**API Documentation Sample: TodoList API**

**Table of Contents**

1. Overview
2. Authentication
3. Endpoints
   1. User Endpoints
   2. Task Endpoints
4. Request & Response Examples
5. Error Handling
6. Best Practices
7. Versioning and Updates
8. Conclusion

**1. Overview**

**API Name:** TodoList API  
**Base URL:** https://api.todolist.com/v1

The **TodoList API** is a RESTful web service designed to manage user accounts and tasks within a simple to-do list application. This API enables developers to:

* Create, read, update, and delete tasks.
* Manage user registration and authentication.
* Organize tasks with optional due dates and priority levels.

**Target Audience:**

* Developers integrating a to-do list feature into their own applications.
* Teams building productivity apps requiring task management functionality.

**Supported Data Formats:**

* **JSON** for request and response bodies.

**Assumptions & Prerequisites:**

* Users must register to create and manage tasks.
* All requests require **HTTPS** for data security.

**2. Authentication**

**2.1 Registration and Login**

* **Public Endpoints** (no token required):
  + POST /users/register
  + POST /users/login

After a successful login, the server returns a **JSON Web Token (JWT)** in the response. This token must be included in the Authorization header of all subsequent requests.

**2.2 Usage of JWT Token**

* **Header Format:** Authorization: Bearer <token>
* **Token Expiration:** 24 hours from issuance, by default.
* **Token Refresh:** Endpoint available if needed (POST /users/refresh-token).

**3. Endpoints**

**3.1 User Endpoints**

**3.1.1 Register User**

POST /users/register

* **Description:** Creates a new user account.
* **Request Body (JSON):**
* {
* "username": "string",
* "email": "string",
* "password": "string"

}

* **Response:**
  + **201 Created** on success, with a userId returned.
  + **400 Bad Request** if the username or email is already taken.

**3.1.2 Login User**

POST /users/login

* **Description:** Authenticates an existing user and returns a JWT token.
* **Request Body (JSON):**
* {
* "email": "string",
* "password": "string"

}

* **Response:**
  + **200 OK** on success, returns a token and userId.
  + **401 Unauthorized** if credentials are invalid.

**3.1.3 Refresh Token (Optional)**

POST /users/refresh-token

* **Description:** Refreshes an expired or soon-to-expire JWT.
* **Headers:** Requires current valid token in Authorization: Bearer <token>.
* **Request Body (JSON):** None (token is provided in header).
* **Response:**
  + **200 OK** on success, returns a new token.
  + **401 Unauthorized** if the current token is invalid or missing.

**3.2 Task Endpoints**

**Note:** All Task endpoints require a valid JWT token.

**3.2.1 Create a Task**

POST /tasks

* **Description:** Adds a new task to the authenticated user’s to-do list.
* **Headers:** Authorization: Bearer <token>
* **Request Body (JSON):**
* {
* "title": "string",
* "description": "string",
* "dueDate": "YYYY-MM-DD",
* "priority": "LOW|MEDIUM|HIGH"

}

* **Response:**
  + **201 Created** if the task is added successfully, including a taskId.
  + **400 Bad Request** if required fields are missing.

**3.2.2 Retrieve All Tasks**

GET /tasks

* **Description:** Fetches all tasks for the authenticated user.
* **Headers:** Authorization: Bearer <token>
* **Query Parameters (Optional):**
  + ?status=completed|pending|all (default is all)
  + ?priority=LOW|MEDIUM|HIGH|all (default is all)
* **Response:**
  + **200 OK**, returns an array of Task objects.

**3.2.3 Retrieve Single Task**

GET /tasks/{taskId}

* **Description:** Retrieves a specific task by ID.
* **Headers:** Authorization: Bearer <token>
* **Response:**
  + **200 OK** with Task details.
  + **404 Not Found** if taskId does not exist for the user.

**3.2.4 Update a Task**

PUT /tasks/{taskId}

* **Description:** Updates an existing task.
* **Headers:** Authorization: Bearer <token>
* **Request Body (JSON):**
* {
* "title": "string",
* "description": "string",
* "dueDate": "YYYY-MM-DD",
* "priority": "LOW|MEDIUM|HIGH",
* "completed": true

}

* **Response:**
  + **200 OK** if the task is updated successfully.
  + **404 Not Found** if taskId does not exist for the user.

