API Documentation

Base URL

```
http://localhost:3001/api
```

Authentication

All protected endpoints require a JWT token in the Authorization header:

```
Authorization: Bearer <jwt_token>
```

Response Format

All API responses follow this format:

```json
{
  "message": "Success message",
  "data": { ... },
  "error": "Error message (if applicable)"
}
```

Authentication Endpoints

Register User

POST `/auth/register`

Register a new user account.

Request Body:

```json
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123"
}
```

Response:
```json
{
  "message": "User registered successfully",
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "USER",
    "status": "ACTIVE"
  },
  "token": "jwt_token"
}
```

Login User
POST `/auth/login`

Authenticate user and receive JWT token.

Request Body:
```json
{
```

```
  "email": "john@example.com",
  "password": "password123"
}
```

Response:
```json
{
  "message": "Login successful",
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "USER",
    "status": "ACTIVE"
  },
  "token": "jwt_token"
}
```

Get Current User
GET `/auth/me`

Get current authenticated user information.

Headers:
```
Authorization: Bearer <jwt_token>
```

Response:

```json
{
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "USER",
    "status": "ACTIVE",
    "profilePicture": "path/to/image.jpg"
  }
}
```

Refresh Token
POST `/auth/refresh`

Refresh JWT token.

Headers:
```
Authorization: Bearer <jwt_token>
```

Response:
```json
{
  "token": "new_jwt_token"
}
```

User Management Endpoints

Get All Users

GET `/users`

Get list of all users (Admin only).

Headers:

```
Authorization: Bearer <admin_jwt_token>
```

Query Parameters:

- `page` (optional): Page number for pagination
- `limit` (optional): Number of users per page

Response:

```json
{
  "users": [
    {
      "id": "uuid",
      "name": "John Doe",
      "email": "john@example.com",
      "role": "USER",
      "status": "ACTIVE",
      "createdAt": "2023-01-01T00:00:00Z"
    }
  ],
  "total": 10,
  "page": 1,
  "totalPages": 2
```

}
```

### Get User by ID
**GET** `/users/:id`

Get specific user information.

**Headers:**
```
Authorization: Bearer <jwt_token>
```

**Response:**
```json
{
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "USER",
    "status": "ACTIVE",
    "profilePicture": "path/to/image.jpg",
    "createdAt": "2023-01-01T00:00:00Z"
  }
}
```

Update User
PUT `/users/:id`
```

Update user information.

Headers:
```
Authorization: Bearer <jwt_token>
```

Request Body:
```json
{
  "name": "Updated Name",
  "email": "updated@example.com"
}
```

Response:
```json
{
  "message": "User updated successfully",
  "user": {
    "id": "uuid",
    "name": "Updated Name",
    "email": "updated@example.com",
    "role": "USER",
    "status": "ACTIVE"
  }
}
```

Update User Status
PATCH `/users/:id/status`

Update user status (Admin only).

Headers:
```
Authorization: Bearer <admin_jwt_token>
```

Request Body:
```json
{
  "status": "DISABLED"
}
```

Response:
```json
{
  "message": "User status updated successfully",
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "status": "DISABLED"
  }
}
```

Upload Profile Picture
POST `/users/:id/profile-picture`

Upload user profile picture.

Headers:
```
Authorization: Bearer <jwt_token>
Content-Type: multipart/form-data
```

Request Body:
```
profilePicture: <file>
```

Response:
```json
{
  "message": "Profile picture uploaded successfully",
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "profilePicture": "path/to/new-image.jpg"
  }
}
```

Todo Management Endpoints

Get Todos
GET `/todos`

Get todos with optional filters.

Headers:

```
Authorization: Bearer <jwt_token>
```

Query Parameters:

- `type` (optional): "assigned" | "created" | "all"
- `status` (optional): "PENDING" | "IN_PROGRESS" | "COMPLETED"
- `search` (optional): Search term for title/description

Response:

```json
{
  "todos": [
    {
      "id": "uuid",
      "title": "Complete project",
      "description": "Finish the todo management system",
      "status": "IN_PROGRESS",
      "dueDate": "2023-12-31T23:59:59Z",
      "order": 1,
      "createdAt": "2023-01-01T00:00:00Z",
      "createdBy": {
        "id": "uuid",
        "name": "Admin User",
        "email": "admin@example.com"
      },
      "assignedTo": {
        "id": "uuid",
        "name": "John Doe",
```

```
      "email": "john@example.com"
    },
    "files": [],
    "_count": {
      "files": 0
    }
  }
 ]
}
```

Get Todo by ID
GET `/todos/:id`

Get specific todo information.

Headers:
```
Authorization: Bearer <jwt_token>
```

Response:
```json
{
  "todo": {
    "id": "uuid",
    "title": "Complete project",
    "description": "Finish the todo management system",
    "status": "IN_PROGRESS",
    "dueDate": "2023-12-31T23:59:59Z",
    "order": 1,
```

```
    "createdBy": {
      "id": "uuid",
      "name": "Admin User"
    },
    "assignedTo": {
      "id": "uuid",
      "name": "John Doe"
    },
    "files": [
      {
        "id": "uuid",
        "filename": "document.pdf",
        "originalName": "Project Document.pdf",
        "size": 1024000,
        "mimeType": "application/pdf"
      }
    ]
  }
}
```

Create Todo
POST `/todos`

Create a new todo.

