```
    "postinstall": "npm run install-all",
    "install-all": "cd adminfrontend && npm install && cd ../backend && npm install && cd
../frontend && npm install"
  },
  "author": "",
  "license": "ISC"
}
```

This is package.json in the main folder where all there frontend and backend exist so that i can run all of them . Now assuming that this whole files will be given in another computer to use that as server provide me proper steps to do in that computer to make server and run them properly

To set up and run your project on another computer (server), follow these steps:

Prerequisites

1. **Node.js**: Ensure Node.js is installed on the server. You can download it from Node.js official website.
2. **MySQL**: Install MySQL on the server if your backend relies on it for database operations.

Steps to Deploy and Run the Project

1. Clone the Project

Clone your project repository from your version control system (e.g., GitHub) onto the server:

bash

```bash
git clone <repository-url>
cd ctuniversityresult  # Navigate into your project directory
```

2. Set Environment Variables

Create a `.env` file in the root directory of the project to store environment-specific variables. These should include database credentials and any other configuration settings used in your project. Here's an example:

plaintext

```
# .env file

# Backend environment variables
HOST=localhost
SQL_PORT=3306
USER=root
PASS=password
```

```
DATABASE=student_results
PORT=4000
```

Replace the values with appropriate configurations based on your server setup.

3. Install Dependencies

Install the required dependencies for all three parts of your project (frontend, backend, and adminfrontend):

```bash
npm run install-all
```

This command, defined in your `package.json`, will navigate to each subdirectory (`adminfrontend`, `backend`, and `frontend`) and install dependencies specified in their respective `package.json` files.

4. Build Admin Frontend (if necessary)

If `adminfrontend` requires a build step (e.g., using webpack or similar):

```bash
cd adminfrontend
npm run build
cd ..
```

Ensure the build step generates static files that can be served by your backend.

5. Start the Application

Now, you can start the application using the following command, which utilizes `concurrently` to run frontend, backend, and adminfrontend concurrently:

```bash
npm start
```

This command, defined in your `package.json`, starts:

- **Frontend**: Runs `npm start` in the `frontend` directory.
- **Backend**: Runs `npm start` in the `backend` directory.
- **Admin Frontend**: Runs `npm run dev` (or your specified command) in the `adminfrontend` directory.

Additional Considerations

- **Database Setup**: Ensure MySQL is properly configured with the database and credentials specified in your `.env` file. You may need to create the database schema (`student_results`) if it doesn't exist.
- **Firewall and Security**: Configure firewall rules and security settings on the server to ensure it's accessible only through secure channels.
- **Environment-specific Configurations**: Adjust configurations (like API URLs, CORS settings) in your frontend and backend as per the server environment.
- **Monitoring and Logging**: Implement monitoring and logging to track application behavior and errors on the server.
- **Process Management**: Consider using process managers (e.g., PM2) to keep your Node.js applications running in the background and automatically restart them on failure or server restarts.

By following these steps, you should be able to deploy and run your `ctuniversityresult` project on a server successfully. Adjustments may be necessary based on specific server configurations or additional project requirements.