**3.2.5 Delete a Task**

DELETE /tasks/{taskId}

* **Description:** Removes a task from the user’s list.
* **Headers:** Authorization: Bearer <token>
* **Response:**
  + **204 No Content** on successful deletion.
  + **404 Not Found** if taskId does not exist.

**4. Request & Response Examples**

**4.1 Create a Task (Request)**

POST /tasks

Content-Type: application/json

Authorization: Bearer <token>

{

"title": "Buy Groceries",

"description": "Buy milk, eggs, and bread.",

"dueDate": "2024-01-15",

"priority": "MEDIUM"

}

**4.2 Create a Task (Response)**

{

"taskId": "abc123",

"title": "Buy Groceries",

"description": "Buy milk, eggs, and bread.",

"dueDate": "2024-01-15",

"priority": "MEDIUM",

"completed": false,

"createdAt": "2024-01-01T10:00:00Z"

}

**4.3 Retrieve All Tasks (Response)**

[

{

"taskId": "abc123",

"title": "Buy Groceries",

"description": "Buy milk, eggs, and bread.",

"dueDate": "2024-01-15",

"priority": "MEDIUM",

"completed": false,

"createdAt": "2024-01-01T10:00:00Z"

},

{

"taskId": "xyz789",

"title": "Finish Assignment",

"description": "Complete the math homework.",

"dueDate": "2024-01-20",

"priority": "HIGH",

"completed": false,

"createdAt": "2024-01-02T09:30:00Z"

}

]

**5. Error Handling**

1. **400 Bad Request**
   * Triggered when required fields are missing, invalid JSON is provided, or a parameter format is incorrect.
2. **401 Unauthorized**
   * Returned if the JWT token is missing or invalid. Users must re-authenticate or refresh their token.
3. **403 Forbidden**
   * Not used in all configurations, but may appear if a user attempts to access resources owned by another user.
4. **404 Not Found**
   * Triggered when the requested resource (e.g., taskId) does not exist.
5. **409 Conflict**
   * Returned when an attempt to create or update a resource results in a duplicate or conflict (e.g., two tasks with the same unique title if the system enforces uniqueness).
6. **500 Internal Server Error**
   * Indicates a server-side failure. Check logs for further diagnostics.

**Tip:** Include descriptive error messages in JSON format where possible to help clients diagnose issues.

**6. Best Practices**

1. **Use HTTPS**: Always send requests over a secure connection.
2. **Handle Tokens Securely**: Store JWT tokens on the client side in secure storage (e.g., HTTP-only cookies or secure token storage for mobile apps).
3. **Pagination**: If returning a large list of tasks, consider implementing pagination via query parameters (e.g., ?limit=10&offset=0).
4. **Rate Limiting**: If your API is public, you may want to enforce rate limits to prevent abuse.
5. **Data Validation**: Validate user input both client-side and server-side to avoid malformed requests.
6. **Logging and Monitoring**: Track request logs, errors, and performance metrics.

**7. Versioning and Updates**

* **Current Version**: v1
* **Future Changes**:
  + Endpoint path changes are introduced via /v2, /v3, etc.
  + Deprecation notices or warnings should be returned in response headers (e.g., Deprecation: true).
  + Backward compatibility considerations should be maintained until EOL for older versions.

**8. Conclusion**

The **TodoList API** offers a flexible yet straightforward interface for user and task management. By following the best practices outlined in this document, developers can build a robust integration that supports task creation, retrieval, and updates in a secure, efficient manner. Feel free to extend this API with additional features such as **reminders**, **task sharing**, or **progress tracking** according to your application’s needs.

For additional information or troubleshooting, please refer to our online documentation or contact our support team at **support@todolist.com**.

**Author:**  
*Frederick James (Technical Writer)*  
*Email: fwalterjames@gmail.com | GitHub:* [*https://github.com/fwalterjames*](https://github.com/fwalterjames)

**Last Updated:** *2/3/2025*

**End of Document**