Headers:
```
Authorization: Bearer <jwt_token>
```

Request Body:

```json
{
  "title": "New Todo",
  "description": "Todo description",
  "assignedToId": "user_uuid",
  "dueDate": "2023-12-31T23:59:59Z"
}
```

Response:

```json
{
  "message": "Todo created successfully",
  "todo": {
    "id": "uuid",
    "title": "New Todo",
    "description": "Todo description",
    "status": "PENDING",
    "dueDate": "2023-12-31T23:59:59Z",
    "order": 1,
    "createdBy": {
      "id": "uuid",
      "name": "Current User"
    },
    "assignedTo": {
      "id": "uuid",
      "name": "Assigned User"
    }
  }
}
```

```

### Update Todo
**PUT** `/todos/:id`

Update existing todo.

**Headers:**
```
Authorization: Bearer <jwt_token>
```

**Request Body:**
```json
{
  "title": "Updated Todo",
  "description": "Updated description",
  "status": "COMPLETED",
  "dueDate": "2023-12-31T23:59:59Z"
}
```

Response:
```json
{
  "message": "Todo updated successfully",
  "todo": {
    "id": "uuid",
    "title": "Updated Todo",
    "description": "Updated description",
    "status": "COMPLETED"
```
```

```
  }
}
```

Delete Todo

DELETE `/todos/:id`

Delete a todo.

Headers:

```
Authorization: Bearer <jwt_token>
```

Response:

```json
{
  "message": "Todo deleted successfully"
}
```

Reorder Todos

PATCH `/todos/reorder`

Update the order of multiple todos.

Headers:

```
Authorization: Bearer <jwt_token>
```

Request Body:

```json
{
  "todoUpdates": [
    {
      "id": "uuid1",
      "order": 1
    },
    {
      "id": "uuid2",
      "order": 2
    }
  ]
}
```

Response:

```json
{
  "message": "Todos reordered successfully"
}
```

Get Assignable Users

GET `/todos/users/assignable`

Get list of users that can be assigned todos.

Headers:

```
Authorization: Bearer <jwt_token>
```

```

**Response:**

```json
{
  "users": [
    {
      "id": "uuid",
      "name": "John Doe",
      "email": "john@example.com",
      "status": "ACTIVE"
    }
  ]
}
```

File Management Endpoints

Upload Files

POST `/files/upload/:todoId`

Upload files to a specific todo.

Headers:

```
Authorization: Bearer <jwt_token>
Content-Type: multipart/form-data
```

Request Body:

```

files: <file1>, <file2>, ...
```

Response:
```json
{
  "message": "Files uploaded successfully",
  "files": [
    {
      "id": "uuid",
      "filename": "document_123456.pdf",
      "originalName": "Project Document.pdf",
      "path": "uploads/todos/document_123456.pdf",
      "size": 1024000,
      "mimeType": "application/pdf",
      "todoId": "todo_uuid"
    }
  ]
}
```

Get All Files
GET `/files`

Get all files accessible to the current user.

Headers:
```
Authorization: Bearer <jwt_token>
```
```

Response:

```json
{
  "files": [
    {
      "id": "uuid",
      "filename": "document_123456.pdf",
      "originalName": "Project Document.pdf",
      "size": 1024000,
      "mimeType": "application/pdf",
      "createdAt": "2023-01-01T00:00:00Z",
      "todo": {
        "id": "uuid",
        "title": "Complete project",
        "status": "IN_PROGRESS",
        "createdBy": {
          "id": "uuid",
          "name": "Admin User"
        },
        "assignedTo": {
          "id": "uuid",
          "name": "John Doe"
        }
      }
    }
  ]
}
```

Get Todo Files

GET `/files/todo/:todoId`

Get all files for a specific todo.

Headers:
```
Authorization: Bearer <jwt_token>
```

Response:
```json
{
  "files": [
    {
      "id": "uuid",
      "filename": "document_123456.pdf",
      "originalName": "Project Document.pdf",
      "size": 1024000,
      "mimeType": "application/pdf",
      "createdAt": "2023-01-01T00:00:00Z"
    }
  ]
}
```

Download File
GET `/files/download/:id`

Download a specific file.

Headers:
```
```

Authorization: Bearer <jwt_token>
```

**Response:**

File download with appropriate headers.

### Delete File

**DELETE** `/files/:id`

Delete a file.

**Headers:**

```
Authorization: Bearer <jwt_token>
```

Response:

```json
{
  "message": "File deleted successfully"
}
```

Error Responses

400 Bad Request

```json
{
  "error": "Validation error message",
  "details": [
    {

```
      "field": "email",
      "message": "Invalid email format"
    }
  ]
}
```

401 Unauthorized

```json
{
  "error": "Authentication required"
}
```

403 Forbidden

```json
{
  "error": "Insufficient permissions"
}
```

404 Not Found

```json
{
  "error": "Resource not found"
}
```

500 Internal Server Error

```json
{
```

```
  "error": "Internal server error"
}
```

Rate Limiting

API endpoints are rate-limited to prevent abuse:

- Authentication endpoints: 5 requests per minute per IP
- General endpoints: 100 requests per minute per user
- File upload endpoints: 10 requests per minute per user

Pagination

Endpoints that return lists support pagination:

- `page`: Page number (default: 1)
- `limit`: Items per page (default: 10, max: 100)

Response includes pagination metadata:

```json
{
  "data": [...],
  "pagination": {
    "page": 1,
    "limit": 10,
    "total": 50,
    "totalPages": 5,
    "hasNext": true,
    "hasPrev": false
  }
}
```