**Event Management API with Database Connectivity**

**Objective:**

Create an event management API that allows users to register, log in, and manage events. Users can create events, invite other users, and manage RSVPs.

**Requirements**:

1. User Registration and Authentication:
   1. POST /register: Register a new user.
   2. POST /login: Authenticate a user and return a JWT token.
2. Event Management:
   1. POST /events: Create a new event (title, description, date, time).
   2. GET /events: Get all events for the authenticated user.
   3. PUT /events/:id: Update a specific event.
   4. DELETE /events/:id: Delete a specific event.
3. Invitations and RSVPs:
   1. POST /events/:id/invite: Invite users to an event.
   2. POST /events/:id/rsvp: RSVP to an event.
4. Optional
   1. Implement an endpoint to upload images for events using Multer.

**Technical Requirements:**

1. Node.js and Express.js:
   1. Set up an Express server.
   2. Use middleware for JSON parsing and JWT authentication.
2. JWT (JSON Web Tokens):
   1. Implement JWT-based authentication. Protect the /tasks endpoint to ensure only authenticated users can access it.
3. Database Connectivity:
   1. Connect to either MongoDB or MySQL to store users and tasks.
      1. For MongoDB, use the mongoose library.
      2. For MySQL, use the mysql2 or sequelize library.
4. Optional:
   1. Use Multer to handle file uploads.

**Instructions:**

1. Database Setup:
   1. Set up MongoDB or MySQL for storing user and task data. Provide a simple setup script or instructions for setting up the database.
   2. For MongoDB, create User and Task schemas using Mongoose.
   3. For MySQL, create User and Task tables with appropriate fields.
2. Database Configuration:
   1. Configure the application to connect to the database using environment variables for the connection details (e.g., DB\_HOST, DB\_USER, DB\_PASS, DB\_NAME).
3. Error Handling:
   1. Implement basic error handling and return appropriate HTTP status codes and messages for different error scenarios (e.g., user not found, task not found, unauthorized access).
4. Testing:
   1. Provide a script or instructions for testing the API using Axios. Demonstrate making requests to the registration, login, and task endpoints.
5. Documentation:
   1. Document the API endpoints, including request and response formats, in a README file.
   2. Include instructions for setting up the database and running the application.

**Deliverables:**

1. Source code for the Node.js application.
2. Instructions for setting up and running the application.
3. Instructions or scripts for setting up the database (MongoDB or MySQL).

**Evaluation Criteria:**

1. Correctness and completeness of the implementation.
2. Proper use of JWT for authentication.
3. Ability to connect to and interact with a database.
4. Error handling and response statuses.
5. Clarity and organization of the code.
6. Quality of the documentation.
7. (Optional) Proper implementation of file upload using Multer.