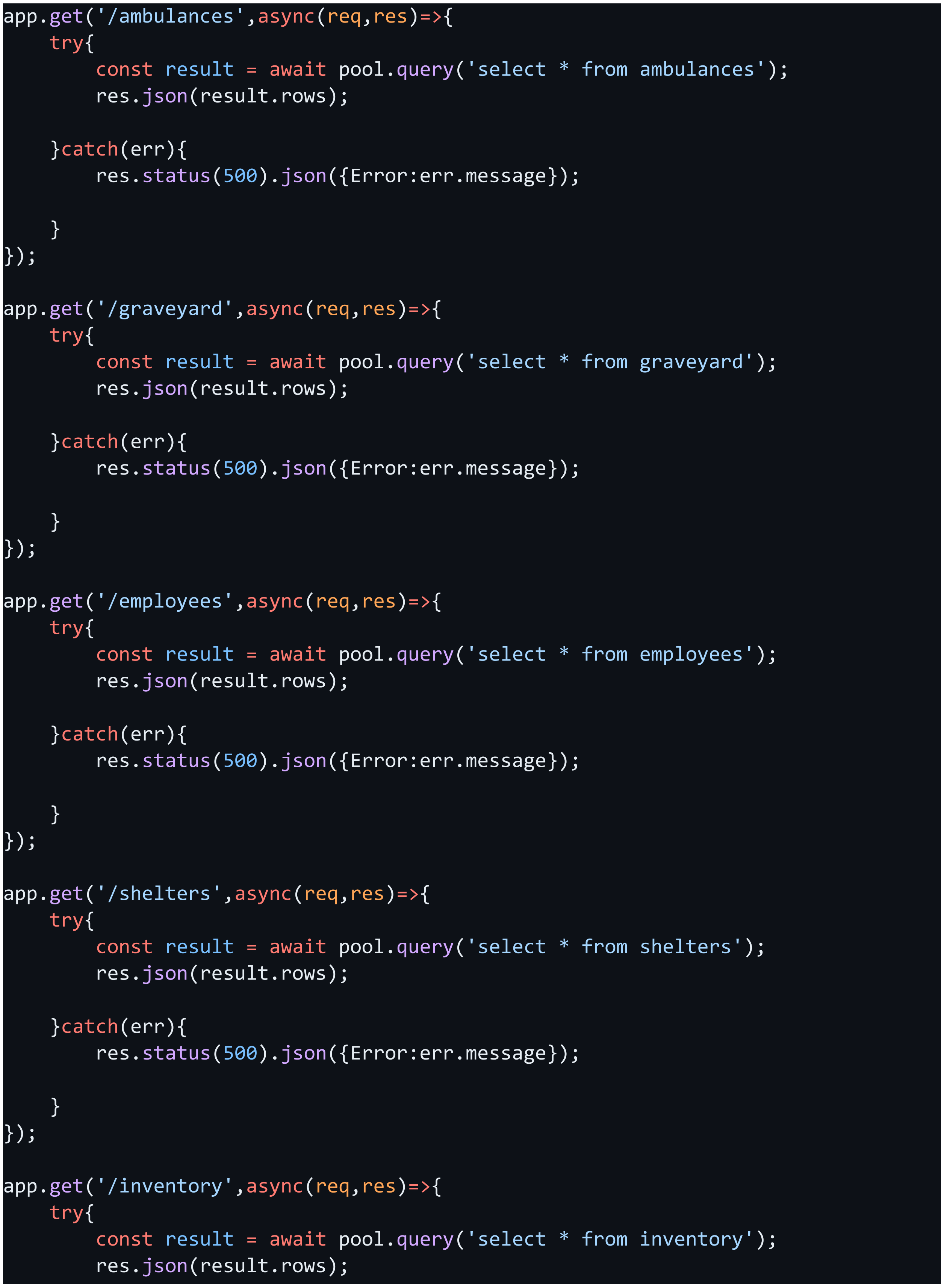
ERD





Get APIs

```
app.get('/donors',async (req,res)=>{
    try {
        const result = await pool.query('select * from donor');
        res.json(result.rows);
    } catch (err) {
        res.status(500).json({Error:err.message});
    }
});

app.get('/donations',async(req,res)=>{
    try{
        const result = await pool.query('select * from donations');
        res.json(result.rows);

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

app.get('/orphanges',async(req,res)=>{
    try{
        const result = await pool.query('select * from orphanages');
        res.json(result.rows);

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

    }
});

app.get('/orphans',async(req,res)=>{
    try{
        const result = await pool.query('select * from orphans');
        res.json(result.rows);

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

    }
});

app.get('/services',async(req,res)=>{
    try{
        const result = await pool.query('select * from services');
        res.json(result.rows);

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

    }
});

app.get('/volunteers',async(req,res)=>{
    try{
        const result = await pool.query('select * from volunteers');
        res.json(result.rows);

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

    }
});
```



```
app.get('/ambulances',async(req,res)=>{
    try{
        const result = await pool.query('select * from ambulances');
        res.json(result.rows);

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

    }
});

app.get('/graveyard',async(req,res)=>{
    try{
        const result = await pool.query('select * from graveyard');
        res.json(result.rows);

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

    }
});

app.get('/employees',async(req,res)=>{
    try{
        const result = await pool.query('select * from employees');
        res.json(result.rows);

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

    }
});

app.get('/shelters',async(req,res)=>{
    try{
        const result = await pool.query('select * from shelters');
        res.json(result.rows);

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

    }
});

app.get('/inventory',async(req,res)=>{
    try{
        const result = await pool.query('select * from inventory');
        res.json(result.rows);
    }catch(err){
        res.status(500).json({Error:err.message});

    }
});
```

