**1. Introduction**

* **Purpose of the API**: Provide a clear overview of what the API does. (e.g., "This API allows administrators to manage users, groups, and notifications within the system. It provides authentication, authorization, user CRUD operations, and group management functionality.")
* **Overview of Key Features**:
  + User registration and authentication (JWT, cookies).
  + Admin operations (create, update, delete users).
  + Group creation, member management, and notifications.

**2. Data Models**

* **User Model**:
  + Description: Represents a user in the system with basic details like username, password, and role.
  + Fields: firstname, lastname, username (unique), password, role.
* **Group Model**:
  + Description: Represents a group with a name, description, and members.
  + Fields: groupName, description, admin (user who created the group), members (array of user references).
* **Notification Model**:
  + Description: Stores notifications for user actions.
  + Fields: recipient, sender, message, date.

**3. API Endpoints**

* **POST /sign**: User registration (creates a new user with hashed password).
* **POST /login**: User login (authenticates and generates a JWT token).
* **PATCH /update**: Update user profile information (password, role).
* **DELETE /delete/:username**: Delete a user by username.
* **POST /createGroup**: Create a new group (admin only).
* **POST /addMember**: Add a member to an existing group (admin only).

**4. Authentication and Authorization**

* **JWT Authentication**:
  + Users authenticate with a JWT token. The token is issued upon successful login and expires in 1 hour.
* **Role-Based Access Control (RBAC)**:
  + Admin users can create, delete, or modify other users and groups.
  + Regular users can only update their own profile.

**5. Security Considerations**

* **Password Storage**: Use bcrypt for hashing passwords.
* **JWT Expiration**: Tokens expire after 1 hour to limit their validity.
* **HTTP-only Cookies**: Tokens are stored in cookies marked as httpOnly to protect against XSS attacks.

**6. Scalability Considerations**

* **MongoDB**: The database scales well for large datasets and supports high read/write operations.
* **Indexing**: Indexes on fields like username and groupName improve query performance.

**7. Error Handling**

* Provide meaningful responses for different error conditions (e.g., user not found, invalid password).
* Use appropriate HTTP status codes:
  + 201 Created for successful creation.
  + 400 Bad Request for invalid data.
  + 404 Not Found for missing resources.
  + 500 Internal Server Error for unexpected errors.

**8. Conclusion**

* Summarize the system’s security, scalability, and maintainability features.