Project Description

Rice Bikes Repair Dashboard Assignment

Bike Repair Transactions Dashboard

You work for **Rice Bikes**, a local bike repair shop that tracks repair transactions, customers, and bike details in their system. The shop needs a centralized dashboard for viewing transactions in a clean, organized format.

The shop manager wants an internal webpage that displays all bike repair transactions in a table format, including customer and bike information. The page must be sorted by transaction date (most recent first).

You are a full-stack engineer responsible for building both:

- A backend API to surface the required data
- A frontend webpage that consumes the API and renders the table

The manager does not care about the specific tech stack, but your solution should be clean, readable, and easy to run locally.

Assignment

1. Create a Repository

Create a new public GitHub repository for this project.

The repository must contain:

- Backend code
- Frontend code
- Setup instructions in a README
- Database schema (migration file or SQL script)
- Sample seed data

---

2. Backend Requirements

Build a backend service that:

- Connects to a Postgresql database (any instance works, but our current setup runs on version 17)
- Exposes a REST API endpoint:

GET /api/transactions

This endpoint must:

- Return all repair transactions
- Include nested or joined customer and bike information
- Be sorted by `transaction_date` in descending order (most recent first)

Example Response Shape

```
[
 {
   "transaction_id": 101,
   "transaction_date": "2026-01-15",
   "total_cost": 85.00,
   "customer": {
     "id": 5,
     "first_name": "Alex",
     "last_name": "Nguyen",
     "email": "alex@example.com"
     “phone_number:” 0000000000
   },
   "bike": {
     "id": 9,
     "make": "Trek",
     "model": "Domane",
   }
 }
]
```

Backend Expectations

- Clean project structure
- Clear separation of concerns (routes, services, models, etc.)
- Input validation where appropriate
- At least one automated test (unit or integration)
- Clear instructions in README for running the backend locally

You may use any backend framework (Node, Django, Spring Boot, etc.).

3. Frontend Requirements

Build a webpage that:

- Fetches data from `/api/transactions`
- Displays the data in a table format
- Shows the following columns:

| Date | Customer Name | Email | Bike | Service | Cost |

The table must:

- Be sorted by date (oldest first)
- Render correctly with seeded data
- Be readable and organized

Bonus (optional):

- Add client-side sorting
- Add basic styling
- Add pagination

4. How We Should Run It

Your README must include:

- Tech stack used
- Setup instructions
- Database setup instructions
- How to seed data
- How to run backend
- How to run frontend
- Example curl command to test API

Example:

curl http://localhost:3000/api/transactions

5. Submission Instructions

Once complete:

1. Push your repository to GitHub.
2. Ensure the repository is public.
3. Email the link to:

ricebikes@gmail.com

The email subject line should be:

[Your Full Name] – Rice Bikes Repair Dashboard

In the email body, include:

- A short explanation of your tech stack
- Any assumptions you made
- Instructions if anything non-standard is required to run

Evaluation Criteria

We are looking for:

- Clean, readable code
- Thoughtful database design
- Correct sorting behavior
- Clear API structure
- Proper joins between transactions, customers, and bikes
- Working end-to-end functionality
- Clear documentation

This assignment is not meant to be tricky. We care about correctness, clarity, and maintainability more than flashy UI.

Seed SQL Page

```sql
-- ============================================
-- Rice Bikes Repair Database Setup & Seed
-- PostgreSQL Version
-- ============================================

-- Drop tables if they exist (safe to re-run)
DROP TABLE IF EXISTS repair_transactions;
DROP TABLE IF EXISTS bikes;
DROP TABLE IF EXISTS customers;

-- ============================================
-- Create Tables
-- ============================================

CREATE TABLE customers (
    customer_id SERIAL PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    phone_number TEXT
);

CREATE TABLE bikes (
    bike_id SERIAL PRIMARY KEY,
    make TEXT NOT NULL,
    model TEXT NOT NULL
);

CREATE TABLE repair_transactions (
    transaction_id SERIAL PRIMARY KEY,
    bike_id INT NOT NULL,
    customer_id INT NOT NULL,
    total_cost NUMERIC(10,2) NOT NULL,
    transaction_date DATE NOT NULL,
    CONSTRAINT fk_bike
        FOREIGN KEY (bike_id)
        REFERENCES bikes(bike_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_customer
        FOREIGN KEY (customer_id)
        REFERENCES customers(customer_id)
);

-- ============================================
-- Seed Customers
-- ================================================

INSERT INTO customers (first_name, last_name, email, phone_number) VALUES
('Alex', 'Nguyen', 'alex.nguyen@email.com', '7135550101'),
('Maria', 'Lopez', 'maria.lopez@email.com', '7135550102'),
('James', 'Carter', 'james.carter@email.com', '7135550103'),
('Sophie', 'Kim', 'sophie.kim@email.com', '7135550104'),
('Daniel', 'Reed', 'daniel.reed@email.com', '7135550105');

-- ================================================
-- Seed Bikes
-- ================================================

INSERT INTO bikes (make, model) VALUES
('Trek', 'Domane'),
('Specialized', 'Allez'),
('Giant', 'Defy'),
('Cannondale', 'Synapse'),
('Santa Cruz', 'Hightower');

-- ================================================
-- Seed Repair Transactions (10 total)
-- ================================================

INSERT INTO repair_transactions (bike_id, customer_id, total_cost, transaction_date) VALUES
(1, 1, 89.99, '2026-01-05'),
(2, 2, 120.00, '2026-01-08'),
(3, 3, 45.50, '2026-01-10'),
(4, 4, 200.00, '2026-01-12'),
(5, 5, 75.25, '2026-01-15'),
(1, 1, 150.00, '2026-01-18'),
(2, 3, 60.00, '2026-01-20'),
(3, 2, 95.75, '2026-01-22'),
(4, 5, 180.00, '2026-01-25'),
(5, 4, 55.00, '2026-01-28');

-- ================================================
-- Verification Query (Sorted by Date Desc)
-- ================================================

SELECT
    rt.transaction_id,
    rt.transaction_date,
    rt.total_cost,
    c.first_name,
    c.last_name,
    b.make,
    b.model
FROM repair_transactions rt
JOIN customers c ON rt.customer_id = c.customer_id
JOIN bikes b ON rt.bike_id = b.bike_id
ORDER BY rt.transaction_date DESC;
```