Quiz API Documentation

1 Overveiw

This Quiz API is a backend system designed using **Node.js** and **Express.js**, with **MongoDB** as the database. The API allows users to create, manage, and attempt quizzes, while also handling authentication and authorization.

2 Technologies Used

- **Node.js** (Runtime Environment)
- **Express.js** (Web Framework)
- **MongoDB** (Database via Mongoose)
- **JWT (JSON Web Tokens)** for Authentication
- **bcrypt.js** for Password Hashing
- **CORS** for Handling Cross-Origin Requests
- **Morgan** for Request Logging

3 How to Run This Project

3.1 Prerequisites

Ensure you have the following installed:

- **Node.js** (v16+ recommended)
- **MongoDB** (Local or Cloud-based like MongoDB Atlas)

3.2 Installation

Clone the repository and navigate into the project folder:

```
git clone <repo-url>
cd quiz-api
npm install
```

3.3 Setup Environment Variables

Create a `.env` file in the root directory and add:

```
MONGO_URI=<your_mongodb_connection_string>
JWT_SECRET=<your_secret_key>
PORT=5000
```

3.4 Running the Server

For development (hot reload enabled):

```
npm run dev
```

For production:

```
npm start
```

4 API Endpoints

4.1 User Routes (/api/users)

- `POST /register` - Register a new user
- `POST /login` - Login a user
- `GET /me` - Get current user details (Auth required)
- `PUT /me` - Update profile (Auth required)
- `GET /quizzes` - Get quizzes created by the user
- `GET /attempts` - Get user quiz attempts

4.2 Quiz Routes (/api/quizzes)

- `GET /` - Fetch all quizzes
- `POST /` - Create a new quiz (Auth required)
- `PUT /:id` - Update a quiz (Quiz owner only)
- `DELETE /:id` - Delete a quiz (Quiz owner only)
- `POST /:id/questions` - Add a question
- `POST /:id/attempt` - Start an attempt

4.3 Attempt Routes (/api/attempts)

- `POST /:id/submit` - Submit quiz attempt
- `GET /:id` - Get attempt details

5 Dependencies

The project relies on the following Node.js packages:

```
{
  "bcryptjs": "^3.0.2",
  "colors": "^1.4.0",
  "cors": "^2.8.5",
  "dotenv": "^16.4.7",
  "express": "^4.21.2",
  "jsonwebtoken": "^9.0.2",
  "mongoose": "^8.11.0",
  "morgan": "^1.10.0"
}
```

6 Debugging & Fixes

During the development of this API, I faced several issues, and at first, I resolved them incorrectly, but after further debugging, I found the correct solutions. Here are some of the key problems and my journey to fixing them:

6.1 Users Couldn't Login After Registering

Initial Issue: Users registered successfully, but login failed due to password mismatch.

First Attempted Fix (Wrong): I mistakenly hashed the password twice before storing it.

```
userSchema.pre('save', async function(next) {
    if (!this.isModified('password')) return next();
    this.password = await bcrypt.hash(this.password, 10);
    this.password = await bcrypt.hash(this.password, 10);
    next();
});
```

Final Fix: Removed the extra hashing step.

```
userSchema.pre('save', async function(next) {
    if (!this.isModified('password')) return next();
    this.password = await bcrypt.hash(this.password, 10);
    next();
});
```

7 Contribution

Feel free to contribte - himavanthkar- GitHub.