```
        res.status(200).send("Sucessfull")
    } catch (error) {
        next(error);
    }
})

// Create server
app.listen(process.env.PORT , ()=>{
    console.log(`Server has started on port ${process.env.PORT}`)
})
```

Do the same for backend and also explain their relationship with each frontend

README for Backend (Server-side)

Project Overview

This project provides backend services for managing student result data. It includes functionality for handling CSV data uploads, transforming and storing data in a MySQL database, and fetching student result details based on enrollment numbers.

Features

1. **CSV Data Handling**: Processes CSV data uploads from the frontend.
2. **Data Transformation**: Transforms CSV data into a structured format for database storage.
3. **Database Integration**: Stores student result data in a MySQL database.
4. **Data Retrieval**: Retrieves student result details based on enrollment numbers.
5. **Error Handling**: Provides error handling and response management for various operations.

Project Structure

bash

```
backend/
├── src/
│   ├── app.js
│   ├── database/
│   │   ├── dataFunctions.js
│   │   ├── getData.js
│   │   └── insertResults.js
│   ├── routes/
│   │   └── index.js
│   └── utils/
│       └── csvUtils.js
├── .env
├── package.json
└── README.md
```

Installation and Setup

1. **Clone the Repository**:

```sh
git clone <repository-url>
cd backend
```

2. **Install Dependencies**:

```sh
npm install
```

3. **Set Environment Variables**:
   Create a `.env` file in the root directory with the following variables:

```makefile
HOST=localhost
SQL_PORT=3306
USER=root
PASS=password
DATABASE=student_results
PORT=4000
```

4. **Start the Server**:

```sh
npm start
```

Components and Their Relationship with Frontend

1. `app.js`

- **Description**: Entry point of the backend application. Configures middleware, routes, and starts the server.
- **Relationship**: Handles HTTP requests from the frontend, routes them to appropriate endpoints (`/addFile`, `/addResult`, `/getData/:regNo`).

2. `dataFunctions.js`

- **Description**: Contains functions for reading CSV data, transforming data, creating database tables, and establishing MySQL database connections.
- **Relationship**: Used by `app.js` to handle CSV uploads (`/addFile` endpoint), create database

3. `insertResults.js`

- **Description**: Handles insertion of transformed result data into the MySQL database.
- **Relationship**: Called by `/addResult` endpoint in `app.js` to insert student result data received from the frontend.

4. `getData.js`

- **Description**: Fetches student result data from the MySQL database based on enrollment numbers.
- **Relationship**: Endpoint `/getData/:regNo` in `app.js` invokes `getDataMiddleware` function to retrieve and return student result data to the frontend.

5. `csvUtils.js`

- **Description**: Utility functions for handling CSV data streams.
- **Relationship**: Used internally by `dataFunctions.js` to read and parse CSV data uploaded from the frontend (`readCSV` function).

Data Flow and Block Diagram

Data Flow

1. **CSV Upload**:
   - Frontend uploads CSV file to `/addFile` endpoint.
   - Backend reads and processes CSV data, transforms it into structured JSON format (`transformData`).
   - Transformed data is stored in MySQL database (`insertResults`).
2. **Result Retrieval**:
   - Frontend requests student result data via `/getData/:regNo` endpoint, passing enrollment number as a parameter.
   - Backend retrieves data from MySQL database based on the enrollment number and sends it back to the frontend.

Block Diagram

plaintext

Explanation:

- **Frontend**: Initiates requests to upload CSV data and retrieve student results.
- **CSV Upload**: Sends CSV data to the backend for processing.
- **CSV Data Handling**: Reads and parses CSV data, transforms it, and stores it in the database.
- **Backend**: Receives CSV data, processes and stores it, and retrieves student results upon request.

Running the Project

1. **Start the Backend Server**:

```sh
npm start
```

2. **Ensure MySQL Database is Running**:
   - Ensure MySQL server is running with credentials specified in `.env`.
3. **Frontend Integration**:
   - Frontend should be configured to communicate with backend endpoints (`/addFile`, `/addResult`, `/getData/:regNo`) for CSV uploads, result storage, and retrieval.

Technologies Used

- **Node.js**: Backend JavaScript runtime environment.
- **Express**: Web framework for Node.js.
- **MySQL**: Relational database management system.
- **csv-parser**: CSV parsing library for Node.js.
- **mysql2**: MySQL client for Node.js, supports promises.

Conclusion

This backend project efficiently handles student result data management, from CSV upload processing to structured storage in a MySQL database. By following the provided instructions, developers can seamlessly integrate and extend this backend with the frontend to deliver a robust student result management system.