**Fitness App Backend**

This project is a simple backend for a fitness application built with Node.js and Express. It's designed to accept user details (weight, height, etc.) and then allow the user to upload a video for a fitness assessment.

**Project Structure**

* server.js: The main entry point for the application. It starts the server and sets up all the middleware.
* package.json: Defines the project dependencies and scripts.
* routes/: Contains the API route definitions.
  + userRoutes.js: Handles all API logic for user details and video uploads.
* public/: Contains static front-end files that are served to the user.
  + index.html: The initial form to collect user data.
  + video.html: The page where the user uploads their fitness test video.
* uploads/: This directory is automatically created and will store all the videos uploaded by users.

**How to Run the Application**

**Prerequisites**

You must have [Node.js](https://nodejs.org/) installed on your computer.

**Steps**

1. **Open a terminal or command prompt.**
2. **Navigate to the project directory:**
3. cd path/to/your/desktop/new work
4. **Install the necessary dependencies:** This command reads the package.json file and installs Express and Multer.
5. npm install
6. **Start the server:**
7. npm start
8. **Access the application:** You will see a message in the terminal: Server is running on http://localhost:3000. Open your web browser and go to this address: [http://localhost:3000](https://www.google.com/search?q=http://localhost:3000)

**How It Works**

1. When you visit the homepage, index.html is served. You can fill in your details and click "Proceed to Fitness Test".
2. The form data is sent via a POST request to the /api/details endpoint. The backend server receives this data, logs it to the console, and then redirects you to /video.html.
3. On the video upload page, you can select a video file.
4. When you click "Upload and Finish", the video is sent via a POST request to the /api/upload endpoint.
5. The multer middleware on the server intercepts this request, saves the video file to the uploads/ directory with a unique name, and then displays a success message.