**Auction API Documentation (Markdown)**

**Base URL:** http://localhost:3001/api

**Authentication**

**POST /auth/register**

Registers a new user.

**Request:**

json

{

"email": "user@example.com",

"password": "yourpassword"

}

**Response:**

* **201 Created**

{ "message": "User registered successfully", "user": { "id": 1, "email": "user@example.com" } }

**400 Bad Request**

{ "error": "Email already exists" }

**POST /auth/login**

Authenticates a user and returns a JWT token.

* **Request:**

{ "email": "user@example.com", "password": "yourpassword" }

**Response:**

* **200 OK**

{ "token": "jwt-token" }

401 Unauthorized

{ "error": "Invalid credentials" }

**Auction Items**

**GET /auction/getRecords**

Retrieves all auction items.

* **Headers:**
  + Authorization: Bearer <token>
* **Response:**
  + **200 OK**

[ { "id": 1, "title": "Auction Item 1", "description": "Description of the item", "startingBid": 100, "endDate": "2024-10-31" }, { "id": 2, "title": "Auction Item 2", "description": "Description of the item", "startingBid": 200, "endDate": "2024-11-15" } ]

**POST /auction/auction**

Creates a new auction item.

* **Headers:**
  + Authorization: Bearer <token>
* **Request:**

{ "title": "New Auction Item", "description": "Item description", "startingBid": 100, "endDate": "2024-11-01" }

**Response:**

* **201 Created**

{ "id": 3, "title": "New Auction Item", "description": "Item description", "startingBid": 100, "endDate": "2024-11-01" }

**PUT /auction/updateAuction/**

Updates an auction item.

* **Headers:**
  + Authorization: Bearer <token>
* **Request:**

{ "title": "Updated Title", "description": "Updated description", "startingBid": 150, "endDate": "2024-11-15" }

**Response:**

* **200 OK**

{ "id": 1, "title": "Updated Title", "description": "Updated description", "startingBid": 150, "endDate": "2024-11-15" }

**DELETE /auction/deleteAuction/**

Deletes an auction item.

* **Headers:**
  + Authorization: Bearer <token>
* **Response:**
  + **200 OK**

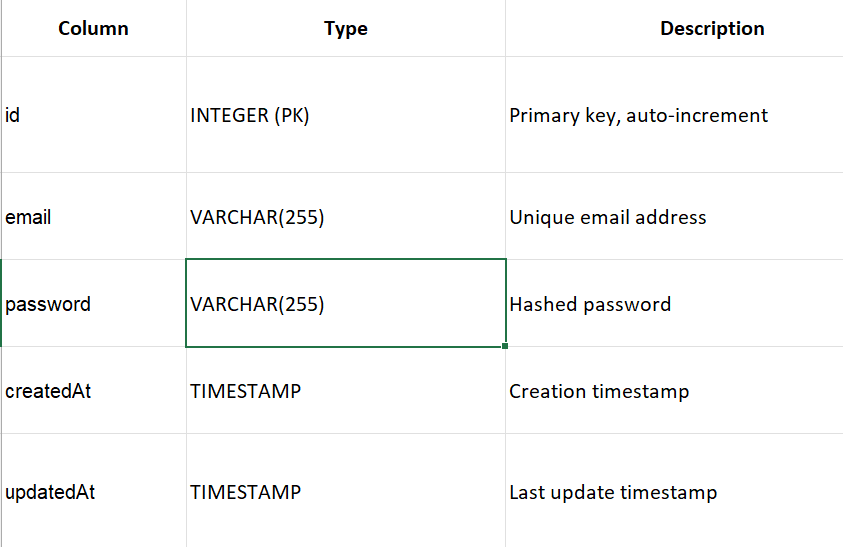
{ "message": "Auction item deleted successfully" }

**2. Database Schema Documentation**

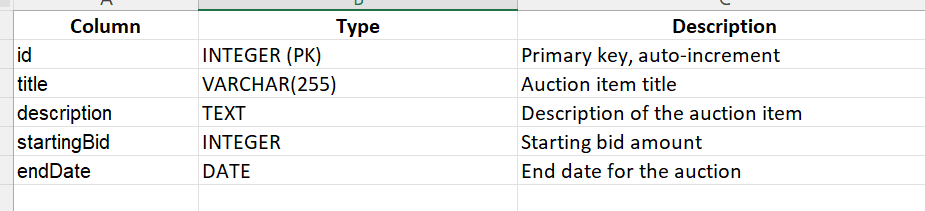
**Database: PostgreSQL**

* **ORM:** Sequelize
* **Tables:** Users, AuctionItems

**Users Table**

****

**AuctionItems** Table

****

**3. Design Decisions**

* **Authentication: JWT-based authentication was chosen for stateless user sessions and scalability.**
* **Sequelize ORM: Chosen for easy interaction with PostgreSQL and efficient handling of SQL queries and database migrations.**
* **Error Handling: Standardized error messages for all API endpoints, providing clear feedback to clients.**
* **Security: User passwords are hashed using Bcrypt to ensure secure password storage, and JWT is used for secure token-based authentication.**
* **CRUD Operations: Auction items can be created, updated, and deleted with authentication, ensuring that only logged-in users can perform these operations.**