# Project Requirement Document: Remote Work Collaboration Platform

Introduction The objective of this project is to develop a collaborative platform for remote work, which includes a web application and APIs to facilitate efficient remote team collaboration and project management. The platform will enable teams to manage projects, tasks, documents, communication, and user roles in a remote work environment. It will provide various features and APIs for different user roles to perform specific actions.

## Project Scope

The project will involve the development of the following entities:

* User: Represents remote team members.
* Project: Describes a set of tasks and activities.
* Task: Represents individual work items and assignments within a project.
* Document: Records shared files and documents.
* Chat: Facilitates team communication and messaging.
* Role: Describes predefined roles, including Project Manager and Team Member.

User:

1. **UserID**: A unique identifier for each user.
2. **Username**: The name chosen by the user for logging into the system.
3. **Password**: Encrypted password for user authentication.
4. **Email**: The email address of the user.
5. **FirstName**: The first name of the user.
6. **LastName**: The last name of the user.
7. **RoleID**: The identifier of the role assigned to the user (links to Role entity).
8. **LastLogin**: Timestamp of the last time the user logged into the system.
9. **IsActive**: A boolean flag indicating whether the user account is active.

Role:

1. **RoleID**: A unique identifier for each role.
2. **RoleName**: The name of the role, such as “Administrator”, “Project Manager”, or “Team Member”.
3. **RoleDescription**: A brief description of the role and its responsibilities.

Project:

 **ProjectID**: A unique identifier for each project.

 **ProjectName**: The name of the project.

 **ProjectDescription**: A brief description of the project.

 **StartDate**: The date when the project starts.

 **EndDate**: The estimated date when the project is expected to end.

 **Status**: The current status of the project (e.g., “Not Started”, “In Progress”, “Completed”).

 **ProjectManagerID**: The identifier of the user who is the project manager (links to User entity).

Task:

 **TaskID**: A unique identifier for each task.

 **TaskName**: The name of the task.

 **TaskDescription**: A brief description of the task.

 **StartDate**: The date when the task is scheduled to start.

 **DueDate**: The date by which the task should be completed.

 **Status**: The current status of the task (e.g., “Not Started”, “In Progress”, “Completed”).

 **AssignedToUserID**: The identifier of the user to whom the task is assigned (links to User entity).

 **ProjectID**: The identifier of the project that the task is part of (links to Project entity).

Document:

 **DocumentID**: A unique identifier for each document.

 **DocumentName**: The name of the document.

 **UploadDate**: The date when the document was uploaded.

 **UploadedByUserID**: The identifier of the user who uploaded the document (links to User entity).

 **ProjectID**: The identifier of the project that the document is part of (links to Project entity).

## Functional Requirements

## 3.1 Authentication and Authorization

Authentication API (Anonymous Access): Develop an API to allow users to log in and obtain an authentication token. This API should be accessible without authentication.

## 3.2 User Management

User APIs: Create APIs to:

* Get a list of all users.
* Get a user by their user ID.
* Create a new user.
* Delete a user.
* Update user details.
* Assign User to a Role.
* These APIs can only be accessed by users with the "Administrator" role. Anonymous requests and users with other roles will not have access.

## 3.3 Role Management:

Develop APIs to:

* Get a list of all existing roles.
* Get a role based on role ID.

These APIs can only be accessed by users with the "Administrator" role. Anonymous requests and users with other roles will not have access.

There are two roles in the application: "Administrator" and "Project Manager."

## 3.4 Project Management

Project APIs: Develop APIs to:

* Create a new project.
* Edit project details.
* Delete a project.
* Get a list of all projects.
* Get a single project based on project ID.
* These APIs can only be accessed by users with the "Project Manager" role.

## 3.5 Task Management

Create APIs to:

* Add a new task within a project.
* Edit task details.
* Delete a task.
* Get a list of all tasks within a project.
* Get a single task by task ID.
* These APIs can be accessed by users with the "Project Manager" and "Team Member" roles.

## 3.6 Document Management

Develop APIs to:

* Upload and share documents within a project.
* Edit document details.
* Delete a document.
* Get a list of all documents within a project.
* Get a single document by document ID.
* These APIs can be accessed by users with the "Project Manager" and "Team Member" roles.

## 3.7 Communication and Chat

Create APIs for team communication and messaging within projects.

* Send and receive chat messages.
* Create chat groups within a project.
* Share files and documents within chat.
* Get chat history.
* These APIs can be accessed by users with the "Project Manager" and "Team Member" roles.

## Non-Functional Requirements

### 4.1 Security Implement secure token-based authentication and authorization.

Ensure data privacy and protection for sensitive project and document information.

### 4.2 Miscellaneous

Implement middleware for the following functionalities:

* Authentication Middleware: Check if the token associated with the incoming request is valid or not.
* Authorization Middleware: Enforce role-based access control. Check if the incoming request is associated with any roles in the claim; otherwise, reject the request.
* Logging Middleware: Record and log API requests and responses. This middleware will log important information about each API request, such as request method, endpoint, user, and timestamps. It helps in monitoring and debugging the system.
* Exception middleware: Handle exceptions gracefully. This middleware will catch unhandled exceptions and provide standardized error responses to clients. It ensures that errors are well-documented and do not expose sensitive information.

## Conclusion

The Remote Work Collaboration Platform project aims to create a secure and efficient system that allows remote team members to collaborate on projects, tasks, documents, and communication. The APIs have been categorized based on user roles to ensure proper access control and security. The project should adhere to industry best practices and meet all the specified requirements.