Create APIs

```
app.post('/donorscreate', async (req, res) => {
  const { donor_id, name, contact } = req.body;
  try {
    await pool.query(
      'INSERT INTO donor (donor_id, name, contact) VALUES ($1, $2, $3)',
      [donor_id, name, contact]
    );
    res.status(201).json({ donor_id });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/ambulancescreate', async (req, res) => {
  const { ambulance_id, license_plate, status } = req.body;
  try {
    await pool.query(
      'INSERT INTO ambulances (ambulance_id, license_plate, status) VALUES ($1,
$2, $3)',
      [ambulance_id, license_plate, status]
    );
    res.status(201).json({ ambulance_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/donationscreate', async (req, res) => {
  const {donation_id, donor_id, donation_date,donation_amount} = req.body;
  try {
    await pool.query(
      'INSERT INTO donations (donation_id, donor_id,
donation_date,donation_amount) VALUES ($1, $2, $3, $4)',
      [donation_id, donor_id, donation_date,donation_amount]
    );
    res.status(201).json({ donation_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/employeescreate', async (req, res) => {
  const {employee_id,name,salary } = req.body;
  try {
    await pool.query(
      'INSERT INTO employees (employee_id,name,salary) VALUES ($1, $2, $3)',
      [employee_id,name,salary]
    );
    res.status(201).json({employee_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/graveyardcreate', async (req, res) => {
  const {graveyard_id, location, total_plots,status} = req.body;
  try {
    await pool.query(
      'INSERT INTO graveyard (graveyard_id, location, total_plots,status) VALUES
($1, $2, $3, $4)',
      [graveyard_id, location, total_plots,status]
    );
    res.status(201).json({ graveyard_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/inventorycreate', async (req, res) => {
  const {item_id, name,quantity,category} = req.body;
  try {
    await pool.query(
      'INSERT INTO inventory (item_id, name,quantity,category) VALUES ($1, $2,
$3, $4)',
      [item_id, name,quantity,category]
    );
    res.status(201).json({item_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/orphangescreate', async (req, res) => {
  const {orphanage_id,name,location } = req.body;
  try {
    await pool.query(
      'INSERT INTO orphanages (orphanage_id,name,location) VALUES ($1, $2, $3)',
      [orphanage_id,name,location]
    );
    res.status(201).json({orphanage_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/orphanscreate', async (req, res) => {
  const {orphan_id, name, age,orphanage_id} = req.body;
  try {
    await pool.query(
      'INSERT INTO orphans (orphan_id, name, age,orphanage_id) VALUES ($1, $2,
$3, $4)',
      [orphan_id, name, age,orphanage_id]
    );
    res.status(201).json({ orphan_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/servicescreate', async (req, res) => {
  const {service_id, name} = req.body;
  try {
    await pool.query(
      'INSERT INTO services (service_id, name) VALUES ($1, $2)',
      [service_id, name]
    );
    res.status(201).json({ service_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/shelterscreate', async (req, res) => {
  const {shelter_id, location, capacity,type} = req.body;
  try {
    await pool.query(
      'INSERT INTO shelters (shelter_id, location, capacity,type) VALUES ($1, $2,
$3, $4)',
      [shelter_id, location, capacity,type]
    );
    res.status(201).json({ shelter_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.post('/volunteerscreate', async (req, res) => {
  const {volunteer_id, name, skill,service_id} = req.body;
  try {
    await pool.query(
      'INSERT INTO volunteers (volunteer_id, name, skill,service_id) VALUES ($1,
$2, $3, $4)',
      [volunteer_id, name, skill,service_id]
    );
    res.status(201).json({ volunteer_id});
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

Graph Get Api

```
app.get('/donationbymonth', async (req, res) => {
  try {
    const result = await pool.query(`
      SELECT TO_CHAR(donation_date, 'YYYY-MM') AS month, SUM(donation_amount) AS total
      FROM donations
      GROUP BY month
      ORDER BY month;
    `);
    res.json(result.rows);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```