### **Project Planning Document**

#### **Title: Todo List Application Project Plan**

### **1. Introduction**

The Todo List Application is designed to help users manage their tasks effectively. It allows users to create, read, update, and delete todos, manage groups, and assign tasks within groups. The application will support user authentication and role-based access control to ensure data security.

### **2. Functional Requirements**

1. **User Management**
   * User registration, login, and logout.
   * View and update user profile.
2. **Todo Management**
   * Create, read, update, and delete todos.
   * Mark todos as completed.
3. **Group Management**
   * Create, join, and leave groups.
   * Assign todos to groups.
   * Manage group members (admin and members).
4. **Role-Based Access Control**
   * Admin and user roles.
   * Admin can manage groups and group members.
   * Users can manage their own todos and view group todos.

### **3. Non-Functional Requirements**

* **Performance**: The application should handle up to 1000 concurrent users.(not an immediate requirement)
* **Security**: User data should be encrypted, and secure communication protocols should be used.
* **Usability**: The application should be user-friendly and responsive, accessible from various devices.

### **4. High-Level Architecture**

* **Backend**: Spring Boot application with RESTful APIs.
* **Frontend**: React application.
* **Database**: MySQL and postgres data storage.
* **In-Memory Caching Database**: Redis
* **Authentication**: OAuth2 for secure user authentication.

### **5. Milestones and Timeline**

1. **Week 1**: Setup project, implement user authentication.
2. **Week 2**: Implement todo management.
3. **Week 3**: Implement group management.
4. **Week 4**: Implement role-based access control, finalize frontend, testing, deployment.

### **Data Models and Database Design**

#### **Title: Todo List Application Database Design**

### **1. ERD Diagrams**

(Placeholder for your actual ERD diagram)

### **2. Table Structures**

* **User**
  + id: Integer (Primary Key)
  + username: String (Unique)
  + password: String
  + roles: String
* **Role (Optional , if requirement occurs)**
  + id: Integer (Primary Key)
  + name: String (Unique)
* **Group**
  + id: Integer (Primary Key)
  + name: String
  + admin\_id: Integer (Foreign Key to User)
* **Todo**
  + id: Integer (Primary Key)
  + title: String
  + completed: Boolean
  + startTime: LocalDateTime
  + endTime: LocalDateTime
  + userId: Integer (Foreign Key to Group)
* **User\_Role (Optional , if requirement occurs)**
  + user\_id: Integer (Foreign Key to User)
  + role\_id: Integer (Foreign Key to Role)
* **Group\_Members**
  + group\_id: Integer (Foreign Key to Group)
  + user\_id: Integer (Foreign Key to User)

### **3. Relationships**

* One-to-many relationship between User and Todo.
* Many-to-many relationship between User and Role.**(Optional , if requirement occurs)**
* One-to-many relationship between Group and Todo.
* Many-to-many relationship between Group and User through Group\_Members.

### **API Design**

#### **Title: Todo List Application API Documentation**

### **1. Introduction**

This document provides details of the RESTful APIs for the Todo List Application, including endpoints, request and response formats, and authentication requirements.

### **2. Authentication**

The application uses OAuth2 for secure authentication. Users must obtain an access token to access protected endpoints.

### **3. Endpoints**

* **User Endpoints**
  + POST /api/users - Create a new user.
  + POST /api/auth/login - User login.
  + GET /api/users/{id} - Get user details.
* **Todo Endpoints**
  + GET /api/todos - Get all todos.
  + POST /api/todos - Create a new todo.
  + PUT /api/todos/{id} - Update a todo.
  + DELETE /api/todos/{id} - Delete a todo.
* **Group Endpoints**
  + POST /api/groups - Create a new group.
  + GET /api/groups - Get all groups.
  + POST /api/groups/{id}/members - Add a member to a group.
  + GET /api/groups/{id}/todos - Get todos of a group.

### **Security Design**

#### **Title: Todo List Application Security Design**

### **1. OAuth2 Implementation**

* **Grant Types**: Authorization Code, Refresh Token
* **Token Endpoint**: /oauth/token
* **Authorization Endpoint**: /oauth/authorize

### **2. Role-Based Access Control (RBAC)**

* **Roles**:
  + **Admin**: Can manage groups, members, and todos within groups.
  + **User**: Can manage personal todos and view group todos.
* **Permissions**:
  + **Admin**:
    - Create, update, delete groups.
    - Add and remove group members.
    - Manage todos within groups.
  + **User**:
    - Create, update, delete personal todos.
    - View group todos.

### **3. Security Protocols**

* **Data Encryption**: Use HTTPS for secure data transmission.
* **Password Storage**: Store passwords using a strong hashing algorithm (e.g., bcrypt).
* **Access Control**: Implement access control checks in the backend to ensure users can only access their data or data they have permissions for.